Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ ΜΑΘΗΣΗΣ

Υλοποιητικό Project

Στεφανιώρος Μιχαήλ
A.M. 1072774
Email: up1072774@upnet.gr

Πάτρα, 4ο Έτος

Περιβάλλον υλοποίησης

Ως γλώσσα υλοποίησης ορίστηκε η Python. Η εργασία υλοποιήθηκε σε Jupyter notebook στο VS Code και, επιπλέον, υπάρχουν και δύο python scripts, όπου το ένα μετατρέπει τα δεδομένα από csv σε xlsx και το άλλο εγκαθιστά τις απαραίτητες βιβλιοθήκες.

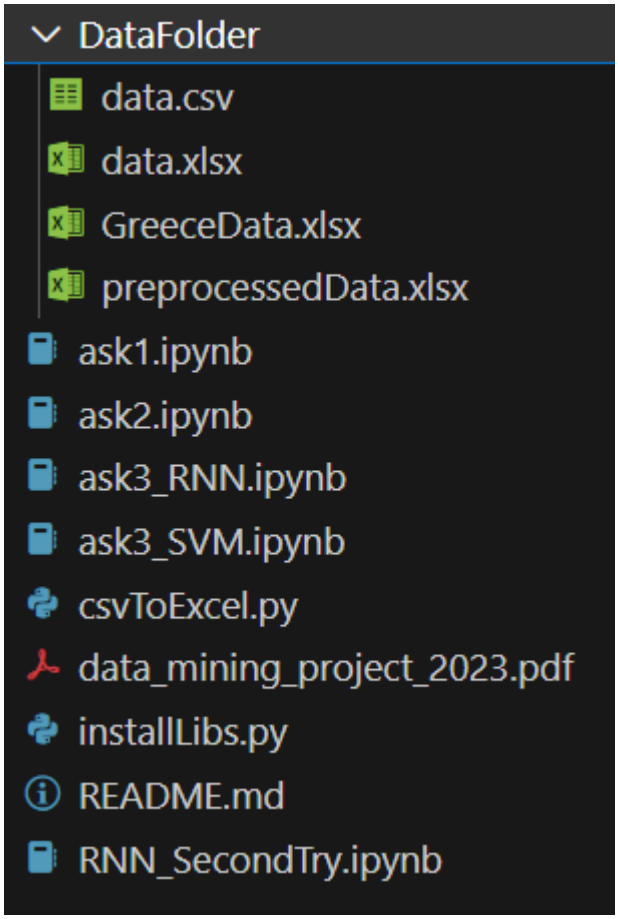
Βιβλιοθήκες/Dependencies

Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι:

1. **Pandas:** για χειρισμό δεδομένων (πχ από csv αρχείο)
2. **Numpy:** για περίπλοκους υπολογισμούς μέσα από τις δομές δεδομένων που προσφέρει.
3. **Matplotlib:** για δημιουργία γραφικών παραστάσεων
4. **Seaborn:** βιβλιοθήκη χτισμένη πάνω στην matplotlib για περίπλοκες γραφικές παραστάσεις όπως heatmaps
5. **Scikit-learn (sklearn):** βιβλιοθήκη που χρησιμοποιείται για μηχανική μάθηση, καθώς προσφέρει modules για classification, clustering, regression και άλλα. Τα modules που χρησιμοποιήθηκαν είναι:
    - **SimpleImputer from sklearn.impute:** για χειρισμό missing values
    - **KMeans from sklearn.cluster:** για clustering
    - **StandardScaler from sklearn.preprocessing:** για να κάνει scale τα δεδομένα, το οποίο είναι απαραίτητο βήμα στην προεπεξεργασία
    - **SVR from sklearn.svm:** Support Vector Regression και είναι μια κλάση που υλοποιεί support vector machine-based models
6. **Tensorflow, Keras:** Είναι ένα open-source machine learning framework από την Google. Χρησιμοποιείται για την εκπαίδευση μοντέλων μηχανικής μάθησης. Το keras είναι ένα API του Tensorflow, το οποίο διευκολύνει την κατασκευή νευρωνικών δικτύων

Εγκατάσταση

Αφού αποσυμπιέσετε το παραδοτέο (αρχείο .rar) σε όποιο directory σας βολεύει, θα δείτε έναν φάκελο, ο οποίος θα περιέχει όλα τα απαραίτητα αρχεία της εργασίας.



Για να εγκαταστήσετε τα απαραίτητα dependencies, μπορείτε είτε να τρέξετε το αρχείο installLibs.py(*) ή να ανοίξετε το command prompt (cmd) και να εκτελέσετε την εντολή:
pip install και το όνομα της βιβλιοθήκης (για κάθε βιβλιοθήκη).
Πχ: pip install numpy.

*Σημείωση: Το script installLibs.py δεν είναι δοκιμασμένο σε κάποιον υπολογιστή χωρίς τις βιβλιοθήκες για να βεβαιωθώ σίγουρα για την ορθότητά του. Όταν το τρέχω εγώ, δουλεύει κανονικά και βγάζει μήνυμα ότι είναι ήδη εγκατεστημένες όλες οι βιβλιοθήκες.

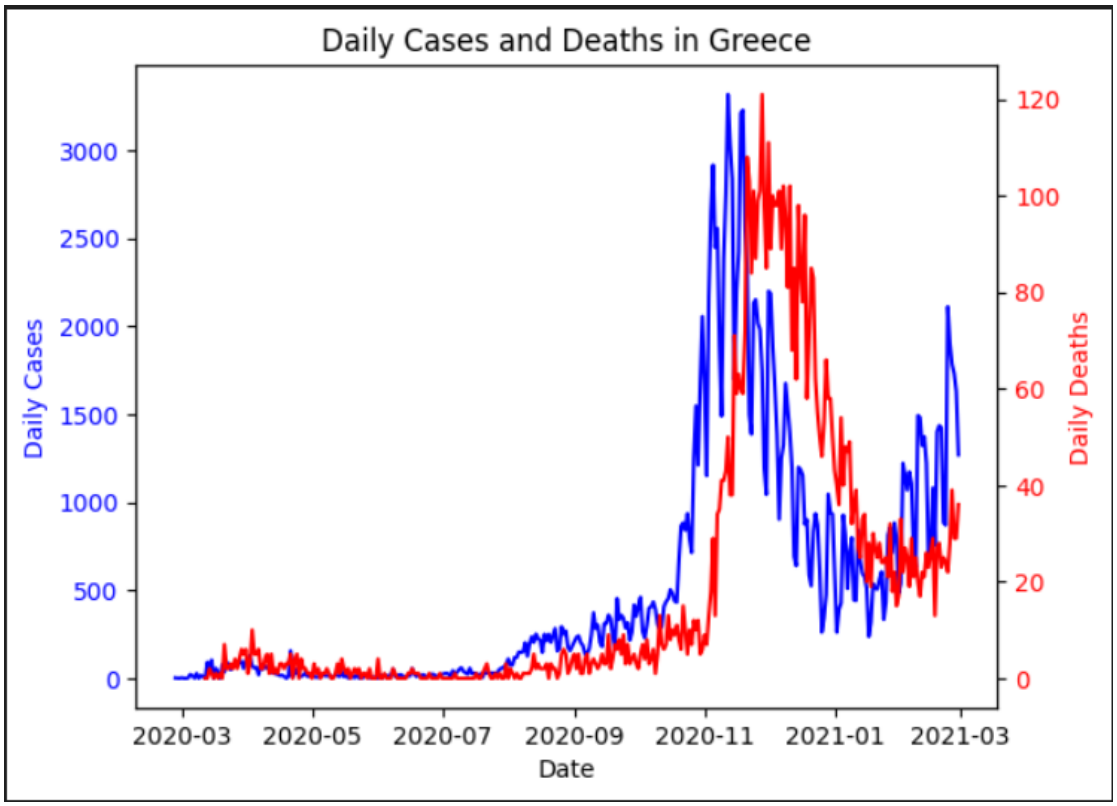
Τέλος, για κάθε άσκηση εκτελείτε το αντίστοιχο αρχείο .ipynb. Το αρχείο

csvToExcel.py δημιουργήθηκε για την καλύτερη απεικόνιση των δεδομένων και εκτελέστηκε μέσα στους κώδικες των ασκήσεων για την δημιουργία των αρχείων .xlsx, και έπειτα έγινε commented.

Ερώτημα 1

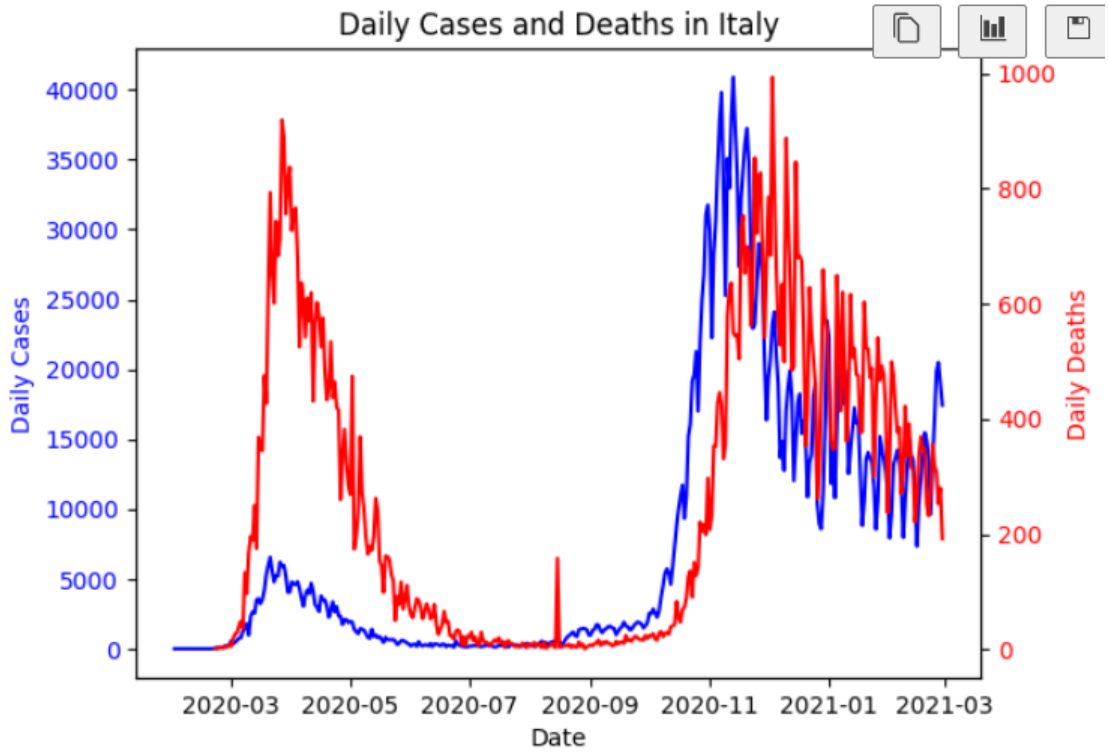
Αρχικά, να αναφερθεί ότι όλες οι συναρτήσεις, για τις οποίες θα γίνει λόγος παρακάτω για την συγκεκριμένη άσκηση, εκτελούνται στην main().

Η πρώτη άσκηση ζητούσε κατανόηση των δεδομένων μέσω γραφικών παραστάσεων. Επομένως, ξεκινώντας δημιούργησα μια συνάρτηση countryData(country), η οποία δέχεται ως όρισμα το όνομα μιας χώρας και επιστρέφει τα daily cases και τα daily deaths. Τα γνωρίσματα αυτά δεν υπάρχουν στα αρχικά δεδομένα και υπολογίζονται ως η διαφορά των cases και deaths δύο συνεχόμενων ημερών. Έπειτα, γίνονται plot σε κοινή γραφική παράσταση ώστε να μπορέσουμε να συγκρίνουμε τα δύο γνωρίσματα, μέσω της συνάρτησης plotData(df). Τα αποτελέσματα είναι τα εξής:



Στην παραπάνω εικόνα, βλέπουμε τα κρούσματα και τους θανάτους καθημερινά στην Ελλάδα. Παρατηρούμε ότι σε όλη την περίοδο, οι θάνατοι, αν και λιγότεροι, είναι αρκετοί σε σχέση με τα κρούσματα αν το

δούμε με ποσοστιαία ματιά. Επιπλέον, είναι φανερό ότι στο τέλος του 2020, τα δύο γνωρίσματα κορυφώνονται, όπως φυσικά το καταλάβαμε κι εμείς με την δεύτερη καραντίνα. Από εκεί και μετά υπάρχει μια πτώση. Όσο προχωράμε χρονικά στην καραντίνα προς τον Μάρτιο και όσο πλησίαζε το Πάσχα, ο κόσμος είχε κουραστεί από το lockdown και για αυτό έχουμε και πάλι μια αύξηση στα κρούσματα.

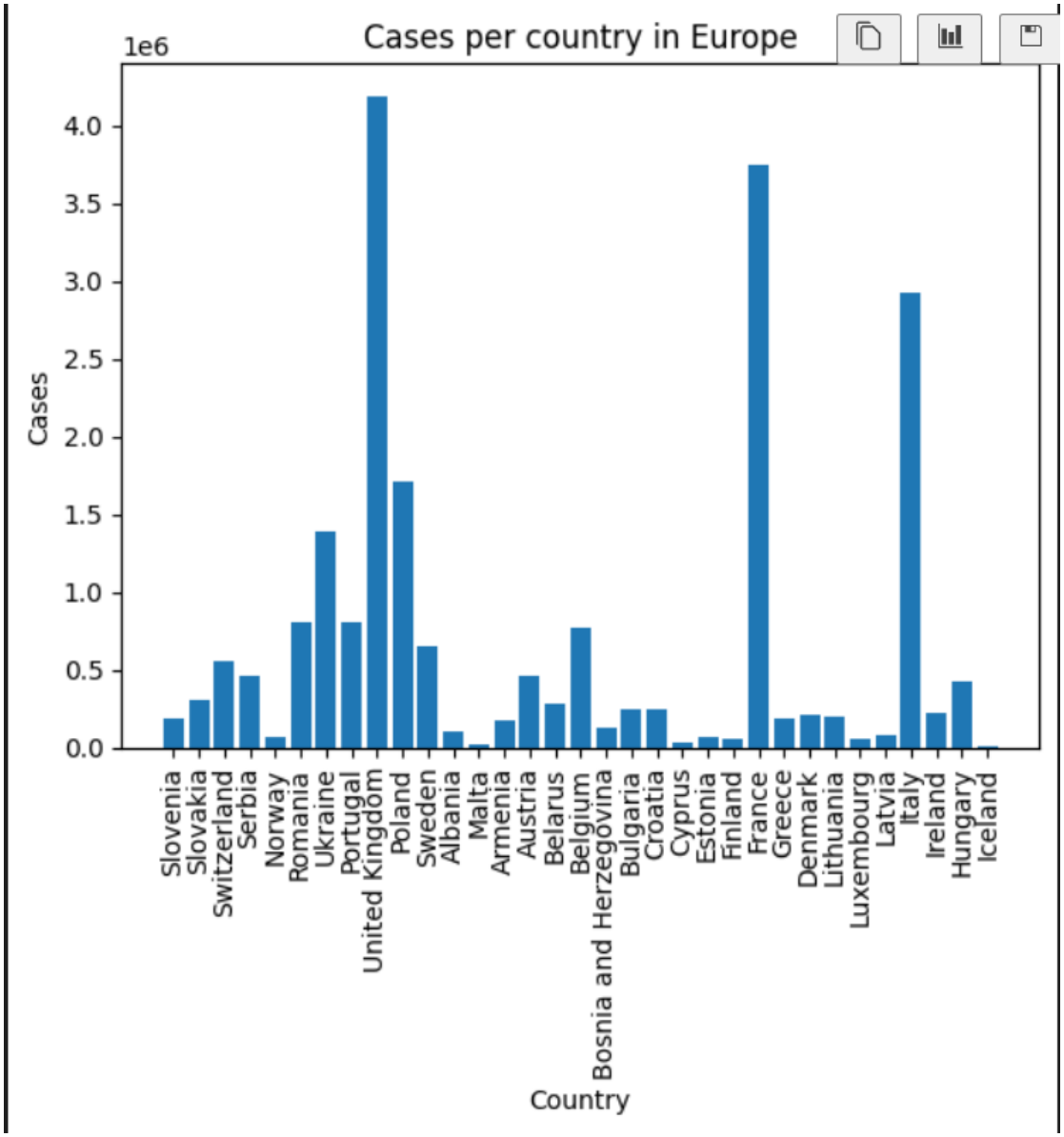


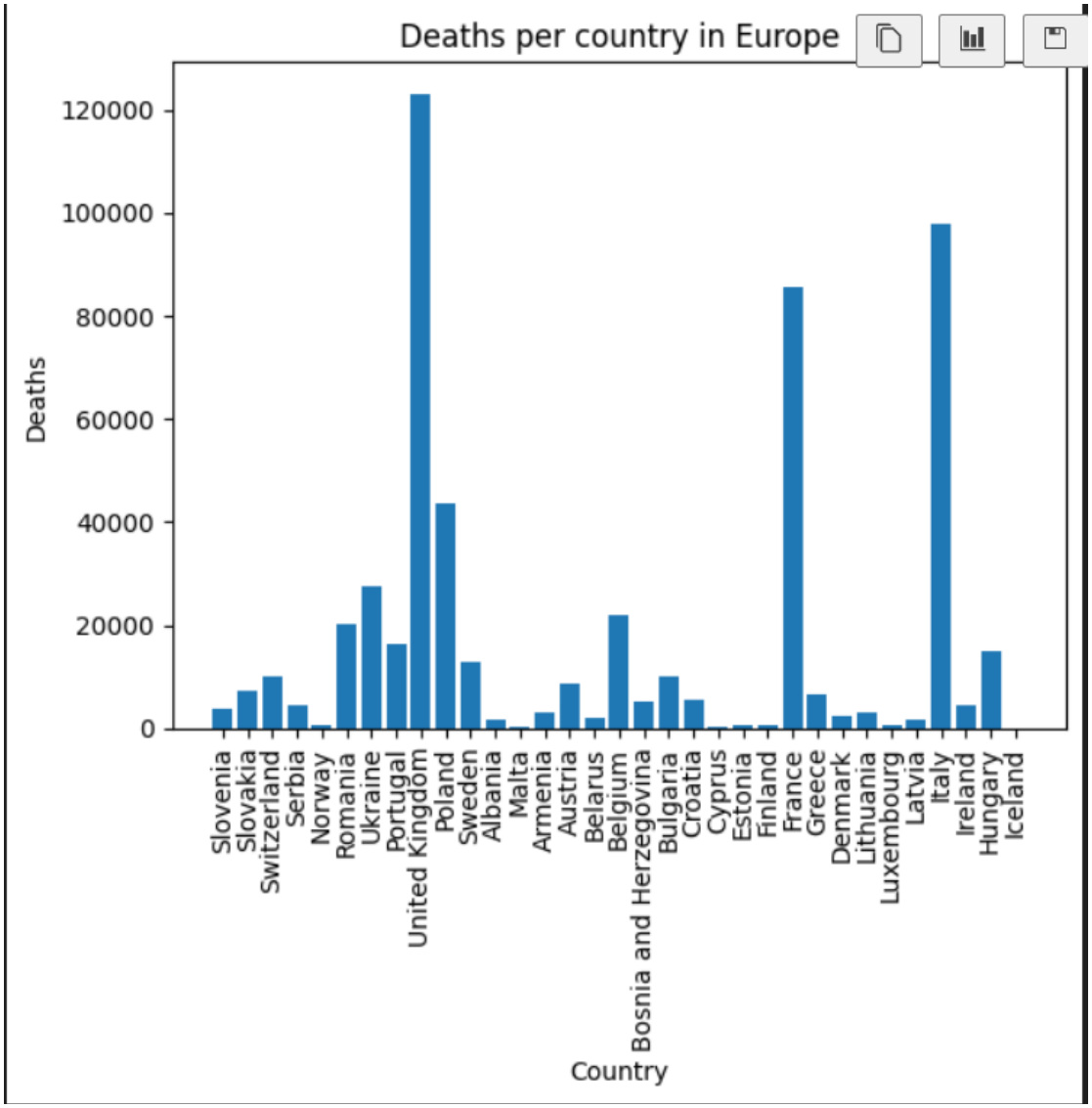
Στην συνέχεια, παρουσιάζονται τα ίδια γνωρίσματα για την Ιταλία, τον γείτονά μας. Είναι προφανές ότι δεν χειρίστηκαν καθόλου καλά την πανδημία, καθώς από την αρχή τα κρούσματα και οι θάνατοι ξεπερνάνε αυτά της Ελλάδας σε όλη την περίοδο. Ωστόσο, το κοινό χαρακτηριστικό στις δύο χώρες είναι ότι στα τέλη του 2020 γίνεται πτώση στα δεδομένα.

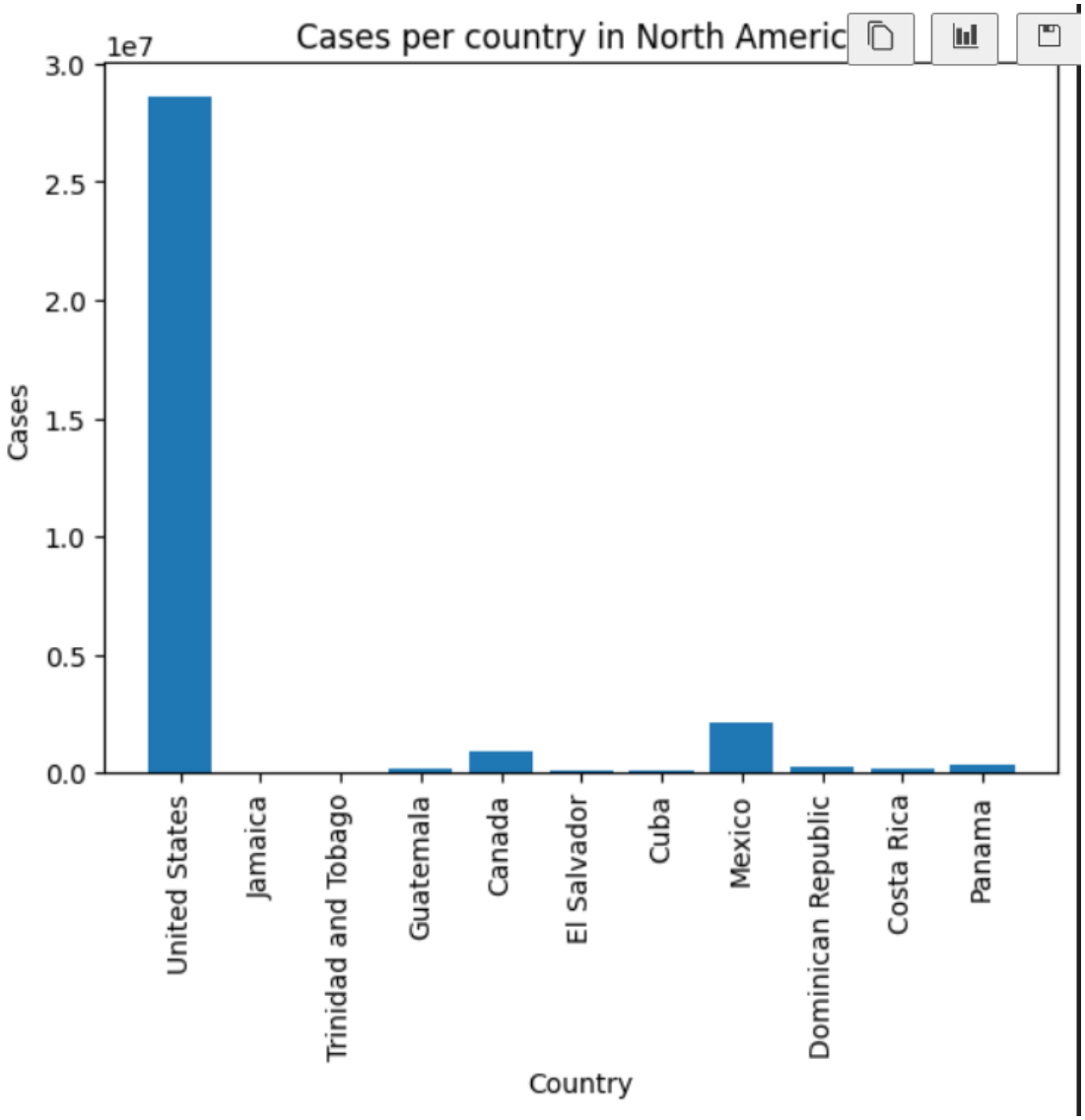
Σε επόμενη φάση, δημιουργώ ιστόγραμμα με τους θανάτους και τα κρούσματα για όλες τις χώρες μιας ηπείρου. Αυτά πραγματοποιούνται με την συνάρτηση countriesPerContinentData() και plotCountriesPerContinentData(data, continent). Για να πάρω τα συγκεκριμένα δεδομένα, κάνω groupBy τα cases και τα deaths με βάση το entity και το continent. Στην συνέχεια, τοποθετώ τα δεδομένα αυτά σε μια np array και τα επιστρέφω.

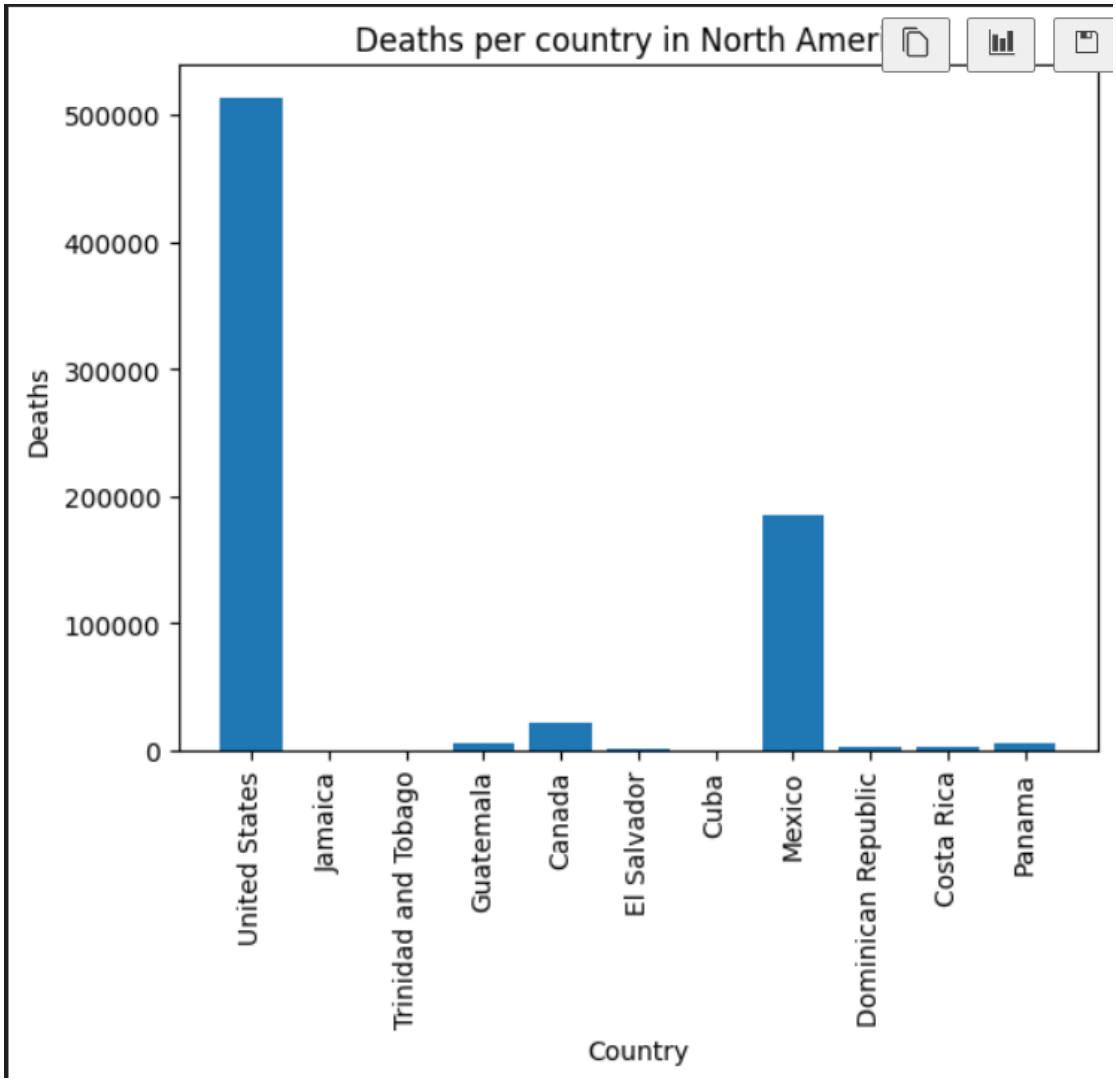
Από την άλλη, για τα plots γίνεται απλή χρήση της matplotlib.

Παρακάτω, παρουσιάζονται τα ιστογράμματα για την Ευρώπη και την Βόρεια Αμερική.

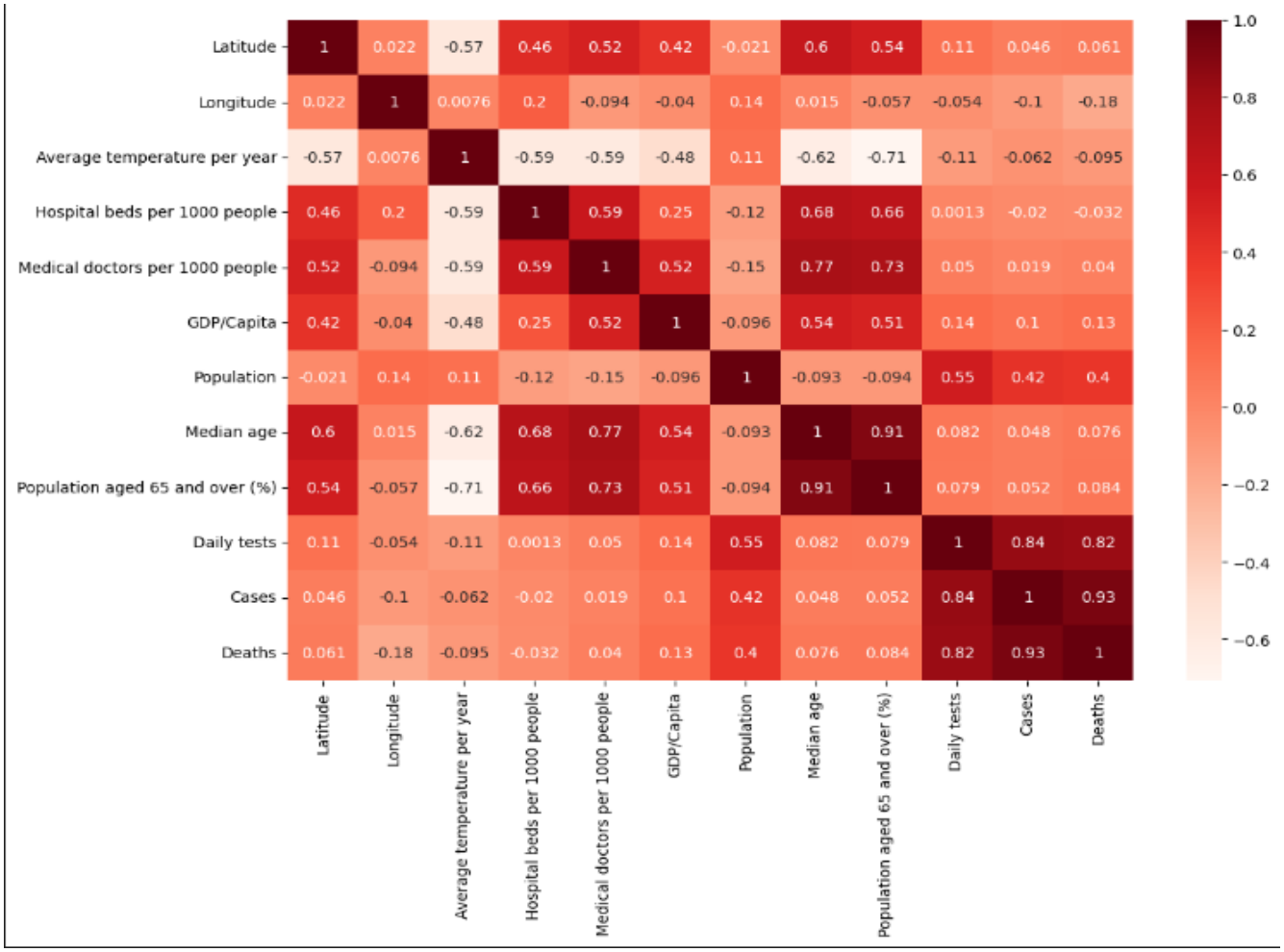



Deaths per country in Europe
Deaths
120000
100000
80000
60000
40000
20000
0
Slovenia
Slovakia
Switzerland
Serbia
Norway
Romania
Ukraine
Portugal
United Kingdom
Poland
Sweden
Albania
Malta
Armenia
Austria
Belarus
Belgium
Bosnia and Herzegovina
Bulgaria
Croatia
Cyprus
Estonia
Finland
France
Greece
Denmark
Lithuania
Luxembourg
Latvia
Italy
Ireland
Hungary
Iceland
Country


Cases per country in North Americ
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Cases
United States
Jamaica
Trinidad and Tobago
Guatemala
Canada
El Salvador
Cuba
Mexico
Dominican Republic
Costa Rica
Panama
Country



Σε τελικό στάδιο, μέσω της συνάρτησης correlationData(), δημιουργώ ένα heatmap το οποίο δείχνει τις συσχετίσεις μεταξύ των κλάσεων των δεδομένων.



Ερώτημα 2

Η δεύτερη άσκηση ζητάει να γίνει clustering στις χώρες με βάση κάποια χαρακτηριστικά που δείχνουν το πόσο καλά αντιμετωπίσανε τον covid. Πρώτο και βασικό βήμα στην διαδικασία αυτή είναι το preprocessing. Οπότε, αρχικά ελέγχω όλες τις στήλες για ελλιπή δεδομένα με την συνάρτηση checkMissingData().

```
RangeIndex: 38472 entries, 0 to 38471
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Entity                          38472 non-null  object
 1   Continent                       38472 non-null  object
 2   Latitude                        38472 non-null  float64
 3   Longitude                       38472 non-null  float64
 4   Average temperature per year    38472 non-null  int64
 5   Hospital beds per 1000 people   38472 non-null  float64
 6   Medical doctors per 1000 people 38472 non-null  float64
```

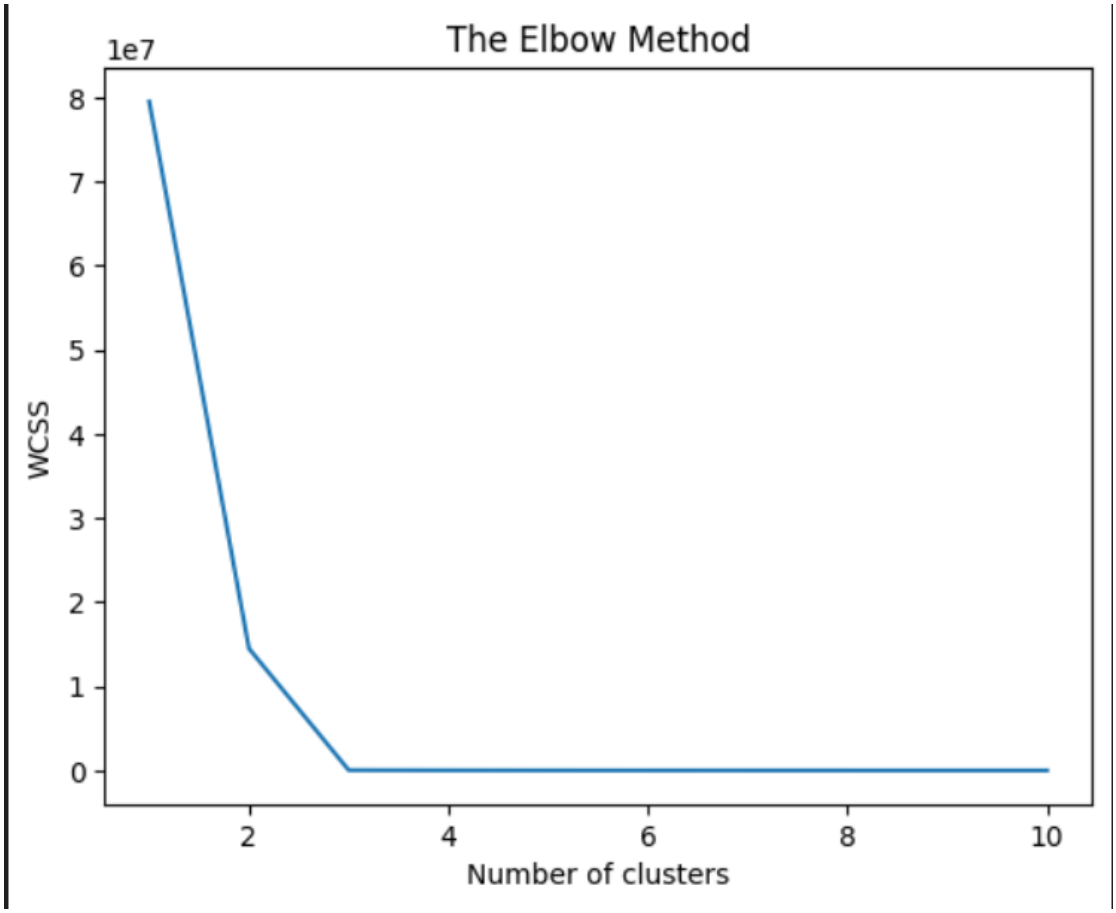
```
 7   GDP/Capita                        38472 non-null  float64
 8   Population                        38472 non-null  int64
 9   Median age                        38472 non-null  int64
 10  Population aged 65 and over (%)   38472 non-null  int64
 11  Date                              38472 non-null  object
 12  Daily tests                       30577 non-null  float64
 13  Cases                             38218 non-null  float64
 14  Deaths                            34862 non-null  float64
```

Από το αποτέλεσμα της μεθόδου, παρατηρώ ότι λείπουν δεδομένα από τα daily tests, cases, deaths. Επομένως, χρησιμοποιώ έναν SimpleImputer της βιβλιοθήκης sklearn.impute για να αντικαταστήσω τις NaN τιμές με την μέση τιμή των υπόλοιπων δεδομένων.

```
RangeIndex: 38472 entries, 0 to 38471
Data columns (total 15 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Entity                            38472 non-null  object
 1   Continent                         38472 non-null  object
 2   Latitude                          38472 non-null  float64
 3   Longitude                         38472 non-null  float64
 4   Average temperature per year      38472 non-null  int64
 5   Hospital beds per 1000 people     38472 non-null  float64
 6   Medical doctors per 1000 people   38472 non-null  float64
 7   GDP/Capita                        38472 non-null  float64
 8   Population                        38472 non-null  int64
 9   Median age                        38472 non-null  int64
 10  Population aged 65 and over (%)   38472 non-null  int64
 11  Date                              38472 non-null  object
 12  Daily tests                       38472 non-null  float64
 13  Cases                             38472 non-null  float64
 14  Deaths                            38472 non-null  float64
```

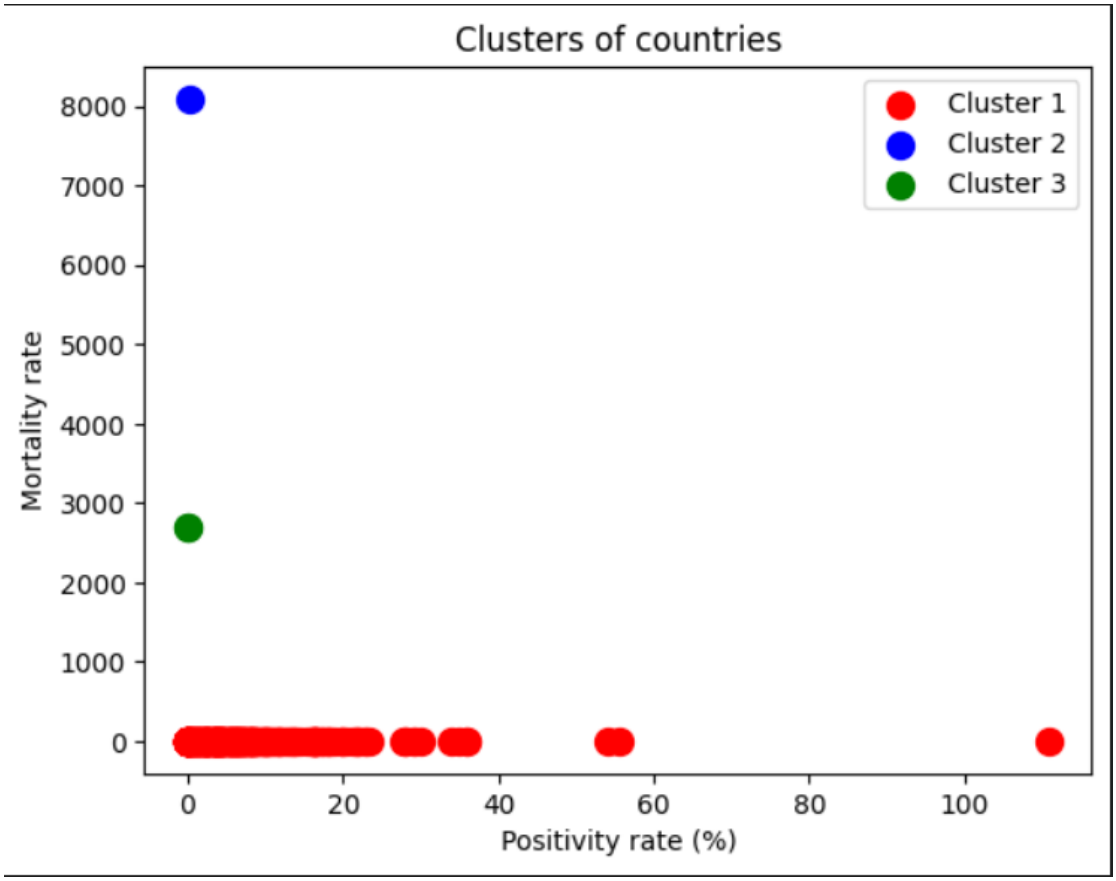
Τελικά, βλέπουμε ότι τώρα δεν λείπουν δεδομένα.

Σε επόμενο στάδιο, προσθέτω στήλες για positivity rate και mortality rate, οι οποίες θα χρησιμοποιηθούν για το clustering. Για να βρω πόσα clusters χρειάζονται, χρησιμοποιώ την elbow method. Υποθέτω 1 έως 10 clusters και χρησιμοποιώ το WCSS (Within Cluster Sum of Squares).

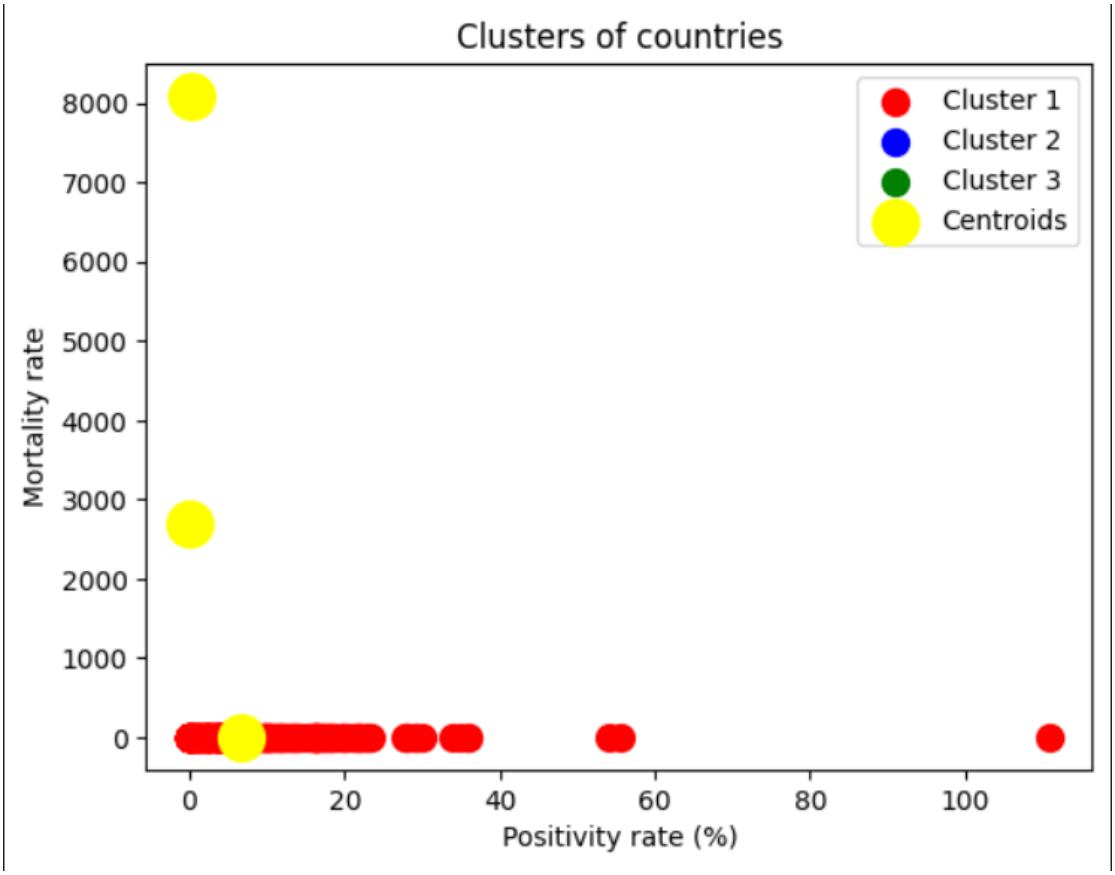


Από την γραφική παράσταση της μεθόδου, είναι φανερό ότι χρειάζονται 3 clusters.

Τελικά, με τον KMeans το clustering γίνεται ως εξής:



Και με τα κεντροειδή:



Παρατηρούμε ότι 2 χώρες ξεχώρισαν ως προς τους θανάτους κρατώντας χαμηλό ποσοστό θετικότητας, ενώ παράλληλα 3 χώρες είχαν υπερβολικά ψηλό ποσοστό κρουσμάτων με λίγους θανάτους.

Ερώτημα 3

Πριν από την δημιουργία οποιουδήποτε παλινδρομητή, πρέπει να γίνει ένα preprocessing. Ακολουθείται η μεθοδολογία της προηγούμενης άσκησης.

RNN

Μετά την προεπεξεργασία των δεδομένων, απομονώνω μόνο τα δεδομένα της Ελλάδας μέχρι το τέλος του 2020, καθώς αυτά μας ενδιαφέρουν. Επιπλέον, κρατάω και τα συνολικά δεδομένα για σύγκριση στο τέλος. Στην συνέχεια, προσθέτω μια στήλη για το positivity percent διότι αυτό θέλω να προβλέψω. Παρακάτω χρησιμοποιώ έναν StandardScaler για να φέρω τα δεδομένα σε όμοιες τιμές και έπειτα τα χωρίζω σε training και test set. Το νευρωνικό δίκτυο RNN δουλεύει έτσι

ώστε να κάνει επαναλήψεις στο μοντέλο βελτιώνοντας τα βάρη και τις απώλειες με το να παίρνει την έξοδο και να την περνάει πάλι σαν είσοδο. Έχει οριστεί στον κώδικα να γίνονται 200 τέτοιες επαναλήψεις.

```
Epoch 1/200
1/1 [==============================] - 4s 4s/step - loss: 7.4284e-04
Epoch 2/200
1/1 [==============================] - 0s 85ms/step - loss: 1.3643e-04
Epoch 3/200
1/1 [==============================] - 0s 116ms/step - loss: 5.7590e-06
Epoch 4/200
1/1 [==============================] - 0s 74ms/step - loss: 1.5320e-04
Epoch 5/200
1/1 [==============================] - 0s 85ms/step - loss: 2.6254e-04
Epoch 6/200
1/1 [==============================] - 0s 112ms/step - loss: 2.3202e-04
Epoch 7/200
1/1 [==============================] - 0s 98ms/step - loss: 1.3107e-04
Epoch 8/200
1/1 [==============================] - 0s 81ms/step - loss: 3.9701e-05
Epoch 9/200
1/1 [==============================] - 0s 112ms/step - loss: 7.4851e-07
Epoch 10/200
1/1 [==============================] - 0s 116ms/step - loss: 1.5910e-05
Epoch 11/200
1/1 [==============================] - 0s 83ms/step - loss: 5.7707e-05
Epoch 12/200
1/1 [==============================] - 0s 87ms/step - loss: 9.1966e-05
Epoch 13/200
...
Epoch 199/200
1/1 [==============================] - 0s 64ms/step - loss: 1.0783e-14
Epoch 200/200
1/1 [==============================] - 0s 65ms/step - loss: 2.1468e-13
```

Παρατηρούμε ότι σε κάθε παλινδρόμηση η απώλεια μειώνεται, σημειώνοντας έτσι βελτίωση στο μοντέλο.
Τελικά, μπορούμε να θέσουμε ως όρισμα μια ημερομηνία, στο format: YYYY-MM-DD, μέσα στην συνάρτηση predictNextDays(date), ώστε το μοντέλο να προβλέψει το ποσοστό θετικότητας μέχρι εκείνη την ημερομηνία. Για την 2021-01-03 έχουμε:

```
Predict positivity percent for the next 3 days:

Date: 2021-01-01
Prediction 1: 2.91 %
Actual value: 2.08 %

Date: 2021-01-02
Prediction 2: 2.9 %
Actual value: 5.46 %

Date: 2021-01-03
Prediction 3: 2.89 %
Actual value: 4.59 %
```

Τα αποτελέσματα αυτά είναι αρκετά ικανοποιητικά, ειδικά αν αναλογιστούμε την περιπλοκότητα του dataset, αφού αφορούσε δεδομένα για μια άγνωστη μέχρι τώρα ασθένεια.

Ο συγκεκριμένος παλινδρομητής υλοποιήθηκε και με δεύτερο τρόπο στο αρχείο RNN_SecondTry.ipynb. Αν δεν επιτρέπεται να βαθμολογηθούν και οι δύο τρόποι, τότε να σημειωθεί **ότι ο βασικός και ο ως προς διόρθωση είναι ο προηγούμενος (αρχείο ask3_RNN.ipynb).** Παρ'όλα αυτά θα αναφερθεί πολύ σύντομα και ο δεύτερος τρόπος.

Το σημείο στο οποίο διαφέρουν οι δύο υλοποιήσεις είναι στην επιλογή δεδομένων για train του μοντέλου. Ενώ προηγουμένως, επιλεγόντουσαν τα δεδομένα μέχρι τα τέλη του 2020, τώρα ο χρήστης δίνει μια ημερομηνία μετά τις 2021-01-01 και το μοντέλο εκπαιδεύεται με τα δεδομένα από 10 μέρες πριν την δοσμένη ημερομηνία μέχρι και αυτήν. Τα αποτελέσματα σε αυτή την περίπτωση είναι:

```
Predict positivity percent for the next 3 days:

Date: 2021-01-01 00:00:00
Prediction 1: 2.1 %
Actual value: 2.08 %

Date: 2021-01-02 00:00:00
Prediction 2: 2.11 %
Actual value: 5.46 %

Date: 2021-01-03 00:00:00
Prediction 3: 2.07 %
Actual value: 4.59 %
```

Παρατηρούμε ότι είναι αρκετά κοντά με την προηγούμενη περίπτωση.

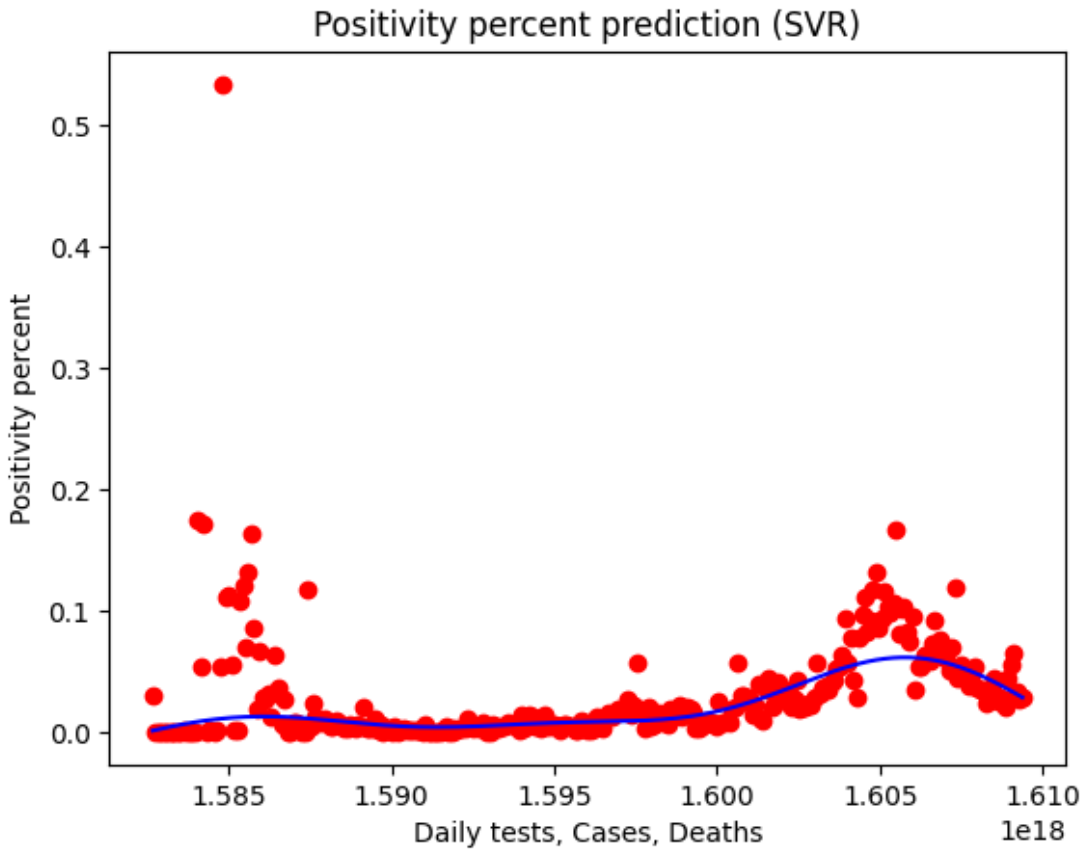
SVM

Όπως αναφέρθηκε και προηγουμένως, η διαδικασία της προεπεξεργασίας είναι ίδια για όλες τις ασκήσεις. Έπειτα, απομονώνω μόνο τα δεδομένα της Ελλάδας μέχρι το τέλος του 2020, καθώς αυτά μας ενδιαφέρουν. Στην συνέχεια, προσθέτω μια στήλη για το positivity percent διότι αυτό θέλω να προβλέψω. Παρακάτω χρησιμοποιώ έναν StandardScaler για να φέρω τα δεδομένα σε όμοιες τιμές. Για την εκπαίδευση του μοντέλου, θέτω ως ανεξάρτητη μεταβλητή τις ημερομηνίες και ως εξαρτημένη μεταβλητή το positivity percent και τα περνάω σε έναν Support Vector Regressor. Για την πρόβλεψη επιλέγω τα δεδομένα για την 2021-01-01 και το αποτέλεσμα που προκύπτει είναι:

Positivity percent on 2021-01-01: 2.71 %

Βλέπουμε, ότι η πρόβλεψη ταιριάζει με αυτήν του RNN μοντέλου, κάτι που επιβεβαιώνει την μέθοδο.

Τελικά, παρουσιάζεται ένα plot με τις προβλέψεις για το SVM:



Όπως ήταν αναμενόμενο, πρόκειται για μη γραμμική παλινδρόμηση, με την καμπύλη να περιλαμβάνει τα περισσότερα σημεία.