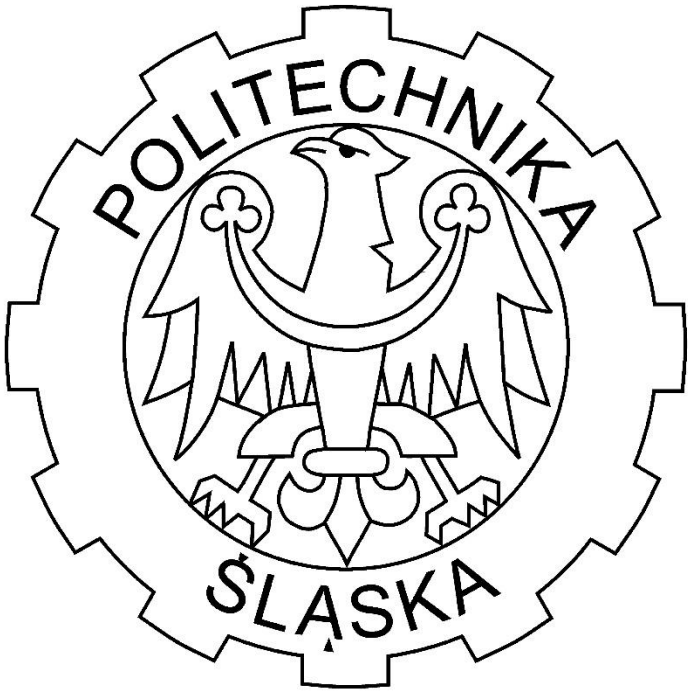
BIOLOGICALLY INSPIRED ARTIFICIAL INTELLIGENCE



Predykcja rozwoju choroby

Krzysztof Kuczyński

Jakub Sądel

INF GKiO3

Wstęp

Celem naszego projektu było stworzenie programu który będzie mógł przewidzieć rozwój chorób zakaźnych na świecie. W naszym przypadku zaczniemy od popularnego ostatnio Covid-19.

Przygotowanie danych

Nasz zbiór z danym został znaleziony na kaggle.com i składa się z 18816 rzędów(294 krajów i regionów przez kolejne 64 dni) i 38 różnych zmiennych takich jak populacja, ilość zarażonych lub odsetek palaczy.

```
#Importing data from github

import pandas as pd
path='https://raw.githubusercontent.com/sondisonda/covid19SpreadForecast/
master/enriched_covid_19_week_2.csv'

df = pd.read_csv(path)
```

Po załadowaniu danych zauważono że niektóre zmienne można usunąć np. ‘id’ bo się dubluje z domyślnym, ‘Date’ ponieważ dane są poindeksowane w odpowiedni sposób, lub ‘Province_State'

```
#Removing unnecessary columns

df.drop(['Id'], 1, inplace=True)
df.drop(['Province_State'], 1, inplace=True)
df.drop(['Date'], 1, inplace=True)
```

Po sprawdzeniu czy w tabeli nie pojawiły się jakieś nieprawidłowości nadszedł czas na podzielenie zbioru na treningowy i testowy. Trzeba pamiętać o odpowiednim podzieleniu zbioru, tak aby nie podzielić danych krajowych jednego państwa w zbiór treningowy i szkoleniowy.

```
#Spliting data in a training set and test set

lines_count = df.shape[0]
lines_per_country = 64

country_count = int(lines_count / lines_per_country)
train_country_count = int(country_count * (2/3))
test_country_count = int(country_count * (1/3))

train_rows_count = int(lines_count * (2/3))
test_rows_count = int(lines_count * (1/3))
print('Country count: ' + str(country_count))
print('Train rows count: ' + str(train_rows_count))
print('Test rows count: ' + str(test_rows_count))
print('Train country count: ' + str(train_country_count))
print('Test country count: ' + str(test_country_count))
```

```
Country count: 294
Train rows count: 12544
Test rows count: 6272
Train country count: 196
Test country count: 98
```

By uprościć odbiór wyniku wykonujemy jeszcze operacje uproszczenia danych gdzie zostawiamy 2 najważniejsze kolumny, a następnie wykonujemy normalizacje w zakresie od 0 do 1.

```
#Data simplification

simplified_df = df[['ConfirmedCases','total_pop']]

simplified_df.shape
```

```
#Data normalization

from sklearn.preprocessing import MinMaxScaler
from numpy import array


scaler = MinMaxScaler()

reshaped = array(simplified_df.ConfirmedCases.values).reshape(-1, 1)
scaler.fit(reshaped)
normalized = scaler.transform(reshaped)
simplified_df['ConfirmedCases'] = normalized


reshaped = array(simplified_df.total_pop.values).reshape(-1, 1)
scaler.fit(reshaped)
normalized = scaler.transform(reshaped)
simplified_df['total_pop'] = normalized

simplified_df.describe()
```

	ConfirmedCases	total_pop
count	18816.000000	18816.000000
mean	0.004613	0.172518
std	0.048129	0.310446
min	0.000000	0.000000
25%	0.000000	0.006149
50%	0.000000	0.025867
75%	0.000202	0.232174
max	1.000000	1.000000

Wygląd danych po tych operacjach

Dane w takiej postaci zapisujemy do odpowiednich zmiennych.

```
#Data assignment

x_train = simplified_df.head(train_rows_count)
x_test = simplified_df.tail(test_rows_count)

y_train = x_train['ConfirmedCases']
y_test = x_test['ConfirmedCases']
```

Dla każdego kraju istnieje ciąg 64 kolejnych dni, który dzielimy na podciągi z 63 krokami. Ostatni krok czasowy będzie używany do prognozowania szkolenia tj : będziemy trenować model do przewidywania liczby przypadków w 64 dniu zgodnie z poprzednimi 63 dniami.

LSTM potrzebuje danych w formacie [próbki, przedziały czasowe i cechy]. Tutaj mamy 196 próbek, 63 kroki czasowe na próbkę i 2 funkcje.

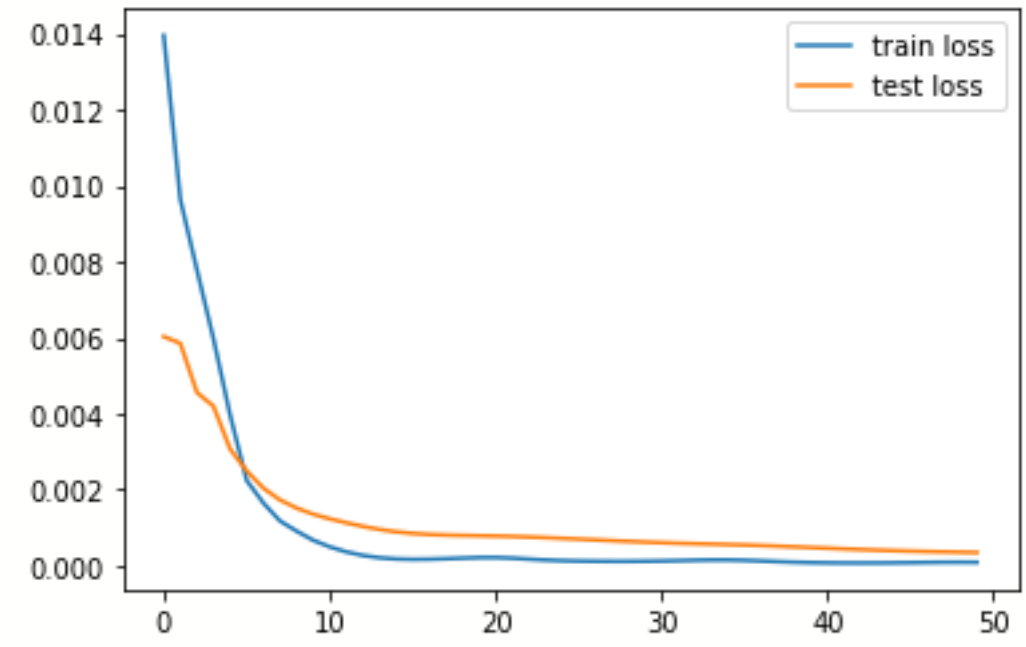
Przed stworzeniem modelu musimy przygotować dla niego odpowiednie dane. Dla testów robimy to analogicznie jak do treningu.

```
#Preparing train data

import numpy as np
from numpy import array

x_days_train = list()
y_days_train = list()
length = lines_per_country
country_count = train_country_count
for i in range(0,length*country_count,length):
    x_day_train = x_train[i:i+length-1]
    x_days_train.append(x_day_train.values)
    y_day_train = y_train[i+length-1:i+length]
    y_days_train.append(y_day_train.values)


x_train = array(x_days_train)
print(x_train.shape)
y_train = array(y_days_train)
print(y_train.shape)

x_train = x_train.reshape(train_country_count, 63, 2)
print(x_train.shape)
```

W naszym projekcie wykorzystaliśmy bibliotekę keras czyli wysoko poziomowego API działającego na tensorflow. Nasz model jest zwykłym modelem sekwencyjnym co oznacza warstwy następują po sobie, w tym wypadku są to rekurencyjna warsta Long Short-Term Memory i „domyślna" Dense. Po dodaniu odpowiednich warstw kompilujemy go, a następnie trenujemy i oceniamy.

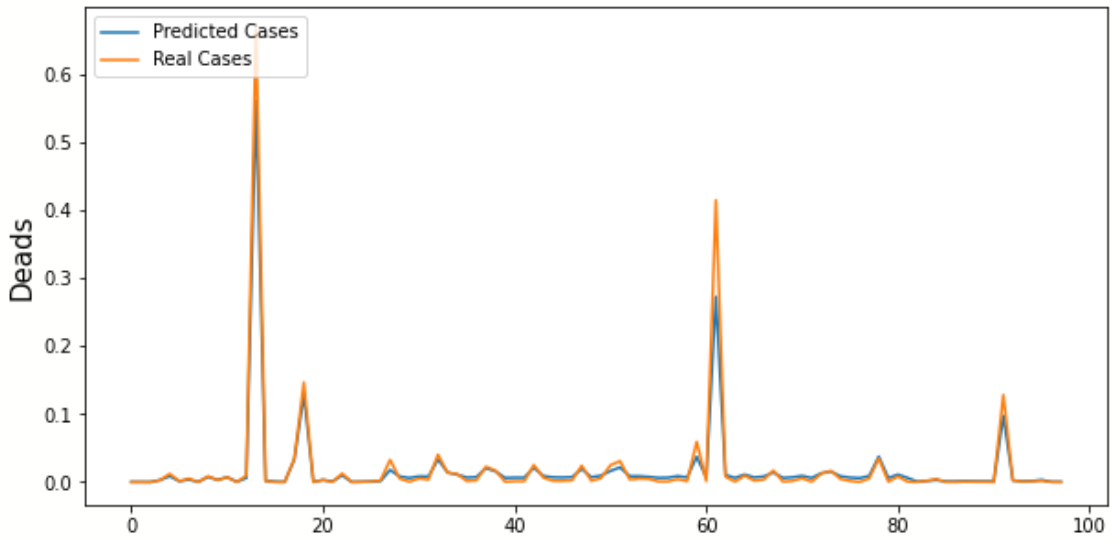
```
#Creating and training  model

from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense


n_steps = 63
n_features = 2
n_batch = 4
model = Sequential()
model.add(LSTM(16, input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

history = model.fit(x_train, y_train, epochs=50, batch_size=n_batch, v
alidation_data=(x_test, y_test),verbose=0, shuffle=False)
```

```
#Evaluating model

import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='test loss')
plt.legend()
plt.show()
```



Na samym końcu porównujemy otrzymane wyniki i zapisujemy model.

```
#Comparing results

import matplotlib.pyplot as plt

predictions = model.predict(x_test)

plt.figure(figsize = (10,5))
plt.plot(range(len(predictions)),predictions, label="Predicted Cases")
plt.plot(range(len(y_test)),y_test, label="Real Cases")
plt.ylabel('Deads',fontsize=15)
plt.legend(loc="upper left")
plt.show()
```




```
#Model save

model.save("model.h5")
```

Wnioski

Jak można zobaczyć po otrzymanym wykresie, obie linie pokrywają się w zdecydowanej większości, ale jest to głównie spowodowane uproszczeniem danych, jak i nie wielkim skomplikowaniem przykładu. Z drugiej strony wydaje się gdyby podejść do problemu z większym zapleczem i próbować go rozwiązać tak by wszystkie dane się pokrywały, to prawdopodobnie spotkałoby się srogi zawód.