

Comparison of two selected ML models for predicting deaths due to COVID-19 outbreak

1st Szymon Krasuski
Warsaw University of Technology
Warsaw

2nd Janusz Mikłuszka
Warsaw University of Technology
Czeladź

Abstract—...

Index Terms—COVID-19, Machine learning, Autoregression.

I. INTRODUCTION

Current coronavirus pandemic started in Chinese Wuhan, in December 2019. On 11 March WHO made the assessment that COVID-19 can be characterized as pandemic which became global problem. On 4 March first case in Poland appeared. From this time number of confirmed cases was steadily increasing. Month later there was 5000 confirmed cases.

II. PROBLEM DESCRIPTION

Using databases of confirmed cases, death cases and recovered cases of COVID-19 we can create model which can give us predictions of future cases. Based on these prediction decisions can be made regarding public restrictions and other regulations.

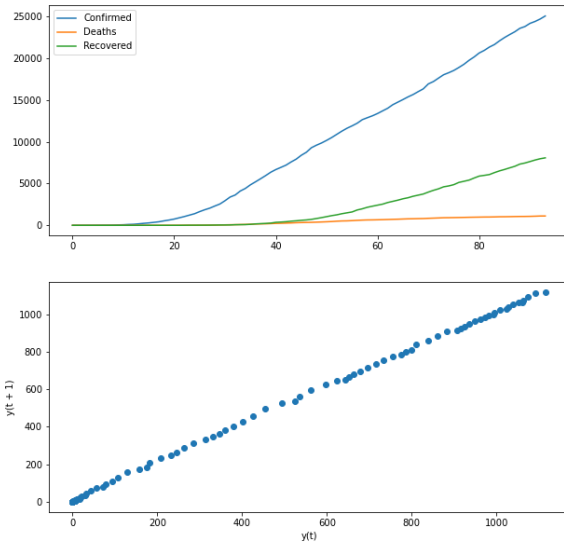


Fig. 1. a) Time series of confirmed, death and recovered cases. b) Lag plot of death cases

The time series plot indicates relationship among the series. Linear shape of lag plot suggest positive autocorrelation

between next values of death cases. This suggest that using autoregression (AR) model is good choice. Randomness of the time series can be checked by autocorrelation plot of the time series.

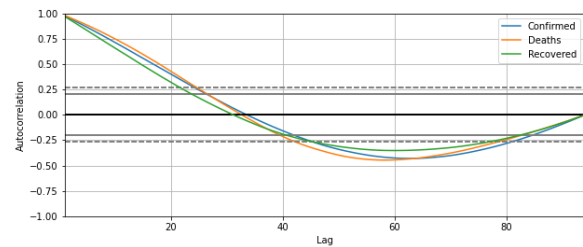


Fig. 2. Autocorrelation of the time series

For non-random time series autocorrelations are significantly non-zero. Above figure shows that lag values about 10 are worth considering. Such high autocorrelation values further proves viability of AR model.

There might be also relation of death cases with confirmed and recovered cases. Pairplot confirms our suspicion showing high correlation with both of them

To take that variables into account during our predictions VAR model can be used. To use VAR model data two assumptions are needed:

- all variables must be Stationary,
- there exist a linear relation between their current and past values

Above figure confirms relation between values. Augmented Dickey-Fuller Test is used to check stationarity. If the result of the test indicates that data is non-stationary, then data cannot be used. Fortunately a difference transform is good way of removing a systematic structure from time series. It can be done by using `pandas.DataFrame.diff().dropna()` method. It can be done second time when ADF test is still negative.

III. DOMAIN IMPLEMENTATION

We are using Python environment to create application which based on provided data and prediction horizon, makes predictions of number of deaths per day.

Project structure is organised into consistent class which implements methods for gathering data, training machine learning model and data visualization.

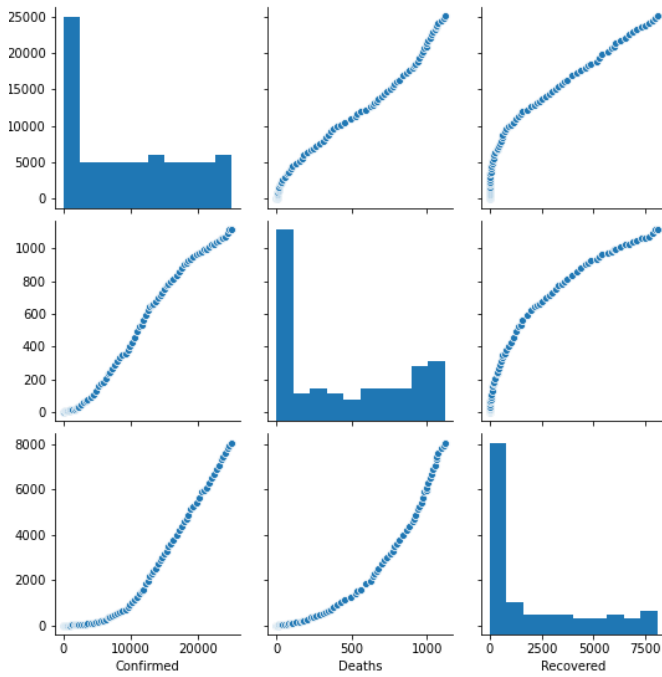


Fig. 3. Autocorrelation of the time series

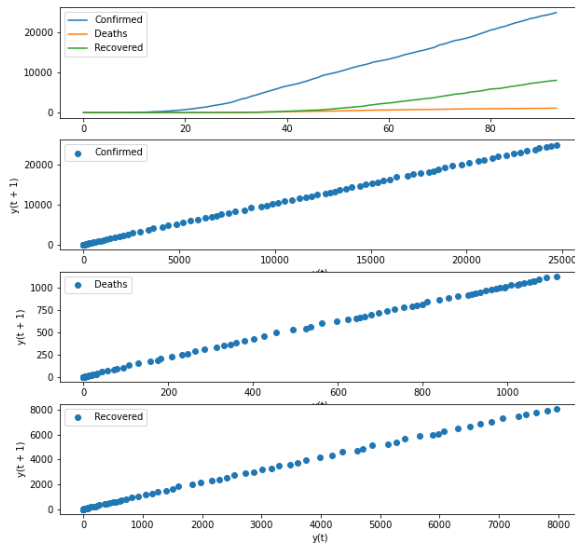


Fig. 4. Autocorrelation of the time series

Data gathering is separate module that allows to download data in the runtime from provided web server or load data from local file. Data format has to be consistent with format provided by user dtandev in repository "coronavirus" published on GitHub platform. Data is presented as summarized numbers per columns starting from 3rd March

2020. Because of that data gathering module on reading raw data has additional steps which detects which one of two supported formats are used in data. Currently project supports data presented in csv files with columns named as data presented in GitHub repositories anuszka/COVID-19-MZ_GOV_PL or dtandev/coronavirus. These two repositories were originally chosen as source of data for machine learning model. Additionally, data is parsed to return only columns containing dates and day-to-day deaths which are calculated from summarized deaths since start of pandemic. Complete data is returned in Pandas DataFrame.

Main class contains methods which implement steps easy to split for external resources e.g. GUI applications:

- *load_data* - implements data gathering model to load data from web or local file. It is possible to provide data also as complete DataFrame during object initialization.
- *set_predict_horizon* - sets dates in which next data points should be predicted. Dates should be datetime object from datetime library.
- *train_AR* - method which trains Autoregression model based on previously loaded data.
- *train_VAR* - method which trains Vector Autoregression model based on previously loaded data.
- *predict* - Because it is possible to alternatively train model with two separate methods then this method uses one of models (the one that was run as last) and predicts next data points up to predict horizon.
- *plt* - this method prints out matplotlib plot with actual data and predicted data points.
- *get_dcc_Graph* - this method returns Dash Core Components Graph with actual data and predicted data which may be used with GUI applications created in Dash

Presented project utilizes Dash library in order to implement user-friendly application.

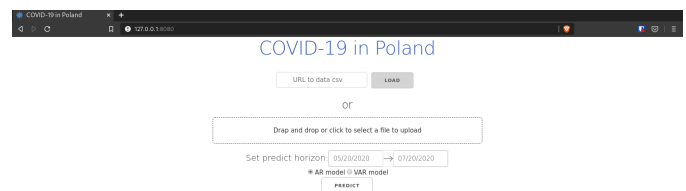


Fig. 5. User welcome screen

User at the first site layout has simple control which allows to load data from pasted URL (by default URL to data from dtandev/coronavirus repository is available to choose from text field) or local file. Afterwards, user has possibility to set data which will be used as predict horizon and finally choose one of two training models.

When user clicks 'Predict' button then below is generated interactive Dash Core Component Graph.

User has always possibility to modify parameters and repeat prediction from any point of process.

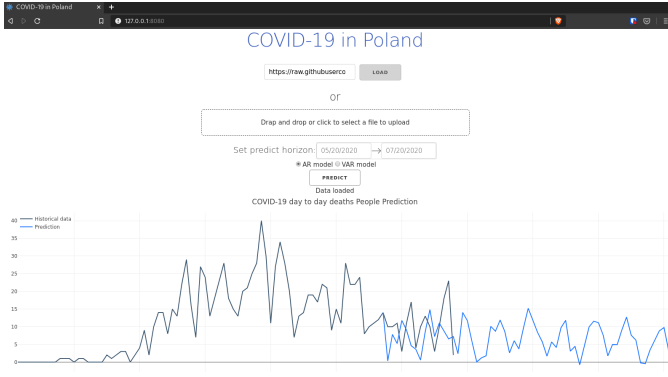


Fig. 6. Application after setting parameters and prediction process

Whole project is covered with unit tests which are used as regression tests for project to make sure that with every code edition all other components still work properly.

Tests are run automatically in every Pull Request created on GitHub platform of project (currently private repository) with CircleCI platform.

Test job was created with tox tool which contains two test jobs.

One runs unit tests and second one run flake8 command which tests code compability with PEP8 standard.

IV. MACHINE LEARNING MODELS

A. *AutoReg*

We are using *AutoReg* function from *statsmodel* library. It is primarily used to describe time-varying processes in nature. This type of model specifies the output variable based on its previous values and stochastic term. Along with moving-average (MA) model it is component of more general autoregressive-moving-average model (ARMA). Dynamic of the autoregressive model of order p $AR(p)$ is given by:

$$X_t = c + \sum_{i=1}^p (\phi_i X_{t-i}) + \epsilon_t \quad (1)$$

where p - order of the model

ϕ - parameters of the model

c - constant

ϵ_t - white noise

B. *ARMA*

It is simirial to AR model. MA part involves modeling the error term as linear combination of error terms occurring contemporaneously and at various times in the past.

Moving average model of order q $MA(q)$ is written:

$$X_t = \mu + \epsilon_t + \sum_{i=1}^q (\theta_i \epsilon_{t-i}) \quad (2)$$

$ARMA(p,q)$ notation refers to $AR(p)$ and $MA(q)$ models:

$$X_t = c + \epsilon_t + \sum_{i=1}^p (\phi_i X_{t-i}) + \sum_{i=1}^q (\theta_i \epsilon_{t-i}) \quad (3)$$

C. *VAR*

Vector autoregression is a stochastic process model used to capture the linear interdependencies among multiple time series. It generalizes univariate AR model by allowing more than one evolving variable. All variables have an equation explaining its evolution based on its own lagged values.

V. RESULTS

...

VI. SUMMARY

...

REFERENCES

- [1] <https://www.sciencedirect.com/science/article/pii/S0047259X85900272>
- [2] https://link.springer.com/chapter/10.1007/978-1-4612-1694-0_12
- [3] <https://journals.ametsoc.org/doi/full/10.1175/JHM517.1>
- [4] <https://github.com/dtandev/coronavirus>
- [5] https://github.com/anuszka/COVID-19-MZ_GOV_PL
- [6] <https://towardsdatascience.com/vector-autoregressions-vector-error-correction-multivariate-model-a69daf6ab618>
- [7] <https://towardsdatascience.com/granger-causality-and-vector-autoregressive-model-for-time-series-forecasting-3226a64889a6>