RESEARCH

Introduction to the profile areas of data sciences: project 6

Rohan Chitte] Natalja Amiridze] Michael Onyebuchi Ohaya

Abstract

Goal of the project: The goal of the project is using Decomposition model and auto-regressive model to analyze and forecast time-series data and thereafter, applying them to longitudinal data to predict Covid-19 cases

Main results of the project: Two methods were described, compared and visualized side by side using the Prophet API and auto-regression model. Relevant information were extracted and visualized using appropriate methods.

Personal key learnings: We learned how to perform time-series using the auto-regression model and also the use of the prophet API decomposition model to visualise, analyze a given time-series data and perform forecast using a data.

Estimated working hours: 12

Project evaluation: 1

Number of words: 2089

1 Scientific Background

Being able to predict certain event of the future has become increasingly important in view of recent activities happening in the world. A time series is defined as a set of observations arranged in chronological order. The time interval between each data point remains constant, such that a sequence of discrete-time data is generated. One aim is to extract meaningful statistics and interesting characteristics from the time series data structure. Thus, investigating the stationary and seasonality is part of the standard protocol when dealing with time series data. It is said to be stationary if it's mean and variance do not change over time, while seasonality can be identified depending on the data, which refers to periodic fluctuations of the values. The second major intention is to perform forecasting. Here, a model is applied to the data to predict future values based on the past observations.

2 Goal

The goal is to understand time series data and is to forecast the COVID-19 outbreak using one Decomposition model and one auto-regressive model (also understand and implement its components) to analyze and forecast time-series data and apply it to longitudinal data from the current Covid-19 case numbers. Furthermore, try different data driven approaches and compare its result of forecasting and predictability to the other models.

3 Data and Preprocessing

For this project we have used RKI data. RKI provides time series data provided by Robert Koch-Institute. Time series data is data that is collected at different points in time. RKI's time series data for COVID is available in different formats such as state wise update on no. of cases and state wise update on no. of deaths in Germany. We have selected to work on "cases-rki-by-state.csv" dataset[1]. The dataset shows daily no. of cases in each state as well as the sum of all the states on each day. The most important feature about this data is that its updated in real time using json. The data reads cases from 03-02-2020 till today. Column "time_iso8601" contains the record of date and time of each day, "sum_cases" records the sum of each state and there are individual state records as well. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. In order to analyze and forecast RKI Data, we used one Decomposition model and one auto-regressive model on it. Before we implemented the models, the data was read by indexing column time_iso8601 using panda's csv reader. Further, the individual state records were removed from the dataset as were only interested in total no. of cases in Germany. To make futher predictions We first splitted the data into train data all the entries except for 14 last days and test data for last 14 days. Then, the train data is fitted to the model using .fit(). Next, forecast method is used to predict the forecast on the data where the result is an array containing the forecast value. These values can be plotted using matplotlib. Additionally, r2 score, mean square error and its root value were calculated using r2_score, mean_squared_error and sqrt functions.

3.1 Decomposition Model

For the decomposition model, we imported seasonal_decompose class from statsmodels.tsa.seasonal, which performs naive seasonal decomposition using moving averages. It takes the time series data as a parameter along with model name and the period. The period is nothing but the period/frequency of the series and the model can be either additive or multiplicative. For our data the multiplicative model is more useful as its seasonal variation increases over time. This seasonal_decompose return an object with attributes Seasional, Trend, Observed and Residual which are then plotted using matplotlib. Similarly, Facebook's Prophet model is also used which extends decomposition model to features such as change point detection, saturating forecasts etc. However, prophet demands columns to be of specific names, namely ds for the time record and y for the sum values. Prophet class imported from fbprophet offers fit function that fits the data to the model, a predict function that makes the prediction on the data and its own plot function that plots the forecast. But first, an object m is created of class Prophet which can then be used to implement its other functions. Automatic changepoint detection is done by passing parameter changepoint_prior_scale to Prophet.

3.2 ARIMA Model

For the second method, we used ARIMA imported from statsmodels.tsa.arima_model. For the classification and forecasting on the timeseries problems there are various maching learning approaches. One among them is by using the autoregressive models. As it focuses on various linear relationships, they perform well on a wide range

of problems. They also perform well if the data is suitably prepared. There are nearly eleven types of autoregressive methods, all lead to forecasting a different time series problem. From these methods we chose the Autoregressive Integrated Moving Average method (ARIMA). The reason which makes this model more significant is that it gives very low residual sum of squares (RSS) which is helpful for building the model and fit it. The lower the RSS, the lesser will be the amount of variance. However, ARIMA models the next step in the sequence as a linear function of the differenced observations and residual errors at prior time steps. It is the combination of both Autoregression (AR) and Moving Average (MA) and helps in differencing pre-processing step of the sequence stationarity called integration. The notation for the model involves specifying the order for the AR(p), I(d) and MA(q) models as parameters to an ARIMA function, e.g. ARIMA(p,d,q). An ARIMA model can also be used to develop AR, MA, and ARMA models.

4 Results

4.1 Prophet

The Prophet Library is composed of the three components trend, seasonality and holiday effects where each of its components is added together with time as its regressor:

y(t) = trend(t) + seasonality(t) + holidays(t) + error(t) where:

Trend models non-periodic changes. This component is the most important one for our analysis since our data has no seasonality like it is the case for i.e. the amount of rain in a certain country over many years.

Seasonality models periodic changes in our data like it is the case for i.e. the amount of rain in a certain country over many years.

Holiday Effects allow the model to include short time intervals with abnormal trend. When modeling Corona in Berlin with no restrictions, this could be a public event like the Karneval der Kulturen where many people from different households interact closely with each other, which would probably lead into a spike in the number of confirmed cases. Since Germany restricted most public events early in the pandemic this component is also not of greater interest for our analysis. The seasonality of non-stationary data can be also visualised using Prophet library (see Figure 1) Prophet forecasts by fitting a curve to the input data that is then prolonged for future values with the three mentioned components. Although, Prophet aims to produce a strong forecast without much hyper-parameter tuning. For our analysis the prediction fit the actual trend of the test data well. By adjusting the change point parameters like *changepoint_prior_scale* and *changepoints* manually it was possible to improve the prediction curve trend of the model (see Figure 2).

4.2 Change Point Detection

The dataset contains 2 main breakpoints (points in the data, where the growth changes). So adding the changepoints in theses regions and setting the *changepoint_prior_prediction scale* lower makes the model prediction curve to be closer to the curve of the confirmed cases see Figures (3) .

This adjustment improves also the statistics values such as MSE and RMSE values (see Table 1). The RMSE is calculated as follows:

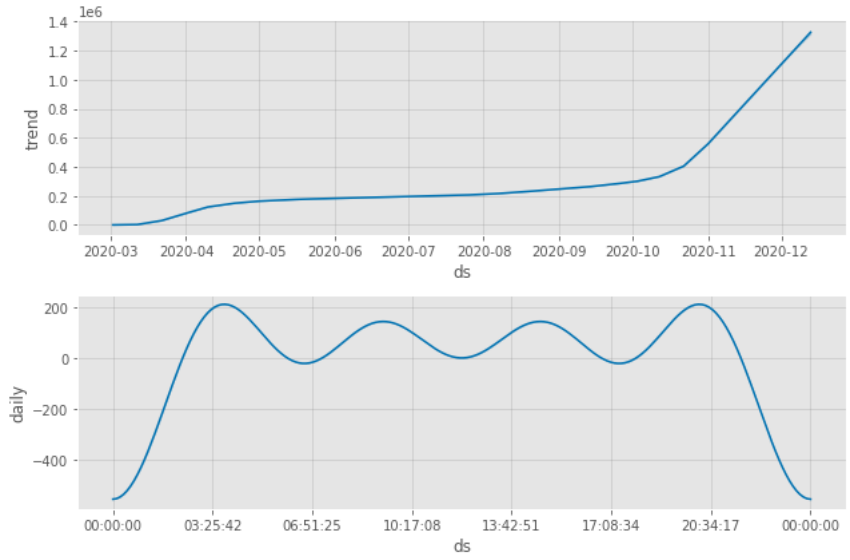

Figure 1 Data Analysis using Prophet library: Daily Seasoning and Trend

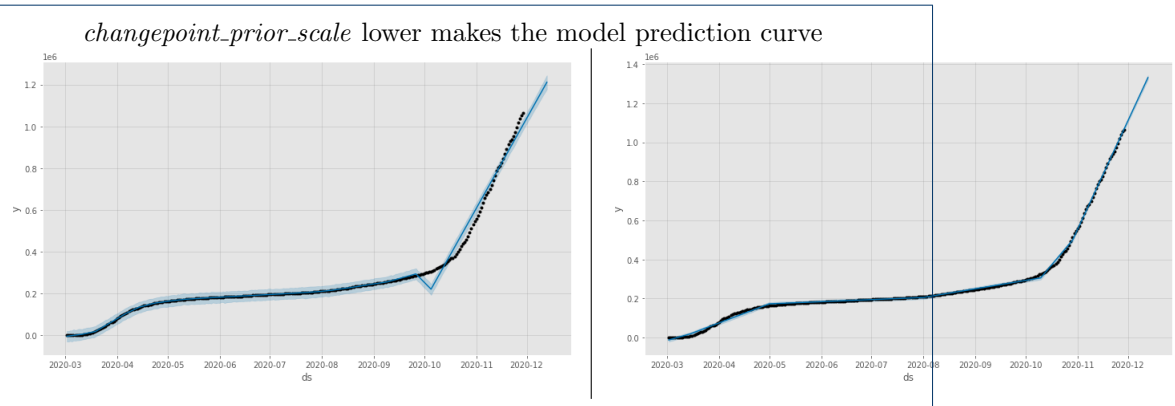

Figure 2 Change Point Detection and Adjusting. **Left:** changepoint_prior_scale=0.9, changepoints: default; **Right:** changepoint_prior_scale: 0.04, changepoints: '2020-03-15','2020-05-01', '2020-08-01','2020-10-10','2020-10-29'

4.3 Autoregressive model: ARIMA

We began by importing the dataset, which contain 289 observations for the confirmed number of cases. Then, the important part is making the time series stationary. In other words the statistical properties (mean, variance) should remain constant over time. Furthermore, we applied a smoothing function to make the time series stationarity (see Figure 4) . Afterwards, we plotted Autocorrelation (ACF) and Partial Autocorrelation (PACF) to understand the parameters (p,d,q) of the ARIMA model (see Figure 5. They basically indicate the correlation between two time instances as well as the degree of association. Finally, we trained a certain duration of the data sets, built the ARIMA model and estimated the Root Mean Squared Error (RMSE) and forecasted the predictions for the test set. Figure 6 represent the ARIMA model plotted for the RKI dataset.

	without changepoint adjustment	with changepoint adjustment
R^2	0.98	R^2: 0.98,
MSE	8.29e-05	7.8e-05
RMSE	0.009	0.0088

Table 1 Comparison of statistics values for model prediction adustment with tuning of changepoint parameters

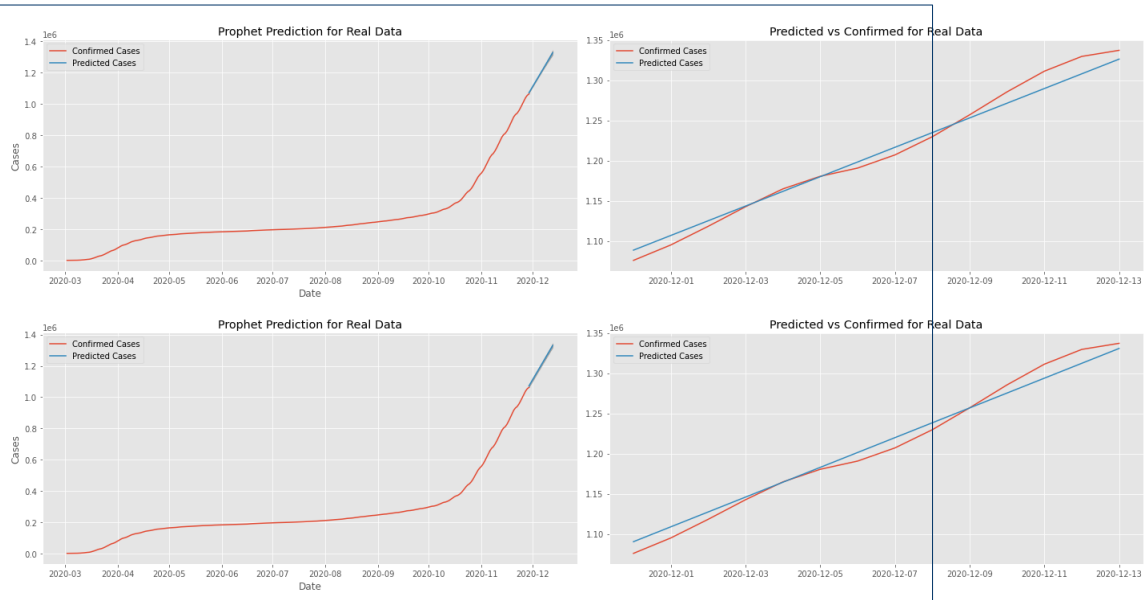

Figure 3 Improvement of prediction curve by adjustment of the changepoint parameters. Model prediction curves (left: all given days; right: 14 predicted days) **Top**: without changepoint adjustment. **Bottom**: with changepoint adjustment. In orange the confirmed cases and in blue the prediction

.

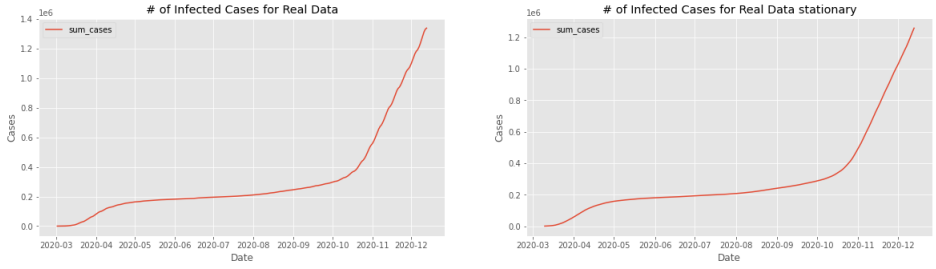

Figure 4 Cases(orange) on RKI actual data for a ARIMA model (y-axis=confirmed cases, x-axis = time period). **Left**: no smooth; **Right**: with smooth

4.4 LSTM neural network

The third method we used was Long Short-Term Memory Networks (LSTM). They are a prominent deep learning method for time series prediction. LSTM are part of Recurrent Neural Networks (RNN). In contrast to common feed forward networks where each layer is only connected to the subsequent layer, RNN has layers connected to itself or even previous layers. This is more closely modeled on the neural connections exhibited by the neocortex and allows the network a greater prediction accuracy for time coded data. The LSTM network is employed by using the python library *keras*. For the prediction a LSTM of the length of the training data with

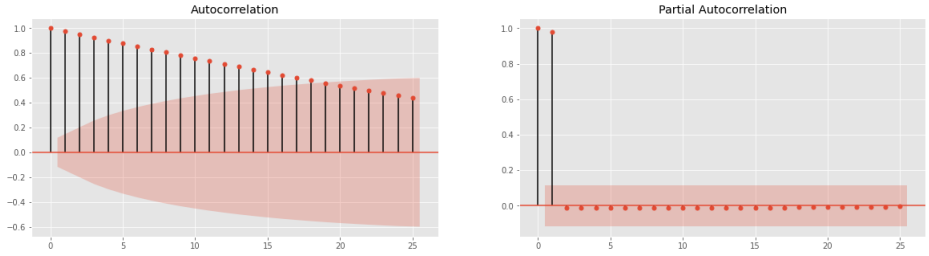

Figure 5 ACF(left) and PACF(right) plots on RKI actual data for a ARIMA model (y-axis=confirmed cases, x-axis = time period)

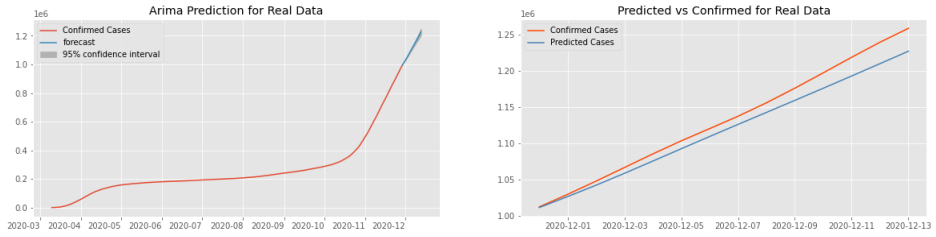

Figure 6 Cases(orange) vs forecast(blue) on RKI actual data for a ARIMA model (y-axis=confirmed cases, x-axis = time period). The gray area represents 95% confidence interval for the 14 days forecast

a simple 1-dense-hidden-layer is created. The model itself is optimized with adam and lossed by the mean_squared_error. The data needed to be preprocessed. The first step of preprocessing is called shifting. In shifting the time series data is shifted by a constant value, dividing the data in a training value and expected value. Subsequently, because LSTM needs to evaluate local differences in the data, each time step is substracted by the shifted value. As in all neural networks a common scalar is used normalizing the data between -1 and 1 with a mean of 0. We plotted the predicted and actual time series data of the last 14 days. For the test set the RMSE is calculated (Figure 7).

4.5 Comparison of three different methods of timeseries prediction

Prophet: In Figure it can be seen that the fit is optimal. The fluctuating trend of the data is captured, the model prediction curve without change point is not absolutely close to the curve of the given data but it doesn't change the prediction results for 14 days (as the MSE and RMSE values and also the prediction curves are very similar with slight improvement by changepoint adjustment.

ARIMA: From the above obtained plots and the R^2 MSE and RMSE values, we can say that the fit is bad for the dataset when compared with other methods used here.

LSTM: LSTM is the "winner" and produced the best RMSE value.

5 Discussion

Often there's a need of using real time data in order to make predictions about future and this is where models like ARIMA, Prophet and LSTM comes in. During

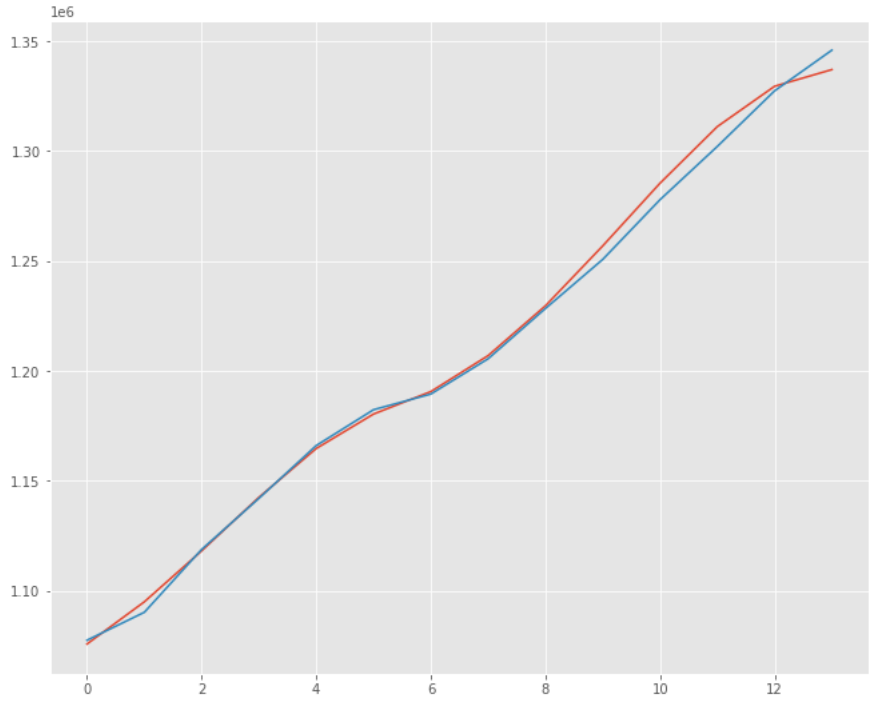

Figure 7 Prediction (orange) vs expected (blue) on RKI actual data for a LSTM model. (y-axis= infected people, x-axis = days)

Statistic figure	ARIMA	Prophet	LSTM
R^2	0.95	0.98	0.99
MSE	2e-03	7.8e-05	2.6e-06
RMSE	9e-03	8.8e-03	1.6e-03

Table 2 Comparison of R^2, MSE, RMSE values for three methods ARIMA, Prophet and LSTM

our study, The results of the forecasting using the distinct time series approaches ARIMA, Prophet and LSTM were evaluated by comparing the forecasting plots with the actual data as reference and analyzing the root mean square values. ARIMA performs well on forecasting stationary data of short periods. Prophet is more advanced than ARIMA and offers also the possibility to identify trends and seasonality. Since seasonality is already proven to exist for other viruses (e.g. influenza), it will be an interesting experiment to analyze the existence of seasonality for COVID-19 on long-term time series data. Such a study would distinguish Prophet's strengths. However, the data we used was non-stationary. That could be an explanation why ARIMA underperformed and showed the biggest RMSE values compared to the other methods. LSTM was the clear winner as it is optimised for dealing with non-stationary data and our results confirmed: LSTM produced the lowest RMSE values for the RKI data. The results further demonstrate that, given data of confirmed COVID-19 cases, LSTM can learn and scale to more or less accurately estimate the amount of the people that will become infected in the future. Naturally, the prediction is only a tendency and needs to be scrutinized very critically. Nevertheless, it is possible to predict a course of the outbreak. And the more data becomes avail-

able, the better the results of the data-driven forecasting methods will be. Therefore ,Such a project fits perfect for a data science related work.

6 Appendix

Name	Work Description
Michael	Abstract, Scientific Background
Rohan	Goal, Data, Some part of results and Some part of code
Natalja	some part of Data Preprocessing, Results, Discussion, Code

Table 3 Task responsibilities

References

1. Jan Gehrcke: COVID-19 Germany (2020). https://raw.githubusercontent.com/jgehrcke/covid-19-germany-gae/master/cases-rki-by-state.csv Accessed Accessed 14 Dec 2020