COMP-592DL | Project in Data Science | Clustering M4 Daily Data for Forecasting

Philipp Beer

May 3, 2021

1 Introduction

This project paper provides information on the impact of time series clustering on a machine learning (ML) forecasting method. Accurate point forecasts (PFs) for time series data are a challenging task for machine learning methods which still get regularly outperformed by much simpler statistical approaches. As shown in [1], all machine learning methods submitted for the M4 competition had lower accuracy than the used combined benchmark method which consists of statistical methods only. Even worse the complexity of machine learning methods adds significant computational costs to a model without providing a useful reward for it.

One question in this context is, whether time series can be combined in such a way that training one model with the appropriate subset simplifies the learning process and increases the point forecast accuracy. Machine Learning models usually have a need for lots of data to be able to perform well in their prediction tasks. For time series a long history is usually a rare commodity and seldomly available. Other approaches are needed if one wants to use ML methods. Therefore, it is interesting to verify whether time series that have similar properties provide additional PF accuracy when trained with the same ML model. A challenge is to choose the right time series properties to simplify the learning task for the algorithm. If done well it should allow for less complex models to achieve better performance by having more data to optimize the model parameters.

1.1 Approach

A large pool of time series is available in the Makridakis competition. For this research project the daily series of the M4 competition was chosen to analyze whether a simple kMeans algorithm can help to cluster time series via its properties such that the same algorithm can perform more accurate forecasts with a high degree of certainty.

Properties

To identify the most relevant time series properties that may be useful in forecasting the tsfresh package developed and described by [2] was used to compute around 800 different metrics ranging from min, max, number of peaks, autocorrelations, and many others. These properties describe the corresponding time series in their behavior. The assumption in this project is that these properties reveal information that can be used to combine time series for forecasting with machine learning algorithms. Similar properties should make the learning process for machine learning models easier and allow to provide more data compared to training models with individual time series. It assumed that it is also more effective than training the model on all available time series.

To find the most relevant properties for a time series the engineered features are filtered via finding the difference between class conditional distributions for different target values. The full mechanism is described in [2]. A features is deemed relevant if the conditional distributions for the feature X are different for the target values Y_1 compared to Y_2; in other words if X and Y are not statistically independent:

$$\exists\, y_1, y_2 \text{ with } f_Y(y_1) > 0, \\ f_Y(y_2) > 0 : f_{X|Y=y_1} \neq f_{X|Y=y_2} \tag{1}$$

Therefore, a target variable needs to be provided in order to measure the relevance. To avoid being impacted by randomness by selecting only a single target variable for each series the time series is transformed into sliding windows and the subsequent value of the time series is used as the target for each window.

After constructing the list of the relevant features for each series the properties where combined into a union of all relevant properties. The features were then extracted once again on the whole time series. This set of engineered features is then used as the dimensions for the clustering algorithm.

Clustering with K-Means

The idea is to find a prototype for each time series to which it belongs. K-means is chosen as forecasting algorithm to compute centroids of similar time series based on the time series properties. With K-Means it is not clear which value K should have. Therefore all k between 2 and 250 are computed and the silhouette scores s of the resulting class associations for each time series are measured:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2}$$

where a is the mean distance to the other instances in the same cluster, and b is the distance mean nearest cluster distance. In consequence values closer to one are better. To avoid picking a cluster that generates good scores by chance the top 3 K identified via silhouette scores are used for forecasting.

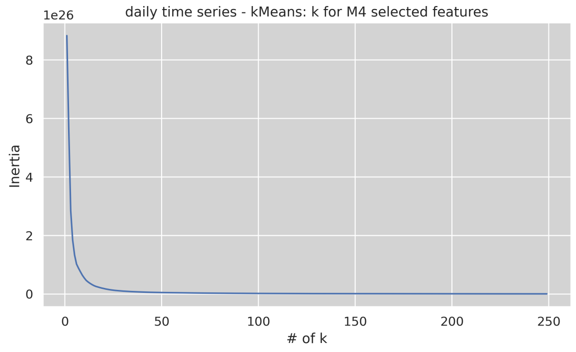

Figure 1: M4 daily K-Means Inertia for different k

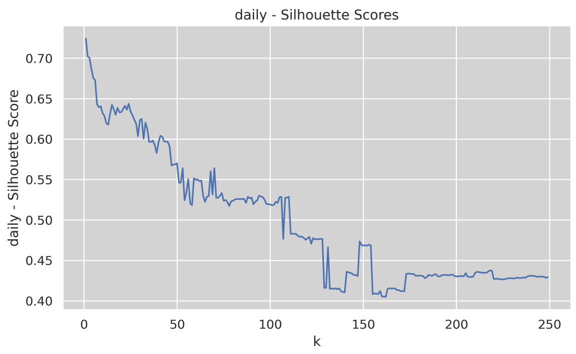

Figure 2: M4 daily K-Means Silhouette Scores for different k

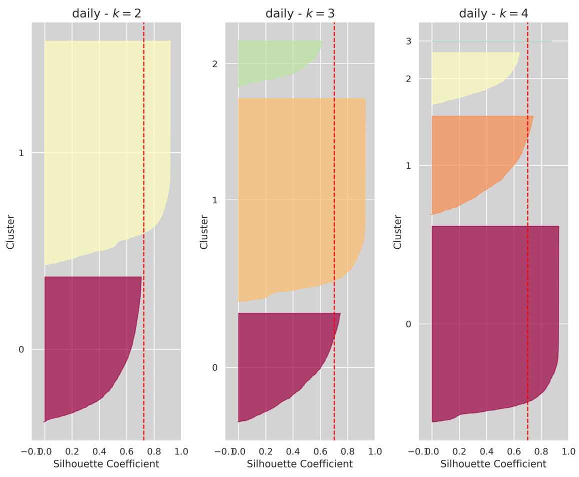

Figure 3: M4 daily K-Means silhouette diagram for top k

Cross-Validation

To increase certainty about the belief that any potential improvement generated via the clustered time series is the result of combining alike time series one has to ensure that the better performance is not due to the specific peculiarities of the data. This can be achieved via cross-validating the forecasting models per trained cluster. In this case a 10 fold cross validation is performed on the daily training data of the M4 competition. Specifically, the training set is increased with every fold from the starting point of the training set. This mimics the real world reality that datasets are continuously growing over time and increasingly more data can be used for training. For each fold the chosen 7 steps ahead forecast is generated and the respective error metrics discussed in 2 are computed and averaged over the different folds. To compare against the real world performance the full training set is separately trained once again and its error metrics computed with the corresponding M4 test set. This allows to show whether the computed cross-validation metrics hold in a real world scenario given the clustering of the series.

Forecasting

To allow for comparison of the clustering performance the following forecasts time series setups are generated: (1) A model with all time series is trained to allow for a base line view of forecasting accuracy. (2) The top 3 kMeans k-clusters identified via the silhouette scores are trained with one model for each cluster to verify whether clustering allows for an improvement over training on the entire dataset. (3) For each k and its corresponding classes a set of the same of amount of classes is generated by drawing random time series from the used data set for each class. The classes also correspond in class size to its counterpart k-clustering, meaning the number of time series inside each particular randomly selected class match the respective number of classes in the kMeans cluster. If the random clusters perform similar to the k-clustered segments one cannot infer that the clustering helped with any possible improvement but rather that less time series per model simplify the learning process.

The machine learning model chosen for the forecasts is a relatively simple neural network with 3 hidden layers. The loss function is defined as mean squared error. Per model 100 epochs are executed and the batch size is set to 128.

2 Benchmarking

For the benchmarking of the results the error metrics of the M4 are employed, namely symmetric mean absolute percentage error (sMAPE):

$$SMAPE = \frac{100}{n} \sum_{t=1}^{n} \frac{F_t - Y_t}{(|F_t| + |Y_t|)/2} \tag{3}$$

where F_t is the forecasted value and Y_t is the actual value at time t. The denominator consists of the sum of absolute values of the forecast and the actual value divided by 2. The second metric is the mean absolute scaled error:

$$MASE = mean\left(\frac{|e_j|}{\frac{1}{T-1}\sum_{t=2}^{T}|Y_t - Y_{t-1}|}\right) \tag{4}$$

where e_j is the forecast error for a given period of J forecasts, defined as $e_j = Y_j - F_j$, where Y_j are the actual values and F_j are the forecasted values. The denominator consists of the mean absolute error (MAE) of the naive forecast, defined as $Y_t - Y_{t-1}$ computed on the training set T from 1 to t. Y_t and Y_{t-1} represent the actual values of the training set.

For the used daily series from the M4 competition a 7-step forecast is generated for 7 consecutive steps increasing the last forecast step to 14 steps ahead. These metrics are employed for both the cross validation computation as well as the comparison to the test test.

3 Challenges

The data pipeline consists of the following major parts: (1) preprocessing, (2) feature extraction and selection, (3) clustering, (4) forecasting, (5) post-processing. Each area entails its own set of challenges.

3.1 Data Preprocessing

Loading and preprocessing the data is required such that the data format matches the expectation of the tsfresh package to be able to compute the time series properties. The M4 dataset is presented in a wide-format layout, whereas tsfresh expects a long format. Preprocessing also included normalizing the data for the utilization via neural network. The *Min-Max feature scaling* method was chosen and computed on the training set per time series. The scaling is computed via:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{5}$$

Not using the test set for the scaling of the values is important as otherwise information leakage from the test set to the training set occurs. In the real world it is unknown whether future values will be larger or smaller than all values observed thus far. An unintended consequence of this computation approach is, that it cannot be guaranteed that the computed values will remain between 0 and 1 - as preferred for the usage within neural networks - between the train and the test set. This issue is made worse via the utilization of cross validation. With this approach even less records are present in the train set increasing the likelihood that the test set contains values that exceed the thresholds used for the normalization.

3.2 Feature Extraction

The tsfresh packages allows for various different metrics to be computed that are separated into various categories (Comprehensive, Minimal, Efficient, Timebased, IndexBased) which can be chosen to be computed. Computational complexity will be discussed in section 3.3. However, due to the constraints on the availability of compute resources the efficient parameter setting was selected as starting point for feature identification for each series. And the relevant subset of those features was computed for all series again.

3.3 Computational costs

The data pipeline described is computationally expensive. In order to be able to train one of the M4 datasets in full a cloud computing unit with 6 vCPU and 32GB of RAM was chosen for this task. The tsfresh feature extraction and selection takes on average 40 seconds per time series. With 4227 time series in the M4 daily dataset this initial step takes close to two days. For the clustering a maximum of 20 clusters is selected to keep the required execution time within an acceptable bound. The neural network has a simple setup with 3 hidden layers. Considering cross-validation for the training each time series is used in training for a different neural network 67 times.

Tsfresh as well as the clustering of scikit-learn already implement multiprocessing to improve the time required for the computations. For the various trained neural network a multiprocessing pool is implemented to train the separate models simultaneously as well.

4 Results

The results indicate that not training all time series with a single model improves the forecasting performance. Training all series via a single model resulted in sMAPE of 4.55 and MASE 4.68. The best clustered metric was k equal to 4 with sMAPE of 4.02 and MASE 4.45. However, this is likely not attributable to the clustering but due to the reduced number of time series used per model as it does not outperform the randomly created clusters. They perform almost identical in the error metrics on the test set. Except for K equal to 2 the clustered series slightly outperform the randomly selected clusters. However, the differences are insignificant.

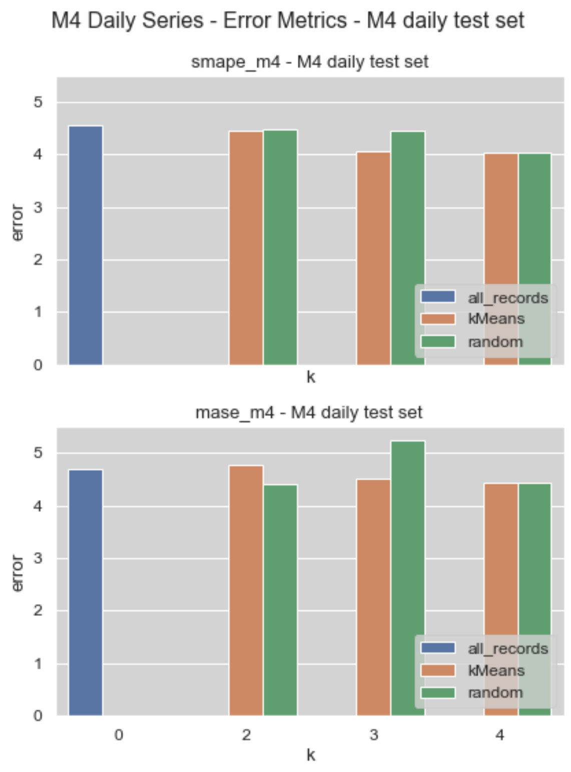

Figure 4: M4 Daily Series - full test set error metrics

The error metrics that are generated as averages from the cross validation steps are worse for the K-Means clusters and indicate that choosing clusters of alike series makes accuracy worse compared to selecting them randomly. For both metrics the randomly selected classes perform better than the series combined via clustering.

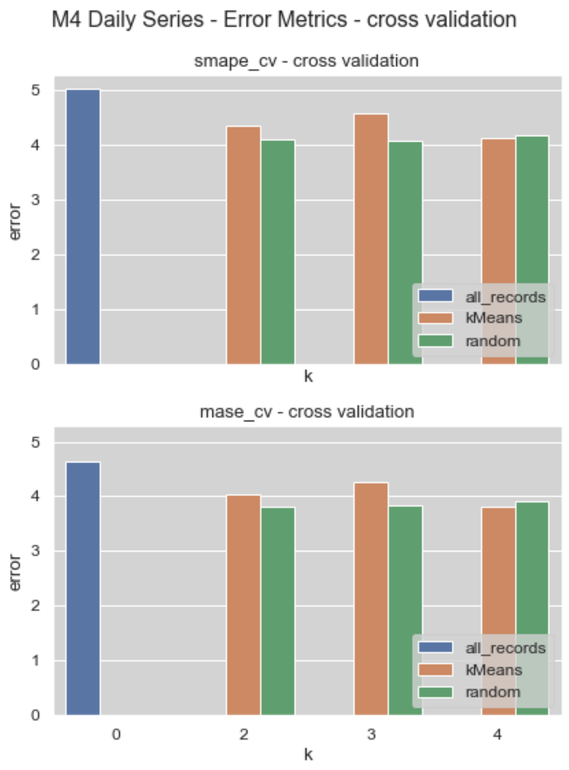

Figure 5: M4 Daily Series - cross validation error metrics

5 Conclusion

The clustering of the time series as presented requires additional consideration before it could add value to forecasting. And within cross-validation the performance for models using clustered series was worse than random combinations of series. Currently the identified time-series properties of the clustered series play only an indirect role with the model training. In particular the forecasting model is built only on the lags 1 to 7 instead of using the identified features for the forecasting process. This means that the properties that are similar were not presented to the forecasting algorithm to actually use as information for learning from the data and hence it was not able to utilize the gained information. Modifying the features in this way would possibly increase the accuracy of the model. But a negative side-effect of the approach would be a significant expansion of the neural network structure as instead of 7 features - the lags - it would be roughly 150 features that the model would have to be trained on.

Clustering also impacts the uncertainty that each model is exposed to. Likely, the clustered time series contain reduced overall noise. A neural network trained on such a cluster is exposed to less noise due to the fact that the time series used for its training are more similar than a random selection. This randomness however is useful in the training data to avoid over-fitting of the model. If the clustering process reduces the randomness in the data it will be negatively impacted by the randomness that any test data set and real-world data will inadvertently present.

Another possible reason for the poor performance of the clustered time series is the selection of the features to be computed. As mentioned in 3.2 the extracted features were restricted to features that can be computed efficiently. Possibly more meaningful properties for the forecasting process were left out and in consequence only less useful features were computed.

Also the K-model selection via silhouette scores possibly is not the best metric for choosing which clusters to use. This could be verified via using different clustering algorithms and selection methods.

6 Outlook

Looking forward the following aspects deserve attention for future analysis. The complete M4 dataset needs be trained using this approach. And the forecast needs to be aligned to the M4 forecasting mechanism of predicting each forecasting point in the test set once. Furthermore, prediction intervals should be computed and compared using this clustering approach.

A meaningful change to the approach would be to exchange the currently used lags in the forecasting method with the features identified in the clustering.

Other clustering algorithms like agglomerative hierarchical clustering and density based methods need to be considered as well. The cross-validation method could be varied to see whether using chunks changes the performance of the computed errors.

It should also be considered whether a meaningful general ranking for time series features can be created that allows to reduce the number of features that need to be computed, based on the forecasting task at hand. Additionally the implementation of the feature computation can be addressed and moved to a more performant framework outside Python.

References

[1] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M4 Competition: 100,000 time series and 61 forecasting methods". In: *International Journal of Forecasting* 36.1 (Jan. 2020), pp. 54–74. ISSN: 0169-2070. DOI: `10.1016/j.ijforecast.2019.04.014`. URL: `http://dx.doi.org/10.1016/j.ijforecast.2019.04.014`.

[2] Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. "Distributed and Parallel Time Series Feature Extraction for Industrial Big Data Applications". In: *CoRR* (2016). arXiv: `1610.07717 [cs.LG]`. URL: `http://arxiv.org/abs/1610.07717v3`.