Forecasting natural gas consumption

Abstract: In this study, the results of forecasting monthly natural gas consumption are presented. The developed methodology uses data describing actual natural gas consumption from Romania and state-of-the-art techniques: linear regression, artificial neural networks [MLP], time series [ARIMA, ARIMAX]. The data was augmented with weather [temperature] factors in order to obtain better results. Customers were split in categories and for each category and for each month a model should be created.

Keywords: Forecasting natural gas consumption, predicting natural gas demand, natural gas forecasting models, naïve model, linear regression, artificial neural networks, multi-layer perceptron, time series, ARIMA, ARIMAX.

Introduction

Our task is forecasting natural gas consumption. It is very important for natural gas distribution companies to forecast their customers' natural gas demand accurately. This is due to the fact that such a company has to specify how much quantity of gas their clients need in the next month, quarter and year. If the values provided are too high or too low, a high cost has to be paid.

In our case, the company had data on four localities from Romania: Miroslava, Panciu, Chirnogi, and Odobești. The records were represented by the consumptions of each client in each month. The data from Miroslava included records from September 2015; Odobești – from March 2016; Panciu – from February 2016; Chirnogi – from September 2016.

Related work

This problem has been tackled in various ways, but, as we will see, the main models and input attributes remain the same. In [1], the author proposes an MLP 22-36-1 with calendar [month, day of month, day of week, hour] and weather [temperature] factors in order to predict hourly and daily gas consumption. In [2], an MLP with 20 hidden units is used with input attributes, such as daily minimal and maximal temperatures, the day in week, consumption for previous day, in order to predict daily gas consumption. In [3], beside the MLP approach, there is also mentioned multiple linear regression as another path in forecasting daily demand. The authors write that MLP is better when interpolating [predicting days similar to ones in its training set] and linear regression is good at extrapolating. That is the reason why they decided to take a weighted combination of these models. New input attributes were created: an indicator variable to distinguish between weekdays and weekends, a Friday indicator etc. In [4], there are three approaches mentioned for daily forecasting: linear regression, ARMAX model and ANN [which outperforms the other two]. Input factors include temperature, price of natural gas and number of gas consumers.

Although those papers refer to hourly and/or daily gas consumption, the key techniques are also suitable for monthly or yearly forecasting, as stated in [5]: using linear regression, MLP, AR(I)MA(X) models and including temperature as an additional input attribute. Another summarized description is given below:

Article	Model	Features
[1]	MLP 22-36-1 It is stated that MLP 22-36-1 model can be successfully used to predict gas consumption on any day of the year and any hour of the day.	calendar (month, day of month, day of week, hour) weather (temperature) factors
[2]	MLP 20 hidden units	- historical weather and consumption data - daily minimal and maximal temperatures were included - it is mandatory to add information about day in week in the training set - consumption for previous day is included in the training set 1 input – neuron for forecasted maximal temperature of the forecasting day 1 input – neuron for forecasted minimal temperature of the forecasting day 1 input – neuron for the consumption in previous day 7 inputs – neurons for the identification of the day in the week. A day was identified using 1, 0, or 0.5: - 1 to appropriate input neuron - other inputs are 0 - Sunday and Saturday with 0.5.
[3]	Most common in this context: multiple linear regression and artificial neural networks. - linear regression extrapolates better than ANNs - ANN intrapolates better => weighted combination of these models gives better results	- HDD – Heating Degree Day - HDDW – Heating Degree Days adjusted for Wind - Previous day(s) demand(s) => autoregressive - A binary indicator variable can be added to the model to distinguish weekdays from weekends. - Friday indicator - day-of-the-week DOW variable to represent the fundamental seven day frequency - includes both the demand and the temperature/HDD for the day seven days ago, unless the day seven days earlier was a holiday - Holidays and days around holidays - Holiday adjustments - Data weighting
[4]	OLS regression ARMAX model ANN model This research concluded that the neural network model with backpropagation outperforms - multiple regression,	- temperature - price of natural gas [most important] - number of gas consumers model 1: - consumption - temperature - squared temperature - price

	- neural network NeuroShell, - and the ARMAX model for natural gas forecasting	- number of consumers final model: - temperature - squared temperature - price - number of consumers - 12 lags of today's consumption
[5]	Summary of existing articles related to this topic	

Table 1: Summary of some approaches

Experimental methodology

a. Models

The purpose of our research activities was the development of models that predict the **monthly** [and yearly] gas consumption **for each client**. The main models we have used so far are:

- Models unrelated to time series: linear regression and ANN [more precisely MLP]
- Time series related models: naïve model, ARIMA and ARIMAX.

Linear regression simply assumes that the predicted value is a linear combination of the input attributes:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m + \varepsilon$$

where y is the predicted value, m is the number of input attributes, x_1, ..., x_m are the input values [of m attributes], β_0, β_1, ..., β_m are the parameters to be set, $\varepsilon \sim N(0, \sigma^2)$ [called white noise]. The method used to estimate the *parameters* is called *OLS* – ordinary least squares.

Artificial neural networks [ANN], in general, and **Multi-Layer Perceptrons [MLP]**, in particular, are constructed using a basic unit called neuron. There is a strong resemblance between a neuron and linear regression. More precisely, linear regression is a neuron of a certain kind [in this context, β_0 is called *bias*]. A neuron has a number of inputs and a single output. Each input has a weight which tells the importance of that particular attribute. Then the corresponding linear combination is computed. The difference from the linear regression model is that the output of the neuron is not represented by the linear combination, but by the value of a function [called activation function] applied to that combination. Those neurons are interconnected and the result is an artificial neural network. MLP is a certain ANN architecture: there is an input layer [of neurons], an output layer and 0, 1 or more in-between layers called *hidden* layers. Each neuron in connected to all and only to the neurons in the next layer. The parameters for this model are represented by the weights of each neuron. The method used to estimate the *parameters* is called *BackPropagation [with Gradient Descent]*. An important note is that this method is non-deterministic [because the weights are random initialized] and, as a result, multiple runs should be taken into consideration. This method can be *optimised* using one of the following variations: momentum, RMSProp, Adam, AdaDelta, AdaGrad etc. To avoid *overfitting*, one can make use of regularization techniques [L1, L2, DropOut etc.]. The *hyper-parameters* of this model are: initialization method, number of hidden layers, number of units in a hidden layer, activation functions, learning rate [or other hyper-parameters from optimization algorithms], hyper-parameters for regularization techniques, loss function, number of epochs, batch size [if mini-batch mode is used].

The **time series** models are client-oriented. For each client, the monthly gas consumption was recorded, thus a time series can be generated. The purpose is to *continue* each plot [as shown later, it is useful for a time series to be plotted], that is to predict the consumption in the following h months. If h equals 1, then the **naïve method** tells us that the predicted value is equal to the value recorded that month last year.

The *AR [auto-regressive] model* assumes that the predicted value is a linear combination of the past p values [p is called *number of lags*], that is:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_p y_{t-p} + \varepsilon_t$$

where y_t is the predicted value [at moment t], p is the number of lags, β_0, β_1, ..., β_p are the parameters to be set, $\varepsilon_t \sim N(0, \sigma^2)$ [called *white noise*].

The *MA [simple moving average] model* assumes that the predicted value is a linear combination of the past q [white] noises, that is:

$$n_t = \alpha_0 + \alpha_1 \varepsilon_{t-1} + \cdots + \alpha_q \varepsilon_{t-q} + \varepsilon_t$$

where n_t is the predicted value [at moment t], q is the number of lags, α_0, α_1, ..., α_q are the parameters to be set $\varepsilon_t \sim N(0, \sigma^2)$ [called *white noise*]. n_t can be seen as a new kind of noise.

The **ARMA** model combines AR and MA models:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_k y_{t-p} + n_t$$
$$y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_k y_{t-p} + \alpha_0 + \alpha_1 \varepsilon_{t-1} + \cdots + \alpha_q \varepsilon_{t-q} + \varepsilon_t$$

There are also *Seasonal AR/MA* models when one looks not only one step behind. ARMA models can only be applied to stationary time series [constant mean, variance and autocorrelation]. One can calculate a difference of a time series in order to make the mean constant ($z_t=y_t-y_{t-1}$) or to make the autocorrelation constant ($z_t=y_t-y_{t-12}$). **ARIMA** models first calculate a difference of the time series and then apply an ARMA model to the new time series. In order to make variance constant, a log transformation or, more generally, a Box-Cox transformation is applied to a time series. **ARIMAX** models also include external factors [just like in a normal linear regression]. The *hyper-parameters* of this model are: p [number of lags for AR], q [number of lags for MA], number for times of differencing the time series [often called d], P [number of lags for Seasonal AR], Q [number of lags for Seasonal MA], number for times of seasonal differencing the time series [often called D], and Box-Cox coefficient, if used.

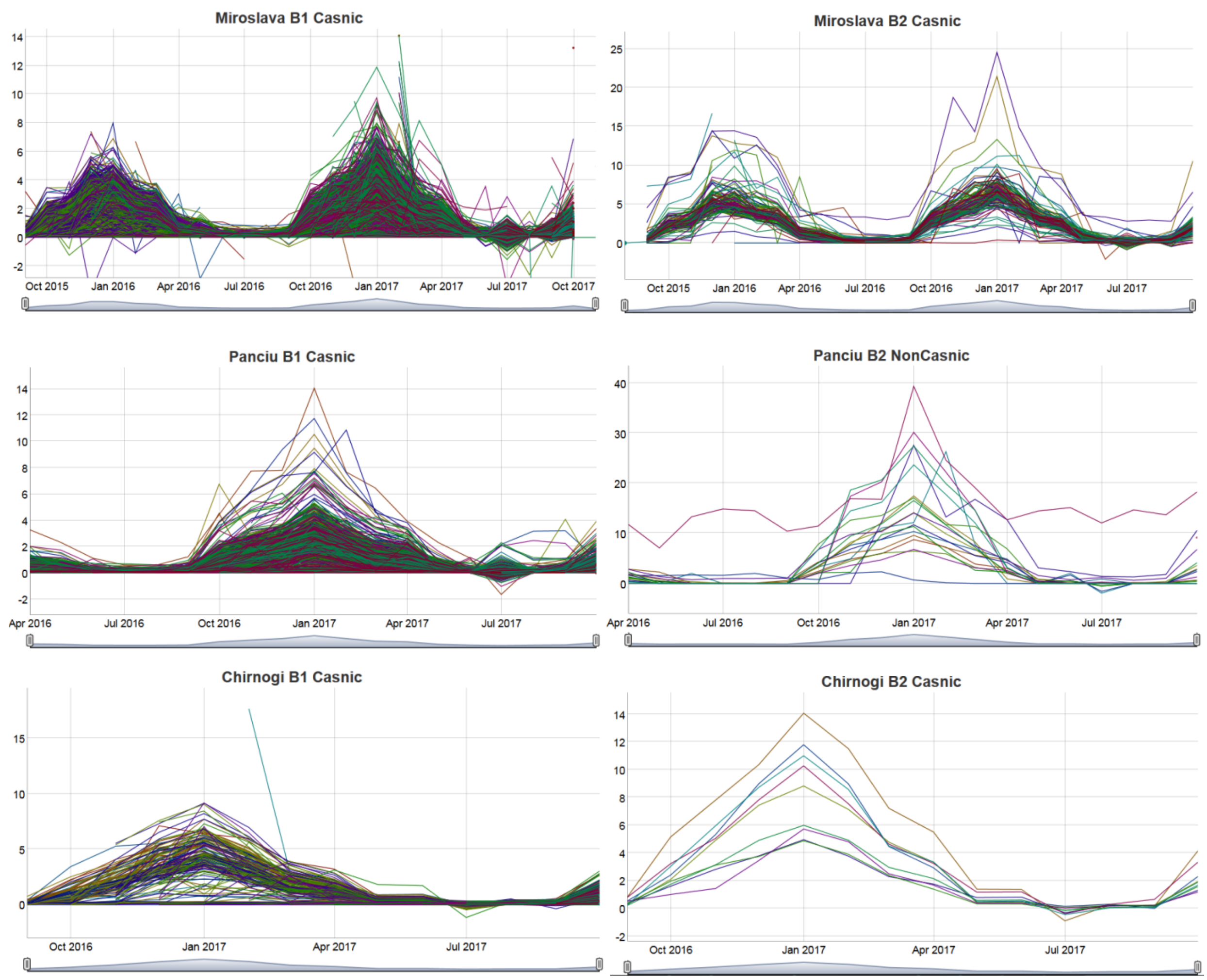
b. Data

Experimental data were represented by the monthly consumptions of clients from 4 localities. Their distribution is given below:

	Miroslava	Panciu	Chirnogi	Odobești	Total
B1-NonHousehold	117	46	14	27	**204**
B1-Household	1690	911	267	631	**3499**
B2-NonHousehold	19	20	0	12	**51**
B2-Household	121	37	9	38	**205**
B3-NonHousehold	6	15	0	5	**26**
B3-Household	0	1	0	6	**7**
B4-NonHousehold	1	0	0	2	**3**
B4-Household	0	0	0	1	**1**
Total	**1954**	**1030**	**290**	**722**	**3996**

Table 2: Number of clients: by category and by locality

The first starting date differs from locality to locality. The earliest dates are from Miroslava starting from September 2015. The next plots show the gas consumptions [for the most representative groups of clients] as time series.



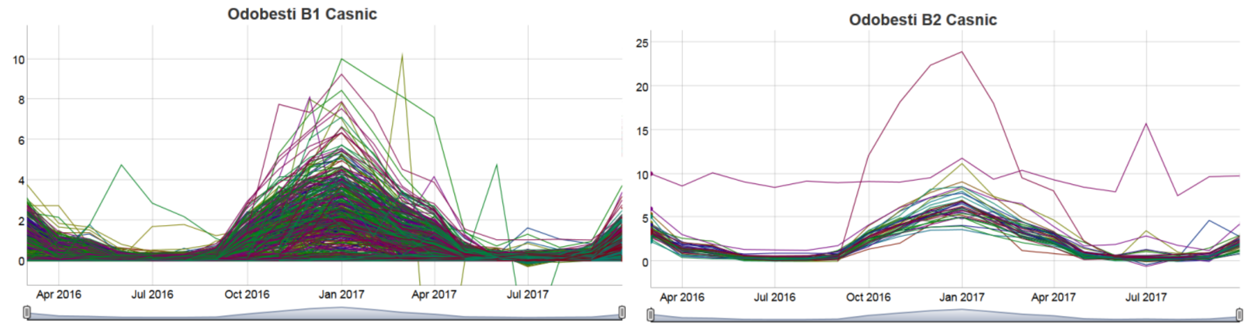


Figure 1: Plots of time series for different categories of clients

As those plots highlight, the data needed to be *pre-processed* in order to remove the negative consumptions [which were a result from bad records]. Moreover, the gas meter was not always read and, in such months, some estimations were made. In the latter case, the errors are corrected in the next months.

Additional data on weather were included. The data for those 4 localities were extracted from the following website which indicates the daily values of temperature: https://www.ogimet.com/gsodc.phtml.en . The stations corresponding to those localities were identified by asking the specialists in meteorology from Faculty of Geography and Geology within *Alexandru Ioan Cuza* University of Iași. This is mandatory because not only the distances between localities and stations matter, but also the landforms. The data from the following stations were taken to be analysed: Miroslava – Iași station, Chirnogi – Oltenița station; Panciu and Odobești – Focșani station. More specifically, we gathered monthly statistics: minimum, maximum, median and average [mean]. In the end we only used the average as a predictor. The temperature is the principal external factor that influences the consumption: for household clients of type B1, the Pearson correlation between the monthly average consumption and the monthly average temperature for the 25 months recorded is -0.96 and for the other household clients is over -0.97. The linear correlation between the average consumption and the monthly average temperature is also significant for some non-household clients [but is reduced in absolute value]. The visible exception is for the only client of type B4. This type of correlation is one extra criterion in dividing the clients in categories. Separate models should be derived for each category.

High accuracy predictions of consumption depend on realistic temperature forecast. The problem is that only the 10-day forecast is reliable. Because of this, our prediction models of consumption were developed and evaluated in two different situations: the ideal case when the real monthly temperatures are available, and the case when no temperature data is known.

Because the first consumption record was on September 2015 and the last on November 2017 [with the possibility of updating the data with December 2017 and January 2018 records], and so the amount of data was not that big, the decision of splitting data in train, validation/development, and test sets was not easy. Ideally, for each month there should be such a tuple (train, validation, test).

For *models unrelated to time series*, the predictor variables were: lagged consumptions [1 or many before the month you want to forecast] and/or type of client [B1, B2, B3, or B4] and/or month when you want to forecast and/or differences [consumption this month minus consumption this month last year; as many as possible] and/or temperature on the month when you want to forecast. For *every client and for every month [of every year]*, those datasets were created. As a result, such a dataset can be identified only specifying the month. The maximum number of lags we considered was 13. This leaded us to the following data splitting [regarding ANN models, 10 runs were considered for October and November 2017 and 1 run for the rest of the months]:

Monthly forecasting [Left aligned] Yearly forecasting on a monthly basis in order to compare results with monthly forecasting [Right aligned]		
Train set	**Validation/Development set**	**Test set**
2016-11-01 *[not for time series]*	2017-11-01	-
2016-10-01 *[not for time series]*	2017-10-01	-
2015-09-01 to 2016-11-01	2016-12-01	Possible December 2017
2015-09-01 to 2016-11-01	2017-01-01	Possible January 2018
2015-09-01 to 2016-12-01	2017-01-01	Possible January 2018
2015-09-01 to 2016-11-01	2017-02-01	-
2015-09-01 to 2017-01-01	2017-02-01	-
2015-09-01 to 2016-11-01	2017-03-01	-
2015-09-01 to 2017-02-01	2017-03-01	-
2015-09-01 to 2016-11-01	2017-04-01	-
2015-09-01 to 2017-03-01	2017-04-01	-
2015-09-01 to 2016-11-01	2017-05-01	-
2015-09-01 to 2017-04-01	2017-05-01	-
2015-09-01 to 2016-11-01	2017-06-01	-
2015-09-01 to 2017-05-01	2017-06-01	-
2015-09-01 to 2016-11-01	2017-07-01	-
2015-09-01 to 2017-06-01	2017-07-01	-
2015-09-01 to 2016-11-01	2017-08-01	-
2015-09-01 to 2017-07-01	2017-08-01	-
2015-09-01 to 2016-11-01	2017-09-01	-
2015-09-01 to 2017-08-01	2017-09-01	-
2015-09-01 to 2016-11-01	2017-10-01	-
2015-09-01 to 2017-09-01	2017-10-01	-
2015-09-01 to 2016-11-01	2017-11-01	-
2015-09-01 to 2017-10-01	2017-11-01	-

Table 3: Train, validation and test sets identified by month

For *models related to time series*, the predictor variables were: lagged consumptions and/or lagged temperatures [1 or many before the month you want to forecast]. This leaded us to the same data splitting as above. As noted, the first two entries are not suitable for time series analysis.

A summary regarding the types of clients and the types of models needed to be developed can be seen in the following tables. For each type of clients there are two types of models that should be created. There are also mentioned the types of models between which one should choose.

Types of models needed to be developed
Without temperature – For the next month
With temperature – For the next month

Table 4: Types of models needed to be developed

Types of clients [only Household clients considered]	
From Miroslava – Correlated with temperature	From Miroslava – Uncorrelated with temperature
From Panciu – Correlated with temperature	From Panciu – Uncorrelated with temperature
From Chirnogi – Correlated with temperature	From Chirnogi – Uncorrelated with temperature
From Odobești – Correlated with temperature	From Odobești – Uncorrelated with temperature

Table 5: Types of clients

Types of models between which to choose	
Models unrelated to time series	**Time series related models**
Linear regression	Naïve model
ANN (MLP)	ARIMA(X)

Table 6: Types of models between which to choose

Experiments

The metrics used to evaluate the models were [y – real consumption value; $\hat{y}$ – predicted value; n – number of clients considered]:

Metric	Formula/Observation
Mean Squared Error	$MSE = mean((y-\hat{y})^2) = \frac{\sum(y-\hat{y})^2}{n}$
Mean Absolute Error	$MAE = mean(\lvert y-\hat{y}\rvert) = \frac{\sum \lvert y-\hat{y}\rvert}{n}$
Median Absolute Error	$MdAE = median(\lvert y-\hat{y}\rvert)$
Linear Correlation between real and predicted values	$Corr = cor(y,\hat{y})$
Absolute Difference between real average monthly consumption and predicted average monthly consumption	An average time series is created and sent as input to the model

Table 7: Metrics used to evaluate a model

The whole process was implemented in the R programming language. The main packages used were:

- **forecast** [with *auto.arima*, *forecast, BoxCox.lambda* functions] [6]: to fit ARIMA(X) models
- **stats** [with *lm* and *predict.lm* functions] [7]: to fit linear regression models
- **keras** [with multiple functions] [8]: to fit MLP models

For *models unrelated to time series*, as mentioned earlier, we tried models *with temperature and without temperature [as a predictor], with and without differences, with and without the month, with or without the type of client*. The *number of lags* considered ranged from 1 to 13 with a step of 2. The format of some input predictors also mattered. There are models with the month specified as a number and models with the month specified as a one-hot vector, e.g. if the month was February there were created 12 predictors; all but the second were set to zero; the second was set to one: 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0. The *type of client* predictor was treated in the same manner [numeric or one-hot vector].

For the *linear regression* model, there were no hyper-parameters to be set.

For the *MLP* architecture, two models were adopted empirically only taking into consideration data generated for validation on October 2017 [the first model for data without temperature and the second for data with temperature]:

<table>
<tr><th>Hyper-parameter (Type)|Model</th><th>MLP _-50-50-1
[2 hidden layers]</th><th>MLP _-30-20-1
[2 hidden layers]</th></tr>
<tr><td>Initialization method</td><td colspan="2">Default:
For weights: Glorot uniform
For bias: Zeros</td></tr>
<tr><td>Activation functions</td><td colspan="2">Hidden layers: ReLu
Output layer: Identity</td></tr>
<tr><td>Optimizer</td><td colspan="2">Adam with default values:
lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08</td></tr>
<tr><td rowspan="2">Regularization</td><td colspan="2">DropOut
1st hidden layer: DropOut rate = 0.5</td></tr>
<tr><td>2nd hidden layer:
DropOut rate = 0.5</td><td>No DropOut</td></tr>
<tr><td>Loss function</td><td colspan="2">MSE</td></tr>
<tr><td>Batch size</td><td>128</td><td>32</td></tr>
<tr><td>Number of epochs</td><td>50</td><td>30</td></tr>
</table>

Table 8: Specific hyper-parameters for the two chosen MLP models

For *models related to time series,* as mentioned earlier, we tried models *with temperature* [ARIMAX] *and without temperature* [naïve and ARIMA]. Moreover, for ARIMA(X), we considered models with and without a Box-Cox transformation, with and without D [number for times of seasonal – here, S=12 – differencing the time series] set to 1, and we also created a model by using the default behaviour of the *forecast* function from *forecast* package in R. In order to set the hyper-parameters p, d, q, P, D [although we set it to 1 manually in certain situations], Q the function *auto.arima* was used [for each client]. This function uses the Hyndman-Khandakar algorithm to find those hyper-parameters. In the call of *auto.arima* function, we set *max.p* and *max.q* to 12, *stepwise* and *approximation* to FALSE. The Box-Cox coefficient was automatically identified by the function *BoxCox.lambda*.

Results

Our research was focused on a specific type of clients [Household – From Miroslava – Correlated with temperature] and the following results are only for this category. For each month we showed only the best model [based on MSE] without temperature and the best model with temperature.

<table>
<tr><th>Model</th><th>MSE</th><th>MAE</th><th>MdAE</th><th>Corr</th><th>Abs. Avg. Diff.</th></tr>
<tr><td colspan="6">Validation on December 2016</td></tr>
<tr><td>MLP _-50-50-1
withTemp: N
Train Period:
2015-09 to 2016-11
#Lags: 13
typeMonth: one hot
typeClient: one hot
withDiff: Y
#Runs: 1
#Records: 765</td><td>0.3798351</td><td>0.4314502</td><td>0.3268276</td><td>0.9227971</td><td>0.234341900</td></tr>
<tr><td>MLP _-50-50-1
withTemp: Y</td><td>0.3801115</td><td>0.4244243</td><td>0.3111872</td><td>0.9167776</td><td>0.049795906</td></tr>
</table>

Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: one hot typeClient: none withDiff: Y #Runs: 1 #Records: 765					
Validation on January 2017					
Linear regression **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: number typeClient: one hot withDiff: Y #Runs: 1 #Records: 796	0.6465580	0.5723214	0.4344356	0.9220211	0.1732643772
Linear regression **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: number typeClient: one hot withDiff: Y #Runs: 1 #Records: 796	0.6465580	0.5723214	0.4344356	0.9220211	0.1732643772
Validation on February 2017					
MLP _-30-20-1 **withTemp: N** Train Period: 2015-09 to 2017-01 #Lags: 11 typeMonth: number typeClient: number withDiff: Y #Runs: 1 #Records: 849	**0.3258726**	0.3245447	0.1960771	0.9180510	0.0150069537
MLP _-30-20-1 **withTemp: Y** Train Period: 2015-09 to 2017-01 #Lags: 11 typeMonth: one hot typeClient: number withDiff: Y #Runs: 1 #Records: 849	0.3559487	0.3384806	0.2096799	0.9095106	0.0015570464
Validation on March 2017					
MLP _-30-20-1 **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: number typeClient: number withDiff: N #Runs: 1 #Records: 837	**0.1694180**	0.2725083	0.1912206	0.9200467	0.1128368908
MLP _-50-50-1 **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 13	0.1767478	0.2715254	0.1724784	0.9132073	0.0213095480

typeMonth: number typeClient: none withDiff: N #Runs: 1 #Records: 837					
Validation on April 2017					
MLP _-50-50-1 **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: number typeClient: one hot withDiff: Y #Runs: 1 #Records: 849	**0.1161291**	0.2531113	0.1965794	0.9153764	0.0195189609
MLP _-30-20-1 **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: none typeClient: one hot withDiff: N #Runs: 1 #Records: 849	0.1292791	0.2601808	0.1998312	0.8982608	0.0195189609
Validation on May 2017					
MLP _-30-20-1 **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 9 typeMonth: none typeClient: number withDiff: N #Runs: 1 #Records: 973	0.02511915	0.0990081	0.06759772	0.6546035	0.0838767070
MLP _-50-50-1 **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 11 typeMonth: number typeClient: none withDiff: Y #Runs: 1 #Records: 913	**0.02357950**	0.1133989	0.09047573	0.6869828	0.0176222206
Validation on June 2017					
Linear regression **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 1 typeMonth: number typeClient: none withDiff: Y #Runs: 1 #Records: 1161	**0.002580885**	0.01317062	0.009098499	0.9664575	0.012162390
Linear regression **withTemp: Y** Train Period: 2015-09 to 2017-05 #Lags: 3 typeMonth: one hot typeClient: none withDiff: Y	0.009466020	0.05861918	0.036816036	0.8649442	0.005596707

#Runs: 1 #Records: 1156					
Validation on July 2017					
Linear regression **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 1 typeMonth: one hot typeClient: none withDiff: Y #Runs: 1 #Records: 1161	0.004868201	0.05178946	0.04446680	0.9647579	0.0475990728
Linear regression **withTemp: Y** Train Period: 2015-09 to 2017-06 #Lags: 1 typeMonth: one hot typeClient: none withDiff: Y #Runs: 1 #Records: 1161	**0.003983435**	0.03877782	0.02984545	0.9647579	0.0178273296
Validation on August 2017					
Linear regression **withTemp: N** Train Period: 2015-09 to 2017-07 #Lags: 1 typeMonth: one hot typeClient: none withDiff: Y #Runs: 1 #Records: 1161	**0.01002398**	0.06686699	0.04711231	0.7967213	0.0021343099
Linear regression **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 3 typeMonth: number typeClient: none withDiff: Y #Runs: 1 #Records: 1161	0.01011272	0.06611762	0.04452621	0.7949303	0.0072216099
Validation on September 2017					
MLP _-30-20-1 **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 1 typeMonth: one hot typeClient: none withDiff: Y #Runs: 1 #Records: 1160	**0.009392238**	0.05245531	4.152874e-02	0.8717407	0.0232097816
Linear regression **withTemp: Y** Train Period: 2015-09 to 2017-08 #Lags: 1 typeMonth: none typeClient: none withDiff: Y #Runs: 1 #Records: 1160	0.012335522	0.07018557	5.854899e-02	0.8841417	0.0298735000

Validation on October 2017					
MLP _-50-50-1 **withTemp: N** Train Period: 2016-10 #Lags: 7 typeMonth: - typeClient: one hot withDiff: N #Runs: 10 #Records: 1155	0.1605943	0.2813402	0.2123464	0.8238079	0.018119066
MLP _-30-20-1 **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 13 typeMonth: number typeClient: none withDiff: N #Runs: 1 #Records: 1008	**0.1467143**	0.2654860	0.1997132	0.8349680	0.096687903
Validation on November 2017					
MLP _-50-50-1 **withTemp: N** Train Period: 2015-09 to 2016-11 #Lags: 11 typeMonth: one hot typeClient: one hot withDiff: Y #Runs: 1 #Records: 1123	**0.2441576**	0.3064871	0.1991085	0.8883574	0.011081485
Linear regression **withTemp: Y** Train Period: 2015-09 to 2016-11 #Lags: 9 typeMonth: number typeClient: one hot withDiff: Y #Runs: 1 #Records: 1150	0.2700347	0.3408615	0.2299223	0.8772789	0.069081164

Table 9: Experimental results: the best two models for each month – with and without temperature

Because the results regarding ARIMA(X) models are not present above, we introduce some of them below:

Model **ARIMA** **[withTemp = N]**	**MSE**	**MAE**	**MdAE**	**Corr**	**Abs. Avg. Diff.**
Validation on October 2017					
Train Period: 2015-09 to 2017-09 typeD: none typeLambda: none #Records: 1158	0.4193413	0.4566618	0.3204010	0.6371408	0.08836832
Train Period: 2015-09 to 2017-09 typeD: none typeLambda:box.cox #Records: 1158	0.6659137	0.6000043	0.5188793	0.4575220	0.53448372
Train Period: 2015-09 to 2017-09 typeD: 1 typeLambda: none #Records: 1158	1.0408777	0.7422500	0.6306670	0.6081971	0.68593220
Train Period: 2015-09 to 2017-09 typeD: 1 typeLambda:box.cox #Records: 1158	1.6023854	0.7650985	0.6120121	0.5245753	0.45529435
Train Period: 2015-09 to 2016-11 typeD: none typeLambda: none #Records: 1090	6.8322081	0.9974210	0.5629849	0.2048947	0.24322983
Train Period: 2015-09 to 2016-11 typeD: none typeLambda:box.cox #Records: 1090	6.6681629	0.8676305	0.3863384	0.1288010	0.03609095
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda: none #Records: 1090	14.5602471	1.4178933	0.7952093	0.2577753	0.72277694
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda:box.cox #Records: 1090	13.9384277	1.3647141	0.7360651	0.2580828	0.72277696

Table 10: ARIMA results on October 2017

Model ARIMAX [withTemp = Y]	**MSE**	**MAE**	**MdAE**	**Corr**	**Abs. Avg. Diff.**
Validation on October 2017					
Train Period: 2015-09 to 2017-09 typeD: none typeLambda: none #Records: 1158	0.2455896	0.3619071	0.2790151	0.7898674	0.273375772
Train Period: 2015-09 to 2017-09 typeD: none typeLambda:box.cox #Records: 1158	0.4005065	0.4391740	0.3383696	0.5367043	0.255035362
Train Period: 2015-09 to 2017-09 typeD: 1 typeLambda: none #Records: 1158	0.4601337	0.4540237	0.3011008	0.6456592	0.107478962
Train Period: 2015-09 to 2017-09 typeD: 1 typeLambda:box.cox #Records: 1158	3886.7337413	2.4691901	0.3062334	0.1708928	0.109419810
Train Period: 2015-09 to 2016-11 typeD: none typeLambda: none #Records: 1090	1.9789074	0.5466996	0.3244657	0.3389013	0.218520357
Train Period: 2015-09 to 2016-11 typeD: none typeLambda:box.cox #Records: 1090	1.9647882	0.5373997	0.3164582	0.3413335	0.218520357
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda: none #Records: 1090	2.4144374	0.9515351	0.6307901	0.1008984	0.754183607
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda:box.cox #Records: 1090	2.4667573	0.9608803	0.6376161	0.0980574	0.754183607

Table 11: ARIMAX results on October 2017

Model ARIMA [withTemp = N]	MSE	MAE	MdAE	Corr	Abs. Avg. Diff.
Validation on November 2017					
Train Period: 2015-09 to 2017-10 typeD: none typeLambda: none #Records: 1158	0.6554094	0.6147501	0.4801120	0.83612771	0.329490872
Train Period: 2015-09 to 2017-10 typeD: none typeLambda:box.cox #Records: 1158	446.9893619	1.6902317	0.6044768	0.06521988	0.248492305
Train Period: 2015-09 to 2017-10 typeD: 1 typeLambda: none #Records: 1158	6.2271071	0.8798864	0.4235224	0.40562785	0.487279437
Train Period: 2015-09 to 2017-10 typeD: 1 typeLambda:box.cox #Records: 1158	24.7541915	1.1890974	0.4637374	0.25256117	0.305636749
Train Period: 2015-09 to 2016-11 typeD: none typeLambda: none #Records: 1090	8.1492590	1.3592469	0.9113088	0.20373432	0.648420597
Train Period: 2015-09 to 2016-11 typeD: none typeLambda:box.cox #Records: 1090	7.7791150	1.2124225	0.7636013	0.14494744	0.853118375
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda: none #Records: 1090	15.0335763	1.4296311	0.6871125	0.34690792	0.563400272
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda:box.cox #Records: 1090	14.3449943	1.3732307	0.6752813	0.35049304	0.563400292

Table 12: ARIMA results on November 2017

Model ARIMAX [withTemp = Y]	MSE	MAE	MdAE	Corr	Abs. Avg. Diff.
Validation on November 2017					
Train Period: 2015-09 to 2017-10 typeD: none typeLambda: none #Records: 1158	3.708000e-01	0.3784989	0.2341639	0.831287993	0.092999434
Train Period: 2015-09 to 2017-10 typeD: none typeLambda:box.cox #Records: 1158	2.245661e+04	5.3750411	0.3974761	0.006898835	0.524683455
Train Period: 2015-09 to 2017-10 typeD: 1 typeLambda: none #Records: 1158	5.345873e-01	0.4580276	0.2831699	0.777938216	0.148471183
Train Period: 2015-09 to 2017-10 typeD: 1 typeLambda:box.cox #Records: 1158	1.578799e+01	0.8623973	0.3243840	0.308394534	0.099438771
Train Period: 2015-09 to 2016-11 typeD: none typeLambda: none #Records: 1090	2.277845	0.5648860	0.2517588	0.471630717	0.005391859
Train Period: 2015-09 to 2016-11 typeD: none typeLambda:box.cox #Records: 1090	2.258035	0.5602095	0.2574237	0.474030504	0.005391859
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda: none #Records: 1090	1.361205	0.7857227	0.5423968	0.525490744	0.565460299
Train Period: 2015-09 to 2016-11 typeD: 1 typeLambda:box.cox #Records: 1090	1.382433	0.7969392	0.5484855	0.523716991	0.565460299

Table 13: ARIMAX results on November 2017

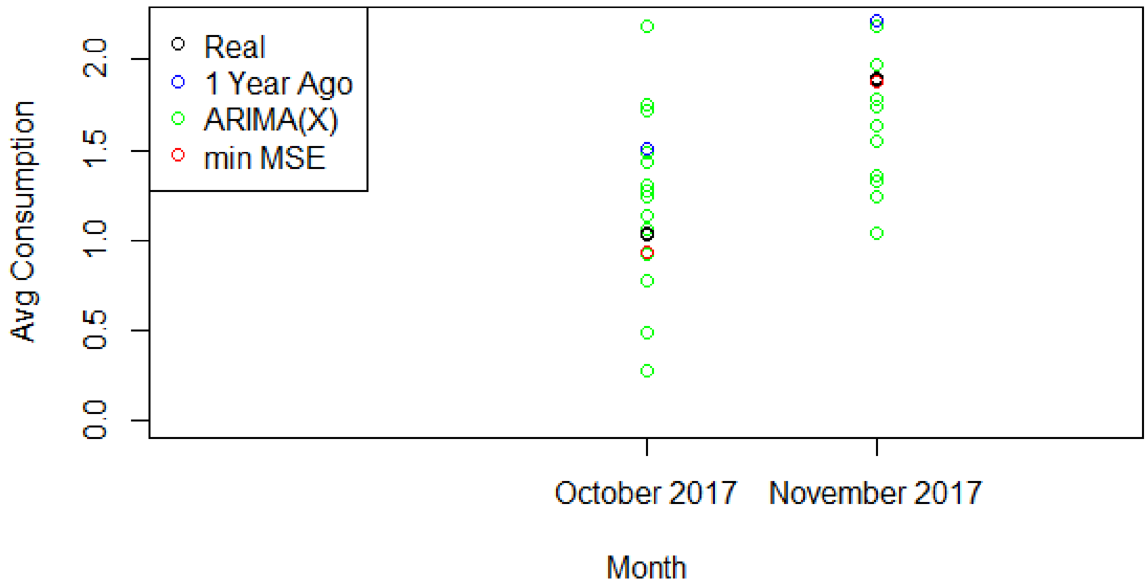


Figure 2: Average consumption time series when forecasting: October 2017 and November 2017

As it can be seen from the above tables, adding temperature data was useful in the case of three months: May, July, and October. Linear regression obtains the best score in the summer. An important note is that the naïve model [which was our benchmark] does not have the lowest MSE in any month. Moreover, we can generalize and say that time series models [including ARIMA(X)] are not suitable in this methodology.

Conclusions and Further Work

The objective of the discussed and presented results of the study was to obtain forecasts of natural gas consumption. The data used was from 4 localities from Romania. The methodology included time series related models [naïve model, ARIMA(X)] and models unrelated to time series [linear regression, MLP]. The main problem was the lack of sufficient data to split it in train, validation and test sets. The highlight was put on monthly forecasts, although yearly forecasts on a monthly basis were also used. The results mainly show that: models unrelated to time series are more suitable in this methodology, the temperature data is useful only for 2-3 months, and linear regression gets the best results for summer months.

As *further work*, this methodology should be applied also to the other categories of clients [household or not; from other localities, not only Miroslava; both correlated and uncorrelated with temperature]. Furthermore, the result on each month could be processed in such a way that the adopted model would be an ensemble [weighted combination] of the top models [ordering them not only by MSE, but also taking into consideration the other computed metrics].

References

1. Jolanta Szoplik, *Forecasting of natural gas consumption with artificial neural networks*

2. Dejan Ivezic, *Short-term natural gas consumption forecast*

3. Steven Vitullo, Ronald Brown, George Corliss, Brian Marx, *Mathematical models for natural gas forecasting*

4. Omer Fahrettin DEMIREL, Selim ZAIM, Ahmet CALISKAN, Pınar OZUYAR, *Forecasting natural gas consumption in Istanbul using neural networks and multivariate time series methods*

5. Bozidar Soldo, *Forecasting natural gas consumption*

6. https://cran.r-project.org/web/packages/forecast/forecast.pdf

7. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html

8. https://cran.r-project.org/web/packages/keras/keras.pdf

9. https://www.datacamp.com/courses/introduction-to-time-series-analysis

10. https://www.datacamp.com/courses/arima-modeling-with-r

11. https://www.datacamp.com/courses/forecasting-using-r