Analyse et visualisation des données de Covid-19 Maroc

Présenté par :

LAGHYATI Moad

Première *année* de *master*– Science des données et Big Data

Sous la supervision de ***Pr. R. Oulad Haj THAMI et Pr .A. Youssfi Allaoui***

Table des matières

Introduction

La **maladie à coronavirus 2019**, ou le **Covid-19**, est une maladie infectieuse émergente de type zoonose virale causée par la souche de coronavirus SARS-CoV-2. Les symptômes les plus fréquents sont la fièvre, la toux, la fatigue et la gêne respiratoire. Dans les formes les plus graves, l'apparition d'un syndrome de détresse respiratoire aiguë peut entraîner la mort, notamment chez les personnes plus fragiles du fait de leur âge ou en cas de comorbidités. Une autre complication mortelle est une réponse exacerbée du système immunitaire inné (choc cytokinique).

Il existe un taux important de formes asymptomatiques. La transmission interhumaine se fait surtout via des gouttelettes respiratoires, postillons comme pour la grippe saisonnière, surtout lors de la parole, de la toux et des éternuements ou lorsque le contact d'une surface contaminée est suivi par celui d'une muqueuse du visage (bouche, nez, yeux). La période d'incubation est en moyenne de 5 à 6 jours, avec des extrêmes pouvant aller de deux à quatorze jours.

Le 2 mars marque le premier cas confirmé de COVID19 au Maroc. Depuis lors, le pays a connu des hauts et des bas et le gouvernement a pris très tôt des mesures préventives.

Dans cet mini projet , nous verrons comment l'épidémie a évolué au fil des mois et répondrons aux questions suivantes:

- Etat de l'épidémie au Maroc.
- Comment les clusters ont affecté chacune 13 régions du Maroc.
- Évaluer les mesures et efforts déployés par le gouvernement marocain depuis le début de l'internement à ce jour.

1 - les outils utilisés :

Un seul jeu de données est utilisé :

- covid19_morocco : contient les cas confirmés, décès, Guérisons et exclus au fil du temps pour chaque région jusqu'au 29-05-2020

	Date	Confirmed	Deaths	Recovered	Excluded	Beni Mellal-Khenifra	Casablanca-Settat	Draa-Tafilalet	Dakhla-Oued Ed-Dahab	Fes-Meknes	Guelmim-Oued Noun	Laayoune-Sakia El Hamra	Marrakesh-Safi	Oriental	Rabat-Sale-Kenitra	So M
0	02/03/2020	1.0	NaN	NaN	28.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	03/03/2020	NaN	NaN	NaN	32.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	04/03/2020	2.0	NaN	NaN	34.0	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	05/03/2020	NaN	NaN	NaN	40.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	06/03/2020	NaN	NaN	NaN	50.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
84	25/05/2020	7532.0	200.0	4774.0	144671.0	112.0	2457.0	585.0	5.0	994.0	42.0	3.0	1321.0	186.0	695.0	
85	26/05/2020	7577.0	202.0	4881.0	153788.0	112.0	2495.0	585.0	5.0	994.0	43.0	3.0	1326.0	185.0	696.0	
86	27/05/2020	7601.0	202.0	4978.0	163112.0	114.0	2506.0	586.0	5.0	997.0	43.0	3.0	1327.0	186.0	697.0	

- Python :
  Est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort.

- Pandas :
  est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en

particulier des structures de données et des opérations de manipulation de tableaux

- Plotly Express :
  Plotly Express est l'interface de haut niveau facile à utiliser de Plotly, qui fonctionne sur une variété de types de données et produit des figures faciles à styliser. Chaque fonction Plotly Express renvoie un objet graph_objects.Figure dont les données et la disposition ont été préremplies en fonction des arguments fournis.

- Folium :
  folium facilite la visualisation des données qui ont été manipulées en Python sur une carte de brochure interactive. Il permet à la fois la liaison de données à une carte pour les visualisations choroplèthes ainsi que la transmission de visualisations vectorielles / raster / HTML riches comme marqueurs sur la carte.

2 - Data Visualisation :

Nous faisons une analyse exploratoire des données sur le premier ensemble de données :

- Nous observons la forme de dataframe (89 lignes et 17 colonnes):

```
cov_df.shape
```

```
(89, 17)
```

- Nous exécutons la fonction info () pour obtenir des informations détaillées sur chaque colonne :

```
cov_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89 entries, 0 to 88
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Date                       89 non-null     object
 1   Confirmed                  83 non-null     float64
 2   Deaths                     72 non-null     float64
 3   Recovered                  71 non-null     float64
 4   Excluded                   80 non-null     float64
 5   Beni Mellal-Khenifra       69 non-null     float64
 6   Casablanca-Settat          72 non-null     float64
 7   Draa-Tafilalet             69 non-null     float64
 8   Dakhla-Oued Ed-Dahab       69 non-null     float64
 9   Fes-Meknes                 70 non-null     float64
 10  Guelmim-Oued Noun          69 non-null     float64
 11  Laayoune-Sakia El Hamra    69 non-null     float64
 12  Marrakesh-Safi             71 non-null     float64
 13  Oriental                   69 non-null     float64
 14  Rabat-Sale-Kenitra         69 non-null     float64
 15  Souss-Massa                69 non-null     float64
 16  Tanger-Tetouan-Al Hoceima  69 non-null     float64
dtypes: float64(16), object(1)
memory usage: 11.9+ KB
```

- Ensuite, nous vérifions le nombre de valeurs NaN dans dataframe:

```
cov_df.isna().sum(axis=0)
```

```
Date                          0
Confirmed                     6
Deaths                       17
Recovered                    18
Excluded                      9
Beni Mellal-Khenifra         20
Casablanca-Settat            17
Draa-Tafilalet               20
Dakhla-Oued Ed-Dahab         20
Fes-Meknes                   19
Guelmim-Oued Noun            20
Laayoune-Sakia El Hamra      20
Marrakesh-Safi               18
Oriental                     20
Rabat-Sale-Kenitra           20
Souss-Massa                  20
Tanger-Tetouan-Al Hoceima    20
dtype: int64
```

- Étant donné que nous avons des valeurs NaN et que nous n'avons pas d'autres statuts de cas qui seront utiles à notre analyse (comme le nombre de cas testés et actifs), nous les créons également.
  Enfin, nous convertissons la colonne Date au format datetime pour mieux la manipuler :

```
#Remplacer les donneés Null par Des 0
cov_df.fillna(value=0,inplace=True)

#Creér nouvelle colonne qui vont nous aider dans notre visualisation
cov_df["Tested"]=cov_df["Confirmed"]+cov_df["Excluded"]
cov_df["Active"]=cov_df["Confirmed"]-(cov_df['Deaths']+cov_df["Recovered"])

#convertir la colonne de date au format datetime pour mieux la manipuler

cov_df["Date"]=pd.to_datetime(cov_df['Date'],dayfirst=True)

cov_df.head()
```

Après les modifications :

	Date	Confirmed	Deaths	Recovered	Excluded	Beni Mellal-Khenifra	Casablanca-Settat	Draa-Tafilalet	Dakhla-Oued Ed-Dahab	Fes-Meknes	Guelmim-Oued Noun	Laayoune-Sakia El Hamra	Marrakesh-Safi	Oriental	Rabat-Sale-Kenitra
0	2020-03-02	1.0	0.0	0.0	28.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2020-03-03	0.0	0.0	0.0	32.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2020-03-04	2.0	0.0	0.0	34.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	2020-03-05	0.0	0.0	0.0	40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	2020-03-06	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Étant donné que la structure du cadre de données ci-dessus n'est pas appropriée pour le plotting (Plotly) parce que que la date doit être l'index et que les régions doivent se trouver dans la même colonne «Région», les étapes suivantes ont été suivies :

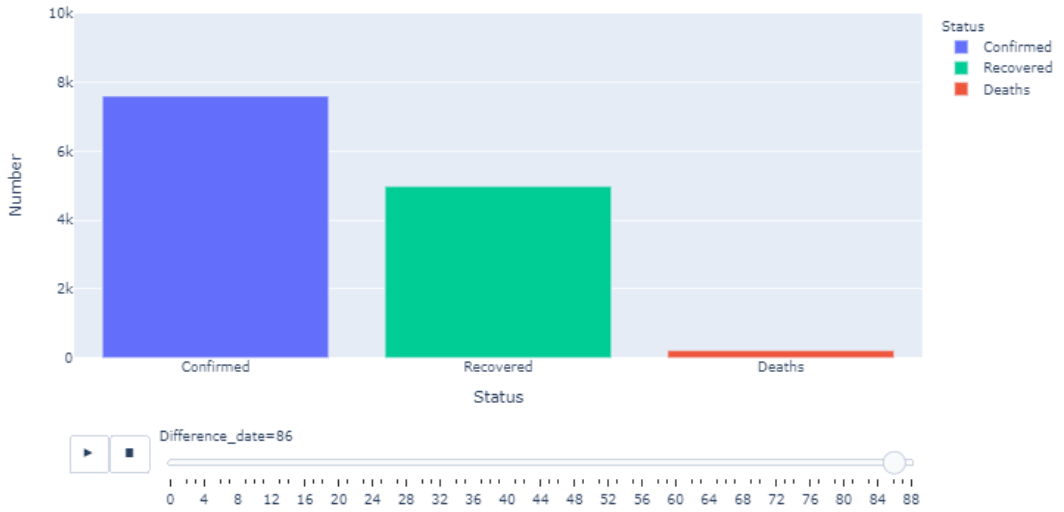
- J’ai fait Melted (le processus de pivotement d'un DataFrame du format large au format long) sur le dataframe.

- J'ai ajouté une colonne "Date de référence" avec le type de données datetime correct en tant que copie de la première valeur de la colonne Date

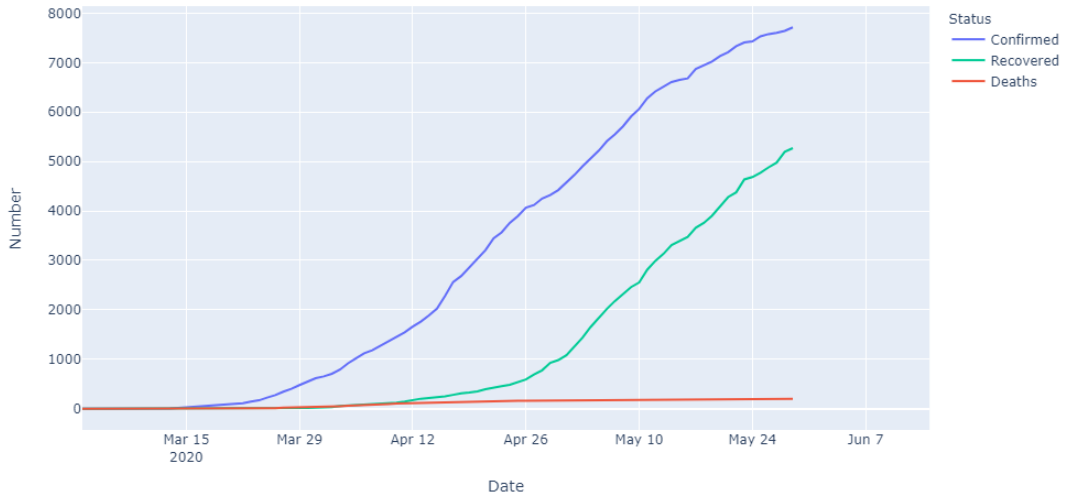
- J'ai créé une nouvelle "date de différence" qui est la différence entre la "date de référence" et la colonne "date", et garde donc une trace du nombre de jours écoulés

```
# melt le datafrmae cov_df pour obtenir une mieux structure de dataframe
updated_cov_df = pd.melt(cov_df, id_vars=['Date'], var_name='Status', value_vars=['Confirmed', 'Recovered', 'Deaths'], value_nam

#conversion de la colonne du timestamp en date type

updated_cov_df['Date'] = pd.to_datetime(updated_cov_df['Date']).dt.date
date_of_reference = datetime.datetime(2020, 3, 2)
updated_cov_df['Date_of_reference'] = date_of_reference
#print(updated_cov_df)
updated_cov_df['Date_of_reference'] = pd.to_datetime(updated_cov_df['Date_of_reference']).dt.date

#création d'une différence de date entre les colonnes date et date_of_references

updated_cov_df['Difference_date'] = updated_cov_df['Date'] - updated_cov_df['Date_of_reference']
updated_cov_df['Difference_date'] = pd.to_numeric(updated_cov_df['Difference_date'].dt.days, downcast='integer')

print(updated_cov_df)

fig = px.bar(updated_cov_df, x="Status", y="Number", color="Status", animation_frame="Difference_date",
             range_y=[0,10000], color_discrete_sequence=['#636EFA', '#00CC96', '#EF553B '])

fig.show()

# plotly
fig = px.line(updated_cov_df, x='Date', y='Number', color='Status', color_discrete_sequence=['#636EFA', '#00CC96', '#EF553B '])

# Show plot
fig.show()
```

En utilisant la bibliothèque Plotly, j'ai tracé le statut Confirmé, Récupéré et Décès et ajouté une option d'animation au tracé :



- Le nombre de cas confirmés a commencé très lentement et a rapidement augmenté avec l'augmentation des décès.
  Le graphique suivant montre une vue plus détaillée des trois états (Confirmed, Deaths, Recovered):



- Suivant le même principe précèdent j'ai fait le Melt sur les Colonnes Testé, Confirmé, décès et la Date comme variable d'identification :

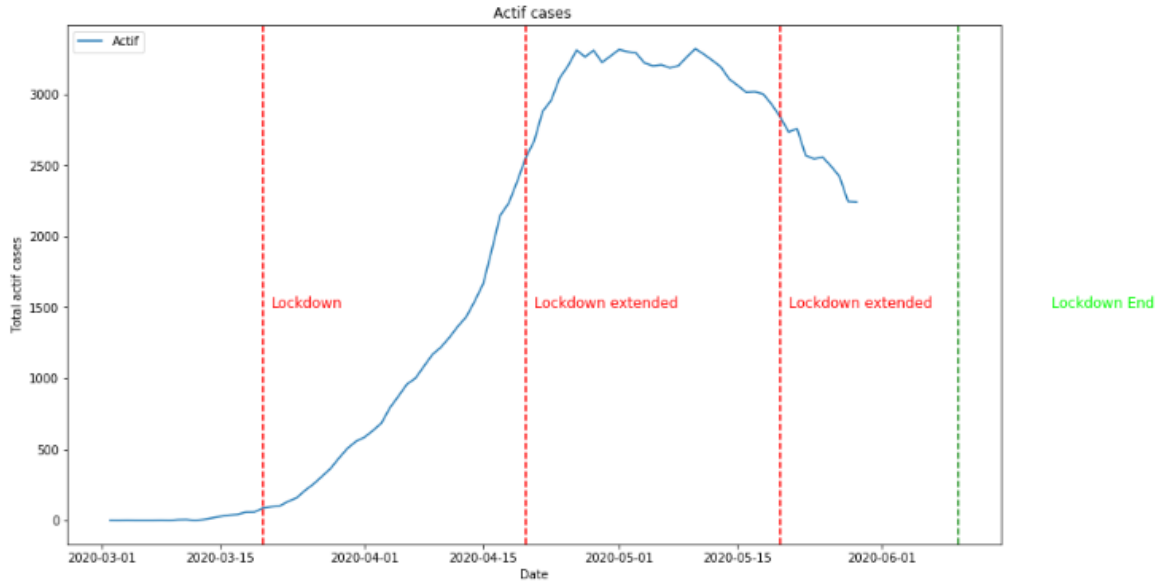
```
updated_cov_df = pd.melt(cov_df, id_vars=['Date'], var_name='Status', value_vars=['Tested', 'Confirmed', 'Deaths'], value_name='

#conversion de la colonne du timestamp en date type

updated_cov_df['Date'] = pd.to_datetime(updated_cov_df['Date']).dt.date
date_of_reference = datetime.datetime(2020, 3, 2)
updated_cov_df['Date_of_reference'] = date_of_reference
#print(updated_cov_df)
updated_cov_df['Date_of_reference'] = pd.to_datetime(updated_cov_df['Date_of_reference']).dt.date

#création d'une différence de date entre les colonnes date et date_of_references

updated_cov_df['Difference_date'] = updated_cov_df['Date'] - updated_cov_df['Date_of_reference']
updated_cov_df['Difference_date'] = pd.to_numeric(updated_cov_df['Difference_date'].dt.days, downcast='integer')

print(updated_cov_df)

fig = px.bar(updated_cov_df, x="Status", y="Number", color="Status", animation_frame="Difference_date",
             range_y=[0,10000], color_discrete_sequence=['#636EFA', '#00CC96', '#EF553B '])

fig.show()

# plotly
fig = px.line(updated_cov_df, x='Date', y='Number', color='Status', color_discrete_sequence=['#636EFA', '#00CC96', '#EF553B '])

# Show plot
fig.show()
```

Résultat :

Le gouvernement marocain a fait de gros efforts pour augmenter le nombre de tests au fil du temps et essayer de contenir le virus le plus rapidement et le plus possible. Alors que le nombre de tests augmente presque exponentiellement, le nombre de cas confirmés a suivi une lente augmentation.

et depuis que le gouvernement oblige les gens à faire de la quarantaine on va voir le nombre totale des cas dans le confinement

```
plt.figure(figsize=(14,8))
sns.lineplot(data = cov_df['Active'],label='Actif')
plt.axvline(dt.datetime(2020, 3, 20),ls = '--',c = 'r')
plt.axvline(dt.datetime(2020, 4, 20),ls = '--',c = 'r')
plt.axvline(dt.datetime(2020, 5, 20),ls = '--',c = 'r')
plt.axvline(dt.datetime(2020, 6, 10),ls = '--',c = 'g')
plt.text(dt.datetime(2020, 3, 21), 1500, 'Lockdown', fontsize=12,color='#FF0000')
plt.text(dt.datetime(2020, 4, 21), 1500, 'Lockdown extended', fontsize=12,color='#FF0000')
plt.text(dt.datetime(2020, 5, 21), 1500, 'Lockdown extended', fontsize=12,color='#FF0000')
plt.text(dt.datetime(2020, 6, 21), 1500, 'Lockdown End', fontsize=12,color='#00FA00')
plt.xlabel('Date')
plt.ylabel('Total actif cases')
plt.title('Actif cases ')
plt.legend()
```



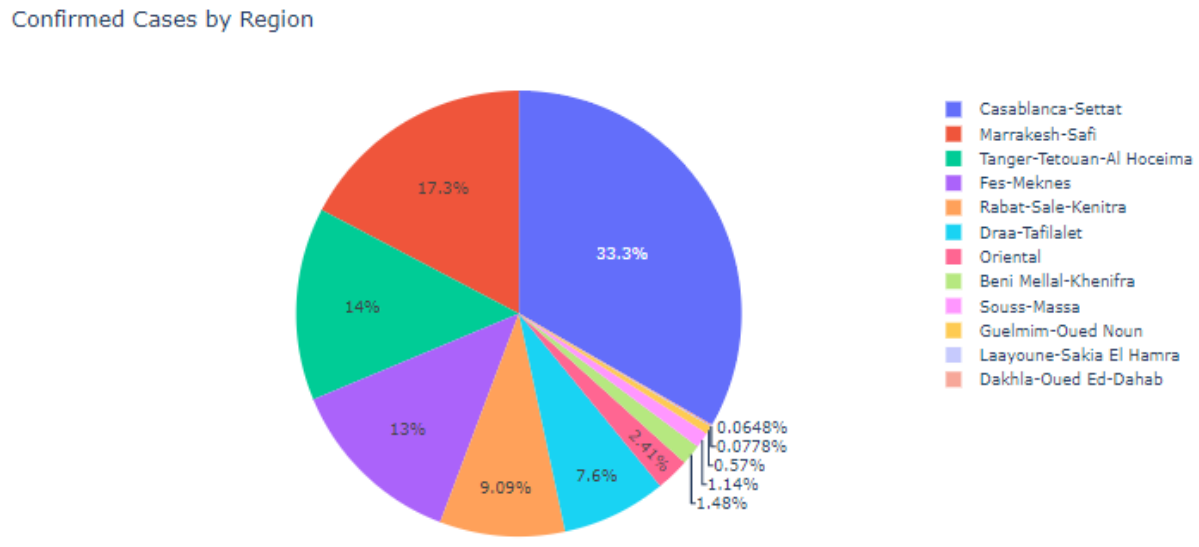
- On constate que le nombre de cas commence à baisser après 2 mois de confinement .

Maintenant, nous allons approfondir les régions spécifiques du Maroc.

Pour mieux comprendre et comprendre le pourcentage et le nombre total de cas dans chaque région, j'ai suivi ces étapes:

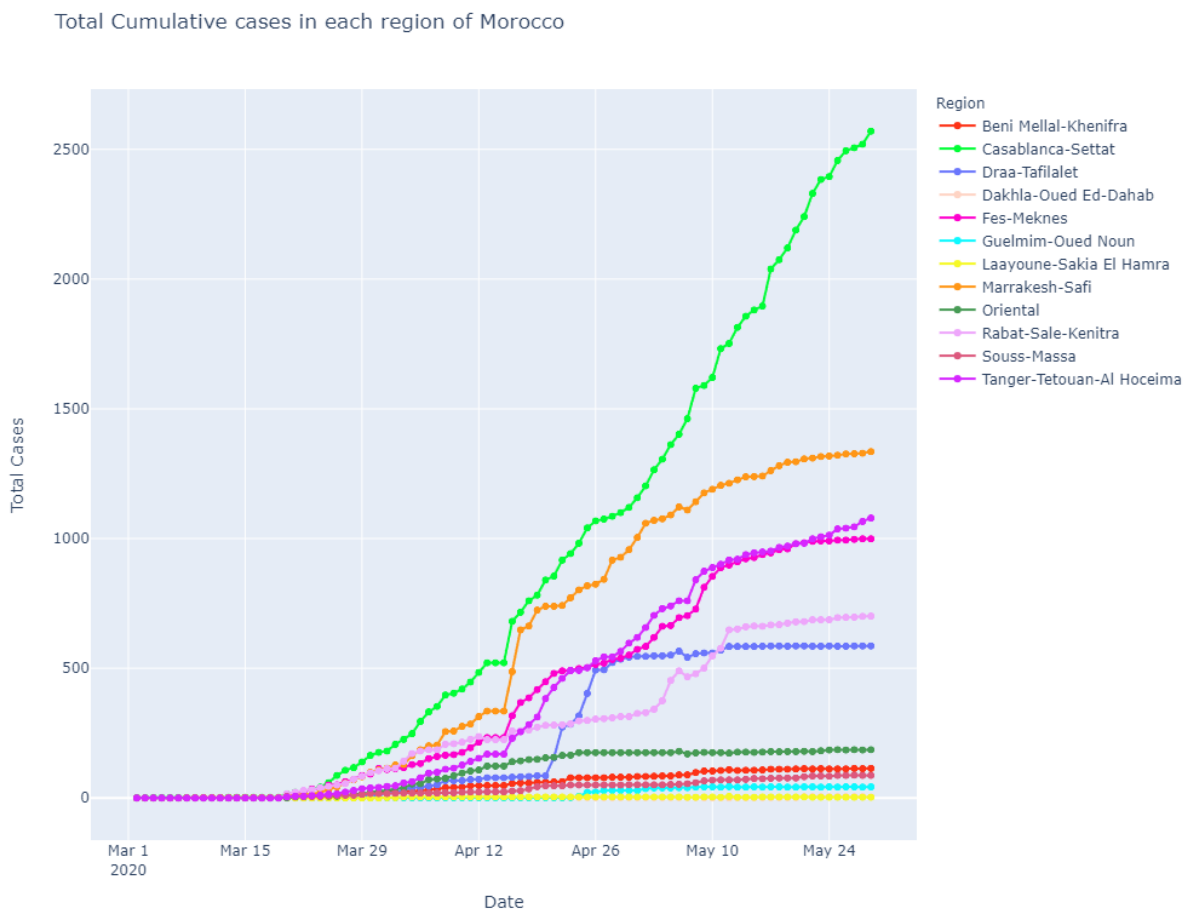
- Sélection des colonnes des régions que je voulais à partir de dataframe par référence d'index
- Visualisé à l'aide d'un Pie Chart

```
import plotly.express as px


# sélectionnez les colonnes dont nous voulons dessiner un graphique à secteurs
columns = cov_df.columns[5:-2]

# regrouper la Dataframe par nombre total de cas, puis les résumer et sélectionner le maximum dans chaque région
new_cov_df = cov_df.groupby(columns.tolist(), as_index=False).sum().max()

print(new_covid_df[:])

fig = px.pie(new_covid_df, values=new_cov_df[0:-6].values, names=columns, title='Confirmed Cases by Region')
fig.show()
```



- Pour avoir une idée de l'évolution du nombre de cas confirmés dans chaque région, j'ai fusionné le cadre de données par régions en tant que variables de valeur et la date en tant qu'identifiant :

```
REGIONS= ['Beni Mellal-Khenifra', 'Casablanca-Settat', 'Draa-Tafilalet',
          'Dakhla-Oued Ed-Dahab', 'Fes-Meknes', 'Guelmim-Oued Noun',
          'Laayoune-Sakia El Hamra', 'Marrakesh-Safi', 'Oriental',
          'Rabat-Sale-Kenitra', 'Souss-Massa', 'Tanger-Tetouan-Al Hoceima']

updated_covid_df = pd.melt(covid_df, id_vars=['Date'], var_name='Region', value_vars=REGIONS, value_name='Number')


fig = px.line(updated_covid_df, x='Date', y='Number', color='Region', width=1000, height=800,
              title="Spike lines active", color_discrete_sequence=px.colors.qualitative.Light24)

fig.update_traces(mode="markers+lines")

fig.update_xaxes(showspikes=True)
fig.update_yaxes(showspikes=True)

fig.update_layout(
    title="Total Cumulative cases in each region of Morocco",
    yaxis_title="Total Cases",
)

# # Show plot
fig.show()
```

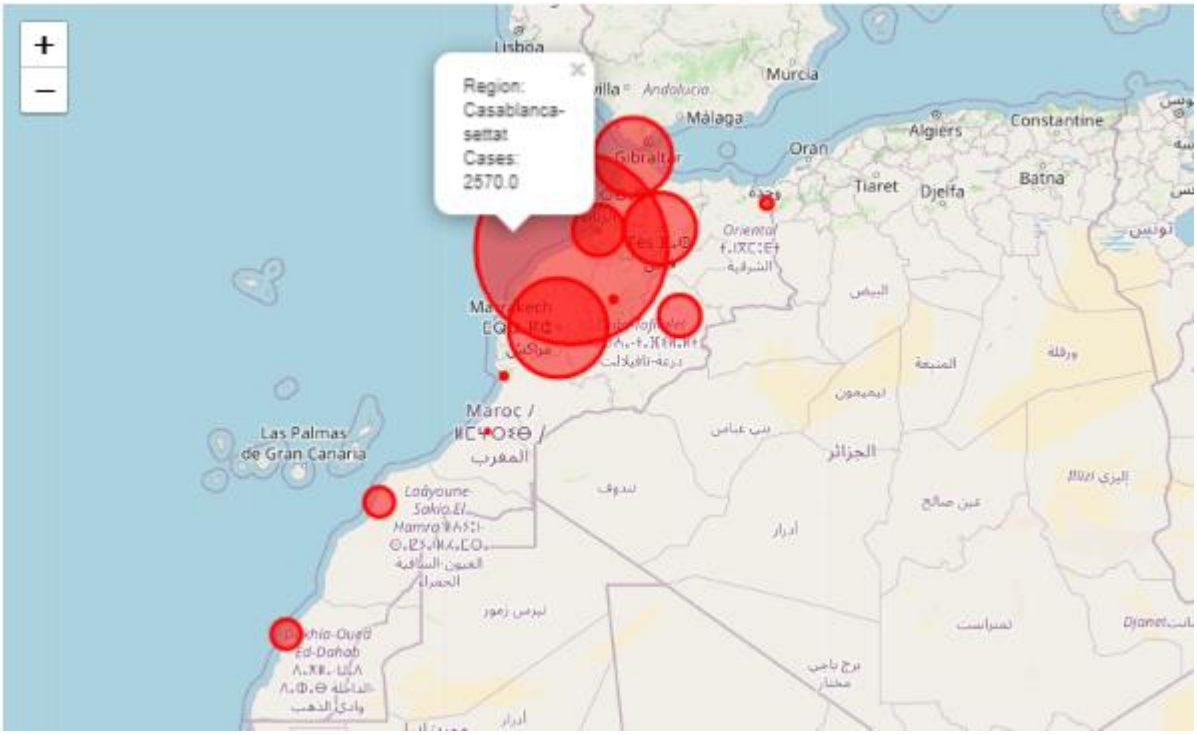


L'évolution du nombre total de cas cumulés dans chaque région et le pic soudain de certaines régions comme Tanger et Marrakech par exemple début avril sont dus aux clusters COVID19 trouvés dans les usines qui ont monté en flèche le nombre de cas dans cette région.

État de chaque région sur MAP avec Folium :

- J'ai créé une plage horaire des 15 derniers jours.
- J'ai sélectionné les lignes de dates correspondantes de la plage de temps que je viens de créer.
- J'ai créé une nouvelle trame de données en utilisant la latitude, la longitude de l'ensemble de données, et la région et le nombre de cas totaux que je viens de calculer dans la plage de temps prédéfinie.

```
from datetime import date, datetime, timedelta

# Get the time range of the last 15 days
DATE = date.today() - timedelta(days=10)
cov_df['Date'] = pd.to_datetime(cov_df['Date']).dt.date

# Select corresponding date rows
row = cov_df[cov_df.Date == DATE]

row = row[REGIONS].iloc[row.shape[0]-1,:]

# Make Data as Dataframe
df = {
    'latitude': [32.320296, 33.590000, 31.932124, 23.692847, 34.037120, 28.986355, 27.158507, 31.630000, 34.680000, 34.020000, 3
    'longitude': [-6.380926, -7.610000, -4.424149, -15.938255, -5.002964, -10.057375, -13.207878, -8.000000, -1.910000, -6.83000
    'Region': row.index.to_list(),
    'Ncases': list(row.values)
}
df = pd.DataFrame.from_dict(df)

# Create Map
fig = folium.Figure(width=800, height=500)
latitude, longitude = 29.852972, -12.763558
CovidMap = folium.Map(location=[latitude, longitude], zoom_start=5)

# Add Title
title_html = f' <h3 align="center" style="font-size:20px"><b>Map of Covid-19 Cases in Morocco Regions By {DATE}</b></h3>'
CovidMap.get_root().html.add_child(folium.Element(title_html))

for lat, lon, Region, Ncases in zip(df['latitude'], df['longitude'], df['Region'], df['Ncases']):
    folium.CircleMarker(
        [lat, lon],
        radius=.1 * Ncases//4,
        popup = ('Region: ' + str(Region).capitalize() + '<br>'
                 'Cases: ' + str(Ncases) + '<br>'),
        color='red',
        fill_color = "red",
        fill=True,
        fill_opacity=0.5
        ).add_to(CovidMap)

fig.add_child(CovidMap)
```



3 – Prédiction avec la Model Régression :

Prétraitement sur les données pour appliquer le model :

```
print(cov_df)
df_Maroc = cov_df[['Date', 'Confirmed']]
df_Maroc =df_Maroc[20:]
print(df_Maroc)
df_Maroc = df_Maroc.reset_index(drop=True)
df_Maroc_reg = df_Maroc.copy()
df_Maroc_reg = df_Maroc_reg.set_index('Date')
df_Maroc_reg = df_Maroc_reg[20:]
df_Maroc_reg.index = pd.to_datetime(df_Maroc_reg .index)
print(df_Maroc_reg.index)
```

Initialisation du modèle :

```
x = np.arange(len(df_Maroc_reg)).reshape(-1, 1)
y = df_Maroc_reg.values
```

```
print(x)
print(y)
```

```
[[ 0]
 [ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]
 [11]
 [12]
 [13]
 [14]
 [15]
 [16]
```

Ajuster les données (former le modèle) :

```
from sklearn.neural_network import MLPRegressor
model = MLPRegressor(hidden_layer_sizes=[32, 32, 10], max_iter=50000, alpha=0.0005, random_state=26)
_=model.fit(x, y)
```

```
C:\Users\mlagh\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:1342: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
```

Prédire :

```
from datetime import date, timedelta
test = np.arange(len(df_Maroc_reg)+7).reshape(-1, 1)
pred = model.predict(test)
prediction = pred.round().astype(int)
week = [df_Maroc_reg.index[0] + timedelta(days=i) for i in range(len(prediction))]
dt_idx = pd.DatetimeIndex(week)
predicted_count = pd.Series(prediction, dt_idx)
```

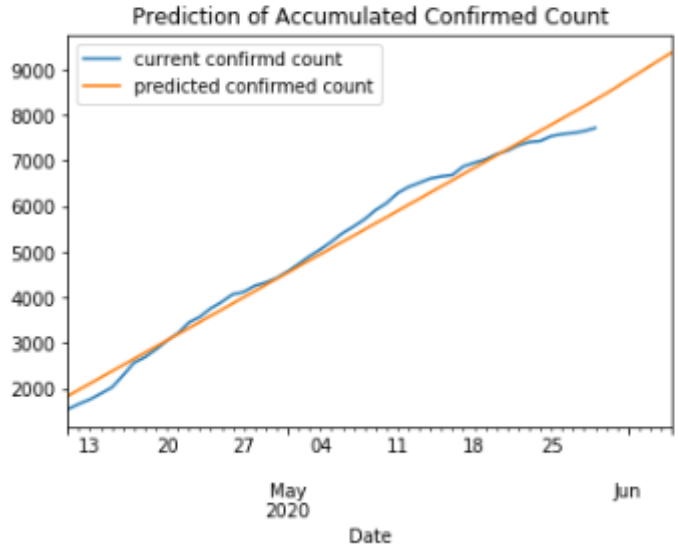
```
predicted_count.tail()
```

```
2020-06-01    8770
2020-06-02    8921
2020-06-03    9072
2020-06-04    9223
2020-06-05    9374
dtype: int32
```

évaluation du modèle :

```
pd.plotting.register_matplotlib_converters()
```

```
df_Maroc_reg.plot()
predicted_count.plot()
plt.title('Prediction of Accumulated Confirmed Count')
plt.legend(['current confirmd count', 'predicted confirmed count'])
plt.show()
```



CONCLUSION :

- Tout au long de ce billet, l'état de l'épidémie de COVID19 au Maroc a été analysé avec une vue d'ensemble et également avec une vue régionale pour mieux comprendre comment chaque région marocaine a été affectée.

- De plus, l'apparition de nouveaux clusters dans les différentes régions a eu un impact massif sur le nombre total de cas et de décès dans chaque région du Maroc.

- Globalement, le gouvernement marocain a pris des mesures drastiques en multipliant le nombre de tests et en maîtrisant le virus en fermant les routes entre les différentes villes et régions.

- Personnellement je pense que la meilleure solution c'est d'augmenter le nombre des tests dans tous les secteurs du Maroc sur tous les grand villes par exemple Casablanca, Rabat, Tanger et dans les usines aussi même si les personnes Ils ne présentent aucun symptôme. ça coût chère pour l'état mais c'est la meilleure solution.