Machine Learning algorithms for air pollutants forecasting

Marius Dobrea, Andreea Bădicu, Marina Barbu, Oana Șubea, Mihaela Bălănescu, Geroge Suciu, Andrei Bîrdici, Oana Orza
Research & Development Department
BEIA Consult International
Bucharest, Romania

Ciprian Dobre
Department of Computers
University POLITEHNICA, Bucharest, Romania
Bucharest, Romania

***Abstract*— Air pollution represents an issue that raises many concerns nowadays, as it has various negative effects on the environment and the economy worldwide. Because of the rapid urbanization, cities are suffering from polluted air, so it is important to predict future air quality. For this purpose, new applications of artificial intelligence should be employed. In this paper, we will present several Machine Learning algorithms, the possible software that can be used for them and the applications used in the field of air quality. Based on the research in the field, we propose SVR, ARIMA and LSTM, 3 Machine Learning models, which can be used to predict air pollution. These algorithms have been tested using time-series for PM_{10} and $PM_{2.5}$ particles. The results showed that SVR and ARIMA algorithms are the most suitable in forecasting air pollutant concentrations.**

Keywords—Machine Learning, Air Pollution, Forecasting, Time Series

I. INTRODUCTION

Air pollution is an issue that concerns a lot of people nowadays and has a significant influence on human health worldwide. It has a great impact on human well-being, the environment and the economic advancement around the world. Recent studies [1] have shown that in 2015, 6.5 million premature deaths worldwide were caused by air pollution. Its impact on health depends on the concentration of the pollutant and the exposure levels. Particulate Matter (PM) represents one of the most important pollutants in regard to the effects on health. Among the best-known PMs are the PM_{10} (PM with a diameter lower than 10 m) and $PM_{2.5}$ (PM with a diameter lower than 2.5 m). Even in small concentrations, these PMs can have many adverse effects on human health [2]. Besides the type of pollutant and its concentration, the duration and the frequency of the exposure are also important factors in the negative impact on human's well-being [3].

Considering that traditional air quality prediction methods require more computational power for the estimation of pollutant concentration, many people are trying to apply Artificial Intelligence (AI) algorithms (machine learning, deep learning), which can lead to better results. There is an increased interest in the machine learning methods for forecasting non-linear time series information, such as meteorological and pollution data. Machine Learning (ML) is a data technique which teaches a computer to create a model using training data. It is a subfield of artificial intelligence and enables software applications to be increasingly precise in predicting results. ML can review a wide range of data and discover patterns and specific trends.

The literature on this topic proposes a wide range of tools for ML classifiers to deal with time-series data. This paper will consider three models of ML algorithms to forecast air pollution and will assess their performance on forecasting air pollutant concentrations. The models used are SVR, ARIMA and LSTM and will be described in the next paragraphs. Support Vector Regression (SVR) is a version for Support Vector Machine (SVM) algorithm characterized by the use of kernels, hyper planes, boundary lines and support vectors [4]. It is shown to be an effective tool in real-value function estimation and its application presented in chapter 4.1. Chapter 4.2. focuses on the application of the Autoregressive Integrated Moving Average (ARIMA) model, which is a class of models that describe a time series based on its past values [5]. It is characterized by the parameters p (lag order), d (degree of differencing) and q (order of moving average). Long Short-Term Memory (LSTM) is a neural network model used for classification, processing and predicting based on time series data, in which there are lags of unknown duration between important events [6]. The algorithm is characterized by a cell, an input gate, an output gate and a forget gate. The application of this model in the present paper is presented in chapter 4.3.

II. RELATED WORK

Authors in article [7] use 50 time series for monthly temperature and precipitation in Greece, and they want to analyse the quality of the forecast. The proposed methods are based on two ML algorithms: one of them uses a Neural Network (NN), and the other uses SVM. From the experiment, it results that by using the SVM algorithm, it is possible to obtain a better performance. In the end, they concluded that, based on their score, there is no relationship between the parameters that

result from the time series and the forecast quality. In their paper [8], the authors propose an online learning algorithm for estimating Auto-Regressive Integrated Moving Average (ARIMA) models. ARIMA models are one of the most general classes of models for forecasting time series. The presented method involves the use of recursive formulation in an online learning environment. According to the experiments, the authors conclude that the proposed online ARIMA model is probably as good as the best ARIMA model. Consequently, the estimation of the parameters can be done online in a scalable and efficient way. In article [9], the authors develop a hybrid method based on ML algorithms, which is supposed to detect jumps in financial time series. The model inputs slow frequency market data and uses a convolutional neural network and LSTM (Long Short-Term Memory). The limitations of the model are also mentioned (very high computational complexity, hence high computation time). Paper [10] presents different deep learning models for predicting future PM_{10} value. The results suggested that the performance of Gated Recurrent Unit (GRU) network is better than the Recurrent Neural Networks (RNN) and LSTM networks. They also discovered a pattern for PM_{10} concentration. In the article [11], the main goal is to obtain a method based on machine learning for $PM_{2.5}$ concentration, using regression modelling with Scikit-learn. The results showed that traffic density is directly correlated with air pollution, especially in the earlier hours of the day: the highest rate of pollution was reached at eight o'clock in the morning. Secondly, the meteorological factors influence the accuracy of the prediction. It was concluded that the best prediction model should be a hybrid data source that includes gas monitoring. The study demonstrated that the performance of the prediction depends on different factors such as traffic, the meteorological factors and the number of trace gases that the hybrid source is based on.

III. DATASET

In order to find the best Machine Learning algorithm to use for the forecasting of air pollution, several analyses were preliminary performed on a dataset that includes the concentration values of PM_{10} and $PM_{2.5}$ air pollutants (Fig. 1).

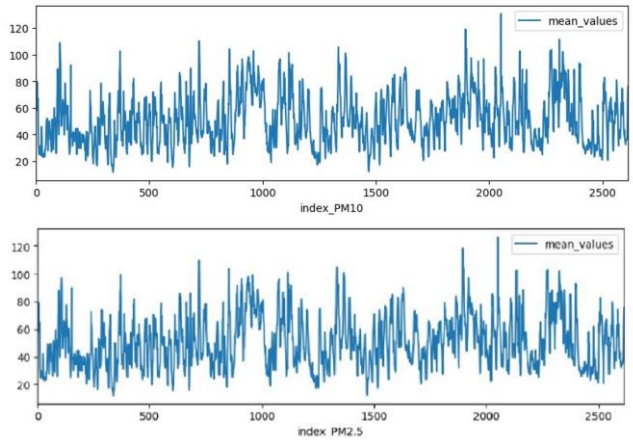

Fig. 1. Concentration values of PM_{10} and $PM_{2.5}$

These parameters have been measured using Libelium sensors every 15 minutes, in Bucharest (Beia office) between 1st of November 2018 – 28th of January 2019 [12]. As it can be observed from the Figure above, the values for the concentration of PM_{10} and $PM_{2.5}$ are very similar.

The first stage of this research consisted in a data analysis that was based on the comparison with requirements of standard data (for example, the values of the concentration must be a null or a positive value). Secondly, a preliminary analysis was made, where more statistical descriptive methods were used. Some of these methods are mean value, variation, or standard deviation. Humidity threshold was selected using the Pearson correlation coefficients, the dimension of the sub-datasets and the ratio between PM_{10} and $PM_{2.5}$.

The next step implied the computation of the average values for the registered data after their validation and for each parameter was obtained a dataset containing 2133 values. Moreover, in **Error! Reference source not found.** is presented the first analysis of the statistical parameters performed on this dataset. As the results show, there is a high variability of the humidity and of the PMs concentrations [12].

TABLE I. STATISTICAL PARAMETERS VALUES

Statistical parameter	Measured parameter				
	PM_{10} $(\frac{ug}{m^3})$	$PM_{2.5}$ $(\frac{ug}{m^3})$	Temperature(°C)	Pressure (Pa)	Humidity(%)
Mean	50.81	50.35	23.66	100984	34
Median	48.45	47.96	23.44	101124.4	33.32
Standard error	0.37	0.37	0.01	15.61	0.08
Sample Variance	359.39	351.85	0.41	637127.7	15.21
Minimum value	11.79	11.74	22.85	98605.14	24.47
Maximum value	130.7	126.7	26.18	102865.9	49.27
Confidence level for mean	±0.73	±0.72	±0.02	±30.61	±0.15

IV. EXPERIMENTS

Depending on some external factors like traffic, weather and others, the pollution level can vary significantly so that by the end of October 2018 till the beginning of February 2019 the average values of PM_{10} and $PM_{2.5}$ pollutants for each hour oscillate very much.

A. SVR MODEL

1) Application of the algorithm

The data is stored in the X and y variables. Variable y contains the average values of PM_{10} concentration (pollutant measured in a specific hour), and X contains the indexes to that value, as it is displayed in the example below. Using the Numpy library, the data is shaped in the appropriate way to be used in methods from the Scikit-learn library.

Example of the code used to store the data in X and y variables:

```
X, y = [], []
for i in range(len(PM10)):
    X.append(i)
    y.append(PM10[i][3])

y_data = y
y = ['%.2f' % elem for elem in y_data]
X = np.array([X]).T
y = np.array(y).ravel()
y = np.array(y)
```

To split the data into train and test dataset, it was used a method from Scikit-learn library, which allows choosing the test of train size as a percent. The resulted data is stored in test and train variables.

Example of the code where the data is split into train and test dataset:

```
X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size=0.2)
```

2) Selection of gamma and C parameters

The method used to choose the most suitable values for gamma and C parameters was represented by the value of the correlation coefficient method (which must be as high as possible) and the mean squared error method (which has to be as low as possible). In order to obtain those values, it was necessary to test several values for each parameter using the .fit() method. This method can take two arguments: X_train, which represents the indexes for the training dataset, and y_train, which represents the target values for the training dataset. These are real numbers in regression and the function returns an object stored in the "svr_rbf" variable. Then, after the model is trained, it can be used to make predictions. In order to predict the data, the object created before is used by applying the .predict() method resulting the y_pred variable based on the test dataset. When the prediction is done, the test dataset is compared with the predicted dataset by using the correlation coefficient and calculating the mean squared error.

Example of code used to predict data and to compare the resulted dataset with the test dataset:

```
gamma_list = ['auto', 0.5, 0.3, 0.1, 0.05, 0.04,
0.03,0.02,0.01, 0.005, 0.004, 0.003,0.002,0.001]
c_list = [1, 10, 1e2, 1e3, 1e4, 1e5]

for c in c_list:
    for g in gamma_list:
        svr_rbf = SVR(kernel='rbf',
C=c,gamma=g).fit(X_train, y_train)
        y_pred = svr_rbf.predict(X_test)
        y_pred = ['%.2f' % elem for elem in
y_pred]
        y_pred = [float(e) for e in y_pred]
        corr_coef = np.corrcoef(y_test,
y_pred)[0,1]
        mse = mean_squared_error(y_test, y_pred)
        print("for C = ", c, " and gamma = ", g,
" correlation coefficient is ", corr_coef,
              " and mean squared error = ", mse)
```

3) Results of the algorithm

After applying the algorithm using different values for gamma and C it was chosen the most appropriate values for the coefficients of the model, more exactly C = 100 (or 1e2), gamma = 0.1, kernel='rbf', a correlation coefficient of 0.966 and a mean squared error which equal to 25.3. In Fig. 2 is presented a comparison between the estimated data and the measured data.

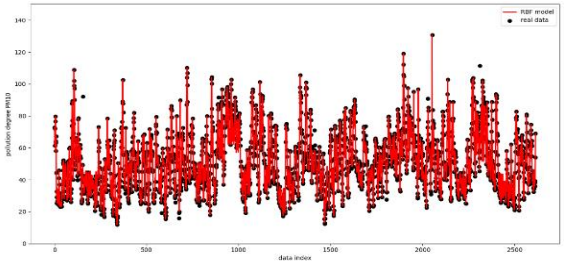

Fig. 2. Comparison between estimated data and measured data

4) Observations

After running the algorithm, for the polynomial kernel, the model wasn't built even after 30 minutes. For C = 10000 (or 1e4) and C = 100000 (or 1e5), the time to create the model took more than 20 minutes. For the same values for gamma and C parameters, the results could be different depending on the training dataset because of the train_test_split method which split the data randomly each time.

B. ARIMA Model

1) Application of the algorithm

The ARIMA algorithm has been used in the process of estimating the parameters PM_{10}, respectively $PM_{2.5}$. 80% of the data set was used for the purpose of training the Machine Learning method. The rest of 20% of data was employed for testing the algorithm in order to estimate the next value, according to the algorithm presented below:

```
series      =      read_csv('file.csv',      header=0,
parse_dates=[0],      index_col=0,      squeeze=True,
date_parser=parser)

X = series.values
size = int(len(X) * 0.80)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
   model = ARIMA(history, order=(5,1,0))
   model_fit = model.fit(disp=0)
   output = model_fit.forecast()
   yhat = output[0]
   predictions.append(yhat)
   obs = test[t]
   history.append(obs)
   print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

For each estimation, the absolute deviation will be calculated and the real (measured) value will be added to the active data set. By employing this method, the average of the set will be modified with each step. The return of the ARIMA algorithm is presented below:

```
predicted=31.517350, expected=29.401999
predicted=36.911160, expected=34.691999
predicted=36.469997, expected=37.681998
predicted=30.688989, expected=32.451999
predicted=31.873749, expected=31.535999
predicted=31.305169, expected=28.096000
predicted=27.492168, expected=27.797500
predicted=27.930571, expected=27.735999
```

```
predicted=34.024295, expected=32.325000
predicted=32.334449, expected=33.146000
predicted=33.602038, expected=34.793999
predicted=35.424062, expected=35.745000
predicted=36.107573, expected=34.813999
predicted=34.594491, expected=35.084000
```

2) Results of the algorithm

ARIMA model was applied using the same dataset as for the SVR algorithm. The results obtained using ARIMA algorithm are presented in Fig. 3.

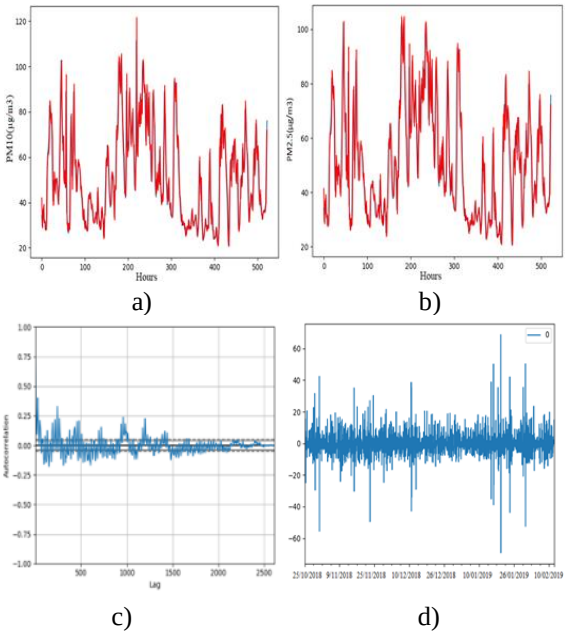

Fig. 3. The results obtained using the ARIMA model for estimating the concentrations of PM_{10} (a), $PM_{2.5}$), the degree of autocorrelation of the algorithm (c), the absolute deviation (d)

The correlation coefficient between the measured values of the concentration of PM_{10} and the estimated ones is 0.921, and for $PM_{2.5}$ parameters is 0.935.

C. LSTM Model

1) Vanilla LSTM Model

Long Short-Term Memory Model (LSTM) Model is a kind of RNN. This model is suited for processing the correlation within time series for both short and long term. The leading innovation of LSTM is the memory cell c_t that acts as an accumulator of the state information. Several self-parameterized controlling gates are accessing, writing and clearing the cell. If the input gate is activated, a new input is provided and the cell is loaded with the information from the new input. If the forget gate f_t is activated, the past cell status c_{t-1} will be no more taken into consideration. Even if the latest cell output c_t will be transmitted to the final state h_t, it is controlled later by the output gate o_t [13]. A LSTM model with a single layer of LSTM units and an output layer used for prediction is called Vanilla LSTM. The architecture of a Vanilla LSTM algorithm can be seen Fig. 4 [14].

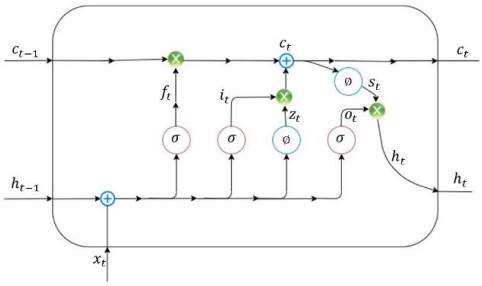

Fig.3. Representation of a vanilla LSTM block structure and its internal information forward flow

2) Application of Vanilla LSTM algorithm

As it was mentioned before, a Vanilla LSTM model is characterized by the fact that the LSTM model has a single hidden layer of LSTM units and another output layer which is used for the prediction process. In this application was used the LSTM model from the layer's library of Keras module. This module can be imported from TensorFlow framework. A Vanilla LSTM model can be defined as it is illustrated in the code below:

```
# define model
model = Sequential()
model.add(LSTM(50,activation='relu',
input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')
```

For this experiment, it was used a univariate series, which means that there is only one feature (n_features=1), for one variable, in this case one variable for PM_{10} and one variable for $PM_{2.5}$. In this way it was defined a model which has an output layer used to predict a one numerical value and 50 LSTM units in the hidden layer. The mean column of the csv file for each PM parameter was used as input sequence in format of a list of values. It is presented in the sequence below:

```
# define input sequence
raw_seq = list(dataset_pm2_5['media_valori'])
```

After the data was processed and prepared the fit() method was applied. Vanilla LSTM algorithm uses the Adam version of stochastic gradient descent based on the mean squared error. Fitting stage is necessary to be able to make a prediction. The way in which fit() function is used is illustrated in the sequence of code below:

```
# fit model
model.fit(X,y,epochs=200, verbose=0)
```

3) Results of the algorithm

The last step to be performed is the prediction one. In order to predict the next value of the sequence it is necessary to provide an input. In this case it was provided a dataset of 23 values and it was expected the prediction of the 24th value. For PM_{10} pollutant was provided the first 23 values from the dataset. These values are presented in the sequence of code below.

```
   # demonstrate prediction
   x_input = array ([72.78399734, 72.07199707,
61.25749779, 61.4219986,
76.73399811, 79.6059967, 64.15999985, 58.09199905,
63.07399902, 67.22666677, 44.46199875, 32.70399933,
29.05599899, 25.27499962, 25.60399933, 26.6279995,
25.3324995, 26.4859993, 27.71199951,
31.55999947,28.41999912, 27.93199921,
24.259999sssss47])
   x_input = x_input.reshape((1, n_steps,
n_features))
   yhat = model.predict(x_input, verbose=0)
   print(yhat)
```

Based on these values, after the first prediction, the model gave as output the next value as equal to *25.366713*, while the expected value was *25.26*. As it can be observed the two values are very close. After performing a 2nd prediction, using the same 23 values, the predicted value was *33.58304*, and after the 3rd prediction the obtained value was *24.5032*.

For $PM_{2.5}$ the first 23 values used as input for the prediction stage are presented in the sequence of code below:

```
# demonstrate prediction
x_input = array ([72.46999,71.90800, 61.13999,
60.39599915, 76.55199738, 79.315998, 63.994999,
57.931999, 62.66399, 64.82333, 40.195998,31.84199,
28.861999, 24.48499, 25.32399,26.17599,  25.0625,
25.97399979, 27.27199936, 29.9239994, 27.99999905,
27.6659996, 23.88599968 ])
x_input = x_input.reshape((1, n_steps, n_features))
yhat = model.predict(x_input, verbose=0)
print(yhat)
```

Using the values presented above the 24th predicted value was: *24.167881*, while the expected output (the real 24th value of the dataset) was *24.3725*. One more prediction test was realised, and the obtained result was: *25.13438*.

CONCLUSIONS

After performing a comparative study of the methods used for data analysis and of the Machine Learning algorithms used for estimating the atmospheric pollutants (PM_{10} and $PM_{2.5}$), it was demonstrated that SVR and ARIMA algorithms are the most suitable in forecasting the air pollutants concentrations, because the correlation coefficient was 0.966 and 0.921 respectively for PM_{10} concentration.

For these methods, it was developed and tested specific algorithms using a dataset that was gathered during the period between the 1st of November 2018 and 28th of January 2019. The results have shown that both algorithms can be used in such examples of air pollution forecasting.

In the future, we want to forecast the air pollutants concentrations for 24 hours with a correlation coefficient greater than 85% using the two proposed algorithms.

ACKNOWLEDGMENT

We would like to express our gratitude to Tel-MONAER project (subsidiary contract no. 1223/22.01.2018, from NETIO Project ID: P 40270, MySMIS CODE: 105976), FarmSustainaBL project (contract no. 119/2019, from ERANET-ERAGAS-ICT-AGRI3 program) and WINS@HI project (PN-III-P3-3.5-EUK-2017-02-0038) for the work presented in this paper.

REFERENCES

[1] P.J. Landrigan, et al. “The Lancet Commission on pollution and health”. The Lancet, 391(10119), pp.462-512, 2018.

[2] K.H. Kim, E. Kabir, and S. Kabir. “A review on the human health impact of airborne particulate matter”. Environment international, 74, pp.136-143, 2015.

[3] A.Y. Watson, R.R. Bates, and D. Kennedy. “Assessment of human exposure to air pollution: methods, measurements, and models”. In Air pollution, the automobile, and public health. National Academies Press (US), 1988.

[4] E. Fradinata, Z.M. Kesuma, S. Rusdiana, and N. Zaman. “Forecast Analysis of Instant Noodle Demand using Support Vector Regression (SVR)”. In IOP Conference Series: Materials Science and Engineering (Vol. 506, No. 1, p. 012021). IOP Publishing, april 2019.

[5] J. Contreras, R. Espinola, F.J. Nogales, and A.J. Conejo. “ARIMA models to predict next-dayelectricity prices”. 2003.

[6] F.A. Gers, J. Schmidhuber and F.Cummins. “Learning to forget: Continual prediction with LSTM”. 1999.

[7] G. Papacharalampous, H. Tyralis, and D. Koutsoyiannis. “”Univariate time series forecasting of temperature and precipitation with a focus on machine learning algorithms: A multiple-case study from Greece. Water resources management, 32(15), pp.5207-5239, 2018.

[8] C. Liu, et al. “”Online arima algorithms for time series prediction. In Thirtieth AAAI conference on artificial intelligence, February 2017.

[9] J.F.A. Yeung, et al. “”Jump detection in financial time series using machine learning algorithms. Soft Computing, pp.1-13, 2019.

[10] V. Athira, P. Geetha, R .Vinayakumar, K.P. Soman. “”DeepAirNet: Applying recurrent networks for air quality prediction. Procedia computer science, 132, pp.1394-1403, 2018.

[11] Y. Rybarczyk and R. Zalakeviciute. “”Regression models to predict air pollution from affordable data collections. Machine Learning—Advanced Techniques and Emerging Applications, 2018.

[12] M. Balanescu, I. Oprea, G. Suciu, M.A. Dobrea, C. Balaceanu, R.I. Ciobanu, and C. Dobre. A Study on Data Accuracy for IoT Measurements of PMs Concentration. In 2019 22nd International Conference on Control Systems and Computer Science (CSCS) (pp. 182-187). IEEE, May 2019.

[13] S.H.I. Xingjian, et al. “”Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems (pp. 802-810), 2015.

[14] A. Nigri, S. Levantesi, M. Marino, S. Scognamiglio, and F. Perla. “”A Deep Learning Integrated Lee–Carter Model. Risks, 7(1), p.33, 2019.