Course-ID:	MA-INF 4306
Course:	Lab Development and Application of Data Mining and Learning Systems: Data Science and Big Data
Term:	Winter 2020/2021
Supervisor(s):	Michael Mock, Till Schulz, Florian Seiffarth, Daniel Trabold, Dorina Weichert, Pascal Welke

Clustering of Covid-19 Time series

Lisa Pucknat   Alexander Zorn

March 13, 2021

Abstract

The corona virus dictates how many people have to live their everyday life. Finding similar case developments between countries and forecasting Covid-19 cases may help in overcoming the pandemic by acting foresighted. For this purpose, we developed a framework that is capable of clustering time-series data of Covid-19 cases and attempted to predict future cases using the results.

1 Introduction

Many diseases follow patterns which can be extracted and used for preventing a severe disease course. The sheer impact of the ongoing Covid-19 pandemic gives rise to compare case developments of different countries and also to learn what containment actions seem to be promising and which are not. In our work we try to get a first overview of case developments without focusing on more complex and hard to model data such as the political situation or the health system. For that, we clustered trends of cases for each country. During the clustering of time-series data several challenges occurred such as noise and high dimensionality. In the next step, we used timespan clustering results for predicting the next day. For this, neural networks were implemented and tested against shallow forecast methods. Lastly an effort was made to visualize the results.

1.1 Related Work

Ali et al., 2019 defined the time-series data structure and summarized clustering, classification and forecasting methods. They discuss problems and present popular approaches used in different literature. This paper was used to get a general overview of time-series. Furthermore, Zarikas et al., 2020 used daily Covid-19 case reports from several countries over 3 months and concentrated on hierarchical clustering without an additional forecasting purpose.

1.2 Input format requirements

Our framework is able to analyse Covid-19 data provided in the following format. We assume the data to be in tabular form (Pandas DataFrame or in CSV format). Each row

corresponds to a Covid-19 case (and death) report of a country for one specific day. Detailed column set and datatype requirements can be found in the *Readme.md* file of the Lab GitHub project[1]. The Covid-19 dataset used in this lab is from the *European Centre for Disease Prevention and Control* 2021 (ECDC). The ECDC's Epidemic Intelligence team collected cases on a daily basis until $14/12/2020$, then switched over to weekly case reports since the beginning of the pandemic, where over 500 relevant sources are screened and processed. A detailed analyzation of that dataset can be found in subsection 4.1.

2 Foundations

2.1 Time series

Time-series (**TS**) data gets defined as an "ordered collection of observations or sequence of data points made through time at often uniform time intervals" (Ali et al., 2019), which adapts to our application as convoluted and scaled case and death numbers per fixed interval. We formally denote a time series as $y_1, \ldots, y_T$.

2.2 1D Convolution

A (discrete) 1D convolution of a vector (or time series) v of dimension d with a kernel k is defined by: $(v * k)[j] = \sum_{i=-d}^{d} v[j-i]k[i]$. This can be visually explained by sliding the kernel over v and point wise multiplying the kernel with the current section of v. We mainly use a kernel of variable length l and value $(1/l, ..., 1/l)$. The result is the same as an l day moving average over the time series.

2.3 Metric

When dealing with time-series data simple day by day comparisons of different series might not be appropriate because of temporal shifts between the data. This gives rise for using different kinds of dissimilarity measurements. We use the euclidean distance (**ED**), where the distance between two time-series is calculated by matching corresponding days. In addition, we use Dynamic Time Warping (**DTW**), where the general idea is to align the TS points so that similar curvatures are mapped together when calculating the distance. This has the advantage of overcoming shifts between the data. This is achieved with a Dynamic Programming algorithm. A local cost matrix is computed, stating the distances between each combination of points from different time-series. An alignment path with minimal cost is then computed, which has the condition that the first and last points are mapped to each other. Points cannot be mapped backwards, therefore only forward steps are performed. This is done in quadratic asymptotic runtime, which makes it a tedious task for large time-series. For a detailed description we refer to Müller, 2007. We therefore only use DTW for complete time-series clustering, where shifts can occur. Further we do not use DTW for forecasting, since our forecasting approaches optimally need the same development especially in the last section of the time-series.

2.4 Clustering

KMeans This method has been chosen because of its simplicity and speed. Here, the dataset is divided into K groups with a cluster center for each, where a center represents the mean of the tentatively assigned data. After convergence each sample has the smallest

---

[1] Pucknat and Zorn, 2021.

distance to its assigned cluster center. We used the SKlearn implementation of KMeans. *Sklearn, K-Means definition* 2021 provides a detailed description of that method.

KMedoids Similar to KMeans, the KMedoids method partitions the dataset into groups, but instead of calculating a cluster center, a datapoint which represents the cluster best is used. Here we used an implementation from Sklearn extra. A detailed description of the algorithm can be found in *Sklearn extra, K-Medoids definition* 2021

DBSCAN In contrast to KMeans, DBSCAN does not necessarily allocate a cluster to each data point. It is rather looking for dense areas where each data point is surrounded by a minimum number of others within a maximum range. Every data point not belonging to a dense area is considered to be noise. Beside the downside of not clustering each data point, DBSCAN is able to detect non convex clusters in contrast to KMeans. A description of the implementation we used can be found in *Sklearn, DBSCAN definition* 2021.

2.5 Forecast

For predicting future cases we follow two approaches, with and without preceding clustering. For both, simple benchmark methods such as naïve- and seasonal naïve forecast are implemented. A more sophisticated approach is represented by a linear regression model and a Long short-term memory (LSTM) network. Our focus is on predicting the next day $\hat{y}_{T+1|T}$ given it's previous days.

(Seasonal) Naïve Forecast For the naïve forecast method, the next day simply corresponds to the last observed day: $\hat{y}_{T+1|T} = y_T$. A similar approach is the seasonal naïve approach, which assumes a periodicity in the case development. The period length then corresponds to the seasonality S. The next day is therefore given by $\hat{y}_{T+1|T} = y_{T+1-S}$. When utilizing clustering results, the same methodology holds for both, but instead of utilizing the time series, we use the predicted and assigned cluster center time-series $\mathbf{c}$. Then, for the naïve approach $\hat{y}_{T+1|T} = \mathbf{c}_T$ and for the seasonal naïve approach $\hat{y}_{T+1|T} = \mathbf{c}_{T+1-S}$ holds.

Linear Regression The Linear Regression model, forecasting a time series y with T data points, is a $T+1$ dimensional real valued vector $\beta_0 \dots \beta_T$. The output $\hat{y}$ is a linear transformation $\hat{y} = \beta_0 + \sum_{k=0}^{T} y_k \cdot \beta_k$. We used the implementation from Sklearn (see *Sklearn, Linear Regression* 2021).

LSTM A LSTM network is a type of recurrent neural network (RNN), where each LSTM cell produces two outputs and passes those to the chronologically succeeding cell in time. The main difference of an LSTM network to a simple RNN is the number and purpose of the generated output. Additional to a more sophisticated output $\mathbf{h}_{(t-1)}$ of a previous LSTM unit, a long-term state $\mathbf{c}_{(t-1)}$ is introduced, acting as memory. The inner workings of a cell include three gates (forget, input and output) protecting the memory contents from perturbation by irrelevant inputs and likewise protecting the current input from irrelevant memory. For a detailed description we refer to the original paper by Hochreiter and Schmidhuber, 1997.

MAPE We use the Mean Absolute Percentage Error (MAPE) measurement for evaluating forecasts. For each forecast the percentage distance to the correct label and an average over those values over all samples is computed. We use this measure, because the case reports values from different countries are generally very far apart (case reports range from zero over a few thousand to hundred thousand cases per day), so the standard Mean Squared Error would not be a suited measurement. Furthermore, Hyndman and Koehler, 2006 strongly recommend this measure for use-cases like the underlying.

3   Implementation/Workflow

We implemented a Python framework on GitHub (see Pucknat and Zorn, 2021) consisting of three main parts, where each can be executed independently if the requirements are met, i.e. examples are already generated. We paid extra attention that the framework works with different specifications, giving also the opportunity to insert additional preprocessing steps and methods.

3.1   Set Configuration and start Workflow

In the two configuration files (mainly *config.ini*) all necessary adjustments can be made to configure the workflow as desired. There are many possible adjustable variables in the main configuration file where multiple different assignments can be set (e.g number of clusters). Most of the variables can be set to a list of multiple assignments. The started process will then go through any feasible configuration combination of those assignments. That means that many different configuration sets can be tested together. That made it possible to generate the data for e.g. 4.3 in one run. The overview files created in each major workflow step also use the current configuration information set to describe each of the saved files. The motivation behind implementing many of the mechanisms behind those configurations can be found in the Data overview and cleaning subsection. The Appendix A gives a detailed overview over all variables and their possible assignments.

3.2   Data generation and representation

The main purpose of this section is to describe the preparation of the raw data for the ongoing steps. In detail, the data is read in from a saved Pandas DataFrame file or a specified website. Next, we extract the time-series for all countries. On each example a cleaning routine is applied (see 4.1). If specified, the countries are divided into a training- and test set to guarantee a crossover free validation of the prediction. We can switch between complete- and snippet time-series handling, where in the last case each time-series gets sampled into multiple smaller and coherent time-series of fixed length. The maximum number of samples and length can be set in the configuration file. All time-series are standardized by dividing each data point by its maximum value in the time-series, yielding a better comparison between different examples (see 4.2). Furthermore, the data generation step is able to perform a 1-D convolution on each time-series, in detail a moving average with configurable number of days. If there are too few snippets for training a simple data augmentation technique is implemented by adding a copy of each snippet to the snippet set where each data point gets randomly disturbed by a Gaussian with a configurable variance (in percentage of the data point value).

Data Representation Class For a consistent data processing during all stages of the framework, a Snippet- and Example class are implemented. The Example class holds train- and test data, which are in a Snippet format. Each Snippet consists of a time-series $y_1, \ldots, y_T$, a corresponding label y_{T+1} and a placeholder for a forecast value.

3.3 Model Training

During this step a cluster model is fitted to the training data. The desired number of clusters and clustering method can be set in the configuration files, where only valid combinations will be executed. We work with several external libraries (i.e. sklearn[2], sklearn-extra[3] and tslearn[4]), so that the data first has to be pre-proccessed to a suitable format before being fitted to the model. We are able to work with cluster methods KMedoids (ED), KMeans (ED and DTW) and DBSCAN (ED). To formally evaluate different cluster settings the following evaluation methods are included: Silhouette Coefficient[5], Calinski and Harabasz score[6] and Davies-Bouldin index[7] as well as the variance over cluster sizes. Finally, the fitted model is saved and an overview file describing the specifications and scoring values is created or updated for each model.

3.4 Prediction

We experimented with different approaches which are amongst other things two naïve forecast methods and neural networks. For both, tests with and without preceding clustering of time-series snippets are constructed.

LSTM Model and Setup A more complex model is represented by an LSTM network. We stack 3 LSTM cells vertically together for one time step, which leads to a greater model complexity. Finding different trends in the data seems to benefit from the greater complexity. The hidden- and therefore the final output size of an LSTM unit is 64. A simple linear transformation is applied to get one output value. Several experiments were conducted, to find a promising training setup. The model is trained with an adaptive learning rate optimization algorithm (Kingma and Ba, 2014) with a learning rate of $2e-4$, beta values of 0.9 and 0.999, and an stability term of $1e-8$. The model with the smallest loss on a validation dataset, which was created with a $80/20$ split on the train data, is used for prediction on the test data.

Forecast with Clustering To connect the first and the second part of the project, we used timespan cluster results as a basis for our forecasting methods. We chose KMeans in combination with ED, since several tests revealed that KMeans and KMedoids performed similar with regard to the evaluation methods defined in 3.3. DBSCAN is disregarded because new data can and has been classified as noise, making this method unsuitable for our application. As examples we use snippets with a 30 day length and no, 3, and 7 day smoothing. To fit or respectively train the models, a $80/20$ split on a country level is performed, keeping the examples overlap free. A model for each cluster was trained or respectively fitted and used to forecast the test snippets which are allocated to that cluster.

---

[2] *machine learning in Python - scikit-learn 0.16.1 documentation* 2021.   [3] *scikit-learn-extra documentation* 2021.
[4] *tslearn 0.5.0.5 documentation* n.d.   [5] *Clustering, Silhouette Score* 2021.   [6] *Calinski-Harabasz Index* 2021.
[7] *Davies-Bouldin Index* 2021.

Forecast without Clustering   As a second approach, we encapsulated this stage from the first part of the project and used (Seasonal) Naïve Forecast, Linear Regression and LSTM (trained on an $80/20$ split) on the same snippet set combinations as above.

Forecast evaluation   Since we want to compare forecasts with different configuration settings, we compare forecasts of true cases. That means that we are able to invert each of the computations applied to the original label of a snippet (e.g. standardizing or convolution). The inverted forecasts of all test data is evaluated using MAPE (see 2.5). After the forecast evaluation is computed, the updated data is saved and an overview file with the used configurations and the corresponding forecast evaluations.

3.5   Visualization

We implemented several Jupyter Notebooks to visualize the resulting files and results from data generation, model training and prediction. They were used to create all images for the Lab progress meetings and this report. The capabilities of those notebooks include e.g. a geographic visualization of cluster using Plotly or analysing the ECDC data file.

4   Results

4.1   Data overview and cleaning

The following overview over the dataset we worked on was generated using the Jupyter Notebook *Analysing_raw_data_ecdc_df* and it can be used to analyse the dataset towards further needs. The dataset structure of the input data we require is described in 1.2. It follows a detailed description over the ECDC Dataset and its structure we worked on to create the following results.

General dataset overview   The dataset contains $61,353$ entries on a daily bases of 212 different countries. In total there are $70,331,715$ reported Covid-19 cases. The reports range from the $12/31/2019$ to $12/12/2020$ so in total there are 347 days covered by at least one case report. An overview over how many countries reported cases on how many days is given in the histogram in figure 1.

Figure 1: How many countries reported on case numbers for a certain amount of days
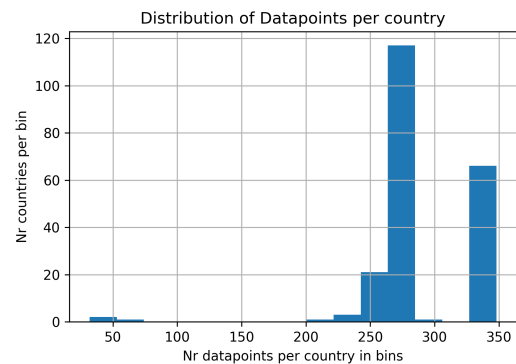
Case data distribution   There are no NAN case reports, but unfortunately there are 17 negative values. Since the number of negative reports is negligible we replaced them with zeros in the Data generation step. Regarding zeros, there are $19,205$ zero case reports which corresponds to a 31.3% share of all entries. Motivated by that result we also examined the distribution of the reports more closely. Figure 2 shows the cumulative distribution function of non-zero case reports. Consequently, there are a lot of small values (about 50% are smaller equal 100). This makes it more difficult to predict, because Covid-19

time series of a country with less than 100 cases are very erratic. Caused by that realization we evaluated the forecast especially on time series $y_1 \ldots y_T$ with labels $y_{T+1} > 100$ or $y_{T+1} > 1,000$.
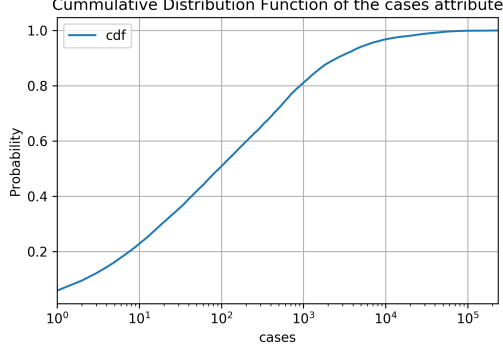

Figure 2: How many countries reported on case numbers for a certain amount of days

To conclude the Data cleaning we examined if there are countries which have a non coherent case report, e.g. there is at least one day with no report between two case report days of one country. Unfortunately, there are 97 countries affected by that and in total there are 469 values missing in between time series. Since many countries (e.g. Germany) have a strong seasonality (e.g. in Germany on a weekly bases) this might be destroyed if we ignored those entries. As a result, we interpreted a missing value in between two reports as a report of zero cases for that day and inserted that entry. Fortunately, there are at least no countries in the dataset which reported more than one value for a single day.

4.2 Cluster Results

Finding cluster methods and suitable parameterizations for those was a challenging task. Initial experiments yielded a strong imbalance in the number of examples assigned to each cluster. Abstracting from the total number of cases and switching to standardize values, resulting in a trend-only analysis, decreased the variance over the example distribution on the clusters drastically. We used DBSCAN for a rough categorization on what number of clusters seem reasonable. With an epsilon value of 2.5 and a minimum number of samples for a point to be considered a core point of 3, 7 cluster were produced, with 60 examples classified as noise. We therefore decided on experimenting further with $5, 7, 10,$ 15 and 20 clusters and concentrating only on KMeans and KMedoids because we did not

Model	No. cluster	Metric	Conv.	Silhouette	Calinski	Davies	Variance
KMeans	5	DTW	7	**0.303**	19.196	3.777	223.04
KMeans	7	DTW	7	0.247	15.621	3.972	119.35
KMeans	5	ED	7	0.244	**55.190**	**1.479**	341.84
KMeans	5	ED	3	0.216	*49.638*	1.582	393.04
KMeans	7	ED	7	0.206	47.552	*1.531*	115.061
KMedoids	5	ED	3	0.118	36.534	2.177	**21.44**
KMeans	15	DTW	3	0.059	7.291	4.632	*26.116*

Table 1: Average cluster results for complete time-series clustering after 5 runs. The two best models in combination with its parameterization are described. Best score is in **bold** and second best in *italic*

find an epsilon value, that classified all country time-series data into reasonable cluster allocations.

In figure 3 an exemplary cluster alignment is depicted, where each subplot represents one cluster. The clusters are computed using KMeans with a 7 day convolution and ED, because in case of using DTW, it is not easy to see if a cluster assignment is plausible. For a general scoring, we use the evaluation metrics stated in 3.3. As one can see from the plots, mostly reasonable clusters are formed, but a high degree of noise still exists with 7 day averaging. The best and second best result for each evaluation metric, calculated corresponding to the distance metric used, is given in table 3. Note, that a Silhouette coefficient of 1, a higher Calinski and Harabasz score, a Davies-Bouldin score close to 0 and a variance roughly near 0 are desired. KMeans slightly outperforms KMedoids and clustering with DTW yield better results in the silhouette coefficient than ED. We can also confirm the assessment of DBSCAN, because smaller cluster sizes of 5 and 7 mostly outperform higher numbers. This statement can also be supported by the fact that Silhouette coefficient scores near 0 indicate overlapping clusters and KMeans with 15 cluster yield a score around that value.
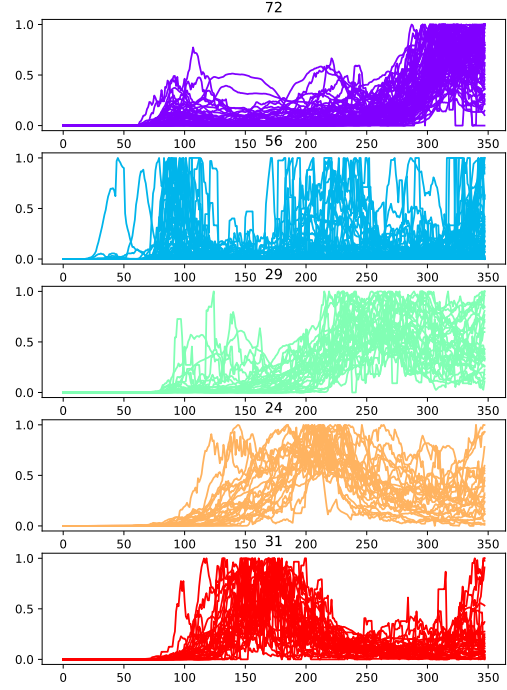

Figure 3: 5 Clusters computed with KMeans, ED, 7 day average; Silhouette coefficient of 0.24, a Calinski and Harabasz score of 55.19, a Davies-Bouldin score of 1.48 and a variance of 341.84
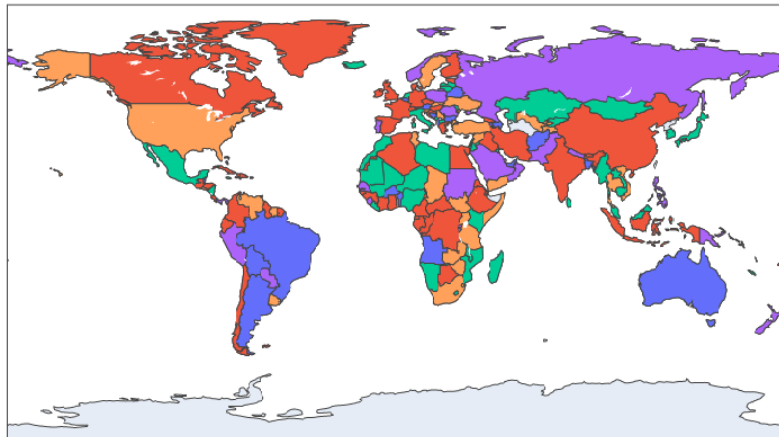
A comparison of cluster sizes can be found in the Appendix B. Figure 4 represents a cluster allocation found with KMeans and DTW. One color, represents one cluster. A shallow statement that we can make is that neighboring countries, especially in Europe, seem to have similar trends, but we are not confident enough to make more profound observations.

Figure 4: Cluster allocation of KMeans, DTW, K=5, 7 day average, resulting in the best found silhouette coefficient score.

4.3 Forecasting Results

As discussed in previous sections, we conducted forecast experiments comprised of two main approaches, namely with and without preceding time-span clustering. For each, we test (seasonal) naïve forecasting against a linear regression model and an LSTM network (see 2.5). Since we use a random $80/20$ split of the dataset the results of each method are depending highly on chance. Therefore we computed each of the methods 5 times and worked with the average of those MAPE values. We discovered that the total number of cases in the true forecast seem to be decisive for the success of the forecast. This might be due to too much unpredictable behavior for small case reports. Therefore, we also consider MAPE on snippets where the true label is larger than 100 or 1000. Table 2 presents the average MAPE results for each method with the best specification found regarding the 1000-measure. Against our expectations, the linear regression model outperforms all forecasting methods including the LSTM. This is surprising because the linear model has no sense of time, but on the other hand linear regression can be solved optimally and is capable of imitating the shallow forecasts, as well as the convolutions. A reason for the mediocre performance of the LSTM against the linear model could be related to too few and too unvaried data[1]. Interestingly to notice is, that preceding clustering does not enhance the results on average. We therefore reason that the chosen models are complex enough to capture different time-series trends. Also, smoothing the data in advance and then inverting the forecast worsens the performance because nearly all methods perform, partially significant, better with true data. This observation was made during multiple tests and can also be noticed from table 2, because all methods performed best with true values.
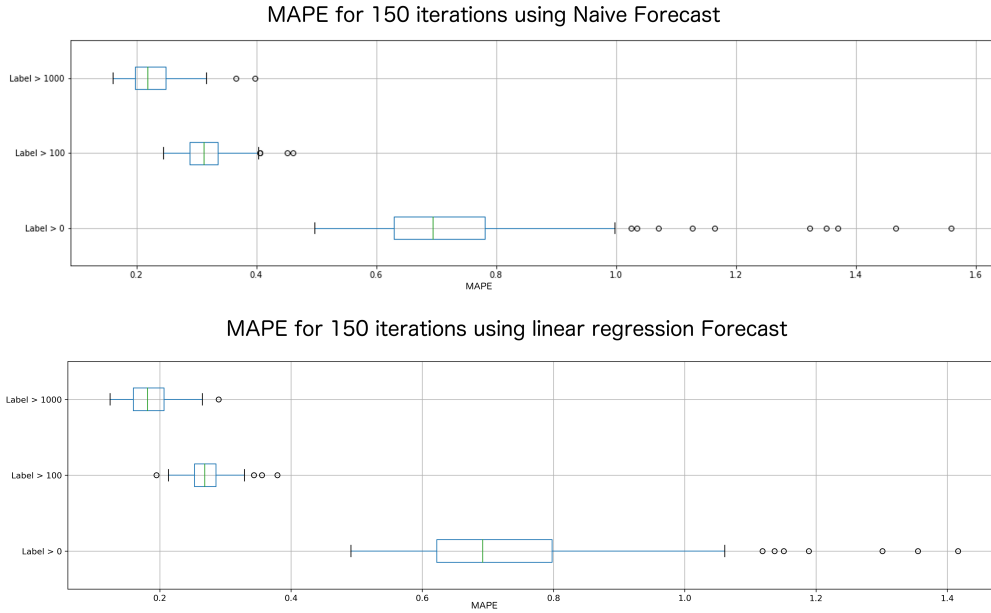

Figure 5: MAPE results of 150 repetitions of the naïve and linear regression forecasting method

As stated above, each single configuration run is influenced by a high degree of randomness. To further investigate this influence we increased the number of iterations for two configurations. The naïve forecasting method and the linear model without preceding clustering, i.e. the model which performed best in the run above. The distribution of

---

[1] We train with $6,700$ snippets generated from 170 randomly chosen different countries.

Method	Avg.	Avg. from 100	Avg. from 1000
Naïve forecast	0.685	0.326	0.203
Naïve w. cluster	0.853	0.383	0.307
Seasonal naïve forecast	0.703	0.339	0.247
Seasonal naïve w. custer	0.919	0.425	0.292
Lin. Regression	0.783	0.281	0.180
Lin. Regression w. cluster	0.676	0.290	0.187
LSTM	0.701	0.292	0.196
LSTM w. cluster	0.715	0.305	0.193

Table 2: Average forecast results for all methods tested. The result for the best configuration is represented. A score of 0.5 means a mean MAPE of 50%

the corresponding MAPE evaluations of each iteration is visualized using 3 boxplots in figure 5. One can see that even with a high variance in one run the linear model clearly outperforms the naïve forecasting method on average in the $1,000$-measure with an average MAPE value of 18.6%. Where the naïve forecast achieves an average MAPE value of 22.5%. In conclusion, one can say that an average deviation of around 23% seems to be attainable with simple models, but exceeding 18% demands more training data or further additional information[8].

5   Conclusion and Outlook

In this Lab we proposed a flexible framework with which we processed, clustered, and predicted developments of Covid-19 cases. We found that simple clustering methods in combination with a more sophisticated distance measurement yield good allocation results. Additionally, we recognized that trends of individual pandemic courses are hard to precisely predict, especially for a small number of cases, and gave reasoning for it. Still, when filtering for higher case numbers very shallow methods are able to achieve an average deviation of around 23% from the true label and we were also able to surpass this result with a linear regression model, reaching an average deviation of around 18%. We assume that to achieve even better performance values, more input information or task specific models are needed, since we tested a variety of models.

For future work, the ECDC dataset also contains death reports where a mortality analysis and forecast could be performed, which is more complex to analyse since there are even more zero and low value entries. Further, additional information such as the political situation or temperature are important factors to forecast ongoing behaviors. Since our framework is designed in a modular way, we are able to attach various kinds of additional information to each snippet. Those could be used for additional similarity measurements, introducing a greater complexity and therefore more precise results.

---

[8] It is to be noted that reported case values don't always portray the correct value since especially in the beginning of the pandemic, Corona tests were not broadly available.

Annotation   We would like to emphasize that this project was carried out in strong collaboration. Therefore we want to accent that each of us takes full responsibility for the content.

References

Ali, M. et al. (2019). "Clustering and Classification for Time Series Data in Visual Analytics: A Survey". In: *IEEE Access* 7, pp. 181314–181338. DOI: 10.1109/ACCESS.2019.2958551.

Calinski-Harabasz Index (2021). URL: https://scikit-learn.org/stable/modules/clustering.html#calinski-harabasz-index (visited on 03/04/2021).

Clustering, Silhouette Score (2021). URL: https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient (visited on 03/04/2021).

Davies-Bouldin Index (2021). URL: https://scikit-learn.org/stable/modules/clustering.html#davies-bouldin-index (visited on 03/04/2021).

European Centre for Disease Prevention and Control (2021). URL: https://www.ecdc.europa.eu (visited on 03/04/2021).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hyndman, Rob J. and Anne B. Koehler (2006). "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* 22.4, pp. 679–688. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL: https://www.sciencedirect.com/science/article/pii/S0169207006000239.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

machine learning in Python - scikit-learn 0.16.1 documentation (2021). URL: https://scikit-learn.org/ (visited on 03/04/2021).

Müller, Meinard (2007). "Dynamic time warping". In: *Information retrieval for music and motion*, pp. 69–84.

Pucknat, Lisa and Alexander Zorn (2021). *Lab2020-21ClusteringCovidData*. https://github.com/mlai-student/Lab2020-21ClusteringCovidData.

scikit-learn-extra documentation (2021). URL: https://scikit-learn-extra.readthedocs.io/ (visited on 03/04/2021).

Sklearn extra, K-Medoids definition (2021). URL: https://scikit-learn-extra.readthedocs.io/en/latest/modules/cluster.html#k-medoids (visited on 03/04/2021).

Sklearn, DBSCAN definition (2021). URL: https://scikit-learn.org/stable/modules/clustering.html#dbscan (visited on 03/04/2021).

Sklearn, K-Means definition (2021). URL: https://scikit-learn.org/stable/modules/clustering.html#k-means (visited on 03/04/2021).

Sklearn, Linear Regression (2021). URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (visited on 03/11/2021).

tslearn 0.5.0.5 documentation (n.d.). URL: https://tslearn.readthedocs.io/.

Zarikas, Vasilios et al. (2020). "Clustering analysis of countries using the COVID-19 cases dataset". In: *Data in Brief* 31, p. 105787. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.105787. URL: https://www.sciencedirect.com/science/article/pii/S2352340920306818.

A   Detailed Configration capabilities Overview

Variables_cfg.ini

- ecdc_dataset_url, dataset_filename, use_dataset_from_file (yes, no): Possibility to provide input data and which to choose.

- log_filename: specifying the filename for the logging output

- replace_negative_values_w_zero: replace negative case values by zero

- add_missing_values_inside_ts: Fill missing case reports with zero everywhere where a country did not report a case number after its first and before its last report.

- Run_data_generation, Run_model_training, Run_model_prediction (each (yes, no)): Decide to run the major steps in the workflow (e.g. If the desired data is already generated in the current output folder that step can be left out).

- number_iterations_for_each_setting_combination (int): Run each config combination several times to get more stable solutions

config.ini

- Data generation

  - test_country_share: set the amount of test data
  - complete_cluster (yes,no): decide whether to work with complete time series (for clustering) or use snippets.
  - examples_snippet_length: Determine the snippet length
  - examples_no_snippets: Give an upper bound on how many snippets to be sampled from each country.
  - divide_by_country_population (yes,no): divide each datapoint by its country population
  - do_smoothing, nr_days_for_avg: Use 1-D convolution and determine the number of days for smoothing.
  - do_data_augmentation, percent_varianz: Do Data augmentation with a given variance.
  - add_additional_info: Capability to add further information to each snippet (e.g. Population of corresponding country). Currently not in use.

- Model training

  - models, metric: specify the set of clustering methods and time series metric to be trained.
  - n_clusters: specify the number of clusters a clustering method is allowed to use.

- Model prediction

  - forecast_evaluation_function: specify the function to evaluate the forecast results of all test set snippets.
  - forecast_function: determine the function used to forecast all test set snippets.

B Overview over performance of cluster number

No. cluster	Silhouette	Calinski	Davies	Variance
5	0.174	35.036	2.629	353.640
7	0.146	28.593	2.700	224.315
10	0.118	22.623	2.599	142.938
15	0.098	17.678	2.531	80.201
20	0.088	15.166	2.339	57.337

Table 3: Results for complete time-series clustering. All cluster results are grouped by the number of clusters and averaged. Higher number of clusters seem to be overlapping.