Hyperautomation (Automated Decision-Making as Part of RPA)

Volker Liermann, Sangmeng Li, and Johannes Waizner

1 Introduction

1.1 Initial Situation

Robotic Process Automation (RPA) is part of every digitalization strategy in banks and insurance companies. RPA has in recent years launched a fundamental new paradigm on how to automate and optimize processes.

RPA has been particularly successful in automating processes across systems (avoiding system discontinuities) and processes with clear "clicking" patterns. The automation potential reaches a limit when a manual decision (by a human) is still needed in the process.

In Fig. 1, the difference between classic and modern RPAis shown. Modern RPA widens classic RPA by the aspect of automated decision-making (some sources call this "hyperautomation" see Gartner, Gartner Top 10 Strategic

V. Liermann (✉) · S. Li · J. Waizner
ifb SE, Grünwald, Germany
e-mail: Volker.Liermann@ifb-group.com

S. Li
e-mail: Sangmeng.Li@ifb-group.com

J. Waizner
e-mail: Johannes.Waizner@ifb-group.com

V. Liermann and C. Stegmann (eds.), *The Digital Journey of Banking and Insurance, Volume II*, https://doi.org/10.1007/978-3-030-78829-2_16

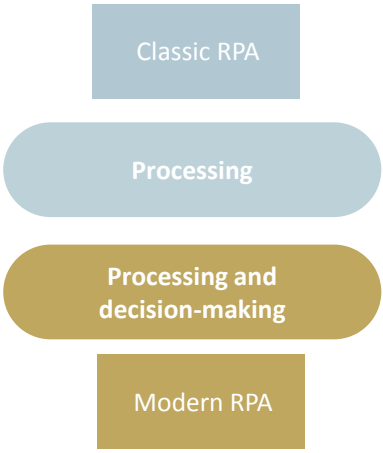

Fig. 1 Modern RPA (© ifb SE)

Technology Trends for 2020 (2019) and Gartner, Gartner Top Strategic Technology Trends for 2021 (2020)).

The adding of (automated) decision-making is not a one-step process. The recent projects have shown that a step-by-step approach delivers the best results and the highest acceptance. In Sect. 2, the idea of how to address this challenge is illustrated in more detail.

1.2 Structure of the Article

Section 2 introduces the general idea and the step-by-step approach to higher levels of automation. RPA and the software tool UiPath are very briefly introduced in Sect. 3. Section 4 discovers the machine learning aspects needed in the context of this use case. The following Sect. 5 gives insights into a practical application, i.e., how to integrate machine learning (Python-based model) into RPA (UiPath). Section 6 summarizes the key findings.

2 General Idea

In Sect. 1.1, we highlighted that a step-by-step automation of the processes works best. In this section, we will discuss different structures for taking these steps.

In Fig. 2, the five-step decision automation model is shown. The different levels show how to move step by step from manual decisions to autonomous

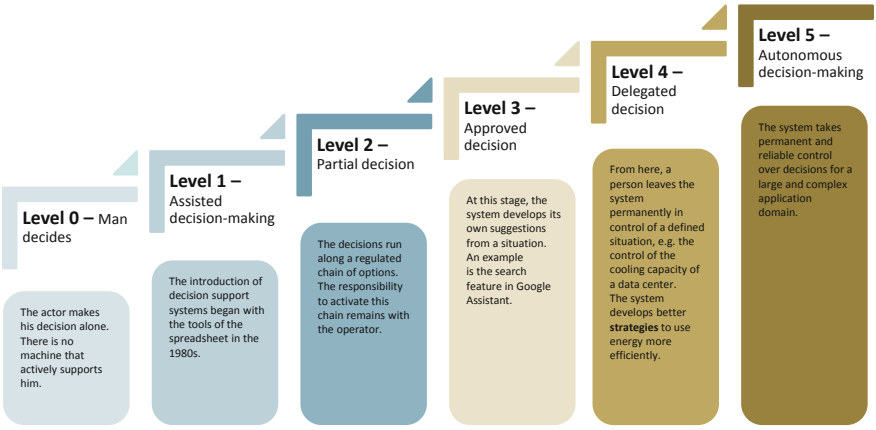

Fig. 2 Five-step decision automation model (© ifb SE)

decision-making. The five-step decision automation model is dealt with in detail (Bitcom 2017).

The automation of a process beyond classic RPA is challenging and needs to be approached step by step. While Fig. 2 gives a detailed structural view, the main blocks are described in Fig. 3.

The foundation for any model to work is formed in level A "learning." The relevant data is collected for a model calibration. The labels are especially hard to find in a structured form. Classic RPA can help with this task because RPA can seamlessly collect and save decisions made by a human in a user-friendly way. In traditional processes, this decision-driving information is—in most

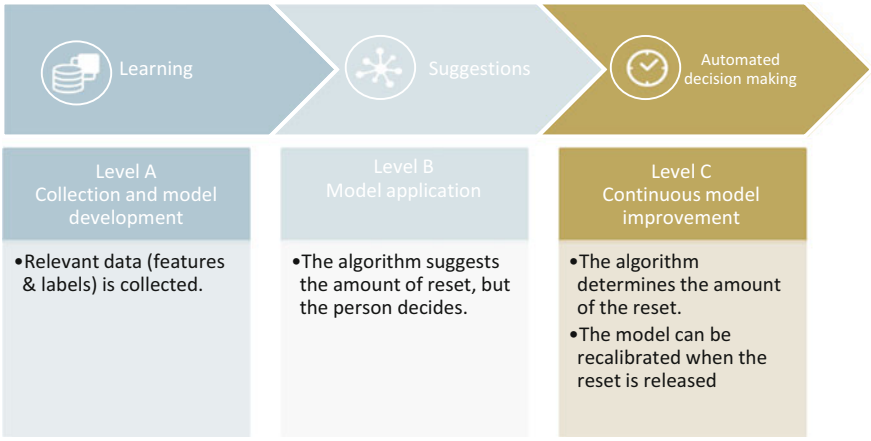

Fig. 3 Blocks for a step-by-step implementation (© ifb SE)

cases—lost after the process management has used it to select the further steps.

To ensure the correctness of the proposed decisions and to ensure the person involved has a good understanding of the status of the process automation level, the model proposes a decision in level B "suggestions," but the human can overrule the model decision. When the model has proposed reasonable decisions for a long time, then the last step (level C) can be entered. The process can run fully automated.

3 RPA and UiPATH

UiPath is one of several tools that can implement RPA procedures. It draws its strength from a graphic user interface that not only lets you string together abstract commands but even simply record mouse clicks that encompass and bring together the control of many different programs by a plethora of third parties. It is therefore easy to create a classic RPA robot that contains, for example, steps going through OfficeSuite products, web browsers, and SAP distributions all at once.

Regarding modern RPA as outlined in Fig. 1, UiPath offers, for example, the integration of external programming languages like Python that can assist the human user in the decision-making process by providing appropriate, custom algorithms. More details are given in the next section.

4 Machine Learning/Data Analysis

4.1 From Data Collection to Full Automation

Figure 3 highlights three steps or phases in the development of a fully automated system. Even in such a system, the data collection process, which is the key part of level A, does not stop. Data collection and usage can therefore themselves be structured into three similar phases. Let us take a use case of, for example, announcements of future invoices arriving at a motor vehicle leasing company. Said announcements present an approximate price corresponding to a specific vehicle and maybe contain additional information like billing addresses, etc., for a purchased vehicle. This is necessary for the leasing company to make provisions to be able to comply with future payments. However, the announced price may be merely a preliminary value and even faulty at times. The receiver of the announcement, henceforth called the user,

will have to decide the actual amount of provisions that the company needs to make. This can range from simply approving the announced amount or double checking it against a list price and selecting the correct one, through to remembering and analyzing previous purchases.

As good and informed decisions are based on sufficient amounts of adequate data, the first step is the pure collection of such data.

Phase one will therefore be a period in which relevant data will be collected, manually processed, and evaluated until one achieves a dataset of input values, e.g., announced invoiced amounts, and validated output data. The latter here are the amounts that are validated, approved, and eventually paid. While the collection of the input data can be automated at this point, the generation of the output data is still a fully manual process.

Once the first phase is completed to a satisfactory degree, one can start to incorporate more automation into the process. In step one, not only was the decision-making excluded from automation, the at times cumbersome research of plausible pricing and comparison to empirical values was a manual process as well. The next phase tackles the latter issue.

Phase two uses the data collected in phase one to aid the user in his or her decision-making process by predicting and proposing a monetary amount for provisioning.

This is the point where hyperautomation enters the playing field. Simply collecting data, pushing it through a static process and ultimately pasting it into a sought-after format is the traditional or standard form of automation. In hyperautomation, a combination of different tools comes into play (Maddox 2019). Here, we combine RPA with a machine learning model, which analyzes the previously collected and prepared empirical data. Triggering the use of that model automatically within the traditional RPA framework brings us to a hyperautomated process. The final result of phase two is a semiautomatic procedure, where the user is predominantly just left with the decision-making process itself. That is whether to confirm the amount suggested by the model for the actual provisions, or to choose the amount extracted from the email invoice announcement. Exceptions might be vastly unintuitive values for both the proposed and extracted values. In that case, the user could opt for a third option, which is to do some manual research after all and set the actual amount him or herself. Note that during phase two the data gathering process of phase one continues, albeit in a more automated fashion. The training of the model should then also be updated in regular intervals incorporating the new data.

Phase three aims at further and ideally full automation. The goal is to eradicate a user as an ongoing decision-maker, instead letting the robot work its "magic" to the fullest.

A natural piece of information that is gathered in phase two is how often the user chooses to follow the model prediction and how many times he or she does not. This is a measurement of how well the model works. This, of course, is also influenced by the amount of data the model is based on, which keeps on increasing over the course of time. Assuming there are no fundamental mistakes in the implemented model, we can assume that the rate for choosing the predicted value increases, too. Not only this way, but also through the user's experience, we gain intuition and measurements regarding the quality of the model. This goes somewhat in the direction of cross-validation, i.e., testing the model on new data. Once it reaches a satisfactory degree, one can switch to always accepting the predicted value as the actual value that is reserved for provisions. At this point, only regular backtesting is required to make sure that the predictions and propositions of the model remain accurate.

4.2 Natural Language Processing for Data Extraction

Natural language processing, or NLP, is a field focused on the interaction between computers and human languages (Wikipedia 2020). A brief introduction to the models can be found in Section 4 in Liermann et al., Deep learning—an introduction (2019). Other applications of NLP can be found in Liermann and Schaudinnus, Real estate risk—appraisal capture (2019) and Schröder and Tieben (2021).

In our particular case, NLP can be a valuable asset—maybe even indispensable in practice. The key area for its application would here be the collection of data extracted from incoming emails that announce upcoming invoices. There are ways to force a certain format of the emails, for example by generating said format after submitting a form on a website, but those ways could be cumbersome, less user-friendly, or even redundant and resulting in more work and time for upkeep. Ideally, the sender sends the email in any way he or she likes, and the receiving computer is able to extract the sought-after information automatically. This is where NLP becomes necessary for automation.

NPL is not only a good idea in the final stages of implementation, nor should it be seen as the cherry on a cake. It could make a tremendous impact on automation at the earliest stage already, i.e., the collection of the initial data.

4.3 Features Engineering and Model Training

The dataset presented in this chapter is mockup data but reflects the structure of the real data. We simulate 13,730 datasets and an overview of the first nine samples is provided in Table 1.

The example data shown in Table 1 will be generated by an RPA tool (UiPath in our case). The RPA extracts the relevant data from incoming emails. The data collection process is covered for the user by the automated process. The extraction can be implemented rule-based (see Sect. 4.1) or can use more "error"-tolerant tools like NLP (see Sect. 4.2).

The column "Value" contains the value of invoice announcements. We aim to predict this column (label) based on the other columns (features). In this section, we are going to analyze the label and features and develop a machine learning method at the end. The following analysis is implemented by using R with ggplot2[1] and h2o library (H2O.ai 2019).

4.3.1 Label

The following analysis shows that the column "Value" contains only 12 categories among 13,730 data samples, see Fig. 4. It suggests that we consider it to be categorical (discrete) instead of numerical (continuous), which implies that we need to discover a classification model instead of regression. In the general case, we should extend a label to be continuous.

4.3.2 Features—Correlation

Feature selection is the process of reducing the number of input variables when developing a predictive model, to reduce the computational cost and improve the model performance. Although we do not have many features and samples, we still proceed with the feature selection to introduce a statistical-based approach by using a correlation matrix. Figure 5 illustrates the correlation matrix between all columns given in the dataset, where the correlation values are displayed by using different colors. Correlation-matrix-based feature selection is widely used to develop machine learning models, especially in case of Big Data. Clustering can be performed based on a correlation matrix, where similar variables can be grouped into the same cluster. In other words, the strongly correlated variables will be selected. In our example,

[1] https://ggplot2.tidyverse.org/, *ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphic*.

Table 1 Head of the dataset (© ifb SE)

Company	Object.Type	Object	Value	Address	Post code	City	Country	DUNSNr
Company 52	Airplane	Airbus A340-500	250,000,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Airplane	Boeing 747-100 Transport	650,000,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Tractor	John Deere 9570RX	600,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Truck	Mercedes Actros	500,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Airplane	Airbus A321-200 (Sharklets)	160,000,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Tractor	Case IH Stelger/Quadtrac 600	540,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Tractor	Fendt 1100 MT	400,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Airplane	Boeing 777-200 Passenger Model	550,000,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213
Company 52	Airplane	Boeing 727-100 Passenger Model	150,000,000	Tschamlerstr. 2	6020	Innsbruck	Austria	747,241,213

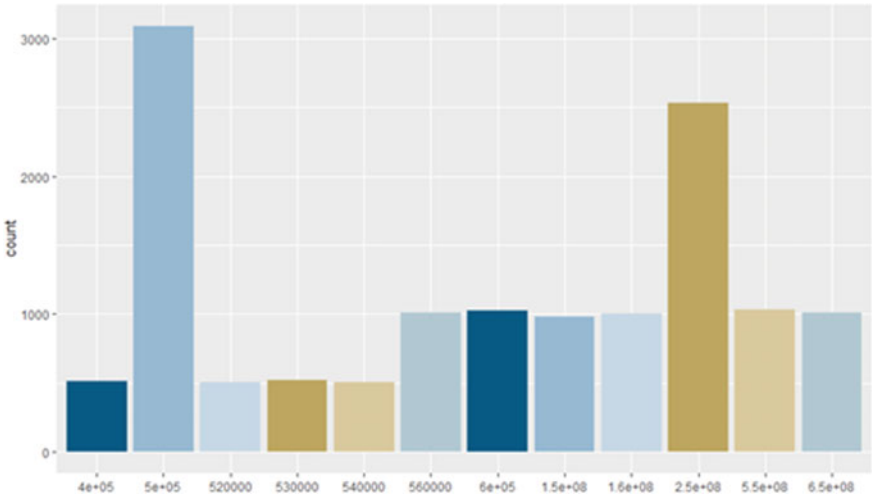

Fig. 4 Overview of label "Value" (© ifb SE)

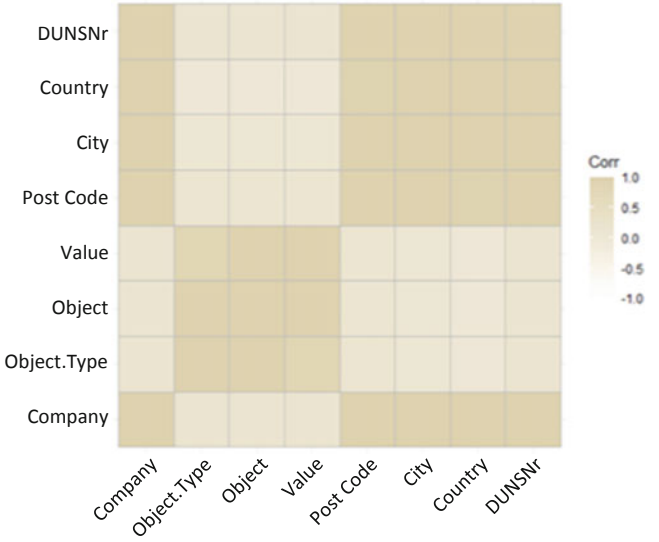

Fig. 5 Correlation matrix (© ifb SE)

we can directly read that the columns "Company," "Address," "Post code," "City," "Country" and "DUNSNr" are strongly correlated, since they are all geographic based. On the other hand, columns "Value," "Object" and "Object.Type" should be grouped together. This indicates that we should select "Object" and "Object.Type" for building the machine learning model.

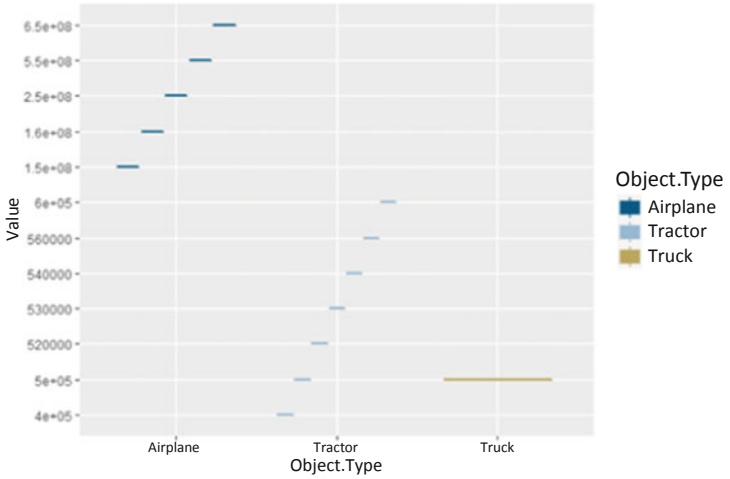

Fig. 6 Object.Type vs. Value (© ifb SE)

4.3.3 Features—Object and Object.Type

In the following, we provide some further statistical analysis for the columns "Object" and "Object.Type." As we already mentioned in other articles (Liermann and Li, Methods of Machine Learning 2021), these classic statistical intuitions should not be ignored. They will provide a deeper look at the dataset and are able to improve the training efficiency during the development of machine learning models.

Figure 6 plots column "Object.Type" against "Value." We are not able to separate the values of the column "Value" finely according to column "Object.Type," since the invoice announcements of category "Airplane" take values across 1.5e+08 to 6.5e+8 and the category "Tractor" covers values between 4e+5 and 6e+5. After considering the column "Object," the separation becomes sharper. As illustrated in Fig. 7, we are nearly able to identify the value of "Value" based on "Object.Type" and "Object." The result of the analysis is not surprising, since we used deterministic conditions to generate datasets, as introduced in Sect. 4.3.

4.3.4 Model Training—Random Forests

One of the traditional classification models—random forest—is chosen. Recalling the statistical analysis provided in the last section, it is unsurprising to see that a small forest with five trees is already able to achieve an excellent accuracy beyond 99.9%. Cross-validation results are shown in Fig. 8. In

Fig. 7 (Object.Type, Object) vs. Value (© ifb SE)

Cross-Validation Metrics Summary:

	mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid	cv_6_valid
accuracy	0.99963856	5.075365E-4	1.0	0.99857855	0.99923897	0.99928266	1.0	1.0
err	3.6141448E-4	5.075365E-4	0.0	0.0014214641	7.61035E-4	7.173601E-4	0.0	0.0
err_count	0.5	0.70710677	0.0	2.0	1.0	1.0	0.0	0.0
logloss	0.040052153	0.0055257706	0.03954867	0.046860714	0.0452101	0.04388479	0.03501095	0.048248984
max_per_class_error	0.010186633	0.015246954	0.0	0.045454547	0.01754386	0.02	0.0	0.0
mean_per_class_accuracy	0.9991511	0.0012705795	1.0	0.9962121	0.998538	0.99833333	1.0	1.0
mean_per_class_error	8.488661E-4	0.0012705795	0.0	0.003787879	0.0014619883	0.0016666667	0.0	0.0
mse	0.006535436	0.0010166838	0.00628277	0.007392936	0.008134246	0.0073786057	0.005536854	0.0076509146
r2	0.99950916	7.6990566E-5	0.9995331	0.99945414	0.99939126	0.99945265	0.9995807	0.9994127
rmse	0.08062427	0.0062506427	0.079263926	0.085982166	0.09019005	0.08589881	0.074410036	0.0874695

Fig. 8 Random forest: cross-validation result (© ifb SE)

practice, we might face more complicated data structures and need to use more advanced models.

5 Practical Application: Machine Learning UiPATH

In this section, we present an example technical implementation of the previously circumscribed use case. As the RPA tool we use *UiPath Studio Pro 2020.6.0-beta.93 Community License* from the company *UiPath*. This offers the integration of the Python programming language as an external tool.

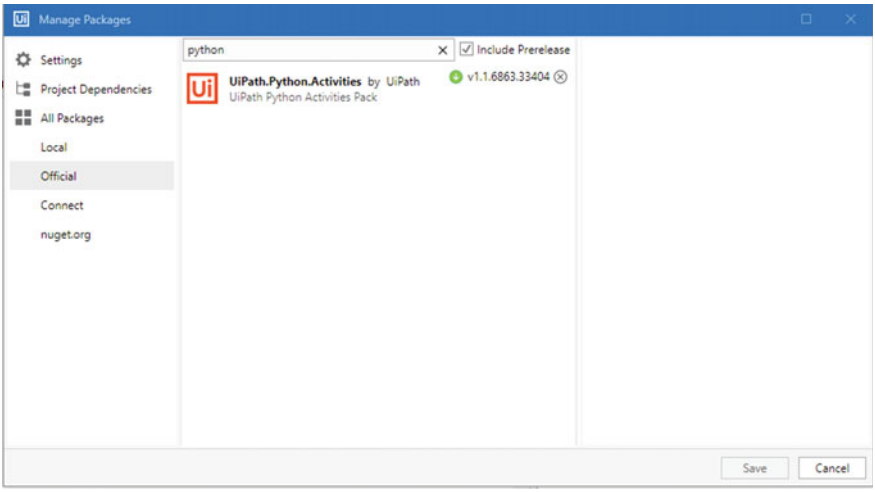

Fig. 9 UiPath.Python.Activities module in the UiPath Packages manager (© ifb SE)

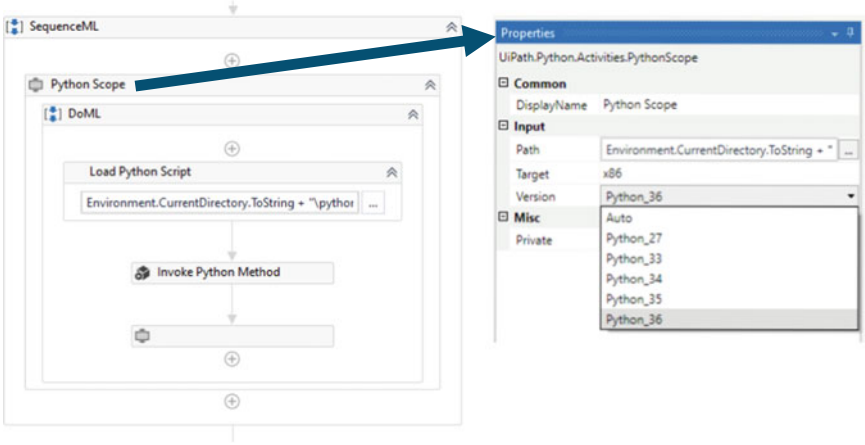

Fig. 10 Python scope and properties in UiPath (© ifb SE)

5.1 Python Integration in UiPath

For the technical integration of the externally installed Python programming language, UiPath provides an adapter package module called *UiPath.Python.Activities*, which is found in its Packages manager,[2] see Fig. 9.

Once installed, UiPath provides a so-called *App Invoker* for Python in its listed activities that include objects like *Invoke Python Method*, *Load Python*

[2] Note that we used the English language setting for UiPath.

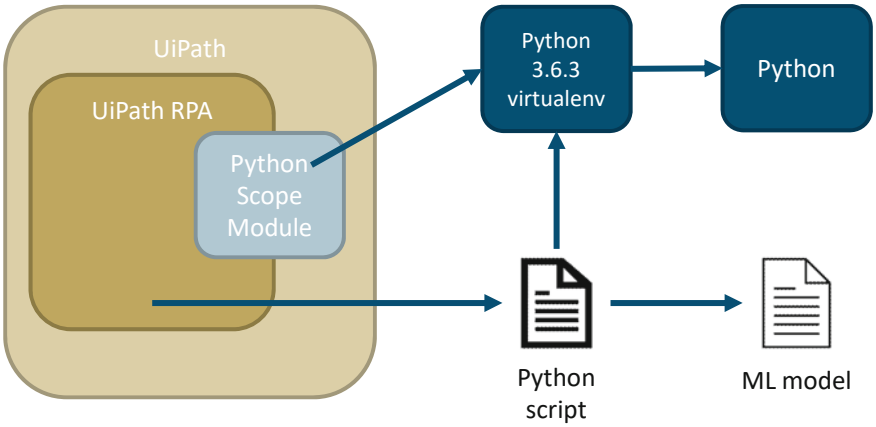

Fig. 11 Architecture of the integration of Python into UiPath (© ifb SE)

Script or *Python Scope*. Python is connected to UiPath via the *Python Scope* environment, see Fig. 10.

To make the connection work, three input attributes are key, namely Path, Target, and Version. The *Path* argument points to the folder containing the externally installed Python interpreter, not the interpreter itself. This can be the globally installed version or a virtual python environment created by virtualenv or the like. The tag *Version* corresponds to the Python version. As you can see in Fig. 10, several versions are supported. We used the latest supported one, version 3.6, as older versions tend to lose support by developers. Python 2.7 has even reached its end-of-life status (Python Software Foundation, Sunsetting Python 2, 2020). Last but not least, the *Target* differentiates between the 32bit and 64bit versions of Python. A Windows executable for Python 3.6.8 is available at (Python Software Foundation, Download 2020).[3]

After the Python scope is set up, an externally programmed Python script can be used via the *Load Python Script* module. This simply specifies the path to the script. Within this script, one defines functions or methods that can then be invoked and used via the *Invoke Python Method* module. Here you can forward parameters from UiPath to be used as input for the Python function. The return values of the function can then be read out by UiPath and processed further. In the scope of this use case, we created machine learning models beforehand, see Sect. 4.3. These models are invoked by the Python script. For a visualization of the interplay of the different components, see Fig. 11.

[3] We used the 32bit version of Python 3.6.8.

5.2 Example in UiPath

As a proof of concept and a technical example, we now sketch the implementation of phase two implemented in UiPath.

At this point, we assume that initial data has been recorded according to phase one. In our example case, it was generated as described in Sect. 4.3. The first step of phase two and ultimately a preparatory step is the model training, which was also covered in Sect. 4.3. The onset is that the user receives an email in MS Outlook like the one shown in Fig. 12.

The UiPath procedure starts by checking whether there are unread emails in the inbox regarding a certain topic, here "Invoice announcement." Upon finding one, its content gets read, processed and the relevant details assigned to UiPath variables. Most important for us are the variables "Value," stating the amount of the invoice, "Object type" and the "Object." At the end of the entire process, UiPath checks for the next unread email until none are left, see Fig. 13.

The detailed automation steps of the data processing are grouped in the "check mail message" box of Fig. 13. There, the logical steps begin by filling in the necessary UiPath variables, in our case the ones mentioned in the previous paragraph. Secondly, a function from a Python script gets invoked with the UiPath variables for the "Object Type" and "Object" as input for the function. The Python function refers to a previously calculated and provided ML

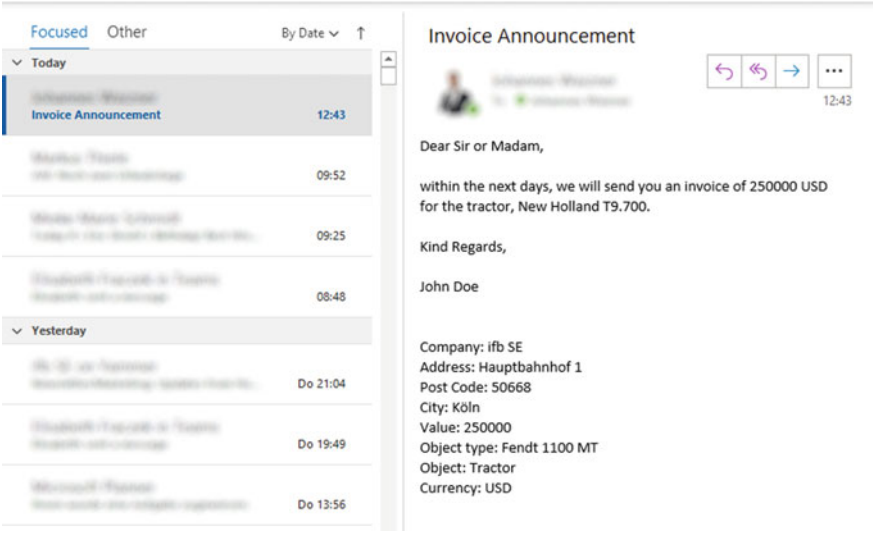

Fig. 12 Email regarding an invoice announcement (© ifb SE)

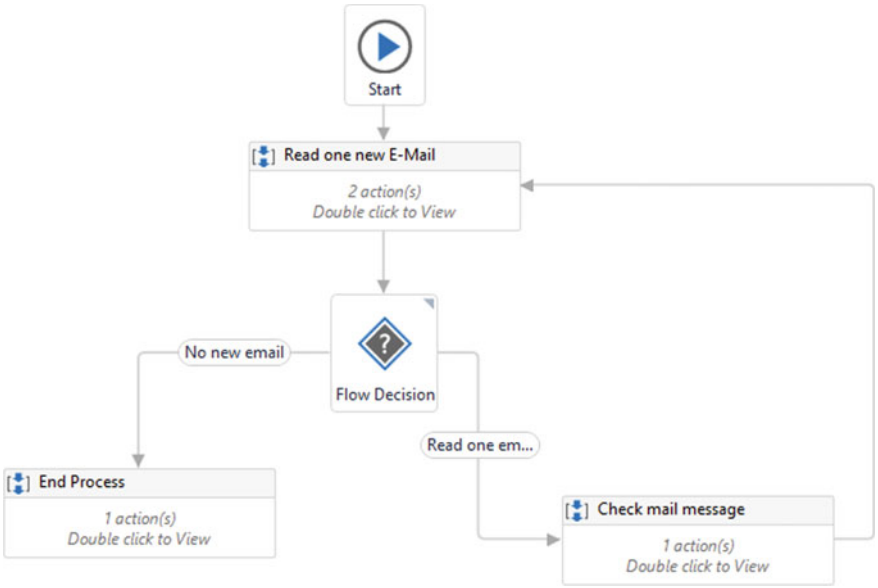

Fig. 13 Coarsest grained flow graph of the automation procedure in UiPath (© ifb SE)

model which yields a predicted value for the invoice that is returned by the Python script to UiPath. The user then gets prompted with two values: the originally announced invoice value from the email and the predicted value from the ML model. Since it is only semiautomatic in phase two, i.e., level B from Fig. 3, the user still needs to decide which one to choose or where he or she would like to use a third, new value as provisions.

6 Summary

The chapter shows a practical example of an important pattern in automating processes. The challenge is to have a step-by-step approach to achieve the benefits of true automation (modern RPA using automated decision-making). Full automation including automated decision-making cannot be implemented as a big bang, in one-step or one project task. The important aspect is to gain and integrate the knowledge of the manual process and its decision-making.

The three-step meta process (see Fig. 3) is mirrored in Bitcom's popular "five-step decision automation model" (see Fig. 2). Label generation can be a tricky task, especially if there is no electronic data history regarding the label. Supervised learning—by definition—only works with labels. The use

of RPA capabilities to collect these important information labels is the crucial point in the approach. Only with an approach that collects the label data "en passant" (without any entries to an additional tool) and stores it electronically without slowing down the manual human process will it be accepted by the original process owner.

The potential of the approach is huge. The automation initiatives do not have to stop when more intelligence is required than simple rule-based approaches can easily cover. Defining a rule requires context knowledge and the definition of the rule is still manual and static (no learning without a human interaction), while machine-learning-based pattern recognition can define the rule autonomously and can adapt and learn from new situations. The only, but important, requirement is that the labels are available. Thus, RPA can automate simple, recurring, and stable tasks. In addition, RPA can collect the labels in the existing process and discover the decision-making patterns (using machine learning). Therefore, the way for a full automation is paved.

RPA is well established in most banks and insurance companies. Now is the right time to start collecting the labels (using RPA) so the cost-saving potential can be tapped just in time.

Literature

Bitcom. 2017. *Künstliche Intelligenz verstehen als Automation des Entscheidens Leitfaden*. Berlin: Bitcom Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.

Gartner. 2019. "Gartner Top 10 Strategic Technology Trends for 2020." *Gartner*. Accessed November 15, 2020. https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020/.

———. 2020. "Gartner Top Strategic Technology Trends for 2021." *Gartner*. Accessed November 15, 2020. https://www.gartner.com/smarterwithgartner/gartner-top-strategic-technology-trends-for-2021/.

H2O.ai. 2019. *h2o.ai Overview*, January 29. Accessed January 29, 2019. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html.

Liermann, Volker, and Norbert Schaudinnus. 2019. "Real Estate Risk—Appraisal Capture." In *The Impact of Digital Transformation and Fintech on the Finance Professional*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillen.

Liermann, Volker, and Sangmeng Li. 2021. "Methods of Machine Learning." In *The Digital Journey of Banking and Insurance, Volume III—Data Storage, Processing, and Analysis*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.

Liermann, Volker, Sangmeng Li, and Norbert Schaudinnus. 2019. "Deep Learning—An Introduction." In *The Impact of Digital Transformation and Fintech on the Finance Professional*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.

Maddox, Teena. 2019. "Top 10 Technology Trends for 2020 Include Hyperautomation, Human Augmentation and Distributed Cloud." *TechRepublic*. Accessed November 15, 2020. https://www.techrepublic.com/article/hyperautomation-human-augmentation-and-distributed-cloud-among-top-10-technology-trends-for-2020/.

Python Software Foundation. 2020. "Download." Python. https://www.python.org/downloads/.

———. 2020. "Sunsetting Python 2." *Python.* Accessed November 15, 2020. https://www.python.org/doc/sunset-python-2/.

Schröder, Daniel, and Marian Tieben. 2021. "Sentiment Analysis for Reputational Risk Management." In *The Digital Journey of Banking and Insurance, Volume II—Digitalization and Machine Learning*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.

Wikipedia. 2020. "Natural Language Processing." *Wikipedia*, November 15. https://en.wikipedia.org/wiki/Natural_language_processing.