Technical Sciences, 2021, 24, 115–133

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE

Yearbook peer-reviewed scientific journal
ISSN 1505-4675
e-ISSN 2083-4527
TECHNICAL SCIENCES
Homepage: https://czasopisma.uwm.edu.pl/index.php/ts/

DOI: https://doi.org/10.31648/ts.7058

APPLYING PYTHON'S TIME SERIES FORECASTING METHOD IN MICROSOFT EXCEL – INTEGRATION AS A BUSINESS PROCESS SUPPORTING TOOL FOR SMALL ENTERPRISES

***Jolanta Litwin*[1], *Marcin Olech*[2], *Anna Szymusik*[3]**

[1]ORCID: 0000-0002-3650-1953
[2]ORCID: 0000-0002-8346-4673
[3]ORCID: 0000-0002-2472-851X
Department of Computer Science
Rzeszow University of Technology

Received 27 August 2021, accepted 14 September 2021, available online 22 September 2021.

K e y w o r d s: time series forecasting, python integration, excel integration.

A b s t r a c t

The paper describes the current state of research, where integration of Microsoft Excel and Python interpreter, gives the business user the right tool to solve chosen business process analysis problems like: forecasting, classification or clustering. The integration is done by using Visual Basic for Application (VBA), as well as XLWings Python's library. Both mechanisms serve as an interfaces between MS Excel and Python to allow the data exchange between each other. Creating the suitable Graphical User Interface (GUI) in Microsoft Excel, gives the business user opportunity to select specific data analysis method available in Python's environment and set its parameters, without Python's programming. Running the method by Python's interpreter can bring the results, which are hard or even impossible to obtain by using Microsoft Excel only. However, the data analysis methods stored in the Python's script, which are available to the business user, as well as VBA source code, must be designed and implemented by the data scientist. Sample, basic integration between Microsoft Excel and Python's interpreter is presented in the paper. To present value-added of the proposed software solution, simple case study according to time series forecasting problem is described, where forecasting errors of different methods available in the Microsoft Excel and Python are presented and discussed. The paper ends with conclusions according to the results of the current researches and suggested directions of further research.

Correspondence: Jolanta Litwin, Zakład Informatyki, Wydział Budowy Maszyn i Lotnictwa, Politechnika Rzeszowska im. Ignacego Łukasiewicza, al. Powstańców Warszawy 12, 35-959 Rzeszów, e-mail: j.litwin@prz.edu.pl

Introduction

There is no doubt that business activity of the enterprise – conducted by business processes execution – is strictly connected to decision making. Nowadays enterprises have to adjust to ever changing market's condition. What is more, they must predict some phenomenon like consumer's requirements and apply the solution, which bring competitive advantage to them.

Successful and effective management of the enterprise is hardly based on the data the enterprise has. The data is used to plan activity of the organization in the daily routine, as well as in the future. Despite of the size of the organization or type of business activity the organization conducts (production, services, banking, commerce), the enterprises always try to predict future events and plan their development by using various prediction technique (WINKOWSKI 2019, KURZAK 2012).

The organizations can use whole palette of the prediction method. Some of them are simple enough that can be developed and implemented by using Microsoft Excel, which is very often used by business staff of the enterprise. What is worth to mention, despite of the fact the "Industry 4.0" conception comes into reality, analysis clearly show, that spreadsheets used for over 30 years, will be still in constant usage (BIRCH et al. 2018).

Of course many drawbacks can be seen when trying to apply Microsoft Excel to "Big Data" directly. However, by using additional data science software, which pre-process and aggregate raw data from "Big Data" repository into smaller and more suitable form, will bring possibility to still use spreadsheet as a simple analysis tool for aggregated data and as a dashboard for publish analysis results. Another way is to force business staff to learn and use rather complicated software and libraries like "BigQuery", "Redshift" or "PySpark", which allow direct access to "Big Data". However, is hard to imagine, the business user will interact with data by using programming-oriented source codes instead of familiar graphical user interface available in Microsoft Excel. It seems, that both worlds will have to integrate together and still co-exist.

The certain problem here is the fact, that sometimes – especially with "Industry 4.0" conception in mind – available data analysis method, which are available or can be implemented under spreadsheet environment are not as accurate as needed. This is mainly connected to the characteristics of the data source. In that case, enterprises can use more sophisticated analytical software e.g. Statistica with Automated Neural Networks mechanism or Online Analytical Processing (OLAP) available in Microsoft SQL Server or in Microsoft Dynamics. Those environments allow the organization to access to more complicated and suitable analysis method. The proposed solution however, has some drawbacks:

– the software license in many cases is expensive – especially when the license is commercial;

– dedicated software analytical solution has – in many cases – a lot of features and built-in tools, which are not necessary during normal business activity of small enterprises;

– the complicated software solution must be deployed in the organization, what is cost- and time-consuming, what is more, there are sometimes conflicts with legacy systems;

– business staff of the organization must be trained in order to be able to work with new software solution.

According to presented disadvantages it is worth to mention, that all of them generate expenses, so often deployment of "big" software supporting system is out of scope of small enterprises, which have to be cost-effective to have competitive advantage.

Potential solution according to data analysis and data visualization for small companies, can be found in "open-source" software projects, which can be successfully applied – under certain conditions – as an alternative to commercial software. One of the well-known environment, which can be used in the proposed area and which is "open-source" is Python. The main reason, the Python can be taken under consideration as an alternative to "paid" software, are connected to the fact, that proposed programming language is very popular nowadays (JANUSCHOWSKI et al. 2019). That popularity is connected mainly to the features of Python and its versatility (SAABITH et al. 2019). Another factor, which can motivate the organization to choose Python, is huge repository of additional libraries, which extend its scope of use widely. Some of the extensions are dedicated to data analysis and visualization area, e.g. "Pandas", "Statsmodels" or "scikit-learn". Proposed libraries give the user access i.a. to different forecasting method or machine learning tools (BROWNLEE 2020).

Of course making software solution in the area of data analysis and visualization in the form of Python's script(s) needs knowledge and ability according to business analytics, as well as Python's programming language. This is certainly, out of scope for typical business user of the enterprise. However, the data science labor's market is steadily growing (DONATI, WOOLSTON 2017), so hiring data scientist, who works as a freelancer, should not be a problem to create integration mechanism and to develop appropriate data analysis method. So, here is the point, where the researches briefly described in the current paper were started.

To sum, it seems that good data analytical solution for business people can be developed in form of Python's script(s), but those scripts have to be executable from "classical" tool, which is Microsoft Excel in the case. The possibility of performing integration between both proposed environments can be very attractive to business user and allows to perform sophisticated data analysis task with control mechanisms shared under traditional spreadsheet.

Researches performed so far were focused on basic integration of Microsoft Excel with Python environment. The main idea here, is to allow the business user to:

– prepare source data set under spreadsheet environment;

– perform initial transformation of the data by using traditional mechanism available under Microsoft Excel;

– send chosen data to the data analysis Python's script(s) by using developed graphical user interface (GUI) under spreadsheet;

– retrieve results generated by the Python's script(s) under Microsoft Excel and use them to support decision process.

In the current paper, the results of the researches and the developing process of integration mechanism between Microsoft Excel and Python interpreter are presented. To prove that kind of integration can be valuable to business user, the short and simple case study has been prepared. In the end of the paper, the pros and cons of current software solution are presented, as well as, further research are described.

Material and Methods

The purpose of the studies is to develop a solution for integrating the Microsoft Excel environment and Python interpreter and to verify the validity of its use especially to business user. An important feature of the solution is its accessibility, ease of use and the possibility for unskilled programming workers to use through integration the analytical methods available in the Python package, such as: ARIMA (AutoRegressive Integrated Moving Average model*)* or Deep Neural Networks. Development of integrating solution according to data scientist point of view is presented in the "Solution development" section of the paper. Applying of proposed solution by the business user is presented in the same section as well.

An example of integration of both environments was presented by completing a case study that addresses the problem of time series prediction. In the presented case study a forecasting process based on data from two time series with different characteristics was conducted. The calculations for the selected, more traditional forecasting methods were performed in the Microsoft Excel, while the Python interpreter executed the ARIMA forecasting method by performing implemented script. After the forecasting results have been obtained, the forecast errors of the methods were calculated and a comparative analysis of the accuracy of used prediction methods was performed.

Data characteristics

The data used in the case study were shared by a small production company, which manufacturing the range of products for the horticultural industry. The data are related to the value of demand determined on the basis of the purchasers' declarations and the unit costs incurred for the production of the selected product. Company name and details of the data provided were classified by the company.

The data provided by the company was recorded at the end of each month for a period of 5 years. Thus, a total of 60 values were obtained, as shown in the charts and analyzed to determine the characteristics of the time series. In the case of the demand, which was firstly considered, an upward trend was observed and there was a clear seasonal fluctuation (Fig. 1).

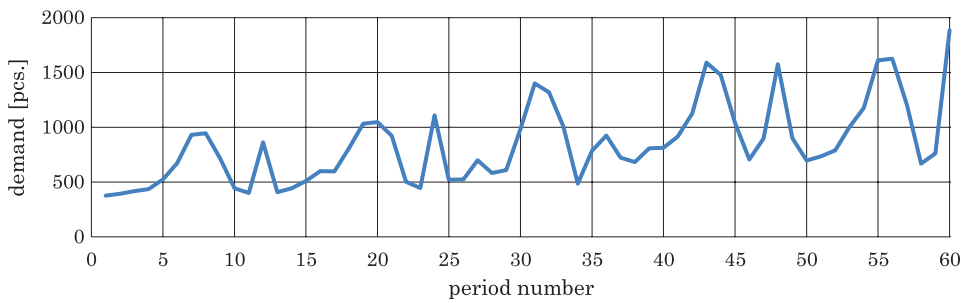

Fig. 1. Observed demand values
Source: own study.

However, in the case of unit costs (Fig. 2) no periodic fluctuations were observed. The only characteristics according to this time series were random fluctuations and, as in the case of demand, upward trends.

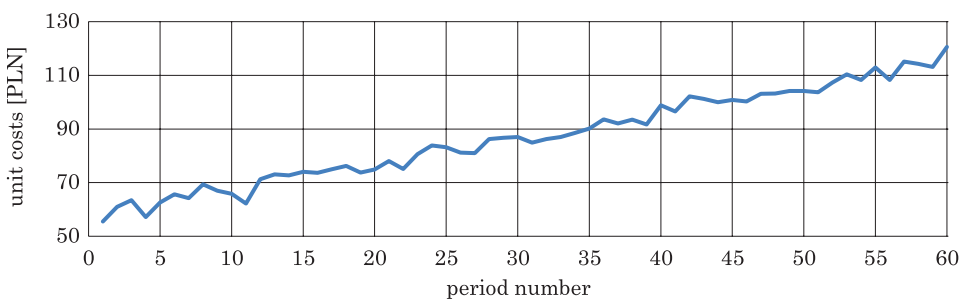

Fig. 2. Observed unit production costs values
Source: own study.

Selection of forecasting methods and metrics

The study uses the following methods of forecasting: naive, moving average, polynomial trend line, autoregression and ARIMA. The first four methods were implemented in a Microsoft Excel environment to illustrate, that some data forecasting methods can be developed by a business spreadsheet user itself. However, the ARIMA method was implemented by the business analyst by using the specific Python's libraries and workflow described by the script. All of the indicated forecast methods were applied and input data related to demand and product's unit costs were used in the prediction of time series.

The purpose of the ongoing case study was to make a forward-looking forecast for the next 12 periods (one year). Due to the fact that the chosen methods of forecasting are well known and described in the literature, their description has been omitted and only the parameters selected for them are indicated in the paper.

According to the naive method, it was assumed that the forecast value in the given period is equal to the value recorded in the time series by one period earlier (SHIM et al. 2012, HYNDMAN, ATHANASOPOULOS 2018). For example, the demand value from the first period is the forecast for the second period. In the moving average method, a lag equal to 6 periods was used, i.e. the arithmetic mean was calculated from the previous 6 parts of the time series (HANSUN, KRISTANDA 2017). In a forecasting using a trend line, a polynomial 6-degree trend line was used. The auto-regression method uses a delay parameter of 6 periods (PENA-SANCHEZ, RINGWOOD 2017). For the ARIMA method, the optimal choice of method parameters, i.e. co-efficients p, d and q (SIAMI-NAMINI, NAMIN 2018) has been obtained by using a grid search (RASCHKA, MIRJALILI 2019).

The indicated parameter values for each forecast method were selected after a series of initial calculations using different values of parameters and comparing the accuracy of the generated forecast. The delay in the moving average method was verified for values of 3 and 6 periods (months), the degree of the polynomial trend line was verified for values between 2 and 6, and the delay in the autoregressive model was verified for values of 3 and 6 months. In the ARIMA method, models were tested by using the following parameters during grid search: p – from 0 to 20, d – from 0 to 2, q – from 0 to 5.

The final values of the parameters for both time series have provided forecasts with the highest accuracy. The accuracy of the methods was measured by calculating forecast error values. Among commonly used forecasting accuracy measures are: the mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE) or mean absolute percentage error (MAPE). All mentioned metrics have a common disadvantage – their values can range between zero and +infinity. Despite, MAPE is one of the most popular measures of the forecast accuracy, especially among industry practitioners. What is more,

it is scale-independent, easy to interpret and intuitive (KIM, KIM 2016). In the presented case MAPE calculated according to formula 1 (CHICCO et al. 2021) was used:

$$\text{MAPE} = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{y_i - y_i^P}{y_i}\right| \cdot 100\ [\%] \tag{1}$$

where:

m – the number of periods available in the time series,
y_i – the actual value for period i,
y_i^P – the forecast value for period i.

After determining the errors of each forecast method by using MAPE metric, a comparative analysis was performed and the most effective method was identified, in addition it was verified whether this method could be implemented directly by the business user in the Microsoft Excel environment, or whether it is necessary to integrate the spreadsheet with Python environment.

The solution development

The preparation of the solution for environments integration, as part of the case study, was performed to provide the business user with easy and convenient access to more complex method of data analysis not available in the Microsoft Excel package. In this case this method was ARIMA that enables forecasting of time series data. The workflow of a data analyst in the process of preparing an environment integration solution included the following tasks:

– review and analysis of input data and business problem;
– selection of an advanced method of forecasting applicable in a Python environment but not available in the Microsoft Excel;
– development of a Python script with source code responsible for the executing the selected forecasting method;
– test and verification of the script operation;
– creation of sheet template for the data in an Excel workbook;
– development of a macro in VBA in the same Excel file;
– adjusting of fragments of Python's script and macro to ensure their correct cooperation.

The source code responsible for executing the ARIMA method (SWAMYNATHAN 2019), on the indicated and temporarily saved data in CSV file, is stored in the Python's script. The script also contains command from "XLWings" library (EHRHARDT et al. 2017). The library allows the developer to return the results of calculations to the Microsoft Excel environment. The detailed source code of the script is presented in the appendix of the current paper. The script itself

was developed under Anaconda environment with Visual Studio Code Editor. The Visual Studio Code Editor is an Integrated Development Environment (IDE) which supports the developer during implementation process of the script (SPEIGHT 2021).

The developed Python script, which realizes the ARIMA predication method is rather typical script, which consist of the following sections:

– load data section, where "Pandas" (NELLI 2018) library is imported and data stored in the "Input_dataset.csv" temporary file are imported as a Python's "DataFrame";

– split data section, where the data stored in "DataFrame" are split into train and test set; at the current version of the script the size of the train is equal to 80% of data, other 20% of data creates test dataset; in the future version of the solution it is planned, that business user itself will use appropriate Graphical User Interface (GUI) elements to select desired size of train and test datasets;

– making ARIMA model section, here main part of the script is implemented; because the ARIMA method needs, that three parameters p, d and q must be set, the grid search has been implemented to help the business user find most suitable parameters; the using "for" loop gives possibility to test whole range of parameters to determine the best ARIMA model; "best" here means the model with lowest mean absolute percentage error (MAPE) calculated between prediction of the chosen model and real data stored in the test set; in the current version of the script the tested parameters are as following: p – from 0 to 20, d – from 0 to 2, q – from 0 to 5, so during execution of the script 378 models were generated and tested against test data to find the optimal set of coefficients;

– generating final prediction results section, in the section the optimal ARIMA model found is used to generate prediction of the examined phenomenon and store the result in the chosen variable; generating the prediction is main goal of the business user;

– exporting prediction results, current section used the "XLWings" library to pass the prediction results stored in the chosen variable to the new sheet of Microsoft Excel named "Forecast", to give the business user ability to check best ARIMA model prediction;

– cleaning section, the last section of the Python's script consists of Python's command, which goal is to remove temporary file "Input_dataset.csv".

The workbook designed in Microsoft Excel consist of a worksheet named "Data" which serves the business user to enter the input data for current issue. Moreover, the worksheet also includes a Graphical User Interface (GUI) element – a button (form control) that launches a developed VBA script enabling a business user to interact with prepared Python's script. The Microsoft Excel software solution view is shown on the Figure 3.

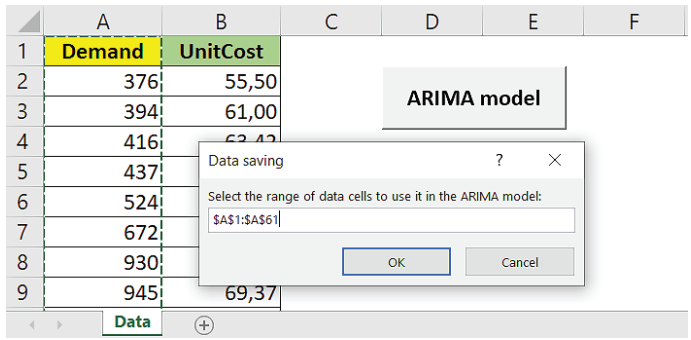

Fig. 3. Data saving in "Data" worksheet
Source: own study.

The macro developed in the Visual Basic Editor (VBE) contains two sections:

– the first one is used for saving the selected data in "Data" worksheet in temporary file named "Input_dataset.csv" as comma separated values (CSV); the business user selects the range of data cells by using specific dialog box (Fig. 3);

– the second one is responsible for proper launch of the Python's interpreter and selection of the script that should be run by the interpreter; in this part of the VBA code, access paths to the interpreter and the script that executes the ARIMA method are assigned to individual variables.

The source code of the VBA macro is available in the appendix of the paper.

Finally, the workflow of the business user according to developed Excel and Python integration solution is shown in Figure 4.

The workflow of a business user while using the developed solution is rather straight-forward and include the following activities:

– open the Excel file with macro provided by business analyst;

– import, copy and paste or enter the data into "Data" sheet;

– run a macro using the button (the graphic element of the graphical user interface);

– indicate the range of cells with data to be used by the Python script in the prediction process;

– review and analysis of the results displayed in the "Forecast" sheet, when Python's interpreter finishes script execution.

To sum, the developed Python's script realizes the prediction, based on data selected in Microsoft Excel by using the best found ARIMA model and bring the results of the prediction process directly to business user in Microsoft Excel sheet by using proposed integration mechanism. All source code for an integration solution (VBA macro and Python script) are presented in the appendix.

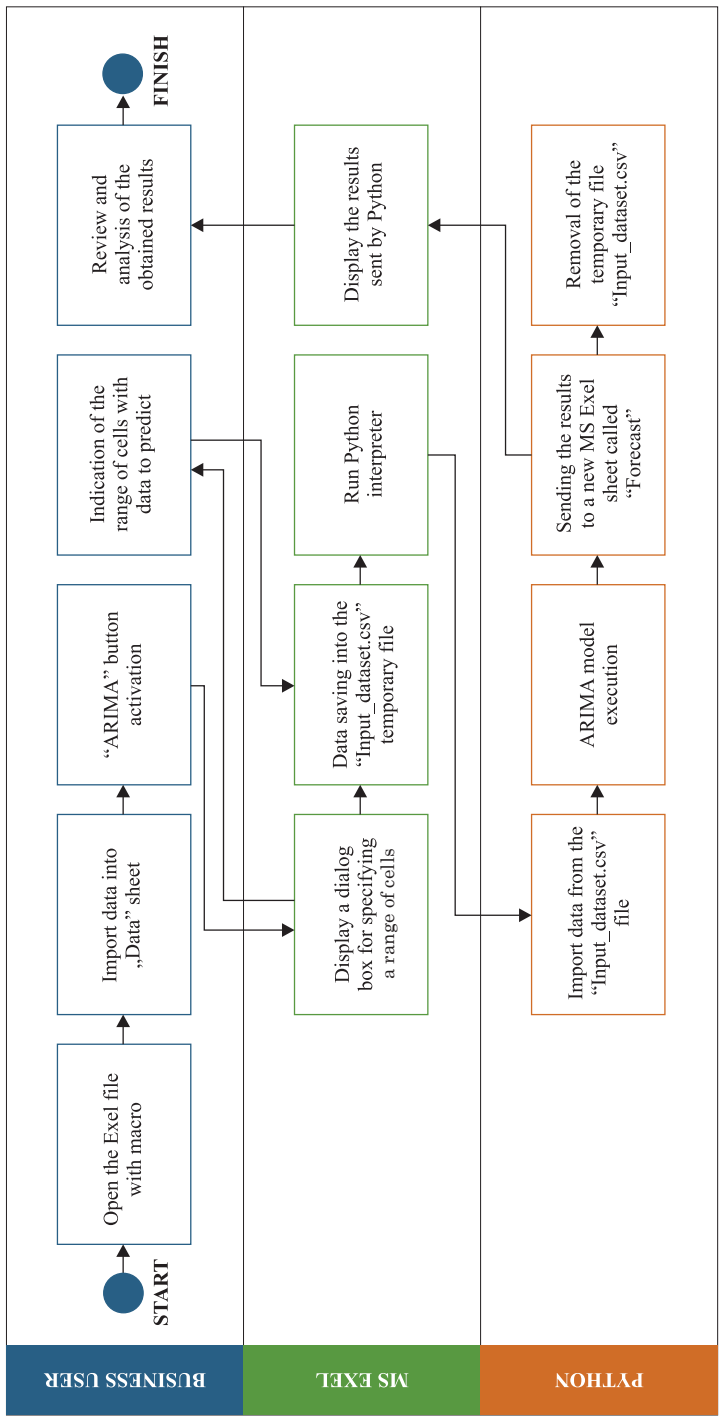

Fig. 4. Activity Diagram showing the business logic of the prepared solution
Source: own study.

Results and discussion

The execution process of developed software support for business user is very simple and intuitive. Business user needs to:

– put all developed files into one directory: the first file is Microsoft Excel's file with embedded VBA macro, which serves as a data input tool, as well as interface, which integrates spreadsheet environment with Python; the other file(s) are Python's script(s), which are developed as a chosen data analysis and/or visualization methods, which brings the final result of analysis back to Microsoft Excel, if all prepared methods are executed;

– open Microsoft Excel file and put all input data, which describes the problem into one column of the prepared "Data" sheet as shown on the Figure 3;

– set the parameters of the developed methods, if able, by using appropriate GUI elements and execute developed "Python" methods by using the button shown on the Figure 3;

– wait until all data analysis methods are executed by the Python's interpreter;

– review the results of the analysis by examining output data sheet called "Forecast" in that case in the Microsoft Excel.

In the following section of the paper, the results of analysis of case study are presented and discussed.

Forecast values for time series related to demand and unit production costs were determined with the use of selected methods and then their accuracy was assessed. The evaluation of a given method consisted of calculating the mean absolute percentage error (MAPE) and indicating the value of the standard deviation of error (SD). The data was divided into training set and test set. These sets consisted of 48 and 12 periods consecutively, regardless of the method chosen. The values of forecast errors for each of the sets and each analysed variable are presented in Table 1 and Table 2. Among the methods, only those were indicated, which business user can develop directly in the Microsoft Excel environment, ignoring the ARIMA method. The results of ARIMA method are presented later in this chapter.

Table 1

Demand and unit costs – training set [%]

Method	Demand		Unit costs	
	MAPE	SD	MAPE	SD
Naive	27.20	28.46	3.17	2.93
Moving average	29.80	24.52	3.72	2.59
Trend line	33.74	23.64	24.55	31.35
Autoregression	24.73	32.13	2.02	1.96

Source: own study.

Table 2

Demand and unit costs – test set [%]

Method	Demand		Unit costs	
	MAPE	SD	MAPE	SD
Naive	30.63	27.23	2.68	2.15
Moving average	36.50	21.92	7.42	4.20
Trend line	110.06	76.06	196.10	50.02
Autoregression	23.15	19.50	1.84	0.94

Source: own study.

After analysing the results presented in tables 1 and 2, it was found, that the most accurate of the forecasting methods was the autoregressive method. According to demand and unit production costs, this method is characterized by the lowest values of forecast errors and standard deviation. The standard deviation value calculated for the demand training set is an exception – in this case it is the largest. Forecasts determined by the autoregression method for each of the time series are shown on the Figures 5 and Figures 6. Due to the delay of six periods used, the method is calculated from period 7th.

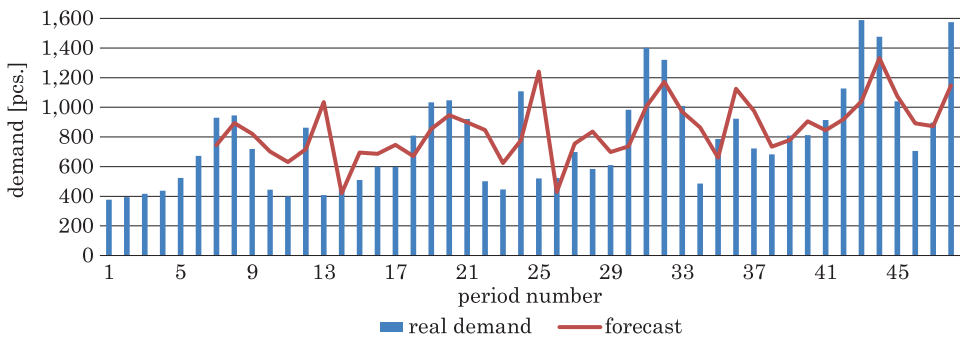

Fig. 5. Historical data and forecast of demand
Source: own study.

The values of the forecasts generated by the autoregressive method were compared with the values returned as a result of running the script containing the ARIMA model. In this case, the forecast errors were calculated for the last 12 periods of the time series, because this part of data was used as a test set of the ARIMA model.

The mean absolute percentage error for the value of demand in the case of the autoregressive method was 23.15%, while in the case of the ARIMA model 10.39%. Thanks to the application of the ARIMA method it is possible

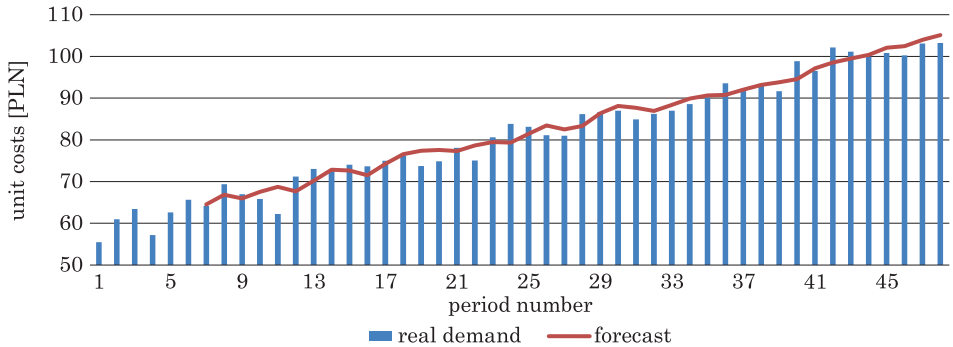

Fig. 6. Historical data and forecast of unit production costs
Source: own study.

to generate a forecast with nearly a half less error. In the case of the values of the unit production costs again the ARIMA model turned out to be more effective, the calculated MAPE was 1.30%, while in the autoregressive method its value was 1.84%. However, here the improvement in the effectiveness of the forecast as measured by the percentage error value is not so spectacular. The forecasts determined by the autoregression method and by the ARIMA model in the given period are shown in Figures 7 and 8 respectively.

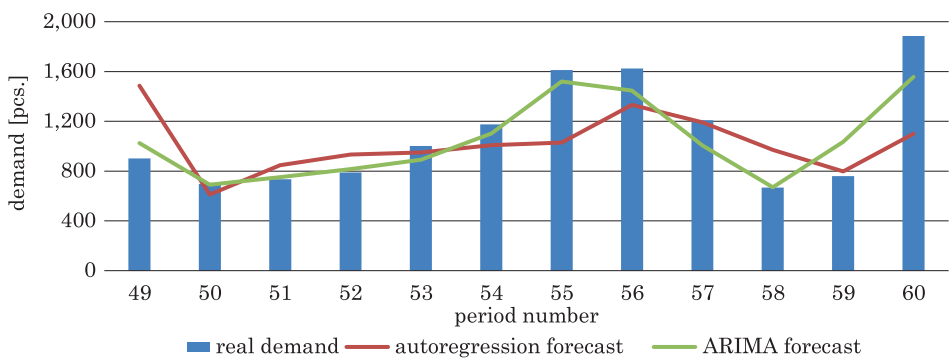

Fig. 7. Comparison of autoregressive and ARIMA forecasting methods for demand
Source: own study.

The analysis of the results carried out as part of the case study confirmed, that of all the prediction methods used, the ARIMA model provides the best results. However, the obtained average forecast error value equal to 10.39% is relatively high, which means that a business user should review other forecasting methods and try to select their parameters properly to achieve better accuracy.

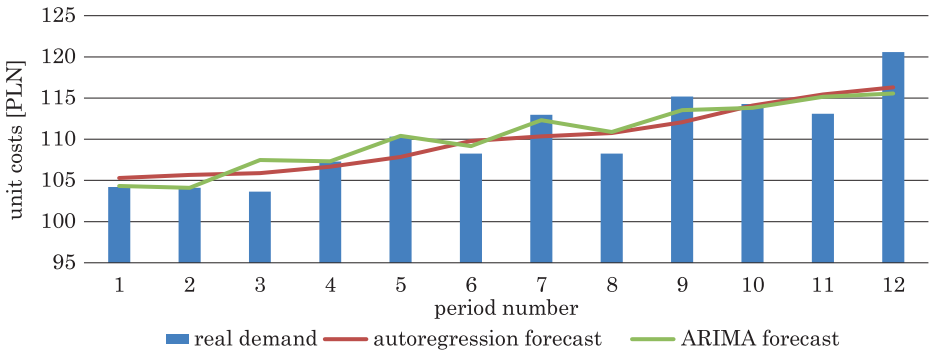

Fig. 8. Comparison of autoregressive and ARIMA forecasting methods for unit production costs
Source: own study.

Summarizing the completed case study, the use of the integration mechanism of the Microsoft Excel and Python environments made it possible to obtain prognostic models more effective, than it is possible to achieve with the use of the Microsoft Excel alone. It is also important, that the many processes of data analysis methods developed in the form of Python's scripts are available in Python's repositories. The business user may applied selected data analysis workflows to any input data stored in a spreadsheet, which describe problem considered in the enterprise, by using proposed integration mechanism, without the need of knowledge the Python or VBA languages.

Conclusions

Researches performed so far, clearly show that building software integration between Microsoft Excel and Python is possible. The result of the integration allows the business user to prepare input data set and then execute sophisticated data analysis method described in Python's script and see final result of the analysis, without Python's programming knowledge and skills. What is more, building appropriate graphical user interface in Microsoft Excel can give the business user possibility to control developed Python's scripts in easy way. The business user is able to select specific methods stored in the scripts or pass specific parameters of the chosen methods to change the way of execution. Moreover, developed integration mechanism can be offered to different business user as a business process analysis support tool using incorporated Python's data analysis and visualization methods.

It is worth to mention, that implementation of proposed solution still needs Python's developer, who will implement or adjust appropriate Python's script

to solve selected business problem. What is more, the developer will have to develop GUI under Microsoft Excel to give the business user ability to control the scripts, as well as, program VBA macros to connect spreadsheet's control elements to Python scripts and methods. So, the Python's developer is essential to create working efficient software tool, but if the development process is finished, the software tool can be used by typical business user, who is able to use Microsoft Excel. Another issue the research team had to face was incompatibility between specific versions of Python's libraries during integration mechanism development. For example, the problem still exists if developer uses newest "Pandas" and "XLWings" libraries. To resolve the problem, developer needs to downgrade problematic libraries to different version, what makes development process of integration solution harder to perform.

Further research according to presented integration mechanism will be oriented to implement different forecasting method e.g. by using neural networks, to improve forecasting results of product's demand prediction. In this case, it will also be necessary to introduce additional measures to ensure a more efficient assessment of the accuracy of forecasts, for example the coefficient of determination (also known as R-squared or R^2) or the symmetric mean absolute percentage error (SMAPE). Another field of interest is building GUI in Microsoft Excel to give business user all control elements needed to fully control implemented Python's methods and their parameters. The more control mechanism the business user have, the developed Python's scripts will be more versatile and adjusted to specific business problem the business user wants to solve. Of course, the integration mechanism can be performed to solve different type of data analysis problems like: classification task, clustering or regression problems by using methods which are unavailable or hard to implement under Microsoft Excel environment. Because the problem described in the current case study was practically oriented, further research can be based on benchmark dataset published by UC Irvine Machine Learning Repository or Kaggle to compare results of proposed solution with some other methods or results presented by other authors in the literature, to obtain optimal business process supporting tool and to show value-added of the proposed solution.

To conclude, research in that field are consider as important, because the final goal of the current researches is to create and describe the whole development process of the integration between both environments. Performing specific activities described in the proposed development process, should result in software sharing most important and accurate data analysis and visualization methods available in Python to Microsoft Excel business user.

References

BIRCH D., LYFORD-SMITH D., GUO Y. 2018. *The Future of Spreadsheets in the Big Data Era*. Proceedings of the EuSpRIG 2017 Conference "Spreadsheet Risk Management". Imperial College, London, UK.

BROWNLEE J. 2020. *Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future*. Machine Learning Mastery, San Francisco, p. 2-5.

CHICCO D., WARRENS M.J., JURMAN G. 2021. *The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation*. PeerJ Computer Science, 7: e623. DOI: https://doi.org/10.7717/peerj-cs.623.

DONATI G., WOOLSTON C. 2017. *Information management: Data domination*. Nature 548: 613–614. DOI: https://doi.org/10.1038/nj7669-613a.

EHRHARDT M., GÜNTHER M., TER MATEN E.J.W. 2017. *Novel Methods in Computational Finance*. Springer, Cham, p. 545.

HANSUN S., KRISTANDA M.B. 2017. *Performance Analysis of Conventional Moving Average Methods in Forex Forecasting*. Proceedings of 2017 International Conference on Smart Cities, Automation & Intelligent Computing Systems. Yogyakarta, Indonesia.

HYNDMAN R.J., ATHANASOPOULOS G. 2018. *Forecasting: Principles and Practice*. 2nd ed. OTexts, Melbourne, p. 57-58. https://otexts.com/fpp2/ (access: 18.06.2021).

JANUSCHOWSKI T., GASTHAUS J., WANG Y. 2019. *Open-Source Forecasting Tools in Python*. The International Journal of Applied Forecasting, 55: 20-26.

KIM S., KIM H. 2016. *A new metric of absolute percentage error for intermittent demand forecasts*. International Journal of Forecasting, 32(3): 669-679.

KOH L., ORZES G., JIA F. 2019. *The fourth industrial revolution (Industry 4.0): technologies disruption on operations and supply chain management*. International Journal of Operations & Production Management, 39(6/7/8): 817-828.

KURZAK L. 2012. *Importance of forecasting in enterprise management*. Advanced Logistic Systems, 6(1): 173-182.

NELLI F. 2018. *Python Data Analytics with Pandas, NumPy and Matplotlib*. 2nd ed. Apress, Rome, p. 143-145.

PENA-SANCHEZ Y., RINGWOOD J. 2017. *A Critical Comparison of AR and ARMA Models for Short-term Wave Forecasting*. Proceedings of the 12th European Wave and Tidal Energy Conference, Kildare, Ireland.

RASCHKA S., MIRJALILI V. 2019. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. 3rd ed. Packt Publishing Ltd., Birmingham, p. 207-211.

SAABITH A.L.S., FAREEZ M.M.M., VINOTHRAJ T. 2019. *Python Current Trend Applications – An Overview*. International Journal of Advance Engineering and Research Development, 6(10): 6-12.

SHIM J.K., SIEGEL J.G., SHIM A.I. 2012. *Budgeting Basics and Beyond*. 4th ed. John Wiley & Sons, Inc., Hoboken, p. 277-279.

SIAMI-NAMINI S., NAMIN A.S. 2018. *Forecasting economic and financial time series: ARIMA vs. LSTM*. http://arxiv.org/abs/1803.06386 (access: 17.06.2021).

SPEIGHT A. 2021. *Visual Studio Code for Python Programmers*. John Wiley & Sons, Inc., Hoboken, p. 3-49.

SWAMYNATHAN M. 2019. *Mastering Machine Learning with Python in Six Steps. A Practical Implementation Guide to Predictive Data Analytics Using Python*. 2nd ed. Apress, Bangalore, p. 234-243.

WINKOWSKI C. 2019. *Classification of forecasting methods in production engineering*. Engineering Management in Production and Services, 11(4): 23-33.

Appendix

VBA source code:

```
Sub startPython()

Dim Rng As Range
Dim WorkRng As Range
Dim xFile As Variant
Dim xFileString As String
Dim newFile As Workbook

Dim objShell As Object
Dim PythonExe, PythonScript As String

'Data saving in csv
On Error Resume Next
xTitleId = "Data saving"
Set WorkRng = Application.Selection
Set WorkRng = Application.InputBox("Select the range of data cells to use
it in the ARIMA model:", xTitleId, WorkRng.Address, Type:=8)

If Err.Number = 424 Then
    Exit Sub
End If

Workbooks.Add
Set newFile = ActiveWorkbook
WorkRng.Copy Application.ActiveSheet.Range("A1")

newFile.SaveAs "D:\...\" & "Input_dataset.csv", xlCSV
newFile.Close

'Running Python script
ActiveWorkbook.Save

Set objShell = VBA.CreateObject("Wscript.Shell")
PythonExe = """C:\...\python.exe"""
PythonScript = "D:\...\arima_model_1.py"
objShell.Run PythonExe & PythonScript

End Sub
```

Pyton's source code:

```
import pandas as pd

#data import
df = pd.read_csv('Input_dataset.csv')

#train and test set
a = df.size
train_data=round(0.8*a)
test_data = a - train_data

train = df.iloc[:-test_data]
test = df.iloc[-test_data:]

#ARIMA parameters test
import warnings
warnings.filterwarnings("ignore")
from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter('ignore', ConvergenceWarning)

from statsmodels.tsa.arima.model import ARIMA
best_error=1;
best_p=0;
best_d=0;
best_q=0;
for q in range(0,6):
    for d in range(0,3):
        for p in range(0,21):
            try:
                model=ARIMA(train,order=(p,d,q), trend='n',
enforce_invertibility=False, enforce_stationarity=False)
                model=model.fit()
                #prediction
                start=len(train)
                end=len(train)+len(test)-1
                forecast = model.predict(start=start,end=end)

                #scoring
                from sklearn.metrics import mean_absolute_percentage_error
                mape=mean_absolute_percentage_error(forecast,test)
                print("Current model: ",p,",",d,",",q," - error: ",mape)
                if (mape<best_error):
                    best_error=mape
                    best_p=p
                    best_d=d
                    best_q=q
            except:
                continue
```

```
print("Best ARIMA model p,d,q - error: ",best_p,",",best_d,",",best_q," -
",best_error)

model2=ARIMA(train,order=(best_p,best_d,best_q),trend='n',enforce_inverti-
bility=False, enforce_stationarity=False)
model2=model2.fit()

#prediction
start=len(train)
end=len(train)+len(test)-1
forecast2 = model2.predict(start=start,end=end)
print('Best ARIMA model: (',best_p,',',best_d,',',best_q,')
- predictions:')
print(forecast2)

#forecast export to Excel file
import xlwings as xw
wb = xw.Book('SWD1.xlsm')
ws2=wb.sheets.add('Forecast')
ws2['A1'].value = forecast2

#delete temporary data file
import os
os.remove('Input_dataset.csv')
```