International Journal of Environment and Geoinformatics 10(2):001-011 (2023)

Reaserch Article

Evaluation of Various Machine Learning Methods to Predict Istanbul's Freshwater Consumption

Mustafa Hekimoğlu*, Ayse Irem Cetin, Burak Erkan Kaya

Kadir Has University Department of Industrial Engineering, 34083 Cibali, Istanbul, TURKIYE

* Corresponding author: M. Hekimoğlu
E-mail: mustafa.hekimoglu@khas.edu.tr

Received 24.01.2023
Accepted 27.03.2023

How to cite: Hekimoğlu, et al. (2023). Evaluation of Various Machine Learning Methods to Predict Istanbul's Freshwater Consumption. *International Journal of Environment and Geoinformatics (IJEGEO)*, 10(2):001-011, doi. 10.30897/ijegeo.1270228

Abstract

Planning, organizing, and managing water resources are crucial for urban areas and metropolitans. Istanbul is one of the largest megacities, with a population of over 15 million. The large volume of water demand and increasing scarcity of clean water resources make long-term planning necessary for this city, as sustained water supply requires large-scale investment projects. Successful investment plans require accurate projections and forecasting for freshwater demand. This study considers different machine learning methods for freshwater demand forecasting for Istanbul. Using monthly consumption data provided by the municipality since 2009, we compare forecasting accuracies of ARIMA, Holt-Winters, Artificial Neural Networks, Recursive Neural Networks, Long-Short Term Memory, and Simple Recurrent Neural Network models. We find that the monthly freshwater demand of Istanbul is best predicted by Multi-Layer Perceptron and Seasonal ARIMA. From the predictive modeling perspective, this result is another indication of the combined usage of conventional forecasting models and novel machine learning techniques to achieve the highest forecasting accuracy.

Keywords: Water Management, Machine Learning, Neural Networks, Autoregressive Models

Introduction

Increasing water demand, accompanied by resource scarcity, is one of the main critical problems of today's societies. Water is not only vital for all living organisms, but it is also crucial for sanitation, agriculture and almost all other economic activities. From the urban water management perspective, the water demand of a city is mainly driven by the size of its population, climatic conditions, distribution and wastewater collection infrastructure, and the level of socio-economic development of a city. Water demand forecasting is the primary input for urban water management studies for all cities (Essien et al., 2019). Water management and planning are highly challenging to megacities like Istanbul, and the results are more sensitive to forecast errors in future water demand estimation.

Istanbul is suffering from water scarcity due to the rapidly increasing population. As Turkey's most populous and industrious city, Istanbul offers many opportunities to locals, refugees, and tourists, such as education, jobs in various areas, and tourism. This stimulates the population growth of the city. Additionally, climate change affects precipitation rates in the surrounding areas of the city, and that is directly related to water levels in the reservoirs and dams (Gazioğlu et al., 1998; Yücel, et al., 2002). All of these factors compel decision-makers to take necessary precautions to overcome Istanbul's water management problems, and a reliable water demand prediction is the key input to designate the water management policy (Goksel, et al., 2006; Burak et al., 2021).

In this paper, we consider Istanbul's medium-term water forecasting problem. Using the historical water consumption data provided by Istanbul Water and Sewerage Administration (ISKI), we employ classical forecasting models, Holt-Winters, Seasonal Autoregressive Integrated Moving Average (S-ARIMA) and Naive forecasting, as well as two different types of Artificial Neural Network methods to obtain consumption predictions for the city. Our results show that among the great variety of models considered for the monthly consumption data of Istanbul, the best result is provided by SARIMA whereas the second-best performance is obtained from Multi-Layer Perceptron (MLP) with our parameterization.

Consideration of classical and novel methods of forecasting freshwater demand forecasting of Istanbul is the main contribution of this study. Interestingly, classical and novel methods perform very close to each other for monthly freshwater demand, indicating the importance of considering existing methods together with the novel machine learning methodologies from the literature (Celik and Gazioğlu, 2022).

This paper consists of six sections. Section 2 briefly reviews the literature and some applications of various machine learning methods used for water demand forecasting. The data set is introduced and explained in Section 3. In Section 4, a detailed presentation of

forecasting models is presented. Validation and verification of our models are given in Section 5. Results and discussion are in Section 6, and the conclusion is given in Section 7.

Water demand forecasting has long been studied, and researchers consider different models and methods for the problem. In the literature, different regressors are considered for predicting water demand in the future. Specifically, Adamowski (2008) used climatic variables, past water demand, and population variables with linear regression models, time series models (ARIMA), and Artificial Neural Networks (ANNs) models. More specifically, the maximum daily temperature, the daily amount of rainfall, and population variables were used in the model (Adamowski, 2008).

Recent papers in this area mostly compare classical models with the current ANN models or just focus on the ANNs and their variations. ANN model was applied by Liu et al., (2003) predict water demand in Weinan City with these parameters, water consumption per month, water and wastewater prices in effect, household size, and household income. Tiwari, and Adamwoski (2013) used Wavelet Neural Networks and Bootstrap Neural Networks on average daily and monthly demand, maximum temperature, and total precipitation parameters and compared their performance with ARIMA and ARIMAX. Caiado (2010) applied classical models such as ARIMA, Holt Winter (HW), and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) with double seasonal univariate time series data and examined the performance of the daily water demand in Granada, Spain.

In addition to classical models and neural networks, machine learning models are used for water demand forecasting in the literature. Smolak et al., (2010) utilize Support Vector Machine (SVM), tree-based models, Random Forest and Extra Trees, and Blind Approach. They compared the models' performance with ARIMA models on seven-day and 24-hour forecasts. Chen (2011) utilizes a Least Squared Support Vector Machine (LS-SVM), a modified version of SVM, and examined its performance on the hourly water demand forecasts with Back Propagate Neural Networks. Bata and Carriveau (2020) consider two different ANNs models, and Nonlinear Autoregressive with Exogenous (NARX) model with historical demand. They examine performances on short-term water demand forecasting (Bata et al., 2020). Altunkaynak et al. (2004) studied TS Fuzzy time series analysis on nine years of monthly water consumption records of Istanbul city.

Moreover, deep learning models such as RNNs are used to forecast water demand models. Guo et al. (2018) utilize ANNs, SARIMA, and the GRUN models. GRUN consists of Gated Recurrent Units, a type of RNN, as the core. GRUN is built on three layers of GRU, a layer of merge layer, a part of many dense layers, and a connection module. Rectified Linear Unit (ReLu) and linear functions were used as activation functions. The model used two water demand datasets collected from different parts (residential or industrial) of the city of Changzhou in China. They examined the performances of the models on two datasets by using 15-minute and 24-hour forecasting periods (Guo et al., 2018). Mu et al. (2020) examined LSTM with ARIMA, support vector regression SVR and Random Forest (RF) models based on predictions of hourly and daily water demands for Hefei City in China. Hu et al., (2019) discussed Convolutional Neural Network (CNNs) and Bidirectional LSTM (Bi-LSTM) hybrid model over CNNs, LSTM, Bi-LSTM, CNN-LSTM and Sparse Autoencoder (SAE) models on urban water demand prediction with historical urban water demand data and meteorological data. Savun-Hekimoğlu et al. (2021) consider an ARIMA model for forecasting the water demand of Istanbul. In our paper, we consider many of the methods in the literature for forecasting the monthly consumption of Istanbul to obtain medium to long-term water demand predictions for the city. To the best of our knowledge, classical and novel methods have never been compared in a water demand forecasting study with monthly consumption data before.

Dataset

In our study, we considered monthly consumption data provided by the open platform of Istanbul Metropolitan Municipality. The dataset includes the monthly freshwater consumption levels of Istanbul between 2009 and 2019. In total, the dataset consists of 132 rows. After a careful review, we decided to remove the value of December 2019 due to inaccurate measurement. The resulting dataset consists of 131 point-univariate time series. In Table 1, descriptive statistics of the dataset are provided.

Table 1. Descriptive statistics of 131 Months of Freshwater Consumption.

Descriptive Statistic	**Value (1000*m^3)**
Mean	76579.1
Standart Deviation	**10269.3**
Minimum	50838.0
Lower Quartile	**68021.5**
Middle Quartile	77285.0
Upper Quartile	**84256.5**
Maximum	96028.0

Seasonality and trend analysis of the water consumption data of Istanbul reveals that water demand has increased over the recent years, and there is a significant seasonality in the dataset. Seasonality analysis is performed with *stl* routine in R Gui. This routine decomposes a time series into its trend and seasonality components using locally polynomial regression fitting (Cleveland et al., 1990). We assume additive model for this decomposition as it is simpler and more effective compared to the multiplicative one. The result of the seasonality decomposition is presented in Figure 1. The increasing trend can be explained by the increasing urban population in Istanbul, whereas cyclic behaviour is mainly due to the effect of temperature. The non-

stationarity of the consumption time series is checked with Augmented Dickey-Fuller Test (ADF). We applied the ADF test using R-Gui and observed that the p-value is close to 1, which shows that our dataset is non-stationary.

Forecasting Water Demand

To obtain water demand forecasts for Istanbul, we examined classical models, (ARIMA and Holt Winter's (HW)), machine learning models (Naïve Bayes), and deep learning models (ANNs and RNNs). The classical models are traditional prediction tools for the time series forecasting because they are easy to use and understand the statistics behind them. More importantly, these tools provide satisfying results. Machine learning and deep learning models present new approaches to time series problems, and they outperform the conventional methods especially in short-term forecasting problems.

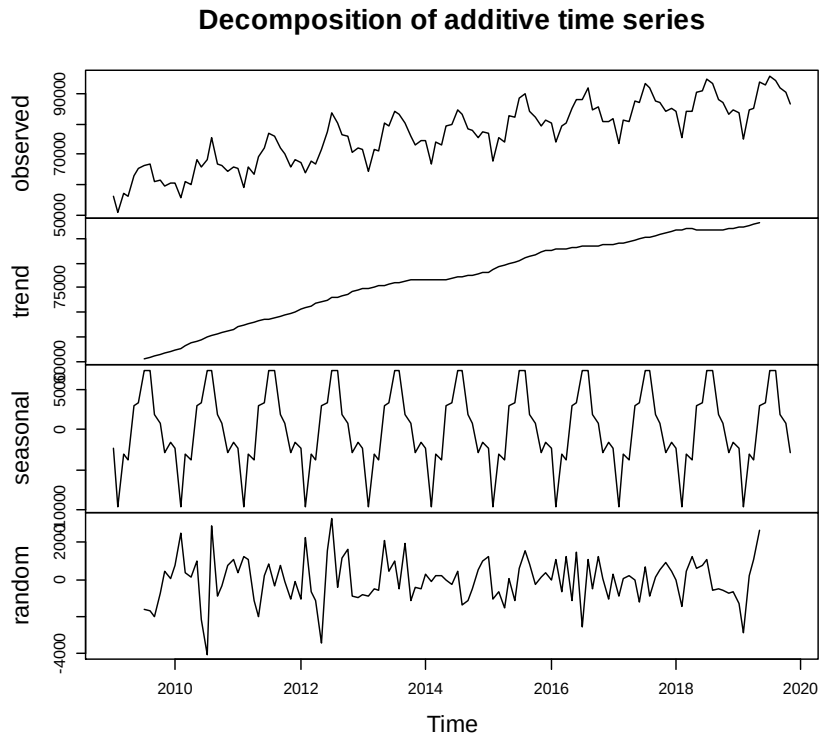

Fig. 1. Decomposed representation of the dataset

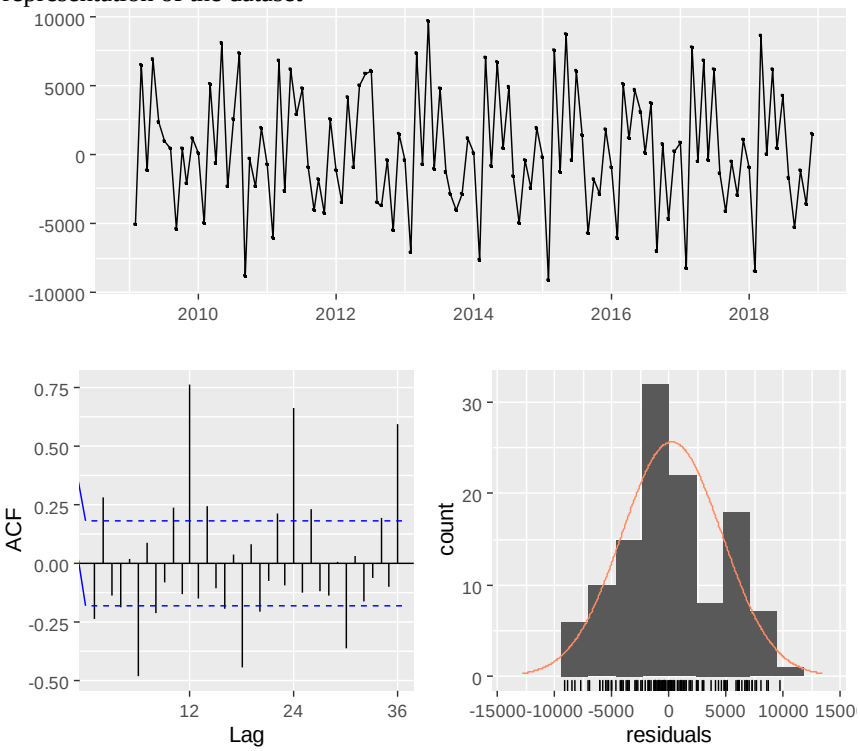

Fig. 2. Autocorrelation function and residual plot.

In contrast to the classical models, these models have more complex mathematical structures, and they can signal issues such as overfitting. Tuning these input parameters and finding the best-fit combination is another optimization problem and require more work time than classical models (Hekimoğlu, 2022). Detailed explanations about the applications of the models can be founded in the appendix. In this section, we first provide a detailed presentation of the classical forecasting models and their applications to water demand prediction for Istanbul. Afterwards, we discuss the application of machine learning techniques for the problem.

When interpreting the reliability of a model Mean Squared Error (MSE), Mean Absolute Error (MAPE), and Root Mean Square Error (RMSE) are used frequently. The mathematical definitions of these accuracy measures are provided in Equations (1-3)

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(\mu - \varepsilon_t)^2, \quad (1)$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}(\mu - \varepsilon_t)^2 \left|\frac{\mu - y_t}{\mu}\right|, \quad (2)$$

$$RMSE = \sqrt[2]{\frac{1}{n}\sum_{t=1}^{n}(\mu - \varepsilon_t)^2}\,. \quad (3)$$

Throughout this study, MSE, MAPE, and RMSE values are also used to measure the performances of forecasting methods. For a reliable method, these values are required to be as much as small. MSE, MAPE, and RMSE values for the naïve method are 251044254, 0.336, and 15844.38, respectively; these can be thought of as very high values intuitively.

Classic Forecasting Models

Autoregressive Integrated Moving Average

The Autoregressive Integrated Moving Average model (ARIMA) is a standard model that is used in statistics, economics, and particularly time series analysis. This model has been a lead model in many areas of time series forecasting for over a half-century. The model is a generalization for the autoregressive (AR) and moving average (MA) methods. These models are suitable for understanding the data and predicting future points. Although models require stationary series, non-stationary series can be transformed into stationary series using the differencing method. Differencing can be defined as computing the difference between consecutive observations, which is also called the random walk method. Differencing helps to stabilize the mean of the series by removing shifts in the data and reducing trend and seasonality. ARIMA is the linear combination of AR and MA models. In AR models, the prediction of the values depends on the linearly combined past values of variables. In Equation (4) AR is represented, where ε_t is error (white noise) value, y_t is predicted value, c is the mean difference between consecutive observations, and ϕ_t is the predictor such like multiple regression. For AR, finding the best p value is essential. p value is called as order of the AR, and AR(p) is called the autoregressive model of order p.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t. \quad (4)$$

Structurally MA is similar to AR. The main difference between the two models is that AR considers previous forecasting errors as predictors instead of the previous values. MA model is presented in Equation (5). As in AR models, finding the right number of predictors is an essential task for building accurate model while avoiding the risk of overfitting. q value is the order of the MA and MA(q) is moving average of order q.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}. \quad (5)$$

Combining differencing with AR and MA, the non-seasonal ARIMA is provided. This combination is linear and represented in Equation (6) where y'_t represents the differenced data. In ARIMA, the integration order, d, should be considered. In ARIMA (p, d, q), p is the number of the lagged terms of y_t, q is the number of lagged terms of ε, and d is the degree of the differencing.

$$y'_t = c + \sum_{j=1}^{p} \phi_j y'_{t-j} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t. \quad (6)$$

By using features explained above, ARIMA can be used for seasonal data. At this point, ARIMA has a new parameter called m, which represents the seasonal part of the model. ARIMA (p, d, q) $(P, D, Q)_m$ represents the seasonal ARIMA (Ho et al., 2002; Contreras et al., 2003; Stevenson, 2007).

Because the handled time series in this study is non-stationary, we applied the seasonal ARIMA. Firstly, the best $p, d,$ and q parameters are found, then Akaike's Information Criteria (AIC), corrected AIC (AICc), and Bayesian Information Criteria (BIC) values are used to check the model. AIC is an assessment method of the goodness of the model in the prediction period and is a useful method in selecting the process of the predictors and order parameters. Additionally, BIC or AICc can be used for the same purpose. We inspected these values of models to observe whether there is a change. Auto-ARIMA function in the python programming language is used to find the best-fitting values of these parameters. A graphical representation of residual values of the resulting ARIMA model is given in Figure 2. Further details on the parameter estimation of the ARIMA model are given in Appendix. We also performed manual search methods for these parameters. More than one well-resulted parameter value was founded in the manual

search, and cross-validation was performed with these parameters.

Holt-Winters

Holt-Winters' model (HW) is a common and simple projection method to deal with trend and seasonality. HW depends on a forecasting equation and three smoothing equations for level (ℓ_t), trend (b_t), and seasonal component (s_t) with corresponding smoothing parameters α, β^* and γ. Additionally, it denotes the frequency of seasonality. HW deals with seasonality with two different methods. The additive method (AM) is preferred when seasonal variations are constant through the series, and the multiplicative method (MM) is used when seasonal variations change proportionally to the level of the series. While the seasonal component in AM is expressed as absolute values, it is expressed as relative terms like percentage in MM (Chatfield, 1978; Chatfield and Mohammed, 1988). AM is presented in Equations (4-6) and MM is represented in Equations (7-10).

$$\hat{y}_{t+\mathrm{h}|\mathrm{t}} = l_t + \mathrm{h}b_t + s_{t+h-m(k+1)}. \quad (7)$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}). \quad (8)$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}. \quad (9)$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}. \quad (10)$$

We applied HW model on the time series data. Firstly, we built the model with default parameters (α=0.3, β=0.1, and γ=0.1 are used when starting optimization and HW function in R programming language to detect optimal α, β, and γ values for optimization) for our data. Then we performed the cross validation to control the goodness of the fit to the model and different data points.

$$\hat{y}_{t+\mathrm{h}|\mathrm{t}} = l_t + \mathrm{h}b_t + s_{t+h-m(k+1)}. \quad (11)$$

$$l_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}). \quad (12)$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}. \quad (13)$$

$$s_t = \gamma\frac{y_t}{(l_{t-1}+b_{t-1})} + (1-\gamma)s_{t-m}. \quad (14)$$

Artificial Neural Networks

Artificial neural networks (ANNs) are the most used deep learning concept. They can be easy to use and understand. Most importantly, they can be adapted for many problems, such as linear-nonlinear models or supervised-unsupervised problems. The idea underlying ANNs is simply the working principle of the human brain and nervous system. ANNs learn and make meaningful assumptions from examples, even if there are errors in the input data (Falah et al., 2019). ANNs are computational models with varied degrees of complexity (Hassoun and Hassoun, 1995). ANNs consist of one input layer, one or more hidden layers, and one output layer. These layers can contain more than one neuron, affecting the model's complexity. The idea of mimicking real neurons was proposed by McCulloch and Pitts (1943). They defined computational neurons with binary threshold unit. The mathematical neuron computes the weighted sum of n inputs and returns 1 if the output of the sum is above a certain threshold value, otherwise 0.

As can be seen in Equation (15), $\theta(.)$ is the unit step function, and w_k is the weight of the k^{th} input value. The last term threshold μ represents another weight w_0 that attached to constant input $i_0 = 1$.

$$y = \theta(\sum_{k=1}^{n} w_k i_k - \mu). \quad (15)$$

ANNs provide many advantages and efficient solutions when traditional methods fail and become impractical. Requiring less formal statistical training, the ability to implicitly detect complex nonlinear relationships between dependent and independent variables, the ability to detect efficient interactions between predictor variables and the availability of multiple training algorithms can be counted in these advantages. When there is a large data set, ANNs give better results in finding patterns in the data set than standard methods. The nonlinear nature of the ANNs offers better results against complex problems than linear techniques. Additionally, identification and learning correlated patterns between input and target values are possible for ANNs, and they are useable to predict the output of new inputs.

Even though ANNs have many advantages, there are still some disadvantages that should be considered. Because of the structure, ANNs have a black box and empirical nature. They can be burdened computationally and prone to overfit. Input-output table of an ANNs model can be without a solid analytical basis because the relationship between input and output variables is not developed by theoretical judgment. The overtraining problem can occur to create a model well in unseen inputs. This causes an over-complex and over-specified model and needs the capacity of the network to exceed free parameters.

It is possible to differentiate neural networks and use different types of them by not only adding multiple hidden layers, nodes and changing functions. In this manner, three types of neural networks are studied and used by Jain et al. (2016). We applied these models to our data with different parameters and observed the reactions of these changes.

Feed-Forward Neural Network

One of the most used types of neural networks is feed-forward neural networks (FFNNs). FFNNs consist of sequential layers of function architecture. In this structure, the outputs of the current layer are inputs of the next layer. They can be single-layered or multi-layered forms based on model design. Single-layered networks are called shallow neural networks (SNNs); meanwhile, multi-layered networks are called deep neural networks (DNNs). SNNs have a simple structure and activation functions, or step activation functions of the nodes; the multiplication of inputs and weights feeds those. In Figure 3, a generic structure for an FFNN model is presented. In general, multi-layered neural network models might have complex architecture,

whereas single-layer neural networks have only one hidden layer between inputs and output (Figure 4). FFNNs is easy to use and maintain, fast and not too complex. However, it is not powerful enough when the problems get complicated because of the lack of dense layers and backpropagation (Tang and Fishwick, 1993).

Various numerical experiments were made to find a model that gives the best results. Firstly, we convert time series data to supervised data because ANNs models need supervised data. We created a supervised data matrix with predictions and real values from the time series data. This matrix is created by shifting the data points one by one. Thus, the previous data point is the input parameter of the model for the current state, and the current data point is the real value to validate the model's output. For the second step, we split the data into train and test sets and built the first model with logarithmic values of the training set. We did experiments on the different models with the data matrix, which we produced in the previous step. Then we performed a parameter search for the number of hidden neurons, decay, the number of iterations, and initial weights.

Next, we applied similar searches on the dataset that shifted by two, three and four points, we compared the fourth parameter search results and observed that eight hidden neurons model with shifted data by three points gave the best result out of these results. Additionally, we applied Box-Cox Transformation to the matrix. We experimented with the models on the transformed data matrix and observed the effects of the transformation on the models.

Multi-Layer Perceptron

Perceptron is the earliest and the simplest version of the ANNs forms. A perceptron forms a single neuron that may have input values and return an output. On the other hand, a multilayer perceptron (MLP) is a more complex structure that can be built on a different number of layers and neurons. MLP is a special type of FFNNs. MLP and FFNNs are fully connected models; any neuron in a layer is connected to all neurons in the next layer. In some definitions, MLPs are defined with the same number of neurons in each hidden layer and with the same activation function across the hidden layers. As shown in Figure 3, weighted input layer values are presented to hidden layers to feed activation functions. Both forward and back propagation are usable for MLP.

Due to the fully connected structure and backpropagation method, MLP is a powerful solution for deep learning problems. However, there are a couple of points that should be considered. One of them is the number of layers and neurons that can affect the speed of the algorithm. Other is the fully connected structure that can be complex and cause maintenance issues.

Speech recognition and complex classification are some problems that can be solved by MLP. We applied MLP in a similar manner to ARIMA and HW. We did not convert time series data to a matrix because the model requires time series data. After splitting the data to train and test sets, we built the first examples of the MLP model and did experiments with them. In these experiments, we observed models' behaviours while changing the number of layers and neurons in each layer. We tried to use deterministic seasonality dummies parameter in our models and observed its effects on the models. We observed 'trg' method for adding seasonality dummies that gave the best results. Additionally, we performed a parameter search on the models and validated the results by using the cross-validation method.

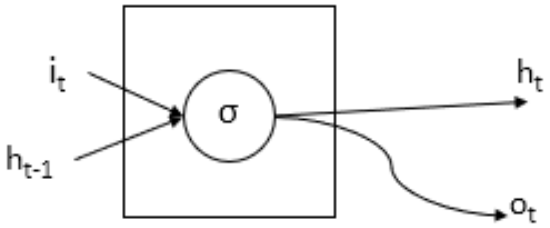

Fig. 3. A Generic Structure of Feed Forward Neural Network (FFNN) Model.

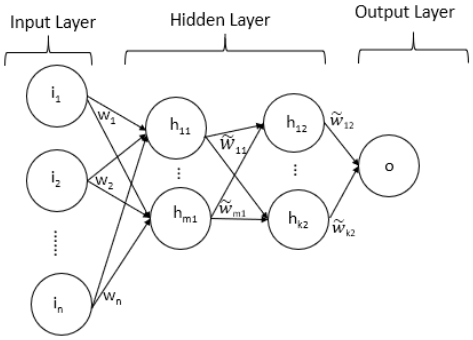

Fig. 4. Structure of a RNNs unit.

Recurrent Neural Networks

Recurrent neural networks (RNNs) are another type of artificial neural network. RNNs are designed by considering the detailed analogy of the brain modules. The primary things that separate from each other are simply the memory property of the RNNs. In the structure of RNNs, there are internal loops that provide to remember output from the previous neurons. These feedback loops are the characteristic feature of the RNNs and make them powerful computational models and universal approximators (Xia et al., 2018). Although ANNs are designed to take a fixed size of the input vector, RNNs can analyze data streams with no predetermined limit and process variable lengths of sequences or even infinite lengths of sequences (DiPietro and Hager, 2020). RNNs can perform well against problems such as speech recognition, sentiment analysis, and image processing. However, RNNs have some modeling weaknesses and computational issues, such as vanishing gradient problems. Gradient descent is an optimization algorithm that is used to find the global minimum of a differentiable function.

RNNs are using this function to find global minimum of the cost function and update the weights for setting up the optimal network. Finding lower gradient takes more

time for the algorithm, and it is more difficult for the model to finalize the results. This problem called vanishing gradient problem and affects to model training negatively. In Figure 5, the structure of a RNN unit is depicted. Long short-term memory units (LSTM) overcome these weaknesses and improve the model. In the structure of the LSTM, there are memory blocks that include memory cells. These self-connected cells store temporal network states with special multiplicative units. These units are called gates. Every memory block includes two types of gates; input and output gates. Input gates control the flow of inputs toward memory cells, whereas output cells control the output flow of cell activations toward the rest of the network. Afterward, forget cells are added to the memory block. Input modulation gates provide a range to the state using the hyperbolic tangent activation function and allow the cell state to forget the memory. This may turn into a weakness and prevent LSTM models from processing continuous input streams (Sak et al., 2014).

Self-recurrent connections of the cell add the internal states as input to the cell after it is scaled by a forget cell; thus memory of the cell resets by forget cell. Peephole connections were added to the architecture of the modern LSTM. Peephole connections learn the exact timing of the outputs from internal cells to the gates of the same cell. LSTM solves the vanishing gradients problem of the RNNs. Time and memory are essential needs for LSTM. Because of the architecture of the model, they need high memory bandwidth. Additionally, they tend to overfit training data.

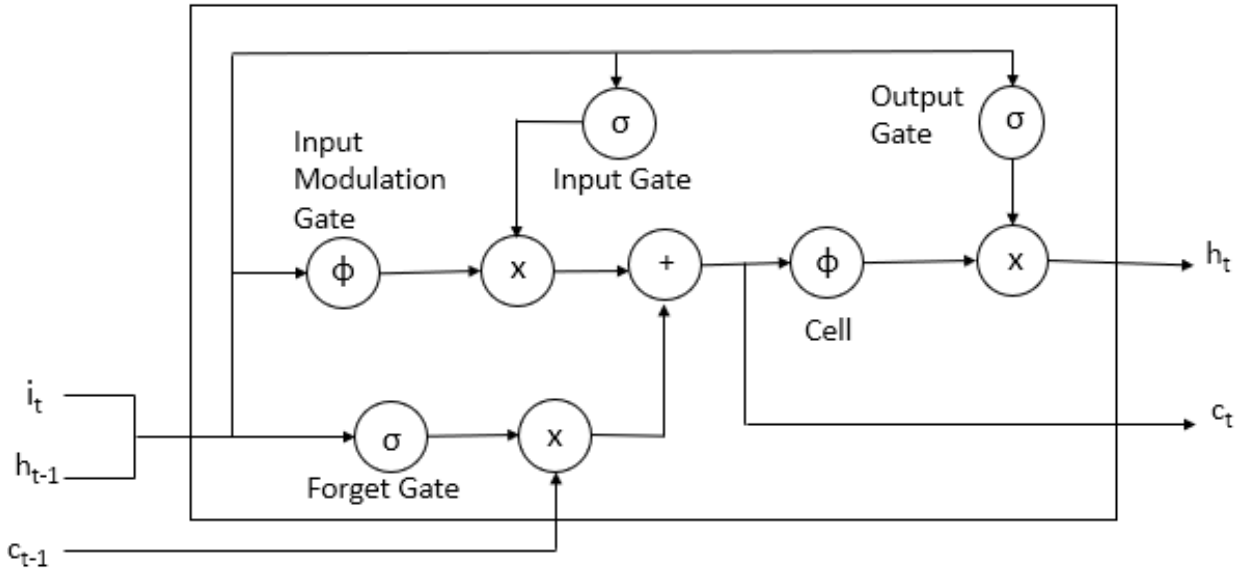

Fig. 5. Structure of an LSTM Unit

Firstly, to be able to apply RNNs models, we convert time series data to the data matrix. This step is required as RNNs models do not work with time series data. Once data is converted to the required shape and scaled selected range, we built the first examples of simple RNNs and LSTM models.

In the early stages of our experiments, we try a different number of time steps as input to investigate the model's behaviour. We create a new data matrix with the best time step value and use it as a time series generator in the next steps. Next, we perform the parameter search on created models to calculate the number of layers and neurons, epoch, batch size activation function, loss function, and optimizer.

Activation functions determine the output of the neurons. In our study, we observed that ReLu activation function fits better than the other functions in the literature. ReLu function returns the maximum number between two numbers that can be shown mathematically as $\max(0, x)$.

In the neural networks, optimizers are used ,to optimize the loss or cost functions to evaluate the predictions of a model mathematically. Predictions are evaluated based on their differences from the real values Among many defined loss functions in the literature, we use mean squared error loss function in our study. Mean squared error function (MSE) is represented mathematically as follows

$$: \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2.$$

In the training process of parametric models, optimizers search over a parameter space to minimize the loss function. In our study, we used Adaptive moment estimation (ADAM) optimizer, an extension of the classical stochastic gradient descent method. ADAM uses first and second-order moments. We observed that functions explained before affected our models better than other alternative combinations in our case (Kingma and Ba, 2015).

After the parameter estimation, we perform cross validation on both simple RNNs and LSTM models. We observe that SRNN with ten hidden neurons in a single layer, gave the best results, and the LSTM model, including three and two neurons in two layers, gave the best results. To avoid the risk of overfitting, we add a dropout regularization parameter to our model. Dropout

regularization provides neurons randomly in each training session with a given probability distribution. In our model training studies, we find that adding a dropout regularization parameter does not affect the results. We apply a differencing method to the data to reach a more effective model. With differencing, we aimed to remove the trend from the data. After this step, we converted new stationary data to supervised data by shifting one or two data points and followed by a normalization step. We build models with same functions, batch size and epoch values.

Validation and Verification

In the literature, different validation techniques are suggested to validate machine learning and deep learning models and verify the results. k-Fold Cross Validation, Leave One Out Cross Validation are only two examples of the most used techniques. In this study, we used cross validation techniques using the original dataset.

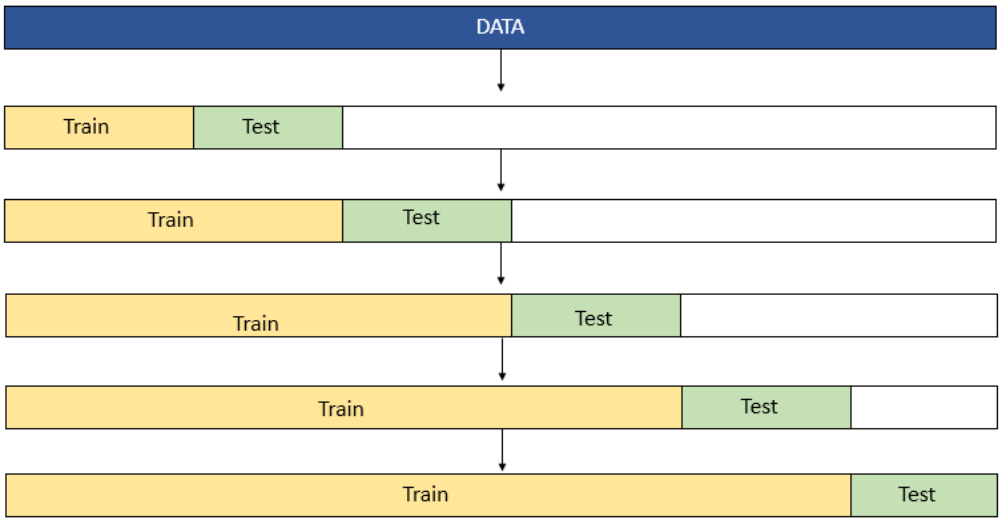

Fig. 6. Time series Cross Validation Process Chart

k-Fold CV technique splits the data into k groups and iteratively use a group as a test set while keeping others in the training set. This process is repeated k times until every k^{th} part left as test. Once this process is complete, the average of these k results represents the predictive power of the model. In time series cross validation, same process is applied with a little adaptation. In the time series cross validation, model is trained with previous observations. As shown in Figure 6, orange parts represent the training sets and pink parts represent test sets (Altunkaynak et al., 2004). In this study, we used 5-fold time series cross validation on our models to verify the results.

Results and Discussion

In Table 2, we summarize performances of the studied models. As shown as in the table, MLP is the best performing model while SARIMA has the second-best performance. Good performance of MLP models can be attributed to its capability of modeling seasonal variations. Similarly, SARIMA model is a higher-dimension version of the ARIMA model incorporating seasonality coefficients The importance of seasonality for forecasting water demand can be observed from the poor performance of ARIMA models, which omit seasonality.

Additionally, as shown in Table 2, the results of the SRNN and LSTM models are not good. All these models, ARIMA with Box-Cox transformation gave the worst results. In our experimentations with different model structures, we consider adding dropout regularization parameter to RNN models to improve their forecast accuracy. Our results reveal that the forecast accuracy of LSTM can be marginally improved with regularization whereas SRNN's forecast results are rather insensitive to this modification.

Table 2. Results of the models.

Cross Validation k=5				
Model	**Detailed Description**	**MAPE**	**MSE**	**RMSE**
MLP	3 layers	0.018	3.8 E+06	1902.9
SARIMA		0.019	3.9 E+06	1857.1
HW		0.022	5.1 E+06	2111.3
ANNs	Box-Cox Transform.	0.043	1.8 E+07	4066.4
SRNN		0.047	2.2E+07	4435.4
SRNN	Dropout regularization	0.049	2.9E+07	4998.6
ANNs		0.058	6.9E+07	6151.1
LSTM	2 layers	0.06	3.8E+07	5776.6
SRNN	2 layers	0.064	4.0E+07	6052.2
LSTM	Dropout regularization	0.087	6.9E+07	7851.2
LSTM		0.11	1.1E+08	9545.1
LSTM	Differencing by 1	0.112	1.3E+08	10068.6
LSTM	2 layers and differencing by 1	0.112	1.3E+08	9912.4
ARIMA	Box-Cox Transform.	0.123	1.0E+08	10150

Conclusion

Istanbul is a megacity including residential, industrial, and agricultural areas, housing more than 16 million people. In addition, the city experiences occasional droughts with increasing frequencies mainly due to climate change. Therefore, it is important to make medium- and long-term plans for ensuring the city's water supply by utilizing the right combinations of infrastructure investments. Planning such investment activities requires accurate forecasts for the city's future water demand.

In this paper, classical models such as ARIMA and HW, and deep learning models such as ANNs, MLP and RNNs models are investigated for forecasting water consumption of Istanbul. ARIMA, HW and MLP require input data as time series, while ANNs and RNNs require input in a matrix form. We train our models by using approximately 95% of the entire data set. After finding satisfying parameters for each model, five-fold cross-validation is applied. Additionally, ARIMA and ANNs models are combined with the Box Cox transformation. We observed that RNNs and LSTM models tend to return inconsistent results. We verified that this instability is mainly due to the non-stationary characteristic of the data. Differencing method is used to alleviate the effect of non-stationarity.

The results of cross-validation were examined for every model. Our results reveal MLP and SARIMA models have successful forecasting accuracy as they can capture the increasing trend and seasonality of water consumption levels. In addition, LSTM and RNNs have reasonable performance on the training set, whereas they perform poorly in cross-validation. After the differencing method, LSTM models produced more stable results. To our knowledge, the literature consists of studies focusing on short-term predictions with classical or deep learning models. In this paper, we consider classical and novel models for the long-term prediction of the freshwater demand in Istanbul. Our predictive models are trained on the data set of monthly freshwater consumption of the city. We observe that a classical forecasting model and a novel machine learning technique outperform the other models. This result strongly indicates the potential of utilizing statistical models and machine learning techniques to achieve the best result.

References

Adamowski, J. F. (2008). Peak daily water demand forecast modeling using artificial neural networks. *Journal of Water Resources Planning and Management*, *134*(2), 119-128.

Altunkaynak, A., Özger, M., Çakmakci, M. (2005). Water consumption prediction of Istanbul city by using fuzzy logic approach. *Water resources management*, *19*, 641-654.

Bata, M. T. H., Carriveau, R., Ting, D. S. K. (2020). Short-term water demand forecasting using nonlinear autoregressive artificial neural networks. *Journal of Water Resources Planning and Management*, *146*(3), 04020008.

Burak, ZS., Bilge, A.H., Ülker, D. (2021). Assessment and simulation of water transfer for the megacity Istanbul, *Phys. Geogr.*, 43(6): 784-808.

Caiado, J. (2010). Performance of combined double seasonal univariate time series models for forecasting water demand. *Journal of Hydrologic Engineering*, *15*(3), 215-222.

Celik, OI., Gazioglu, C. (2022). Coast type based accuracy assessment for coastline extraction from satellite image with machine learning classifiers, *The Egyptian Journal of Remote Sensing and Space Science*, 25 (1), 289-299.

Chatfield, C. (1978). The Holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *27*(3), 264-279.

Chatfield, C., Yar, M. (1988). Holt-Winters forecasting: some practical issues. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *37*(2), 129-140.

Chen, L. (2011). Genetic least squares support vector machine approach to hourly water consumption prediction. *Journal of Zhejiang University (Engineering Science)*, *45*(6), 1100-1103.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., Terpenning, I. (1990). STL: A seasonal-trend decomposition. *J. Off. Stat*, *6*(1), 3-73.

Contreras, J., Espínola, R. Member, S., Nogales, FJ (2003). *ARIMA models to predict next-day electricity prices*, *18*(3), 1014-1020.

DiPietro, R., Hager, G. D. (2020). Deep learning: RNNs and LSTM. In *Handbook of medical image computing and computer assisted intervention* (pp. 503-519). Academic Press.

Essien, E., Jesse, E., Igbokwe, J. (2019). Assessment of Water Level in Dadin Kowa Dam Reservoir in Gombe State Nigeria Using Geospatial Techniques, *International Journal of Environment and Geoinformatics*, 6(1), 115-130. doi.10.30897/ijegeo. 487885.

Falah, F., Rahmati, O., Rostami, M., Ahmadisharaf, E., Daliakopoulos, I. N., Pourghasemi, H. R. (2019). Artificial neural networks for flood susceptibility mapping in data-scarce urban areas. In *Spatial modeling in GIS and R for Earth and Environmental Sciences* (pp. 323-336). Elsevier.

Gazioğlu, C., Yücel, Z.Y., Doğan, E. (1998). Uydu Verileri İle İstanbul Boğazi ve Yakin Çevresindeki İçme Suyu Havzalarina Genel Bir Bakiş., *Büyükşehirlerde atık su yönetimi ve deniz kirlenmesi kontrolu sempozyumu*. 18-20 Kasım 1998,

Goksel, C., Musaoglu, N., Gurel, M., Ulugtekin, N., Tanik, A., Seker, D. Z. (2006). Determination of land-use change in an urbanized district of Istanbul via remote sensing analysis. *Fresenius Environmental Bulletin*, 15(8 A), 798–805. 4

Guo, G., Liu, S., Wu, Y., Li, J., Zhou, R., Zhu, X. (2018). Short-term water demand forecast based on deep learning method. *Journal of Water Resources Planning and Management*, *144*(12), 04018076.

Hassoun, M. H. (1995). *Fundamentals of artificial neural networks*. MIT press.

Hekimoğlu, M. (2022). Markdown Optimization with Generalized Weighted Least Squares Estimation. *International Journal of Computational Intelligence Systems*, *15*(1), 109.

Ho, S. L., Xie, M., Goh, T. N. (2002). A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction. *Computers & Industrial Engineering*, *42*(2-4), 371-375.

Hu, P., Tong, J., Wang, J., Yang, Y., de Oliveira Turci, L. (2019, June). A hybrid model based on NN and Bi-LSTM for urban water demand prediction. In 2019 IEEE Congress on evolutionary computation (CEC) (pp. 1088-1094). IEEE.

Jain, A. K. M., J., Mohiuddin, KM (1996). Artificial Neural Networks: A Tutorial. *IEEE Computer Society (29)*, 31-44.

Kingma, D. P., Adam, B. J. (2015). A method for stochastic optimization. CoRR. 2014; abs/1412.6980. *ArXiv preprint arXiv:1412.6980*.

Liu, J., Savenije, H. H., Xu, J. (2003). Forecast of water demand in Weinan City in China using WDF-ANN model. *Physics and Chemistry of the Earth, Parts A/B/C*, *28*(4-5), 219-224.

McCulloch, W. S. y Pitts, W (1943), A logical calculus of the ideas immanent in nervous activity. *Bull. of Math. Biophysics*, *5*, 116.

Mu, L., Zheng, F., Tao, R., Zhang, Q., Kapelan, Z. (2020). Hourly and daily urban water demand predictions using a long short-term memory based model. *Journal of Water Resources Planning and Management*, *146*(9), 05020017.

Sak, H., Senior, A., Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

Savun-Hekimoğlu, B., Erbay, B., Hekimoğlu, M., Burak, S. (2021). Evaluation of water supply alternatives for Istanbul using forecasting and multi-criteria decision making methods. *Journal of Cleaner Production*, *287*, 125080.

Smolak, K., Kasieczka, B., Fialkiewicz, W., Rohm, W., Siła-Nowicka, K., Kopańczyk, K. (2020). Applying human mobility and water consumption data for short-term water demand forecasting using classical and machine learning models. *Urban Water Journal*, *17*(1), 32-42.

Stevenson, S. (2007). A comparison of the forecasting ability of ARIMA models. *Journal of Property Investment & Finance*, *25*(3), 223-240.

Tang, Z., Fishwick, P. A. (1993). Feedforward neural nets as models for time series forecasting. *ORSA journal on computing*, *5*(4), 374-385.

Tiwari, M. K., Adamowski, J. (2013). Urban water demand forecasting and uncertainty assessment using ensemble wavelet-bootstrap-neural network models. *Water Resources Research*, *49*(10), 6486-6507.

Xia, Y., Xiang, M., Li, Z., Mandic, D. P. (2018). Echo state networks for multidimensional data: Exploiting noncircularity and widely linear models. In *Adaptive Learning Methods for Nonlinear System Modeling* (pp. 267-288). Butterworth-Heinemann.

Yücel, Z.Y. Gazioğlu, C., Doğan, E., Kaya, H. (2002). Uzaktan Algilama ve CBS/B ile Ömerli Baraji ve Yakin Çevresinin İzlenmesi, *Türkiye'nin Kıyı ve Deniz Alanları IV Ulusal Konferansı Bildiriler Kitabı*

Appendix

A.1 Classical Models

We applied classical models by using R. First, we built auto-ARIMA and ARIMA models. Since the data has a seasonal characteristic, we used the seasonal ARIMA model. To find the (p, d, q) parameters of ARIMA, we run the auto-ARIMA routine in R Gui. The auto-ARIMA performs a manual parameter search on different ARIMA models and compares them using Akaike Information Criterion (AIC). A close investigation into the auto ARIMA results reveals that (p,d,q)=(0, 1, 1) gives the best cross-validation result. As the second model, we applied Holt-Winter's model. We find that the additive method suits more than the multiplicative method to the data.

A.2. Feed Forward Network

We applied feed-forward neural networks by using the nnet library in R. In our application, we first applied data preparation steps. Next, we built shallow feed-forward neural network models and ran parameter searches for them. During the parameter search step, we focused on the number of neurons, decay, and maximum iteration parameters. Also, we set Linout and Hess parameters to true because the expected output is linear. Additionally, we tried different values for the decay and maximum iteration, and initial weights parameters. We determined to use zero for initial weights, and decay is 1e-2 with 1000 maximum iteration. We observed that with those parameters, eight hidden neurons yielded the best results on the shifted data by three points.

A.3 Multilayer Perceptron

We used the mlp function from R to build models. We used the data as time series data without changing. We performed a parameter search for the function. In our application, we used deterministic seasonality dummies. From the type of the deterministic seasonality dummies, the 'trg' type gave the best results, meaning the sin-cosine pair is used. After the cross-validation was applied, we observed that model with three layers with two neurons in each gave the best results among the MLP models and all the other models.

A.4 Recurrent Neural Networks

We used Python and Keras library to build our models and do data pre-processing steps. After building the models, we performed a parameter search for the number of neurons, number of layers, activation function, loss function, and optimizer parameters.

According to Keras documentation, ReLu activation function of the library takes four input arguments. These are x, alpha, max value, and the threshold value. x input argument represents the input tensor or variable, and alpha is a float controlling the input slope below the threshold value. While the maximum value represents the largest value the function returns, the threshold input argument represents a float that the function set the lower values to this threshold value or as default to zero.

ADAM optimizer takes five input parameters. These parameters are learning rate, beta 1, beta 2, and epsilon. The learning rate is also called step size and are used to update the model weights. Beta 1 and beta 2 parameters are used as exponential decay rates regarding to the first and second-order moments. Epsilon is a small number that is used to prevent division by zero. This parameter is an addition to the original algorithm.

In addition to these parameters, the input shape for the input layer is an important parameter and must be indicated. For the first model, we used a time series generator which provides generated matrix from the given time series with determined time steps and batch size. We searched for the time steps and batch size. After that, we determined to use a twelve-time step as a number of inputs and one for the batch size. For the models, the input shape is an important parameter and must be determined in the input layer. The number of inputs should be the same as the generator, and the number of features is determined as one because the data is univariate data. Besides, we observed that larger epoch values did not affect the results noticeably in our parameter search for epoch number, and we used smaller epoch values because of performance concerns. We determined the epoch as twenty. Moreover, for the multi-layered models, the return sequence parameter should be set as true. At last, we observed that single-layered SRNN from SRNN models with ten hidden neurons gave the best result with these parameters, while two-layered LSTM with respectively three and two hidden neurons gave the best results from LSTM models. Because of the difference between general prediction results and cross-validation results, we suspected overfitting, and we added a dropout regularization parameter. We tried different values for this parameter. However, it did not affect our models positively. In general, SRNN model has the best results out of RNN models. However, we observed a high variation issue for SRNN and LSTM models and SRNN tends to be affected more from this issue.

In the second method that we used for only LSTMs, we used differenced data, and we converted the data to supervised matrix type manually rather than using the generator. We changed the data preparation step to get rid of the variation issue. Additionally, we set the stateful parameter to true. Stateful parameters are used to save the current state of the neurons and use it in the next training session. After that, we used mostly the same parameters. We only changed the input shape parameters to batch input shape, which is mostly the same as the previous one. We needed to change because we did not use a time series generator anymore. Instead of defining batch size in the generator, we added the batch size to the input shape parameter. As a result, we observed that single-layered LSTM models gave better results than multi-layered results. The model with two hidden neurons in a single-layered model gave the best results. In addition to the superior performance of the single-layered model, the high variation of the results decreases noticeably. Besides, different shifting points do not yield noticeable effects on the results.