**Υλοποιητικό Project Data Mining 2023**

Η γλώσσα προγραμματισμού για την υλοποίηση του Project έχει επιλεχθεί η python. Οι βιβλιοθήκες όπου χρησιμοποιούνται για την παρακάτω υλοποίηση είναι οι ακόλουθες:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

**Περιβάλλον και εγκατάσταση**

Το πρόγραμμα εκτελέστηκε στο περιβάλλον του google colab. Πρόκειται για μια online πλατφόρμα συνεπώς δεν απαιτείται κάποια εγκατάσταση λογισμικού. Για την δημιουργία ενός notebook, αφού μεταφερθούμε στην ιστοσελίδα https://colab.research.google.com , πατάμε new notebook και είναι έτοιμο για χρήση. Σε περίπτωση όπου θέλουμε να ανοίξουμε ήδη υπάρχον notebook, πατάμε στην καρτέλα upload και επιλέγουμε το .ipynb αρχείο όπου επιθυμούμε.

**Εισαγωγή και διάβασμα αρχείου CSV**

Για την εισαγωγή και το διάβασμα του αρχείου CSV εκτελείται ο παρακάτω κώδικας:

# Load the data into a pandas DataFrame

df = pd.read\_csv('data.csv')

# Fill missing values with 0

df = df.fillna(0)

**Προεπεξεργασία δεδομένων**

Αφού διαβάσουμε το CSV αντικαθιστούμε τις NA τιμές, δηλαδή τις τιμές όπου έχουν αφεθεί κενές με την τιμή 0, για να είναι πιο εύκολη η διαδικασία του preprocessing.

Αν παρατηρήσουμε το CSV αρχείο, υπάρχει ένα πρόβλημα με τις τιμές των Deaths και Cases. Ενώ οι τιμές των υπόλοιπων στηλών δείχνουν τα συμβάντα εκείνης της μέρας, τα deaths και τα cases είναι συσσωρευτικά, δηλαδή οι θάνατοι κάποιας μέρας είναι οι θάνατοι που έγιναν εκείνη την μέρα + τους θανάτους της προηγούμενης. Για να μας είναι πιο εύκολη η χρήση του dataset, χρησιμοποιείται ο παρακάτω αλγόριθμος για την εξαγωγή των θανάτων όπου έγιναν μόνο την συγκεκριμένη ημέρα.

# Define a function that calculates daily counts for a single country

def calculate\_daily\_counts(df\_country):

    df\_country['Cases'] = df\_country['Cases'].diff()

    df\_country.at[df\_country.index[0], 'Cases'] = df\_country.at[df\_country.index[1], 'Cases']

    df\_country['Deaths'] = df\_country['Deaths'].diff()

    df\_country.at[df\_country.index[0], 'Deaths'] = df\_country.at[df\_country.index[1], 'Deaths']

    return df\_country

# Split the DataFrame by 'Entity', apply the function to each group, and combine the groups back together

df = df.groupby('Entity', group\_keys=False).apply(calculate\_daily\_counts)

#Preproccess the data to exclude negative deaths and cases

df.loc[df['Deaths'] < 0, 'Deaths'] = 0

df.loc[df['Cases'] < 0, 'Cases'] = 0

Επιπλέον, με μια καλύτερη ανάλυση, παρατηρείται μια ακραία τιμή (outlier) στα δεδομένα όπου οι θάνατοι μιας ημέρας ήταν -1. Συνεπώς στις 2 τελευταίες γραμμές του κώδικα, οποιαδήποτε αρνητική τιμή όπου δεν βγάζει φυσικό νόημα τίθεται 0.

Έτσι, αν τυπώσουμε μια σειρά του dataset θα έχουμε τώρα το παρακάτω, πιο λογικό, αποτέλεσμα:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Στο dataset, υπάρχουν για κάθε χώρα κάποιες μέρες στην αρχή όπου δεν έχουν γίνει tests. Αυτό το θεωρούμε λογικό, καθώς μπορεί στην αρχή να μην γινόντουσαν τα test. Ωστόσο, υπάρχουν ημέρες στην συνέχεια όπου η τιμή των daily tests έχει αφεθεί κενή, ενώ τις προηγούμενες και επόμενες μέρες υπάρχει μεγάλος αριθμός daily tests. Έτσι θεωρούμε ότι είναι ελλιπείς τιμές και πρέπει να συμπληρωθούν. Έχουν αναφερθεί αρκετές τεχνικές για την συμπλήρωση ελλιπών τιμών σε ένα dataset αλλά αυτό που φάνηκε πιο λογικό στην συγκεκριμένη περίπτωση είναι η παρακάτω διαδικασία :

1. Εντοπισμός της πρώτης μέρας όπου υπήρξαν daily tests.
2. Αντικατάσταση των κενών τιμών μετά από αυτή την μέρα με τον μέσω όρο των 4 προηγούμενων ημερών.

Η υλοποίηση του παραπάνω γίνεται ως εξής:

# Set the date column to datetime type

df['Date'] = pd.to\_datetime(df['Date'])

# Sort values by entity and date

df = df.sort\_values(['Entity', 'Date'])

# For each entity, find the first date where 'Daily tests' is not zero

first\_test\_dates = df[df['Daily tests'] != 0].groupby('Entity')['Date'].idxmin()

# For each entity, if there's a first test date, replace the 'Daily tests' values before the first test date with NaN

for entity, idx in first\_test\_dates.iteritems():

    df.loc[(df['Entity'] == entity) & (df.index < idx), 'Daily tests'] = np.nan

# Calculate the 4-day rolling mean for 'Daily tests'

df['Rolling\_Mean'] = df.groupby('Entity')['Daily tests'].transform(lambda x: x.rolling(window=4, min\_periods=1).mean())

# Replace the zero 'Daily tests' values with the rolling mean values

df.loc[df['Daily tests'] == 0, 'Daily tests'] = df['Rolling\_Mean']

#Drop the 'Rolling\_Mean' column

df.drop('Rolling\_Mean', axis=1, inplace=True)

df = df.fillna(0)

**Ερώτημα 1**

Για τον υπολογισμό των βασικών συγκεντρωτικών μεγεθών του dataset έχουμε:

# Group by 'Entity' (country), and calculate the mean for each group

df\_grouped\_mean = df.groupby('Entity').mean()

# Calculate other statistics

df\_grouped\_sum = df[['Entity', 'Daily tests', 'Cases', 'Deaths']].groupby('Entity').sum()

df\_grouped\_max = df.groupby('Entity').max()

df\_grouped\_min = df.groupby('Entity').min()

display(df\_grouped\_sum)

display(df\_grouped\_mean)

display(df\_grouped\_max)

display(df\_grouped\_min)

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Αρχικά, βλέποντας τους μέσους όρους για κάθε χώρα των daily tests, cases και deaths, μας δίνεται μια καλή εικόνα για την εξέλιξη του ιού στην κάθε χώρα. Για παράδειγμα, βλέπουμε ότι η Αλγερία έχει πραγματοποιήσει πολύ μικρό αριθμό test και έχει και αρκετούς θανάτους. Επιπλέον, το Vietnam έχει πραγματοποιήσει παρόμοιο αριθμό daily tests με την Αλβανία αλλά οι θάνατοι του είναι πολύ λιγότεροι.

Εικόνα που περιέχει κείμενο, αριθμός, γραμμή, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, γραμμή, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματαΠαραπάνω φαίνονται οι μέσοι όροι για κάθε μια από τις στήλες για κάθε χώρα. Παρατηρούμε ότι το Vietnam έχει πολύ χαμηλό μέσω όρο θανάτων. Αντίστοιχα, η Αμερική έχει πολύ υψηλό. Βέβαια, λόγος αυτό είναι φυσικά και ο πληθυσμός της κάθε χώρας.

Εικόνα που περιέχει κείμενο, αριθμός, γραμμή, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματαΑντίστοιχα έχουμε τις μέγιστες τιμές της κάθε στήλης ανά χώρα. Ακολουθούν οι ελάχιστες τιμές.

Ο παρακάτω κώδικας μας δίνει κάποια συγκεντρωτικά χαρακτηριστικά (min, 25%, 50%, 75%, max) για κάθε στήλη:

# Group by 'Entity' (country)

df\_grouped = df.groupby('Entity')

# Apply describe() to each group

df\_grouped\_describe = df\_grouped.describe()

#df\_grouped\_describe.reset\_index(inplace=True)

display(df\_grouped\_describe)

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Για τον εντοπισμό των συσχετίσεων μεταξύ των χαρακτηριστικών της κάθε χώρας, κατασκευάζεται ένας correlation matrix

# Calculate the correlation matrix for all the numerical columns

corr\_matrix = df.corr()

# Plot a heatmap of the correlation matrix

plt.figure(figsize=(10, 8))

plt.title('Correlation Matrix Heatmap')

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.show()

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, παράλληλα, τετράγωνο

Περιγραφή που δημιουργήθηκε αυτόματα

Εννοείται ότι η τιμή 1 δείχνει πλήρης συσχέτιση και βρίσκεται στην διαγώνιο του πίνακα. Παρατηρώντας τον παραπάνω πίνακα, μπορούμε να εντοπίσουμε αρκετά ενδιαφέροντα χαρακτηριστικά. Όπως ήταν αναμενόμενο τα daily tests, cases, deaths έχουν πολύ μεγάλη συσχέτιση. Επιπλέον παρατηρούμε ότι ο μέσος πληθυσμός και ο πληθυσμός άνω των 65, έχουν μεγάλη συσχέτιση με τον αριθμό των νοσοκομείων και γιατρών ανά 1000 άτομα. Κάτι που φαίνεται παράξενο είναι η έλλειψη συσχέτισης μεταξύ των νοσοκομείων και γιατρών με τους θανάτους καθώς όπως γνωρίζουμε στην πραγματικότητα αποτέλεσαν κύριο στοιχείο στην εξέλιξη του ιού ( Ελάχιστες ΜΕΘ, έλλειψη χώρου για τους ασθενείς γενικότερα στα νοσοκομεία, έλλειψη προσωπικού. Τα παραπάνω συντέλεσαν σημαντικά στην θνησιμότητα του ιού). Ωστόσο, ίσως μπορεί να εξηγηθεί από το γεγονός ότι ο αριθμός νοσοκομείων και γιατρών παραμένει σταθερός όσο αυξάνονται οι θάνατοι και συνεπώς δεν λαμβάνει μεγάλο συντελεστή συσχέτισης.

Στη συνέχεια, παρατηρούμε τις γραφικές των cases/deaths κάθε χώρας

# Get a list of all unique countries in the data set

countries = df['Entity'].unique()

# Loop through each country and generate the corresponding plots

for country in countries:

    country\_df = df[df['Entity'] == country]

    # Line chart of the daily progression of Covid-19 cases and deaths

    country\_df.plot(x='Date', y=['Cases', 'Deaths'], kind='line', title=f'Daily Progression of Covid-19 Cases and Deaths in {country}', figsize=(10, 8))

    plt.xlabel('Date')

    plt.ylabel('Number of Cases/Deaths')

    plt.show()

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματαΑπό τις γραφικές μπορούμε να παρατηρήσουμε ότι η τάση των γραφικών παραμένει παρόμοια σχεδόν σε όλες τις χώρες. Δίνονται μερικές ως παράδειγμα.

**Ερώτημα 2**

Στο δεύτερο ερώτημα της εργασίας, ζητείται ο χωρισμός των χωρών σε συστάδες με βάση τις επιδώσεις τους στην αντιμετώπιση του ιού. Θα χρησιμοποιήσουμε τον αλγόριθμο k\_means ο οποίος είναι πολύ δημοφιλής σε προβλήματα χωρισμού των δεδομένων σε συστάδες. Οι επιδώσεις των χωρών θα αξιολογούνται με βάση τα εξής νέα κριτήρια: ποσοστό θετικότητας, ποσοστό θνησιμότητας, συνολικός αριθμός κρουσμάτων σε σχέση με τον πληθυσμό της χώρας. Βασικό βήμα στην χρήση του k\_means είναι η κανονικοποίηση των δεδομένων.

# Feature engineering

df\_grouped = df.groupby('Entity').sum()

df\_grouped['Positivity Rate'] = df\_grouped['Cases'] / df\_grouped['Daily tests']

df\_grouped['Mortality Rate'] = df\_grouped['Deaths'] / df\_grouped['Cases']

df\_grouped['Cases Per Capita'] = df\_grouped['Cases'] / df\_grouped['Population']

# Data normalization

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df\_normalized = scaler.fit\_transform(df\_grouped[['Positivity Rate', 'Mortality Rate', 'Cases Per Capita']])

Παραπάνω, κατασκευάζουμε τα νέα χαρακτηριστικά με βάση τα οποία θα χωριστούν οι χώρες σε συστάδες και κανονικοποιούμε τα δεδομένα. Για τον ορισμό του αριθμού των clusters όπου θα δημιουργηθούν από τον αλγόριθμο, χρησιμοποιείται ο αλγόριθμος Elbow Method.

Η μέθοδος του αγκώνα είναι μια ευρετική μέθοδος που χρησιμοποιείται για τον προσδιορισμό του αριθμού των συστάδων σε ένα σύνολο δεδομένων.Η ιδέα είναι η εκτέλεση του αλγορίθμου k-means στο σύνολο δεδομένων για ένα εύρος τιμών του k (num\_clusters, στον άξονα x) και για κάθε τιμή του k να υπολογίζετε το άθροισμα των τετραγωνικών σφαλμάτων (SSE, στον άξονα y).

Πιο συγκεκριμένα, για κάθε τιμή του k, υπολογίζουμε το άθροισμα τετραγώνων εντός της συστάδας (WCSS), το οποίο είναι το άθροισμα της τετραγωνικής απόστασης μεταξύ κάθε μέλους της συστάδας και του centroid της. Διαισθητικά, καθώς αυξάνουμε τον αριθμό των συστάδων, το άθροισμα τετραγώνων εντός της συστάδας θα μειώνεται, επειδή τα δείγματα θα είναι πιο κοντά στα centroid στα οποία αντιστοιχίζονται.

Ωστόσο, το SSE τείνει να μειώνεται προς το 0 καθώς αυξάνουμε το k (το SSE είναι 0 όταν το k είναι ίσο με τον αριθμό των σημείων δεδομένων στο σύνολο δεδομένων, διότι τότε κάθε σημείο δεδομένων είναι η δική του συστάδα και δεν υπάρχει σφάλμα μεταξύ αυτού και του κέντρου της συστάδας του). Επομένως, η ιδέα είναι να επιλέξουμε το k στο οποίο το SSE μειώνεται απότομα. Αυτό παράγει ένα "φαινόμενο αγκώνα" στο γράφημα, και το σημείο αυτό θεωρείται ως ο κατάλληλος αριθμός συστάδων.

distortions = []

K = range(1,10)

for k in K:

    kmeanModel = KMeans(n\_clusters=k).fit(df\_normalized)

    distortions.append(sum(np.min(cdist(df\_normalized, kmeanModel.cluster\_centers\_, 'euclidean'), axis=1)) / df\_normalized.shape[0])

plt.plot(K, distortions, 'bx-')

plt.xlabel('k')

plt.ylabel('Distortion')

plt.title('The Elbow Method showing the optimal k')

plt.show()

Εικόνα που περιέχει κείμενο, γραμμή, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Επιλέγουμε k=4 καθώς για μεγαλύτερους αριθμούς k, δεν έχουμε απότομη πτώση.

# Clustering

kmeans = KMeans(n\_clusters=4, random\_state=0)  # Choose the appropriate number of clusters

df\_grouped['Cluster'] = kmeans.fit\_predict(df\_normalized)

# Analysis

for cluster in df\_grouped['Cluster'].unique():

    print(f"Cluster {cluster}:")

    print(df\_grouped[df\_grouped['Cluster'] == cluster].index.tolist())

Με βάση τον παραπάνω κώδικα, εκτελείται ο αλγόριθμος K\_means.

Στην πρώτη συστάδα :

Cluster 0:

['Albania', 'Argentina', 'Armenia', 'Austria', 'Bahrain', 'Belgium', 'Chile', 'Colombia', 'Costa Rica', 'Croatia', 'Estonia', 'France', 'Ireland', 'Israel', 'Jordan', 'Kuwait', 'Latvia', 'Lithuania', 'Luxembourg', 'Malta', 'Panama', 'Poland', 'Portugal', 'Qatar', 'Romania', 'Serbia', 'Slovakia', 'Slovenia', 'Sweden', 'Switzerland', 'United Kingdom', 'United States']

Στην δεύτερη συστάδα :

Cluster 1:

['Bangladesh', 'Belarus', 'Bhutan', 'Canada', 'Cape Verde', 'Cuba', 'Cyprus', 'Denmark', 'Dominican Republic', 'Ethiopia', 'Finland', 'Ghana', 'Iceland', 'India', 'Indonesia', 'Iraq', 'Jamaica', 'Japan', 'Kazakhstan', 'Kenya', 'Libya', 'Madagascar', 'Malaysia', 'Mauritania', 'Mongolia', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia', 'Nepal', 'New Zealand', 'Nigeria', 'Norway', 'Oman', 'Pakistan', 'Paraguay', 'Philippines', 'Russia', 'Rwanda', 'Saudi Arabia', 'Senegal', 'South Korea', 'Sri Lanka', 'Thailand', 'Togo', 'Trinidad and Tobago', 'Turkey', 'Uganda', 'Ukraine', 'United Arab Emirates', 'Uruguay', 'Vietnam', 'Zambia']

Στην τρίτη συστάδα:

Cluster 2:

['Algeria']

Και τέλος στην τέταρτη συστάδα:

Cluster 3:

['Australia', 'Bolivia', 'Bosnia and Herzegovina', 'Bulgaria', 'Ecuador', 'El Salvador', 'Fiji', 'Greece', 'Guatemala', 'Hungary', 'Iran', 'Italy', 'Malawi', 'Mexico', 'Peru', 'South Africa', 'Tunisia', 'Zimbabwe']

Αρχικά παρατηρούμε ότι η Αλγερία βρίσκεται μόνη της σε μία συστάδα. Θα μπορούσε να είναι μια ακραία περίπτωση και να πρέπει να συνταχθεί σε μια από τις υπόλοιπες συστάδες, ίσως μειώνοντας τον αριθμό των clusters. Ωστόσο, περαιτέρω ανάλυση θα δείξει ότι ορθά αποτελεί μια συστάδα από μόνη της (ίσως και για αυτό τον λόγο να ονομάζεται το dataset covid19-Algeria-and-World-Dataset!).

Για την ανάλυση των αποτελεσμάτων χρησιμοποιούμε τον παρακάτω κώδικα :

# Create a list of features

features = ['Positivity Rate', 'Mortality Rate', 'Cases Per Capita']

# Create a scatter plot for each pair of features

fig, axs = plt.subplots(len(features), len(features), figsize=(15, 15))

# Create a dictionary that maps each cluster to a color

cluster\_colors = {0: 'blue', 1: 'green', 2: 'red', 3: 'purple', 4: 'orange'}  # Adjust based on your clusters

# Create a list of colors for each point in your dataset

colors = [cluster\_colors[cluster] for cluster in df\_grouped['Cluster']]

for i in range(len(features)):

    for j in range(len(features)):

        axs[i, j].scatter(df\_grouped[features[j]], df\_grouped[features[i]], c=colors)

        axs[i, j].set\_xlabel(features[j])

        axs[i, j].set\_ylabel(features[i])

# Create a legend

import matplotlib.patches as mpatches

patches = [mpatches.Patch(color=color, label=f'Cluster {cluster}') for cluster, color in cluster\_colors.items()]

plt.legend(handles=patches, bbox\_to\_anchor=(1.05, 1), loc='upper left')

plt.tight\_layout()

plt.show()

Εικόνα που περιέχει κείμενο, διάγραμμα, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Με βάση το παραπάνω σύνολο γραφικών μπορούμε να εξάγουμε τα εξής συμπεράσματα:

1. Το cluster 1, σημειωμένο με πράσινο χρώμα, τα έχει πάει καλά στην αντιμετώπιση του ιού. Με βάση την δεύτερη γραφική της πρώτης σειράς, είχε χαμηλό mortality και positivity rate. Με βάση την τρίτη γραφική της ίδια σειράς, ο συνολικός αριθμός κρουσμάτων των χωρών όπου βρίσκονται σε αυτή την συστάδα σε σχέση με τον συνολικό πληθυσμό τους είναι μικρός αλλά έχουν και μικρή θετικότητα. Από την τρίτη γραφική της δεύτερης σειράς, παρατηρούμε ότι οι χώρες της συστάδας 1 έχουν και χαμηλό δείκτη αριθμού κρουσμάτων αλλά και σχετικά περιορισμένο δείκτη θνησιμότητας. Ωστόσο, αυτό δεν είναι απαραίτητα καλή επίδοση καθώς θα περιμέναμε αφού δεν έχει πολλά κρούσματα σε σχέση με τον συνολικό πληθυσμό της, να έχουν και αρκετά χαμηλά ποσοστά θνησιμότητας.
2. Το cluster 0 φαίνεται να τα έχει πάει ακόμα καλύτερα στην αντιμετώπιση του ιού. Από την δεύτερη γραφική της πρώτης σειράς βλέπουμε ότι το mortality και το positivity rate είναι περιορισμένα. Από την τρίτη γραφική της πρώτης σειράς, παρατηρούμε ότι ενώ τα κρούσματα σε σχέση με τον συνολικό πληθυσμό είναι πολλά, η θετικότητα είναι μικρή. Επιπλέον, από την τρίτη γραφική της δεύτερης σειράς, φαίνεται ότι ενώ υπάρχουν περισσότερα κρούσματα σε σχέση με τον συνολικό πληθυσμό του clyster 1, έχουν περίπου τις ίδιες τιμές θνησιμότητας. Αυτό σημαίνει ότι έλαβαν καλύτερα μέτρα για την αποφυγή θανάτου των κρουσμάτων.
3. Το cluster 3 τα έχει πάει χειρότερα από τα 2 προηγούμενα clusters (εννοείται ότι εννοούμε τις χώρες που ανήκουν στο cluster 3). Συγκεκριμένα, παρατηρούμε από τις γραφικές ότι έχει υψηλό δείκτη θνησιμότητας σε σχέση με την θετικότητα και ενώ έχει μειωμένο αριθμό κρουσμάτων σε σχέση με τον συνολικό της πληθυσμό, έχει μεγαλύτερο δείκτη θνησιμότητας από τα δύο άλλα clusters. Αυτό δείχνει μια αδυναμία στην πρόληψη του θανάτου από τον ιο.
4. Το cluster 2 αποτελείται μόνο από την χώρα της Αλγερίας. Η Αλγερία τα έχει πάει χειρότερα από τις υπόλοιπες χώρες. Από τις γραφικές παραστάσεις συμπεραίνουμε ότι το ποσοστό θετικότητας ήταν πολύ μεγάλο αλλά και πολύ μεγάλο σε σχέση με τα κρούσματα του πληθυσμού της. Επιπλέον βλέπουμε ότι είχε μεγάλο ποσοστό θνησιμότητας αναλογικά με τα κρούσματα σε σχέση με τον συνολικό πληθυσμό της.

Για μία διαφορετική οπτική των συστάδων, κατασκευάστηκε ένα 3d scatter plot :

fig = plt.figure(figsize=(10, 10))

ax = fig.add\_subplot(111, projection='3d')

scatter = ax.scatter(df\_grouped['Positivity Rate'],

                     df\_grouped['Mortality Rate'],

                     df\_grouped['Cases Per Capita'],

                     c=df\_grouped['Cluster'],

                     cmap='viridis')

# Create colorbar

cb = plt.colorbar(scatter, ax=ax, pad=0)

cb.set\_label('Cluster')

ax.set\_xlabel('Positivity Rate')

ax.set\_ylabel('Mortality Rate')

ax.set\_zlabel('Cases Per Capita')

ax.view\_init(elev=20,azim=45)

plt.show()

**Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα**

Για μια πιο εύκολη κατανόηση των αποτελεσμάτων χρησιμοποιήθηκε ο παρακάτω κώδικας :

# Define success criteria

success\_criteria = ['Positivity Rate', 'Mortality Rate', 'Cases Per Capita']

# Calculate mean success criteria for each cluster

cluster\_performance = df\_grouped.groupby('Cluster')[success\_criteria].mean()

# Print the results

print(cluster\_performance)

Εικόνα που περιέχει κείμενο, απόδειξη, γραμματοσειρά, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρούμε εύκολα ότι η Αλγερία διαθέτει έναν τεράστιο αριθμό θετικότητας με αρκετά μεγάλο αριθμό θνησιμότητας και μικρό αριθμό κρουσμάτων σε σχέση με τον πληθυσμό. Αυτό σημαίνει ότι ενώ υπήρχαν λίγα κρούσματα σε σχέση με τον πληθυσμό, η Αλγερία δεν κατάφερε να μειώσει την θνησιμότητα. Ένας ακόμη λόγος του υψηλού δείκτη θετικότητας είναι ο υπερβολικά χαμηλός αριθμός τεστ όπου έκανε.

**Ερώτημα 3**

Στο τρίτο ερώτημα ζητείται το ποσοστό της θετικότητας στην Ελλάδα 3 ημέρες μετά από την ημερομηνία στην οποία γίνεται η ανάλυση. Για την κατασκευή των παλιδρομητών χρειάζεται αρχικά μια προεπεξεργασία των δεδομένων:

df\_greece = df[df['Entity'] == 'Greece']

# Replace NaNs with 0

df\_greece['Daily tests'].fillna(0, inplace=True)

# Find the first non-zero 'Daily tests' value

first\_nonzero\_tests = df\_greece['Daily tests'].ne(0).idxmax()

# Discard all rows before the first non-zero 'Daily tests' value

df\_greece = df\_greece.loc[first\_nonzero\_tests:]

# Calculate the positivity rate

df\_greece['Positivity Rate'] = df\_greece['Cases'] / df\_greece['Daily tests']

# Fill any missing or infinite values resulting from division by zero with 0

df\_greece['Positivity Rate'] = df\_greece['Positivity Rate'].replace([np.inf, -np.inf], np.nan)

df\_greece['Positivity Rate'].fillna(0, inplace=True)

# Compute the rolling mean with a window size of 4

df\_greece['Positivity Rate 4-Day MA'] = df\_greece['Positivity Rate'].rolling(window=3).mean()

#Identify where the 0 values in the positivity rate are

zero\_pos\_rate\_indices = df\_greece[df\_greece['Positivity Rate'] == 0].index

# Replace these 0 values with the 4-day moving average

df\_greece.loc[zero\_pos\_rate\_indices, 'Positivity Rate'] = df\_greece.loc[zero\_pos\_rate\_indices, 'Positivity Rate 4-Day MA']

Στον παραπάνω κώδικα, κατασκευάζουμε ένα dataframe το οποίο περιέχει μόνο τα στοιχεία της Ελλάδας και στη συνέχεια υπολογίζεται το ποσοστό θετικότητας. Οι πρώτες ημέρες δεν περιέχουν αριθμό daily tests. Ως αποτέλεσμα, κατά τον υπολογισμό του ποσοστού θετικότητας δημιουργούνται NA τιμές. Θα μπορούσαμε απλώς να τις αντικαταστήσουμε με 0 αλλά αυτές οι τιμές μπορούν να μπερδέψουν τον παλινδρομητή. Για αυτό τον λόγο, ξεκινάμε τον υπολογισμό του ποσοστού θετικότητας μετά την πρώτη μη μηδενική τιμή των daily tests.

Για τις τιμές positivity rate όπου προκύπτουν 0 στο dataset, τις αντικαθιστούμε με τον μέσο όρο των 4 προηγούμενων ημερών.

df\_greece['Positivity Rate Future'] = df\_greece['Positivity Rate'].shift(-3)

# 2. Feature engineering

for i in range(1, 4):

  df\_greece.loc[:, f'Positivity Rate Lag {i}'] = df\_greece['Positivity Rate'].shift(i)

df\_greece = df\_greece.dropna()

# 3. Splitting the data

train = df\_greece[df\_greece['Date'] < '2021-01-01']

test = df\_greece[df\_greece['Date'] >= '2021-01-01']

X\_train = train[[f'Positivity Rate Lag {i}' for i in range(1, 4)]]

y\_train = train['Positivity Rate Future']

X\_test = test[[f'Positivity Rate Lag {i}' for i in range(1, 4)]]

y\_test = test['Positivity Rate Future']

Με βάση τον παραπάνω κώδικα δημιουργούμε σετ 3 συνεχόμενων ημερών και τοποθετούμε την τιμή θετικότητας την επόμενη μέρα ως την επιθυμητή έξοδο του μοντέλου μας. Στην συνέχεια ορίζουμε την συνάρτηση απώλειας και το μοντέλο:

def mape\_loss(y\_true, y\_pred):

    return tf.reduce\_mean(tf.abs((y\_true - y\_pred) / tf.clip\_by\_value(tf.abs(y\_true), tf.keras.backend.epsilon(), np.inf)), axis=-1)

# Reshape input to be 3D [samples, timesteps, features]

X\_train = X\_train.values.reshape((X\_train.shape[0], 1, X\_train.shape[1]))

# Initialize the RNN model

model = Sequential()

# Add an LSTM layer with 50 units

model.add(LSTM(100, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

model.add(Dropout(0.5))  # 20% dropout

# Add a Dense layer with 1 unit

model.add(Dense(1))

# Compile the model

model.compile(loss='mse', optimizer='adam')

# Fit the model

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=72, validation\_split=0.2, verbose=2, shuffle=False)

Μετά την τελευταία εποχή έχουμε :

Epoch 50/50

4/4 - 0s - loss: 0.0018 - val\_loss: 0.0050 - 31ms/epoch - 8ms/step

Κάνοντας την πρόβλεψη έχουμε:  
X\_test\_reshaped = X\_test.values.reshape((X\_test.shape[0], 1, X\_test.shape[1]))

predictions = model.predict(X\_test\_reshaped)

plt.figure(figsize=(10,6))

plt.plot(y\_test.values, label='Actual')

plt.plot(predictions, label='Predicted')

plt.xlabel('Test set index')

plt.ylabel('Positivity rate')

plt.legend()

plt.show()

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Αντίστοιχα για την κατασκευή του SVM παλινδρομητή έχω :

# Scale the features

scaler = StandardScaler()

#X\_train\_scaled = scaler.fit\_transform(X\_train)

#X\_test\_scaled = scaler.transform(X\_test)

X\_train\_scaled = X\_train

X\_test\_scaled = X\_test

parameters = {'kernel': ['rbf','poly','linear'],

              'C': [0.1, 1,10,100],

              'gamma': [0.1, 1, 10, 100],

              'epsilon': [0.1, 0.2, 0.3, 0.4, 0.5]}

svm = SVR()

grid\_search = GridSearchCV(svm, parameters)

grid\_search.fit(X\_train\_scaled, y\_train)

best\_parameters = grid\_search.best\_params\_

print("Best parameters: ", best\_parameters)

svm = SVR(kernel=best\_parameters['kernel'], C=best\_parameters['C'], gamma=best\_parameters['gamma'], epsilon=best\_parameters['epsilon'])

svm.fit(X\_train\_scaled, y\_train)

# Make predictions

y\_train\_pred = svm.predict(X\_train\_scaled)

y\_test\_pred = svm.predict(X\_test\_scaled)

# Compute MAE

mse\_train = mean\_squared\_error(y\_train, y\_train\_pred)

mse\_test = mean\_squared\_error(y\_test, y\_test\_pred)

print('Train MSE: ', mse\_train)

print('Test MSE: ', mse\_test)

Για την επιλογή των καλύτερων παραμέτρου του SVM χρησιμοποιούμε την μέθοδο gridsearch για την εύερεση των c, epsilon, gamma και kernel.

Best parameters: {'C': 100, 'epsilon': 0.2, 'gamma': 1, 'kernel': 'rbf'}

Train MSE: 0.0029062381038616396

Test MSE: 0.004997135542045255

Ύστερα, για την κατανόηση των αποτελεσμάτων κατασκευάζουμε τις γραφικές παραστάσεις :

# Predictions

y\_train\_pred = svm.predict(X\_train\_scaled)

y\_test\_pred = svm.predict(X\_test\_scaled)

plt.figure(figsize=(14, 5))

# Plot actual and predicted values of the training set

plt.subplot(1, 2, 1)

plt.plot(y\_train.values, label='Actual')

plt.plot(y\_train\_pred, label='Predicted')

plt.title('Training Set')

plt.legend()

# Plot actual and predicted values of the testing set

plt.subplot(1, 2, 2)

plt.plot(y\_test.values, label='Actual')

plt.plot(y\_test\_pred, label='Predicted')

plt.title('Testing Set')

plt.legend()

plt.show()

Εικόνα που περιέχει στιγμιότυπο οθόνης, γράφημα, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Όπως παρατηρούμε, και οι δύο παλινδρομητές έχουν σφάλμα 0.005 στα test data. Αν και η τιμή αυτή δεν είναι τεράστια, το ποσοστό θετικότητας είναι ήδη μικρός αριθμός και συνεπώς μία τέτοια απώλεια πιθανών να είναι αρκετά σημαντική. Και στους δύο παλινδρομητές παρατηρούμε την δυσκολία πρόβλεψης της μεγάλης ακμής όπου δημιουργείται στην γραφική του testing. Αυτό δεν δημιουργεί μεγάλη εντύπωση καθώς απότομες αλλαγές στα δεδομένα είναι δύσκολο να προβλεπτούν.