Department of Computer Science
Technical University of Cluj-Napoca

Intelligent Systems

Exam 2019-2020

Machine learning's take on Covid-19
Tool: Google Colab, Scikit Learn

Name: Ossian Adrian
Group: 30233
Email: ossianadrian@yahoo.com

Cadru didactic: Groza Adrian

Contents

Chapter 1

Prezentarea temei alese

1. Articolul ales. Problema aleasa pentru proiect. Descrierea problemei

   Articolul ales in cadrul acestui proiect face referinta la reutilizarea algoritmilor AI deja existenti si indreptarea lor catre cauza care ne afecteaza cel mai mult in momentul actual, pandemia.

   Domeniul acestui proiect este situatia globala pandemica COVID-19. COVID-19 este un sindrom respirator viral cauzat de coronavirusul sindromului respirator acut sever 2. Acesta este sindromul implicat în epidemia de coronavirus ce a inceput in anul 2019 in China. Modul primar de infecție la om este transmiterea om-om, care are loc în general prin picături de secreții respiratorii de la persoane infectate care sunt expulzate prin strănut, tuse sau expirație.

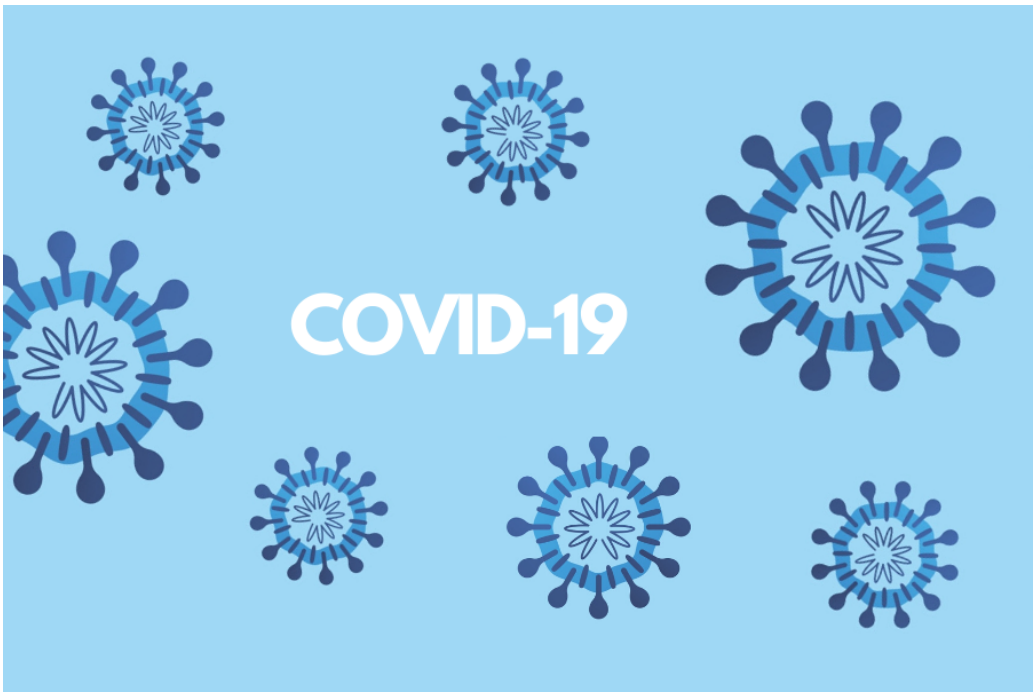

Figure 1.1: Covid-19

2. Scopul acestui proiect

In cadrul acestui proiect se vor prezenta eficienta algoritmilor de ML in predictia situatiei globale Covid-19. Cazul de referinta este cel al Frantei, unde situatia a ajuns sa fie depresionanta. Setul de date ales se poate gasi pe site-ul Kaggle la adresa `https://www.kaggle.com/ashudata/covid19dataset#COVID_Data.csv`. In principal este vorba despre evolutia cazurilor in cadrul pandemiei. Tabelul are 4 coloane importante pe care ne bazam in acest proiect: Cazuri totale, cazuri noi aparute in ziua respectiva, mortalitatea totala si cazuri noi de mortalitate aparute in ziua respectiva. Ne vom folosi de algoritmul KNN pentru a prezice cresterea in cadrul pandemiei. De asemenea, un alt caz de utilizare al acestui proiect il reprezinta prezicerea masurilor care trebuie luate in functie de evolutie.

Chapter 2

Decizii de implementare. Instrumente utilizate. Provenienta datelor.

1. In realizarea acestui proiect s-a folosit platforma Google Colab pentru a facilita compatibilitatea indiferent de sistemul de operare ales. Astfel, codul poate fi vizualizat si rulat chiar si de pe dispozitive mobile.

2. In realizarea acestui proiect au fost folosite date reale preluate de pe site-ul Kaggle. Tabelul este prezentat sub urmatoarea forma:

	A	B	C	D	E	F
1	Country	Date	Confirmed	Death	newConfir	newDeath
2	France	1/1/2020	0	0	0	0
3	France	1/2/2020	0	0	0	0
4	France	1/3/2020	0	0	0	0
5	France	1/4/2020	0	0	0	0
6	France	1/5/2020	0	0	0	0
7	France	1/6/2020	0	0	0	0
8	France	1/7/2020	0	0	0	0
9	France	1/8/2020	0	0	0	0
10	France	1/9/2020	0	0	0	0

Figure 2.1: Data set

- Coloana country reprezinta tara de referinta (in cazul nostru este doar Franta), coloana Date reprezinta data la care a fost dat raportul, coloana confirmed reprezinta numarul de cazuri confirmate pana la data respectiva, coloana death reprezinta numarul de persoane decedate pana la data respectiva, iar coloanele newConfirmed si newDeath reprezinta numarul de cazuri noi aparute respectiv numarul de persoane decedate strict in acea zi.
- Acest set de date etichetate a fost ales pentru versatilitatea pe care o ofera si pentru acuratetea lui.

3. Explicarea alegerii metodei de supervised machine learning.

Metoda de supervised learning machine se potriveste excelent cu problema propusa. Dimensiunea setului de date este relativ redusa si are date strict numerice sau de tip data calendaristica. Aceasta metoda va permite sa invatam sistemul pe un set de antrenare concret oferit, iar dupa aceea, sistemul nu va trebui decat sa prezica corect cateva date de test ce nu au fost date in procesul de invatare.

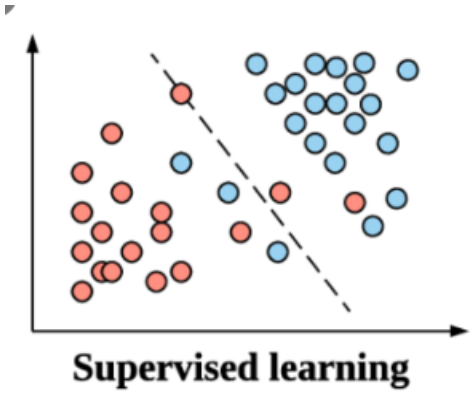

Figure 2.2: Supervised learning

4. Explicarea alegerii regresiei si a algoritmului KNN

Prin definiție, modelul probabilistic prin regresie este creat pentru a aproxima mapping functions (f) din variabile de intrare (X) către un set continuu de variabile de ieșire (Y). Practic, va fi estimată creștearea (sau descresterea) unei funcții.

În recunoașterea modelelor, algoritmul de vecini KNN este o metodă non-parametrică propusă de Thomas Cover, utilizată pentru clasificare și regresie. În ambele cazuri, intrarea constă din k exemple de instruire k cele mai apropiate în spațiul de caracteristici. Rezultatul depinde dacă k-NN este utilizat pentru clasificare sau regresie. In cazul curent KNN este folosit pentru regresie. În regresia KNN, ieșirea este valoarea proprietății pentru obiect. Această valoare este media valorilor k vecini apropiați. k-NN este un tip de învățare bazată pe instanță sau învățare leneșă, unde funcția este aproximativă locală și toată calcularea este amânată până la evaluarea funcției.

O tehnică utilă poate fi alocarea greutăților la contribuțiile vecinilor, astfel încât vecinii mai apropiați să contribuie mai mult la medie decât cei mai îndepărtați. De exemplu, o schemă comună de ponderare constă în acordarea fiecărui vecin a unei greutăți de $\frac{1}{d}$, unde d este distanța față de vecin. Vecinii sunt luați dintr-un set de obiecte pentru valoarea proprietății obiectului. Aceasta poate fi considerată setul de pregătire pentru algoritm, deși nu este necesară o etapă de instruire explicită.

O particularitate a algoritmului KNN este că acesta este sensibil la structura locală a datelor. Algoritmul este suficient de precis pentru problema abordata, deoarece ne folosim de datele zilelor anterioare, astfel KNN reuseste sa aproximeze practic o functie matematica dinamica.

Chapter 3

Explicarea modelului implementat

Codul este împărțit în 5 capitole și este anexat în tabelul Exam 2020 prezent pe platforma facultății UTCN. În cele ce urmează, se vor explica deciziile luate în implementare și se vor explica pașii urmați.

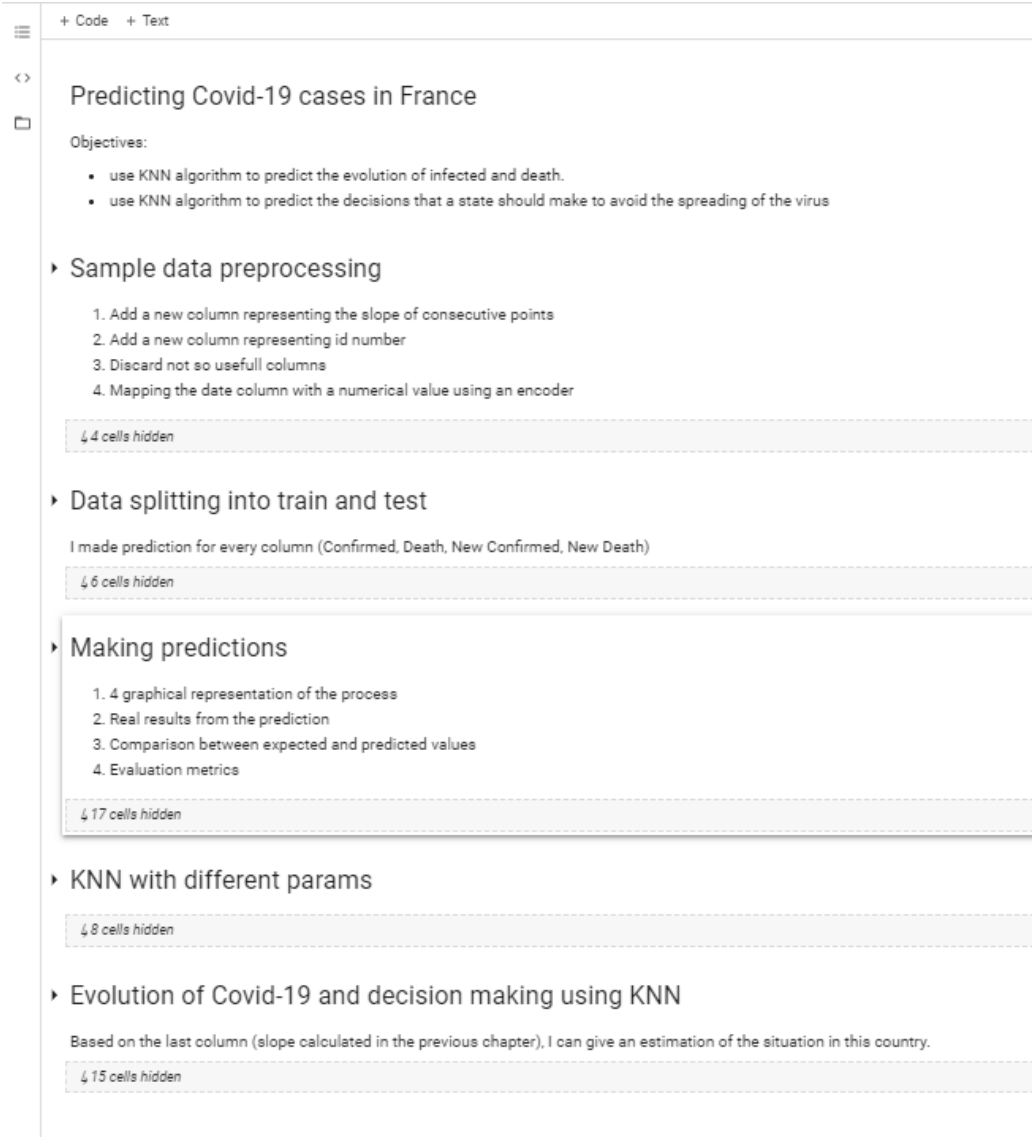

Figure 3.1: Chapters of the code

1. Sample data preprocessing

Primul pas in preprocesarea datelor de intrare este adaugarea unei noi coloane care sa reprezinte un identificator pentru aceste date. Adica fiecarei linii ii va fi asociat un numar de referinta.

Al doilea pas de preprocesare este adaugarea unei coloane noi in tabel ce va reprezenta panta dreptei determinata de punctul curent si punctul anterior. Aceasta a fost calculata folosind formula m = $\frac{y2-y1}{x2-x1}$, unde perechea (x2,y2) reprezinta coordonatele punctului curent, iar perechea (x1,y1) reprezinta coordonatele punctului anterior. Coordonata x va reprezenta data la care a fost adaugat raportul, iar y va reprezenta numarul de cazuri noi aparute in acea zi. Deoarece datele sunt consecutive in setul nostru, valoarea pentru numitorul fractiei va fi mereu 1, iar valoarea numaratorului va fi diferenta de cazuri noi aparute dintre cele doua zile consecutive.

In cel de al treilea pas se va scapa de coloanele nefolositoare. In cazul acesta este vorba doar despre prima coloana reprezentand tara. Proiectul acesta evalueaza situatia doar pentru tara Franta, deci nu este nevoie de specificarea tarii.

Pentru a putea aplica algoritmul KNN, va trebui sa mapam coloana cu data la o valoare numerica. Pentru a realiza acest lucru, se va folosi Label Encoder. Tabelul cu care se va lucra va fi sub urmatoarea forma:

```
[19] #To make this work, I need to encode the date column, because there
     #For that, a label Encoder may be used.

     from sklearn import preprocessing

     labelEncoder = preprocessing.LabelEncoder()

     labelEncoder.fit(myData['Date'])
     myData.loc[:, 'Date'] = labelEncoder.transform(myData['Date'])
```

+ Code + Text

```
[20] myData.head()
```

	Date	Confirmed	Death	New Confirmed	New Death	Nr. Crt	Growth
0	10	0	0	0	0	2	0
1	21	0	0	0	0	3	0
2	24	0	0	0	0	4	0
3	25	0	0	0	0	5	0
4	26	0	0	0	0	6	0

Figure 3.2: Table

2. Data splitting into training set and testing set

Pentru a imparti datele in date de antrenare si de testare, am folosit train_test_split implementat de sklearn. Proportiile de divizare sunt 20% cantitate de testare, 80% cantitate de antrenare.

```
[ ] from sklearn.model_selection import train_test_split


    y = myData.iloc[:, 1].values
    y = y.reshape(-1,1)

    X_usedValues = myData.iloc[:, 0:1].values #only selected the date and the confirmed column
    #Split the date into training set and test set
    X_train, X_test, y_train, y_test = train_test_split(X_usedValues, y, test_size=0.20)

    print(len(X_train))
    print(len(X_test))
```

```
110
28
```

Figure 3.3: Splitting

Tot in acest capitol au fost printate si datele impartite de catre train_test_split pentru a observa ca sunt impartite random. Astfel voi rula mai multe teste si voi face o medie a acuratetilor obtinute.

3. Making predictions

In acest capitol au fost realizate predictiile propriu zise. In prima parte a fost plotat un grafic ce reprezinta modul de functionare a algoritmului KNN. Pentru algoritm au fost alese valorile de 3 pentru numarul vecini, 60 pentru leaf size si algoritmul brute. Rezultatul se poate observa in urmatoarea figura.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor

#Get prediction for number of confirmed persons

n_neighbors = 3 #Number of neighbors chosen is 3. I will also try with other values
for i, weights in enumerate(['uniform', 'distance']):
    knn = KNeighborsRegressor(n_neighbors, weights=weights, algorithm = 'brute', leaf_size = 60) #Initialize regressor
    y_result = knn.fit(X_train, y_train).predict(X_test)
    #Plot a graph to visualize
    plt.subplot(2, 1, i + 1)
    plt.scatter(X_train[:,0], y_train, color='darkorange', label='data')
    plt.plot(X_test[:,0], y_result, color='navy', label='prediction')
    plt.axis('tight')
    plt.legend()
    plt.title("KNeighborsRegressor (k = %i, weights = '%s')" % (n_neighbors,
                                                                weights))

plt.tight_layout()
plt.show()
```

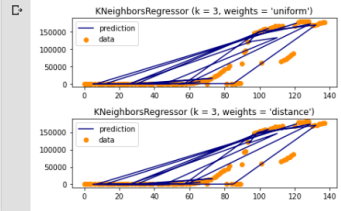

Figure 3.4: Graph KNN

In figura urmatoare se pot compara rezultatele asteptate (stanga) cu cele obtinute (dreapta). Observam ca algoritmul este destul de precis, rezultatele sunt satisfacatoare.

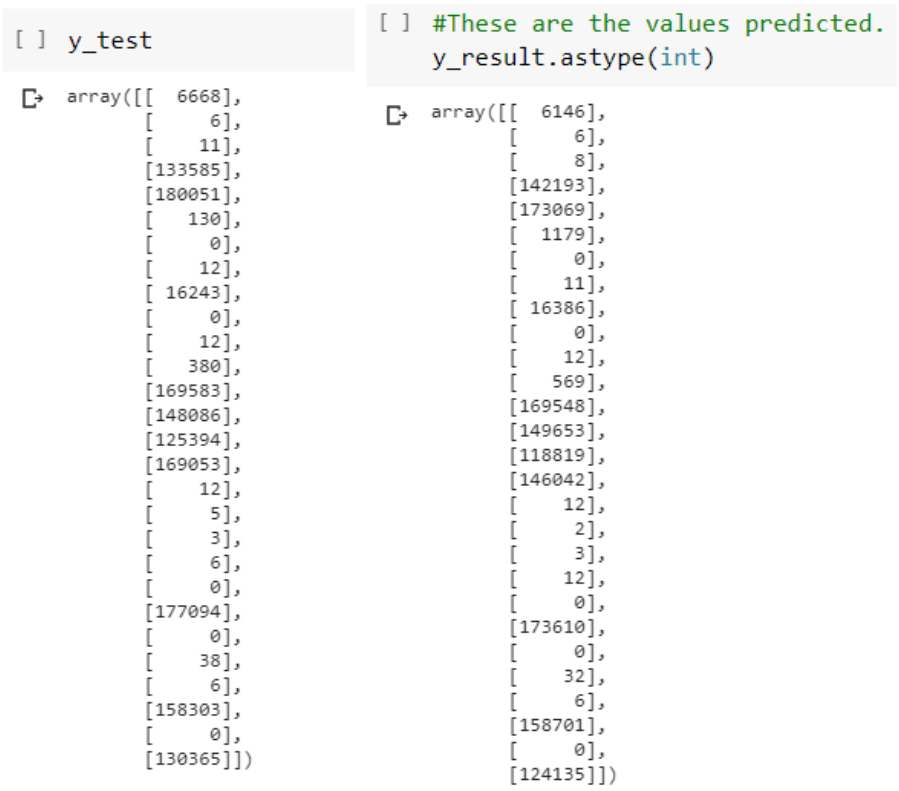

Figure 3.5: Real vs Predicted

Au fost aplicate metrici de evaluare pentru o viitoare comparare a algoritmului folosind diferiti parametri. Pentru cei actuali valori obtinute sunt prezente in figura urmatoare.

```
[ ] #We can use different metrics to evaluate these predictions.
    from sklearn.metrics import explained_variance_score
    explained_variance_score(y_test, y_result.astype(int))
```

```
0.995205033783651
```

```
[ ] from sklearn.metrics import max_error
    max_error(y_test, y_result.astype(int))
```

```
23011
```

```
[ ] from sklearn.metrics import mean_absolute_error
    mean_absolute_error(y_test, y_result.astype(int))
```

```
2100.4285714285716
```

```
[ ] from sklearn.metrics import mean_squared_error
    mean_squared_error(y_test, y_result.astype(int))
```

```
26806362.64285714
```

```
[ ] from sklearn.metrics import mean_squared_log_error
    mean_squared_log_error(y_test, y_result.astype(int))
```

```
0.21478358568590927
```

```
[ ] from sklearn.metrics import median_absolute_error
    median_absolute_error(y_test, y_result.astype(int))
```

```
6.0
```

```
[ ] from sklearn.metrics import r2_score
    r2_score(y_test, y_result.astype(int))
```

```
0.9949101846878372
```

Figure 3.6: Evaluation

4. KNN with different params

Acest capitol are ca scop compararea rezultatelor obtinute folosind diferiti parametri pentru algoritmul KNN. Au fost rulate cate 5 teste pentru fiecare configuratie de parametri si s-a facut media lor. In cele ce urmeaza, se considera urmatoarea notatie pentru parametrii alesi:

(a) Vom nota cu C1 configuratia cu 3 vecini, algoritmul Brute si leaf_size = 60.

(b) Vom nota cu C2 configuratia cu 20 de vecini, algoritmul kd_tree si leaf_size = 30.

(c) Vom nota cu C3 configuratia cu 5 vecini, algoritmul ball_tree si leaf_size = 30.

(d) Vom nota cu C4 configuratia cu 2 vecini, algoritmul brute si leaf_size = 50.

(e) Vom nota cu C5 configuratia cu 60 de vecini, algoritmul brute si leaf_size = 60.

Rezultatele obtinute in urma testelor sunt reprezentate in graficele urmatoare:

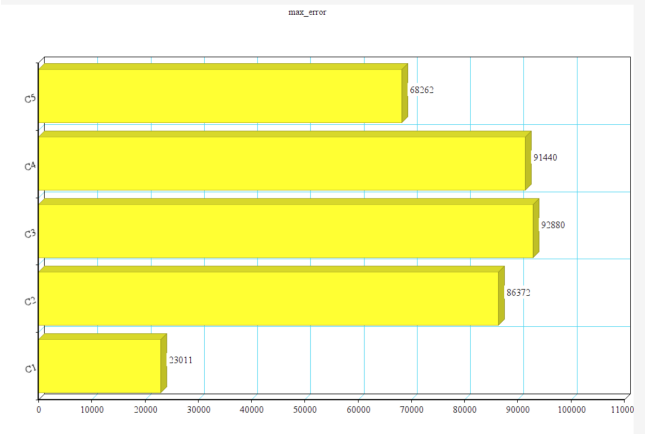

Figure 3.7: Max error

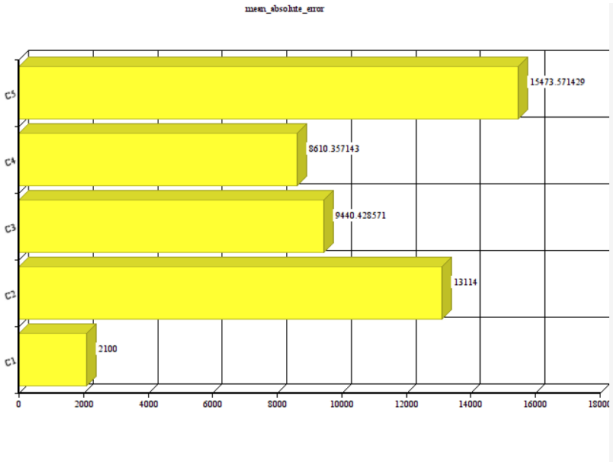

Figure 3.8: Mean absolute error

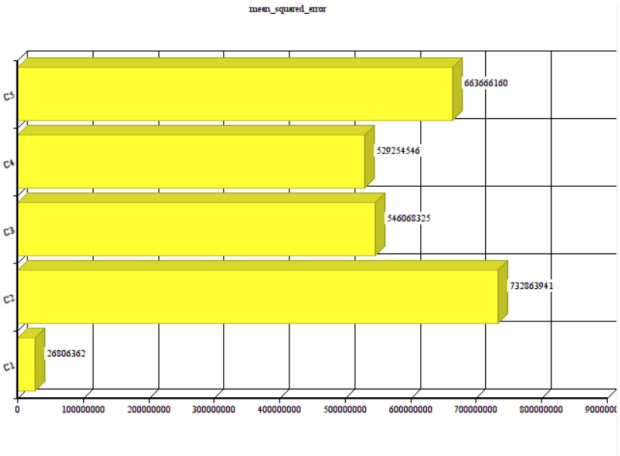

Figure 3.9: Mean squared error

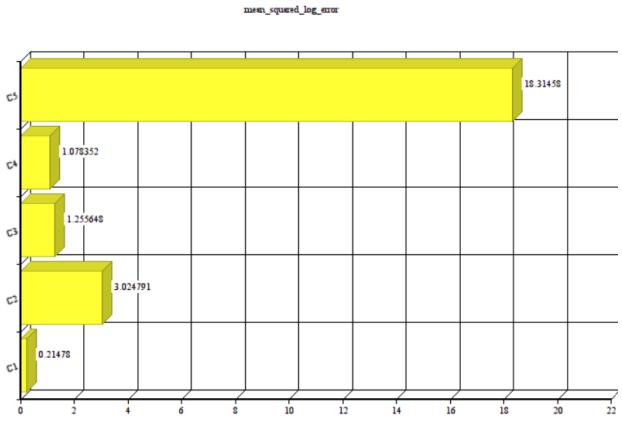

Figure 3.10: Mean squared log error

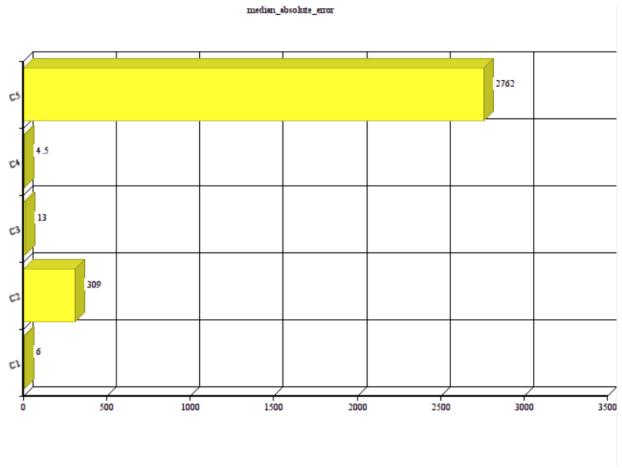

Figure 3.11: Median absolute error

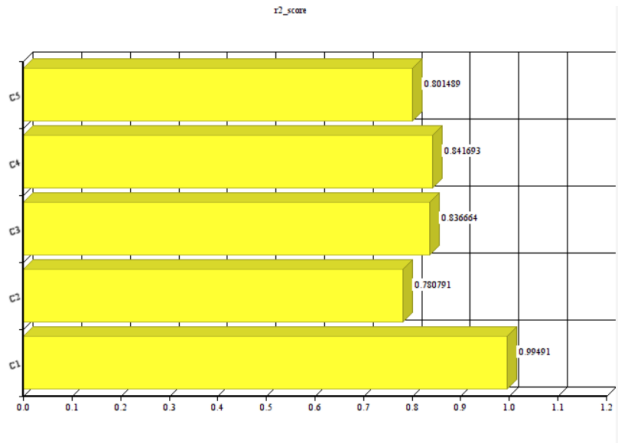

Figure 3.12: r2 score

Se pare ca, in urma testelor efectuate, configuratia C1 folosita initial ramane configuratia cu cele mai bune rezultate. Numarul de vecini si algoritmul utilizat influenteaza semnificativ rezultatele finale. Numarul de vecini ales trebuie sa fie relativ mic pentru a folosi doar vecinii cei mai apropiati, iar algoritmul brute ofera un plus de performanta pe setul meu de date.

5. Evolution of Covid-19 and decision making using KNN

In acest ultim capitol, primul pas efectuat a fost adaugarea unei coloane noi in tabelul initial. O coloana care sa reprezinte starea sau decizia care trebuie luata de catre conducerea statului in functie de evolutia pandemiei. Am notat cu 0 ca fiind starea in care nu trebuie luate masuri, cu 1 starea de alerta, iar cu 2 starea de urgenta. Evolutia fata de ziua anterioara dicteaza luarea deciziilor. Daca se observa o crestere alarmanta, se vor va marca in tabel cu una dintre valorile 1 sau 2.

A fost realizat un grafic care sa evidentieze cresterile alarmante. Acestea pot fi observate in urmatoarea figura reprezentate sub forma 'crestelor' masive.

```
[80] import matplotlib.pyplot as plt

    # A greater value on the height axis means an alarmating growth of the cases on that day
    # We can give one of the following values depending on the state of the growth:
    # - fara stare de urgenta sau alerta if growth < 300
    # - stare de alerta if growth > 300 and growth < 700 for example
    # - stare de urgenta if growth > 700 for example
    # With these, the gouverment can take quick and great decisions for the state.

    plt.plot(y, x, label = "Evolution")
```

```
[<matplotlib.lines.Line2D at 0x7fa551d0e860>]
```

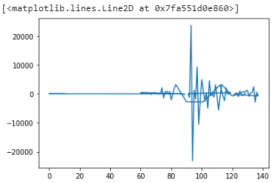

Figure 3.13: Graph of rising cases

Ca si algoritm de predictie a starii sau deciziei, a fost folosit tot KNN. Rezultatele obtinute au fost decente, insa nu perfecte. In unele cazuri cand trebuia semnalata o stare de urgenta, algoritmul a generat o stare de tipul 0, adica fara masuri, ceea ce ar fi dezastruos intr-o ipostaza reala.

```
[83]
    myData.head() #print some of the data to visualize it
```

	Country	Date	Confirmed	Death	New Confirmed	New Death	Nr. Crt	Growth	AlertState
0	France	1/2/2020	0	0	0	0	2	0	0
1	France	1/3/2020	0	0	0	0	3	0	0
2	France	1/4/2020	0	0	0	0	4	0	0
3	France	1/5/2020	0	0	0	0	5	0	0
4	France	1/6/2020	0	0	0	0	6	0	0

Figure 3.14: Tabelul cu noua coloana

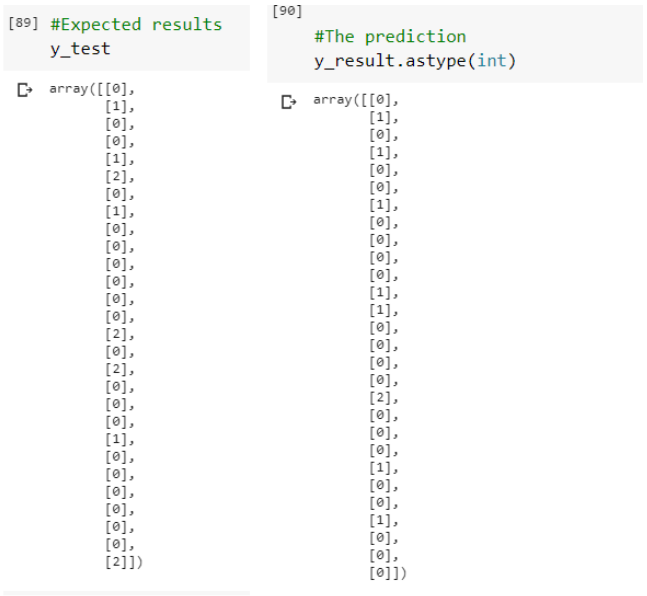

Figure 3.15: Expected vs Predicted

Chapter 4

Concluzii si posibilitati de dezvoltare

In concluzie, as dori sa spun ca ramura de Inteligenta Artificiala este deosebit de importanta in situatiile de urgenta, cum este cea actuala, din mai multe puncte de vedere.

AI poate sa faca aprecieri cu privire la viitor, poate sa ofere diagnosticuri relativ precise pentru radiografii cu raze X, care sunt de asemenea utile in lupta curenta cu pandemia.

Reflectand la munca mea, am observat cat de utili sunt algoritmii de machine learning si am descoperit ca pot produce rezultate reale si valide. Insa parerea mea este ca 'They are not there yet'. Trebuie depusa mai multa munca pe aceasta latura, pentru ca spre exemplu, sistemul de efectuare de decizii la nivel de stat nu poate fi folosit din cauza ca nu este destul de precis.

In final, as dori sa mentionez cateva posibilitati de dezvoltare ulterioara ale acestui proiect. O prima posibilitate ar fi dezvoltarea unor algoritmi de machine learning specifici pentru aceasta ramura. Adica un algoritm custom care sa lucreze bine pe planul de prezicere al deciziilor care trebuie luate in caz de pandemie. O a doua posibilitate de dezvoltare este folosirea unui set de date mai larg, care sa ofere mai multe informatii despre zilele respective precum numarul de bolnavi in stare grava, numarul de persoane izolate la domiciliu etc. Evident, avand mai multe informatii, se vor putea lua decizii mai bune. Hope this is all going to end well! Stay safe.

Bibliography

[1] Science Direct article,
https://www.sciencedirect.com/science/article/pii/S026840122030949X

[2] SkLearn bib,
https://scikit-learn.org/stable/

[3] Link to code in Google Colab,
https://colab.research.google.com/drive/1eCcM05mKTTJLkxkrJtWjXa3Bdm9yLG5K?usp=sharing

[4] Link to code in Datalore,
https://datalore.io/notebook/j6z3qwuh42F1EjTcHzQfsS/5sf7VChQiWEIoAizKQ66OK/