Toward a Framework for Seasonal Time Series Forecasting Using Clustering

Colin Leverger, Simon Malinowski, Thomas Guyet, Vincent Lemaire, Alexis Bondu, Alexandre Termier

Toward a framework for seasonal time series forecasting using clustering

Colin Leverger[1,2], Simon Malinowski[2], Thomas Guyet[3][0000−0002−4909−5843], Vincent Lemaire[1], Alexis Bondu[1], and Alexandre Termier[2]

[1] Orange Labs, 35000 Rennes – France
[2] Univ Rennes, Inria, CNRS, IRISA, 35000 Rennes – France
[3] AGROCAMPUS-OUEST/IRISA - UMR 6074, 35000 Rennes – France
colin.leverger@orange.com

Abstract. Seasonal behaviours are widely encountered in various applications. For instance, requests on web servers are highly influenced by our daily activities. Seasonal forecasting consists in forecasting the whole next season for a given seasonal time series. It may help a service provider to provision correctly the potentially required resources, avoiding critical situations of over- or under provision. In this article, we propose a generic framework to make seasonal time series forecasting. The framework combines machine learning techniques 1) to identify the typical seasons and 2) to forecast the likelihood of having a season type in one season ahead. We study this framework by comparing the mean squared errors of forecasts for various settings and various datasets. The best setting is then compared to state-of-the-art time series forecasting methods. We show that it is competitive with them.

Keywords: Time series · forecasting · time series clustering · Naive Bayesian prediction.

1 Introduction

Forecasting the evolution of a temporal process is a critical research topic, with many challenging applications. One important application is the forecast of future consumer behaviour in marketing, or cloud servers load for popular applications. In this work, we focus on the forecast of *time series*: a time series is a timestamped sequence of numerical values, and the goal of the forecast is, at a given point of time, to predict the next h values of the time series, with $h \in \mathbb{N}^*$. A practical example is the time series of outdoor temperature values for New York: sensors capture temperature values every hour, and the goal of forecasting can be to predict temperatures for the next 24 hours. Many forecasting tasks can be reformulated as time series forecasting, making it an especially valuable research topic.

Forecasting in time series is a difficult task, which has attracted a lot of attention. The most popular methods are Auto-Regressive, ARIMA or Holt-Winters, and come from statistics. These methods build a mathematical model of the time series and focus on predicting the next value ($h = 1$).

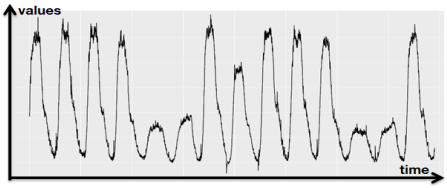

Fig. 1. Seasonal time series example borrowed from [9]. Two weeks of the Internet traffic data (in bits) from a private ISP with centres in 11 European cities. The whole data corresponds to a transatlantic link and was collected from 06:57 on 7 June to 11:17 on 31 July 2005. The time series is obviously seasonal, but the assumption of having one unique periodic pattern seems not suitable in this case.

Many time series, especially those related to human activities or natural phenomena, exhibit *seasonality*. This means that there is some periodic regularity in the values of the time series. In our New York temperature example, there is an obvious 24h seasonality: temperatures gradually increase in the morning, and decrease for the night. Human activity behaviours often exhibit daily and weekly seasonality. Knowing or discovering that a time series is seasonal is a precious information for forecasts, as it can restrict the search space for the mathematical model of the time series. The classical approach to deal with seasonality is called STL [5], it builds a model while taking into account three components: seasonality, trend (long-term evolution of the time series: increase or decrease) and residual (deviation from trend and seasonality). It assumes that the time series exhibits a single periodic behaviour (ex: daily periodicity of temperatures). Holt-Winters and ARIMA models have been extended to deal with seasonality. But they also assume a single and clear periodic behaviour.

However, this assumption is often violated in practical cases. Consider the time series in Figure 1, which shows an Internet traffic measurement for two weeks. While there is indeed a daily periodicity, there are two types of daily patterns: weekday patterns and weekend patterns. This cannot be well captured by the STL framework and the associated statistical methods.

In this article, we address the task of forecasting the next season of a seasonal time series, with regularities in the seasonal behaviours. We propose a framework for that task, in the light of the work in [12]. This framework is based on two steps: (1) the identification of typical seasonal behaviours and (2) the forecasting of the season. It only requires as input the time frame of the seasonality (ex: daily, weekly, etc.), and it determines the main seasonality patterns from the analysis of past data (without any assumption on the number of these patterns). In our previous example, it would automatically identify the weekday pattern and the weekend pattern.

Our contributions are the following:

1. we propose a general framework for seasonal time series forecasting,
2. we provide extensive experiments of our framework with various settings and various datasets,

3. we compare our framework with alternative strategies for time series forecasting.

In the remainder, we present in detail our approach, which is based on a clustering step to determine the seasonal patterns, followed by a classification step to build a "next pattern predictor". Our thorough experiments on real data determines which are the best combinations of clustering and classification method, and show that our approach compares favourably to state-of-the-art forecasting methods.

2 Related work

Time series analysis [3] has become a recent challenge since more and more sensors collect data with high rates. In statistical approaches, statistical models are fitted on the data to predict future values. The nature of the statistical model is chosen depending on the data characteristics (noise, trends, stationarity, etc.). For a wide literature on statistical methods for time series forecasting, we refer the reader to Brockwell's book [3]. Autoregressive methods (AR, ARIMA, etc.) demonstrate a great success in lots of applications. The advantages of those models are the hypothesis simplicity, the computational efficiency during the training phase and the handling of the noise. For instance, in [10], authors use several ARIMA models to predict day-ahead electricity prices. Kavasseri et al. [11] use fractional-ARIMA to generate a day-ahead and two days ahead wind forecasts. They show that the method is much better than the persistence model. But, in these works, methods do not directly handle the seasonal dimension of the data. Indeed, ARIMA expects data that is either not seasonal or has the seasonal component removed, *e.g.* seasonally adjusted via methods such as seasonal differencing [2]. The SARIMA model [8] extends the autoregressive model to deal with seasonality. Such model exhibits autocorrelation at past lags of multiple of season period for both autoregression and moving average components.

In the machine learning communities, one of the objectives is to create data analytic tools that would require fewer modelling efforts for data scientists. Concerning time series forecasting task, machine learning has been used in mainly two ways: use of neural network techniques and of ensemble methods. Neural networks models can be efficient for forecasting tasks, especially when the data is more complex and when the process is non-linear. One drawback of those models is their tendency to over-fit [19], which may cause lower performances. LSTM [7] is a classical neural network architecture used for time series forecasting, but not dedicated to seasonal time series. In [18], the authors use ANN and fuzzy clustering for creating daily clusters that are afterward being used for forecasts. Ensemble forecasting methods and hybrid models are created from several state of the art, independent models, that are mixed to create more complex chains. This strategy is the one proposed in the Arbitrated Dynamic Ensemble [4]. This metalearning method combines different models, regarding to their specifies against target datasets. In [15], authors use both ARIMA and

SVMs models to forecast stock prices problem, to tackle the non-linearity of some datasets.

3 Seasonal time series forecasting

A time series Y is a temporal sequence of values $Y = \{y_1, \ldots, y_n\}$, where $y_i \in \mathbb{R}^d$ ($d \in \mathbb{N}^*$) is the value of Y at time i and n denotes the length of the time series. If $d = 1$, Y is said univariate. Otherwise, Y is said multivariate. We assume here that time series are univariate, regularly sampled and that there is no missing data. We also assume that there is no trend component in the time series[4].

The problem of time series forecasting is a classical problem: given the knowledge of Y up to sample n, we want to predict the next samples of Y, *i.e.* $y_{n+1}, \ldots, y_{n+h}$, where h is called the prediction horizon. Predictions are denoted $\hat{y}_{n+1}, \ldots, \hat{y}_{n+h}$. Quality of the predictions is related to the difference between real and predicted values. Different measures can be used to evaluate the quality of predictions. We will introduce some measures in Section 5.

Let s be the seasonal periodicity of the considered time series. A typical season is a time series $\tilde{y} = \{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_s\}$ of length s. Let $\mathcal{Y} = \{\tilde{y}^{1..k}\}$ be a collection of k typical seasons, then a seasonal time series $Y = \{y_1, y_2, \ldots, y_n\}$ is a univariate time series of length $n = k \times s$ such that:

$$y_i = \tilde{y}^{k_{\sigma(i)}}_{i-s\times\sigma(i)} + \epsilon, \ \forall i \in [n]$$

where $\sigma(i) = \lfloor \frac{i}{s} \rfloor$ is the season index of the i-th timestamp of the series, k_i is type of i-th typical seasonal time series and $\epsilon \sim \mathcal{N}(0, 1)$ is a gaussian noise.

The seasonal time series forecasting is a forecasting of a seasonal time series at an horizon s, *i.e.*, the prediction of the time series values for the whole next season ahead.

4 Framework for seasonal time series forecasting

The general sketch of our approach is composed of two different stages: one for learning the predictive models (see Figure 2) and the other to use this model to predict the next season (see Figure 3). The learning stage is composed of three steps: (i) data splitting, (ii) clustering of the seasons, (iii) training a classification algorithm. The predictive stage is composed of two steps: (i) predict a cluster index, (ii) forecast the next season using the predicted cluster index. We describe in this section all these steps in detail.

Let $Y = \{y_1, \ldots, y_n\}$ be an observed time series with m seasons of length s ($n = m \times s$). We assume that the time series in our possession are equally sampled and that there is always the same number of points s per season. We assume that s is known beforehand: it could be daily, weekly, monthly, or even yearly; and that it does not change over time. Finding the seasonality s is not in the scope of our study.

[4] In practice, it is not a problem, as most time series could be easily decomposed and detrended. Trend components can then be re-applied on the forecasted values.

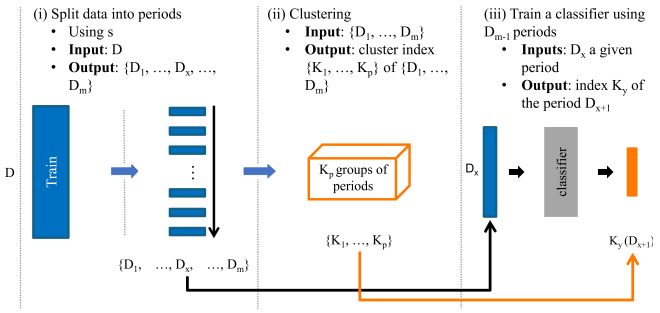

Fig. 2. Illustration of the learning steps.

4.1 Learning stage

The learning stage is composed of three steps (see Figure 2):

(i) The data splitting step consists in constructing a set of m seasons, $\mathcal{D} = \{D_1, ..., D_i, ..., D_m\}$, where D_i represents the subseries corresponding to the i^{th} season of measurements. We assume that observations of the $\mathcal{D}$ are independent.

(ii) The m elements, seasons, of $\mathcal{D}$ are given to a clustering (or co-clustering) algorithm that gathers similar seasons into p groups of typical seasonal time series.

(iii) A probabilistic classifier is then trained to estimate, using the knowledge of a current season D_X, the expected group of next season (season $X+1$ denoted the season just after the season X).

The data splitting step is straightforward (since input datasets are considered to be regularly sampled), for the other step we give just below details on them.

Clustering step: details and algorithms - In this step, a clustering algorithm is used to group the m seasons of $\mathcal{D}$ into p groups. The choice of p will be discussed later at the end of this section. A representative series is computed inside each group. These series represent typical seasons that occur in the dataset. Hence, at the end of this step, every season of $\mathcal{D}$ can be assigned a label (that represents in which group it has been clustered) and p typical seasons are computed. We consider in this paper four different clustering algorithms:

K-Means for time series [13] K-means algorithm aims at creating a partitioning of the data in k clusters by minimising the intra-cluster variance. The use of K-means implies the use of a distance measure between two time series. Two of the major distance measures available for time series are Euclidean, and Dynamic Time Warping (DTW) [17].

K-Shape for time series [16] K-Shape algorithm creates homogeneous and well separated clusters and uses a distance measure based on a normalised version of the cross correlation (invariant to time series scale or shifting). It

can be seen as a K-means algorithm but that uses a shape-based similarity measure.

Global Alignment Kernel K-means (GAK) [6] Kernel k-means is a version of k-means that computes the similarity between time series by using kernels. It identifies clusters that are not linearly separable in the input space. The Global Alignment Kernel is a modified version of DTW.

Co-clustering with MODL [1] MODL is a nonparametric method that uses a piecewise constant density estimation to summarise similar curves. Curves are partitioned into clusters and the curve values are discretised into intervals. The cross-product of these discretisation is an estimation of the joint density of the curves and points.

The choice of the number of clusters is of particular importance in our prediction framework. We use a tuning approach: for each candidate number of clusters, the training set is used to build the overall model. This model is then used to predict the seasons of the validation set. The number of cluster that leads to the lowest error on the validation set is selected. For K-means, K-shape and GAK algorithms, candidates number of cluster are systematically chosen in a pre-defined range $[2, 300]$. Partitions with empty clusters are discarded. On another hand, MODL co-clustering estimates in a nonparametric way the best number of clusters for each input time series. This estimated number usually leads to the best description of input time series, regarding to the coclustering task. However, this model can be simplified to reduce the number of clusters. From now on, the procedure is similar: different number of clusters are evaluated and the overall model that leads to the lowest error on the validation set is kept.

Predicting the cluster index of the next season - A probabilistic classifier is then trained to estimate, using the knowledge of the current season D_{X}, the expected group of next season. To train this classifier, we first apply a clustering model as described above to create groups of similar seasons. A learning set is then created to feed the classifier: each line of the learning set corresponds to time series values of a season D_{X} (explanatory variables) and a target variable which corresponds to the group of the next season (the group of season $D_{\mathrm{X}+1}$). We think that including more data in the classifier would probably be beneficial: data about few past days, and not only the last one, data about national holidays, all sorts of exogene data we can find related to the problem treated; this last point will be studied in future work.

Any classifier could be chosen at this step, the only constraint is that the output of the classifier has to be probabilistic (not only a decision but a vector of conditional probabilities of all possible groups given the input time series). In the Figure 3, $\widehat{K_Y}$ denotes a vector of probabilities ($i.e$ $\{\widehat{K_1}, ..., \widehat{K_p}\}$). In this paper, three different types of probabilistic classifiers are investigated: a Naive-Bayes classifier, a Decision Tree and a Logistic Regression.

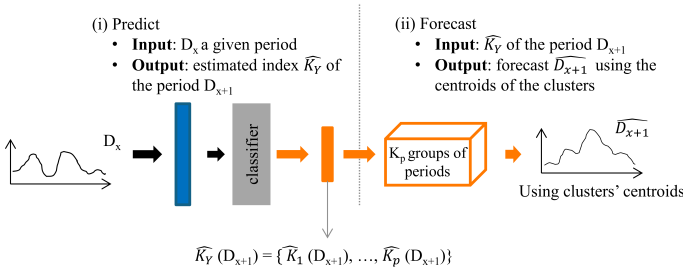

Fig. 3. Illustration of the forecasting steps.

4.2 Forecasting the next season

To forecast the next season, we use the classifier learned above to predict the group index of the next season (using the knowledge of the current season), and representative series of the different groups to generate the prediction of the next season. The forecasting of the next season is done in two steps (see Figure 3). First, the current season (D_X) is given to the classifier. This classifier computes the probabilities of the next season to belong to each group ($\{\widehat{K_1}, ..., \widehat{K_p}\}$). Then, next season is predicted as a weighted combination of the groups centroids, the weights corresponding to the probabilities output by the classifier.

5 Experiments

We have performed different experiments to assess the performance of the proposed method.[5] We were curious about the impacts of the choice of both the clustering and the classification algorithms on the performance of the overall method. We also aimed to find out how our approach compares with classical time series prediction approaches.

We have used nine datasets to assess the performance of the proposed approach. Each dataset is a seasonal time series monitored over many seasons. Eight datasets are open source datasets: five are from the time series data library [9], one is from the City of Melbourne [14] (Pedestrian Counting System) and one is provided by Kaggle[6]. The two other datasets have been provided by Orange company via a project named Orange Money. The first time series provided by Orange represents the number of people browsing a website at a time of the day. The second one is a technical metric collected on one server (CPU usage). These nine time series have different seasonalities (daily, weekly, monthly, quarterly). All the time series have been z-normalised beforehand.

Each dataset is separated into a training set, a validation set and a test set. The training set is composed of 70% of the seasons, while both validation

[5] For reproducible research, code and data are available online `https://github.com/ColinLeverger/IDEAL2019-coclustering-forecasts`.

[6] See: https://www.kaggle.com/robikscube/hourly-energy-consumption

Table 1. Performances of all the possible combination of clustering and classifiers algorithms for our chain, in terms of MSE and MAE. Bold Figures are best MSE/MAE.

	MSE			MAE		
	Naive Bayes	Tree	Logistic Reg.	Naive Bayes	Tree	Logistic Reg.
MODL	0.480	0.581	**0.479**	**0.462**	0.494	0.487
K-means	0.799	0.583	0.492	0.564	0.526	0.492
GAK	0.571	0.638	0.608	0.538	0.576	0.590
K-shape	0.745	0.824	0.852	0.575	0.677	0.693

set and test set are composed of 15% of the seasons. However, no shuffle is done during the separation, as it is important to learn the relations between contiguous seasons. The shuffle would remove those relations, which is not acceptable for a "time split". The validation set is used to select the appropriate number of clusters of the different clustering algorithms. The test set is used to assess the performance of the different approaches.

Two metrics are used in this paper to evaluate the quality of the predictions: the Mean Square Error (MSE) and the Mean Absolute Error (MAE):

$$MSE(y, \hat{y}) = \frac{1}{s} \times \sum_{j=1}^{s} (y_j - \hat{y}_j)^2, \quad MAE(y, \hat{y}) = \frac{1}{s} \times \sum_{j=1}^{s} |y_j - \hat{y}_j|$$

where $y = y_1, \ldots, y_s$ is the ground truth and $\hat{y} = \hat{y}_1, \ldots, \hat{y}_s$ is the prediction. As these two measures represent prediction errors, the lower they are, the more accurate the predictions.

5.1 Overall performance of the approach

As explained in Section 4, our proposed approach is based on a clustering algorithm and a classifier. For comparison purposes, we made use of 4 different clustering algorithms and 3 different classifiers. We want to analyse the performance of these different algorithms. Table 1 gives the average MSE and MAE over the 9 datasets for every possible combination of clustering and classifier.

According to this table, MODL outperforms the other clustering algorithms. Combined with a Naive Bayes classifier, it reaches the best performance for 3 of the 4 settings. On the other hand, K-shape is the worse clustering algorithm for this task. A detailed comparison of the clustering algorithms and the classifiers will be made later in this section.

5.2 Performance of the different clustering algorithms

Figure 4-(a), (b) and (c) compare the performance of MODL against the three other considered clustering algorithms. Each point in these figures represents a combination of experimental settings: clustering algorithms, classifiers, datasets, and metrics. For each of these combinations, the performance is computed according to the considered metrics and inserted in the figure. It shows that MODL

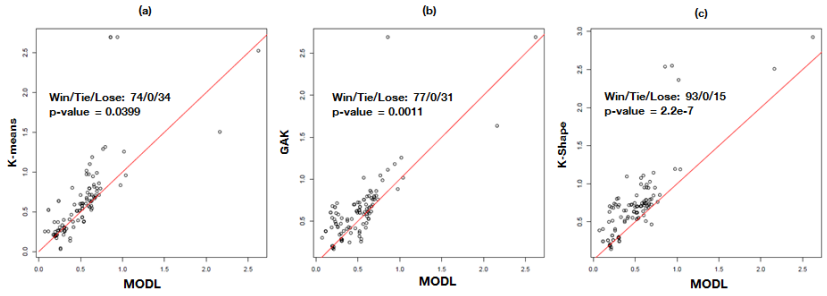

Fig. 4. Comparison between MODL and the other clustering algorithms (K-means, GAK and K-shape). The axis represents an error measure (either MSE or MAE) associated with the clustering algorithm of the axis labels. A point above the diagonal is in favour of the MODL approach.

outperforms the other clustering algorithms. For each comparison (MODL versus another clustering algorithm), a Wilcoxon test has been carried out. The $p-$value of these tests is depicted in each figure. For each comparison, we can see that this probability is less than 0.05, meaning that the performance of MODL is significantly better than the one of the other clustering algorithm. We have also carried out a Nemenyi test to compare the average ranks of the different approaches. The critical diagrams associated with these tests are given in Figure 5. The left-hand side represents the comparison of the average rank according to the MSE measure, while the right-hand side is associated with the MAE measure. It shows that MODL outperforms the three other algorithms, significantly for the MSE measure.

5.3 Performance of the different classifiers

As explained above, an important part of our method is to predict the group (or cluster) of the next season. This prediction is made using a classifier. Once the group is predicted, a weighted combination of different centroids is used to make the forecast. A centroid of a cluster is the Euclidean mean of all its curves. In this section, we first make a performance comparison of the three considered classifiers. Let us compare the performance of the three considered classifiers: a Naive-Bayes classifier, a decision tree, and a logistic regression. The critical diagrams that compare the performance of the three classifiers are given in Figure 5. The left diagram is for the MSE measure and the right one for the MAE measure. We can see that for the MSE measure, the performance of the classifiers are very close (no significant difference between the ranks). For the MAE, the Naive Bayes classifier is better than the two others.

5.4 Comparison with other prediction methods

In this section, we aim at comparing the performance of the proposed approach (PA) with other competitive prediction methods. Following the analysis above,

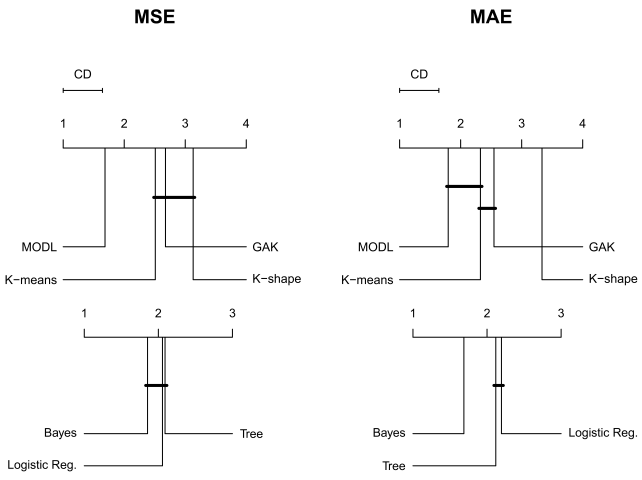

Fig. 5. Critical diagram, comparison of the average ranks of the difference clustering algorithms (on the top) different classification algorithms (on the bottom) in terms of MSE (left) and MAE (right).

we use for our approach the following setting: MODL algorithm is used as the clustering algorithm and the classifier is chosen according to the validation set. We compare the performance of our approach with the following four other prediction methods: mean season, Markov model (MM), autoregressive models (AR) and an algorithm called Arbitrated Ensemble for Time Series Forecasting (AETSF). The mean season is a naive approach that consists in predicting the next season as the average season of the training set. The MM approach has been proposed in [12]. It follows the same framework as the one proposed here. However, the classifier is replaced by a Markov model, whose transition probability matrix is used to predict the cluster of the next season. Auto-regressive models have widely been used for time series forecasting. The order of the regressive process is set to the length of the seasonality of the time series. Finally, the AETSF method has been proposed in [4]. It is an ensemble method that combines prediction of several algorithms based on meta-learners. This method has been shown to reach state-of-the-art performance for time series prediction. We have used four base learners: generalised linear models, support vector regression, random forest and feed forward neural networks.

Table 2 compares the performance of all these approaches for the nine considered datasets and the MSE measure. In terms of average MSE and mean rank, the AETSF is the best method, just in front of the proposed approach. The critical diagram associated with the Nemenyi test is given in Figure 6. The difference between AETSF and our approach is not significant on these nine datasets.

Table 2. Comparison of the proposed approach with other prediction methods.

Dataset	Mean season	MM	AR	AETSF	PA
1	1.2349	0.1800	0.7981	0.1771	**0.1750**
2	3.2345	0.7698	1.7460	**0.3988**	0.5172
3	1.7163	1.1057	0.9060	**0.1189**	0.1924
4	0.9428	0.5701	1.1262	**0.5392**	0.6359
5	1.4766	1.1515	0.9817	**0.1062**	0.2599
6	0.7424	**0.4260**	0.4407	0.6106	0.5633
7	1.2194	1.3411	2.1578	1.3028	**1.0181**
8	1.070	1.5949	0.7294	**0.6226**	0.7187
9	2.0271	0.9542	1.3424	0.2953	**0.2811**
Average MSE	1.5183	1.1365	0.8993	**0.4635**	0.4846
Average Rank	4.44	3.67	3.22	**1.78**	1.89

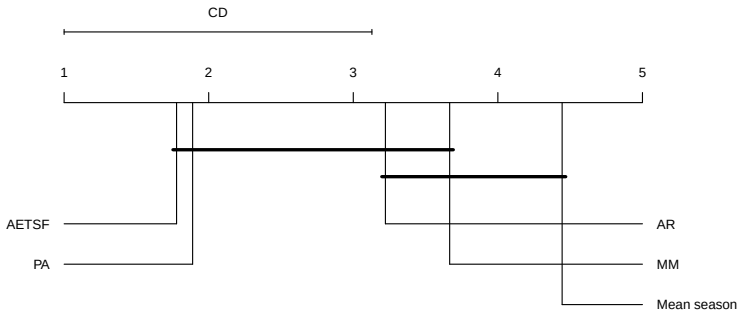

Fig. 6. Critical diagram of the comparison between different prediction approaches.

6 Conclusion

We have proposed in this paper a study of a framework for seasonal time series prediction. This framework involves three steps: the clustering of the seasons into groups, the prediction of the group of the next season using a classifier, and finally the prediction of the next season. One advantage of such a framework is that it is able to produce predictions at the horizon of one season in one shot (*i.e.*, there is no need to build different models for different horizons). We have provided a comparison of different clustering algorithms that can be used in the first step, and a comparison of different classifiers for the second step. Experiment results show that our proposed approach is competitive with other time series prediction methods. For future work, an important point will be to focus on the performance of the classifier. We think that including more exogenous data (example: holidays, social events, weather, etc.) might improve the performance of the classifier, thus the global performance of our framework.

References

1. Boullé, M.: Functional data clustering via piecewise constant nonparametric density estimation. Pattern Recognition **45**(12), 4389–4401 (2012)
2. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)
3. Brockwell, P.J., Davis, R.A., Calder, M.V.: Introduction to time series and forecasting, vol. 2. Springer (2002)
4. Cerqueira, V., Torgo, L., Pinto, F., Soares, C.: Arbitrated ensemble for time series forecasting. In: Proceedings of the Joint European conference on machine learning and knowledge discovery in databases. pp. 478–494 (2017)
5. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: STL: A seasonal-trend decomposition. Journal of official statistics **6**(1), 3–73 (1990)
6. Cuturi, M.: Fast global alignment kernels. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 929–936 (2011)
7. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM (1999)
8. Ghysels, E., Osborn, D.R., Rodrigues, P.M.: Chapter 13 forecasting seasonal time series. Handbook of Economic Forecasting, vol. 1, pp. 659–711 (2006)
9. Hyndman, R.: Time series data library (TSDL) (2011), `http://robjhyndman.com/TSDL`
10. Jakaša, T., Andročec, I., Sprčić, P.: Electricity price forecasting – ARIMA model approach. In: Proceedings of the International Conference on the European Energy Market (EEM). pp. 222–225 (2011)
11. Kavasseri, R.G., Seetharaman, K.: Day-ahead wind speed forecasting using f-ARIMA models. Renewable Energy **34**(5), 1388–1393 (2009)
12. Leverger, C., Lemaire, V., Malinowski, S., Guyet, T., Rozé, L.: Day-ahead time series forecasting: application to capacity planning. In: Proceedings of the 3rd workshop on Advanced Analytics and Learning of Temporal Data (AALTD) (2018)
13. Lloyd, S.: Least squares quantization in PCM. Transactions on information theory **28**(2), 129–137 (1982)
14. Melbourne, C.O.: Pedestrian counting system (2016), `http://www.pedestrian.melbourne.vic.gov.au/`
15. Pai, P.F., Lin, C.S.: A hybrid ARIMA and support vector machines model in stock price forecasting. Omega **33**(6), 497–505 (2005)
16. Paparrizos, J., Gravano, L.: k-shape: Efficient and accurate clustering of time series. In: Proceedings of the International Conference on Management of Data (SIGMOD). pp. 1855–1870 (2015)
17. Petitjean, F., Ketterlin, A., Gançarski, P.: A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition **44**(3), 678–693 (2011)
18. Vahidinasab, V., Jadid, S., Kazemi, A.: Day-ahead price forecasting in restructured power systems using artificial neural networks. Electric Power Systems Research **78**(8), 1332–1342 (2008)
19. Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing **50**, 159–175 (2003)