

A Software Interface for Machine Learning Model Applications

Authors: Antonios Konomos¹, Spiros Chountasis²

¹ *Intrasoft International SA, Athens, Greece*

² *Independent Power Transmission Operator, Athens, Greece*

Corresponding Author: schountasis@admie.gr

Received: January, 2026 Published: March, 2026

ARTICLE INFO

Keywords:

Computer Technologies; Machine Learning; Software Applications; Software Interface

© 2026 by the Authors. This open-access article is distributed under a Creative Commons Attribution (CC-BY) 4.0 license, making research freely available to the public and supporting a greater global exchange of knowledge and human experiments.



ABSTRACT

This paper presents an innovative software interface for the utilization of widely used Machine Learning (ML) algorithms in a unified Python/R programming environment. A novel software model, the Rbox+, is proposed to execute ML algorithms by jointly leveraging the capabilities of the Python and R programming languages. Furthermore, more comprehensive and specialized architecture is made available for integrating ML into enterprise information systems. Unlike conventional ML Application Programming Interfaces (APIs) or isolated Enterprise Resource Planning (ERP) analytics tools, the Rbox+ enables transparent, language-independent execution and validation of ML models while exposing the underlying source code. The proposed approach supports practical applications in enterprise analytics, reproducible research, and enhancing interoperability between ERP systems, analytics platforms, and statistical programming environments. The proposed API has been tested and evaluated using a publicly available benchmark dataset for regression analysis, applying multiple ML models and comparative performance metrics. The obtained results demonstrate improved computational efficiency and scalability.

1.0 Introduction

ML data processing and visualization tools are rarely embedded directly into information systems that are commonly used for managing an organization's core operations (Bejani & Ghatee, 2021). Representative examples of such systems include Microsoft Excel and SAP S/4 Hana ERP. SAP S/4 Hana represents SAP A.G.'s ERP solution and integrates critical enterprise activities such as Financial Accounting, Sales and Distribution, Materials Management, Production Planning, and Human Resources (Song, 2021; Zhao et al., 2021). Despite its widespread adoption, SAP S/4 Hana lacks native advanced analytics, ML execution, and data visualization capabilities. Instead, these needs are typically addressed through separate analytical platforms such as SAP BI or SAP Analytics Cloud (King, 2021; Google, 2020).

Rbox+ is designed to bridge this gap by providing a lightweight and flexible Web API that enables enterprise systems to directly invoke ML and statistical algorithms implemented in Python and R. Nowadays, big data helps to improve the accuracy of the ML models and makes it feasible to virtualize data, so that it may be kept in the cloud or on-premises in the most effective and economical way possible. Major cloud providers deliver ML as a Service (MLaaS), allowing users to train predictive models on this data even if they lack ML expertise or infrastructure (Birnbbaum et al., 2017; Neumann et al., 2018). Customers may outsource their costly ML operations to reliable servers using MLaaS, yet there are still no firm guarantees of service accuracy. The user may merely note that the trained model or prediction result is

well-formed without knowing if the delivered result is accurate (Rajput, & Tiwari, 2023; Shu & Ye, 2022). Unlike traditional ERP analytics solutions or MLaaS platforms, which often operate as closed or proprietary environments, Rbox+ allows enterprise applications to trigger ML computations while retaining full access to the underlying algorithmic logic and execution flow. From a system integration perspective, Rbox+ simplifies interactions between heterogeneous software environments. Through a single web service call, client applications can submit datasets and analysis requests without requiring native support for Python or R. This design enables ML-driven analytics to be embedded into existing enterprise workflows without altering the underlying ERP or spreadsheet-based infrastructure.

Data analysis in an ML context involves multiple stages, including data collection and exploration, data cleaning and preprocessing, feature selection, model training and evaluation, and iterative refinement. The objective is to understand the data, improve its quality, and optimize it for effective model training (Botchkarev, 2018; Brunton & Kutz, 2022). The iterative nature of this process allows for continuous refinement of the analytical pipeline and the resolution of emerging challenges. Ultimately, the goal is to deploy a well-performing model and monitor its performance over time (Hao, 2019; Chourasiya & Jain, 2019). The key frameworks and algorithms in this field have also undergone substantial modifications. Today's technological advancements have updated forms of software engineering and computer architecture (Hamid et al., 2024; Muhammad Imran Sarwar Abbas 2023). Furthermore, large volume of data management stresses the need for more efficient visualization and presentation tools (Jiang et al., 2025; Gandomi & Haider, 2015). Leading open-source programming languages like Python and R, are at the forefront of requisite quantitative computer programs (Chen, 2021; Granger & Perez, 2021; Parkash, 2023). Both programming languages have distinctive characteristics that encompass a wide range of ML algorithms, statistical controls, and robust data visualization features. Being able to correlate data to detect patterns and anomalies can help an organization predict outcomes and improve its operations of business. Appropriate datasets must be applied for the learning process of an ML technique. Because the time at which training converges is unpredictable, it is difficult to anticipate in advance the effort required to train a model. The model training data set, learning parameter settings, and random variables all affect convergence (Hou et al., 2023; Wufka & Canonico, 2026). Web APIs have become the preferred choice for the seamless integration of heterogeneous software systems. The primary objective of this research is to propose the design and implementation of an API (Rbox+) that sends data to the R or Python programming environments and requests the execution of various ML algorithms. Furthermore, statistical tests and graphical representations may also be performed in the R programming language returning the computation results in Python/R Jupyter Notebook format.

The Rbox+ facilitates the integration of both Python and R programming environments with different types and content systems such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) which can consume Web services. The current study aims to further develop an interface for integrating Python/R with the SAP S/4HANA ERP system. SAP S/4 Hana ERP is an ERP business suite based on the SAP HANA in-memory database. The Rbox+ is designed as an "open box" that is receptive to further adjustments, modifications, and improvements. Validation of ML models is very crucial, providing a parameter for software integration and system implementation. The user can review and compare the source code of both Python and R by examining the results of the requested ML and statistical tests. This feature further enhances the transparency of the results by modifying and improving ML algorithms as needed, as their validation becomes more critical than ever.

2.0 The Rbox+ API

Rbox+ is implemented as a generic API that enables the Python and R programming environments to be seamlessly incorporated into any application capable of issuing HTTP requests (Ehsan et al., 2022; Konomos, & Chountasis, 2023). From an integrator's perspective, Rbox+ simplifies integration by minimizing the effort required. The integration consists of a single step: preparing and executing a direct service call via Rbox+'s dispatcher method. This design simplicity and flexibility make Rbox+ highly user-friendly.

From an end-user perspective, Rbox+ generates output as Jupyter Notebook files. Jupyter Notebook is an open-source web application that supports interactive computing by combining executable code, narrative text, equations, and visualizations within a single document. Rather than detailing the general functionality of Jupyter Notebook, this work focuses on its specific role in enhancing the usability, transparency, and reproducibility of Rbox+. By producing results directly in Jupyter Notebook format, Rbox+ preserves the complete computational workflow, including the Python /R source code, intermediate outputs, and results. This integration enables users to review, validate, and reproduce analyses without relying on opaque result summaries. In addition, the notebook format allows analysts to extend or modify the generated code, supporting iterative experimentation and a deeper understanding of the applied ML techniques.

The reproducibility facilitated by Rbox+ is particularly important in research and enterprise analytics contexts, where model validation and auditability are critical. Unlike many ERP analytics tools or MLaaS platforms, which typically provide only aggregated results, Rbox+ exposes the entire execution logic. This allows users to verify assumptions, inspect parameter settings, and assess model robustness, thereby enhancing trust in ML-driven decision support and aligning with best practices in reproducible research. Furthermore, notebook-based output makes Rbox+ suitable for educational use. It can serve as a structured template for teaching ML concepts, enabling students to explore algorithms, visualize results, and connect theoretical concepts with executable code. The Rbox+ combines generic Web API architecture with direct Python/R execution and reproducible Jupyter-based output. This combination distinguishes it from existing ERP analytics solutions, MLaaS platforms, and standalone scripting approaches by offering transparency, interoperability, and extensibility within a unified integration framework.

A wide range of companies, including Google Cloud AI, Amazon Web Services (AWS) AI, IBM Watson and Microsoft Azure (IBM, 2020; Microsoft, 2020), provide cloud-based ML services. These platforms offer collections of APIs that execute inference computations using industry-trained deep neural networks (DNNs) on powerful cloud infrastructures, without requiring developers to possess in-depth knowledge of ML algorithms or resource provisioning.

Despite their advantages, third-party ML APIs still pose several challenges, particularly when integrating ML applications into larger software systems. One approach to making ML more powerful, practical, and interoperable is to develop interfaces that enable algorithm execution independently of programming language and data format. ML algorithms are implemented in diverse ways and are designed to operate on heterogeneous datasets. Owing to differences in semantics, data requirements, and accuracy–performance trade-offs, ML APIs can be difficult to use correctly and efficiently.

In this study, these limitations are addressed by introducing a web-service-based interface that enables researchers and developers to perform statistical analyses and execute ML algorithms in the R and Python environments. The Rbox+ provides a simplified and uniform access layer for ML execution, facilitating integration and comparative analysis across platforms. Table 1 presents the Rbox+ templates developed in Python.

Table 1 - ML Algorithms in Python

Template	ML Algorithms	
	ML Task	Options / Hyperparameters
Decision Trees	Classification / Regression	Descriptives Correlation matrix k-fold Cross Validation
Advanced Trees	Classification / Regression	Descriptives Correlation matrix Bagging Random Forest Extremely Randomized Trees k-fold Cross Validation
Boosting Trees	Classification / Regression	Descriptives Correlation matrix AdaBoost XGBoost LightGBM k-fold Cross Validation
Gaussian Mixture	Clustering	Descriptives Correlation matrix
Mini Batch K-means	Clustering	Descriptives Correlation matrix
Naive Bayes	Classification	Descriptives Correlation matrix k-fold Cross Validation
Stochastic Gradient Descent	Classification / Regression	Descriptives Correlation matrix k-fold Cross Validation
Support Vector Machines	Classification / Regression	Descriptives Correlation matrix Linear kernel RBF kernel Sigmoid kernel Polynomial kernel k-fold Cross Validation

Similarly, Table 2 summarizes the Rbox+ templates developed using the R programming language.

Table 2 - ML & Statistical Algorithms in R

Template	ML Algorithms	
	ML Task	Options / Hyperparameters
K-means	Clustering	Descriptives Silhouette method Gap statistic method Anova
K-prototypes	Clustering	Descriptives Silhouette method Anova Chi-square tests
Hierarchical Clustering	Clustering	Descriptives Anova

Template	ML Algorithms	
	ML Task	Options / Hyperparameters
Partitioning Around Medoids (PAM)	Clustering	Descriptives Silhouette method Gap statistic method Anova Chi-square tests
Clustering Large Applications (CLARA)	Clustering	Descriptives Silhouette method Gap statistic method Anova Chi-square tests
K-Nearest Neighbors	Classification / Regression	Descriptives Correlation matrix Elbow method ROC curve
Linear Regression	Regression	Descriptives Pearson's Correlation R.square value Residuals distribution NCV test Spread Level Plot
Linear Multiple Regression	Regression	Descriptives Correlation matrix Collinearity test K-fold cross-validation Stepwise Regression
Logistic Regression	Classification	Descriptives Correlation matrix Collinearity test Hosmer test (Goodness of fit) Predictive ability ROC curve
Multinomial Logistic Regression	Classification	Descriptives Correlation matrix Evaluate Collinearity Stepwise Regression
Principal Component Analysis (PCA)	Dimensionality Reduction	Descriptives Correlation matrix Correlations Vars – PCs
PCA mix	Dimensionality Reduction	Descriptives Plots
Random Forest	Classification / Regression	Descriptives Correlation matrix
t-Test	Association	Descriptives Kolmogorov-Smirnov Levene's test Mann-Whitney U Boxplots Histogram
Paired t-Test	Association	Descriptives Pearson's Correlation Kolmogorov-Smirnov Mann-Whitney Histograms
One-way Anova	Association	Descriptives TukeyHSD (honest)

Template	ML Algorithms	
	ML Task	Options / Hyperparameters
		significance test) Levene's test (homoskedasticity test) Diagnostic plots Kruskal Wallis
Two-way Anova	Association	Descriptives TukeyHSD Pairwise t-tests Levene's test Normality test
χ^2 test	Association	Descriptives Crosstabs Balloon plot Barplot

3.0 The Rbox+ System Architecture

This section describes the system architecture, which is defined independently of the implementation details. The Rbox+ API runs on a web server that connects to both the R and Python environments and is implemented in PHP. The ML methods included in Rbox+ are accessible to any application capable of handling HTTP requests. The design of Rbox+ follows a conventional three-tier architecture. More specifically, it is based on a client-server model consisting of three distinct processing layers, as illustrated in Fig. 1 below:

- a. Client Tier (service consumer) concerns the calling applications of Rbox+.
- b. API Tier (service provider) handles the requests for the R and Python environments.
- c. Processing Tier executes the requested ML and statistical computations.

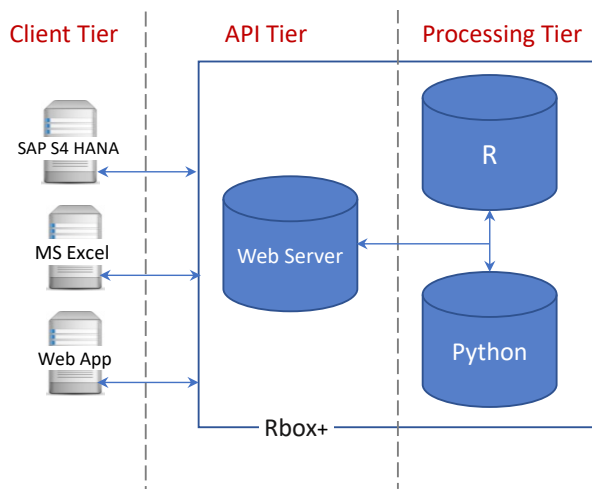


Fig. 1 Rbox+ Three-tier Architecture

All three tiers (Processing, API, and Client tier), operate in accordance with the previously proposed Rbox API and are briefly described in the following section. The API tier functions as a web server that handles HTTP requests via POST method. Most modern programming languages provide built-in mechanisms for converting two-dimensional datasets into JSON structures that can be transmitted as parameters to web services. As a result, invoking Rbox+ is comparable to making a standard function call in most programming environments. A key feature of Rbox+ is its generic calling mechanism, which remains consistent

regardless of the specific ML algorithm, statistical test, or graphical output requested. Integrators are therefore only required to construct a JSON request containing the dataset and the selected analysis, while the remaining integration code can be reused. This approach ensures efficient, flexible, and reusable calls to Rbox+, thereby enhancing the overall integration process. As shown in Table 3 each request carries a JSON structure containing information about the requested ML algorithm (implemented in Python/R), as well as the dataset to which the model will be applied.

Table 3 - JSON Structure

Variable	ML Algorithms	
	Description	JSON sample for t-Test
template	ML / Statistical test	{“template”: “P.NaiveBayes”,
numvars	Variables counter	“numvars”: 2,
numopts	Options counter	“numopts”: 6,
options	Options on/off	“options”: [1,1,1,1,1],
variables	Variables list	“variables”: [
id	x1... xn	{“id”: “X0”,
name	Variable name	“name”: “VEHICLE_TYPE”,
datatype	Variable type	“datatype”: “factor”,
values	Value list	“values”: [“Car”, “Passenger”,...]],
		{“id”: “X1”,
		“name”: “PRICE”,
		“datatype”: “numeric”,
		“values”: [“21.50”, “28.40”, ...]]}]}

The JSON element template specifies the requested ML technique or statistical test. As shown in Table 3, the options parameter includes on/off switches that control the ML algorithm's hyperparameters or enable additional tests for each method. The variables element of the JSON structure contains the attributes id, name, and datatype, along with the corresponding list of values for each variable. The id attribute represents the unique identifier of each variable and ranges from x0 to xn. The numvars parameter specifies the total number of variables, while numopts indicates the number of available on/off switches within the options array. After the execution of the corresponding script, a response is returned to the calling system. This response essentially includes a URL pointing to the results page. The ML outputs are presented in Jupyter Notebook format, allowing end users to view both the generated Python/R source code and the corresponding computational results within a single HTML document. Through this mechanism, Rbox+ dynamically enables or disables hyperparameters for each ML method, thereby reducing the execution time of the corresponding Python/R script and minimizing the overall response time of the HTTP.

4.0 The Rbox+ Features and Results

Rbox+ has been designed as a distinctive learning and analysis tool, acknowledging the researcher's fundamental motivation to understand how algorithmic calculations are performed. By displaying the generated Python/R source code at every stage of the computational workflow, Rbox+ ensures full transparency throughout the entire data processing cycle, including data upload, data manipulation, statistical testing, data analysis, and visualization of the results. This characteristic transforms Rbox+ into a powerful instructional tool that enables developers and researchers to gain a deeper understanding

of ML algorithms and their execution environments, while also supporting the development of Python and R programming skills.

4.1 Sample Dataset and Use-Case Scenario

In this subsection, indicative results produced by Rbox+ are presented. A publicly available dataset, commonly used for educational purposes, is employed to demonstrate the flexibility of Rbox+ in executing multiple ML algorithms and selecting the best-fitting model through comparative analysis. The car sales dataset is available in the Kaggle repository and contains information on vehicle sales across different manufacturers. In addition to sales volume, the dataset includes several explanatory variables, such as resale value, price, engine size, horsepower, wheelbase, width, length, fuel capacity, fuel efficiency, and power performance factor.

For the purposes of the presented use-case scenario, a range of ML algorithms supported by Rbox+ is applied to predict vehicle sales volume. The explanatory variables used in the analysis include engine size, horsepower, wheelbase, width, length, fuel capacity, and fuel efficiency. This type of prediction problem is addressed using multiple regression models, and the predictive performance of each method is evaluated using the R² and adjusted R² metrics (Hu et al., 2025; Lalaoui et al., 2025; Yang et al., 2024).

4.2 Model Selection

After importing the dataset in the Rbox+ Excel Client, the researcher can select the Data Analysis model (Regression, Classification, Clustering or Dimensionality Reduction) and the desired ML method. At this step, it is important to specify the response variable and the independent variables. Additional options can also be enabled, performing further calculations or plots useful during the analysis stage as shown in Fig. 2.

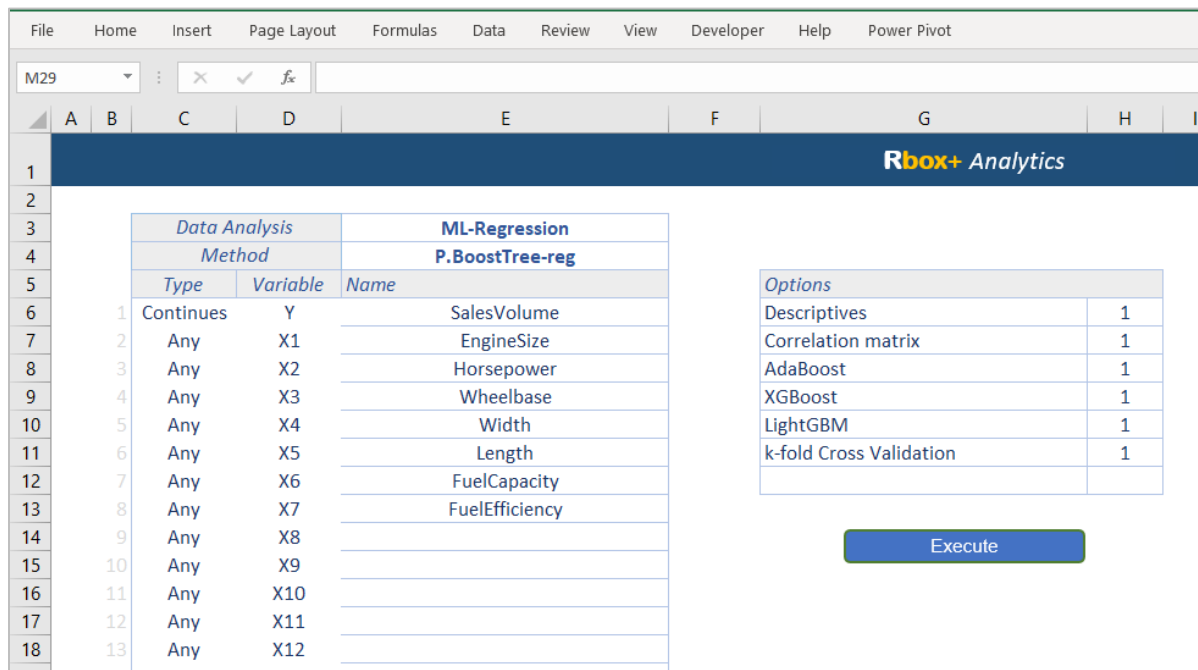


Fig. 2 Rbox+ Excel Client

Moreover, Fig. 3 illustrates the option of using the SAP S4Hana Client. The analyst can easily explore all the available ML Regression methods and trigger the execution accordingly.

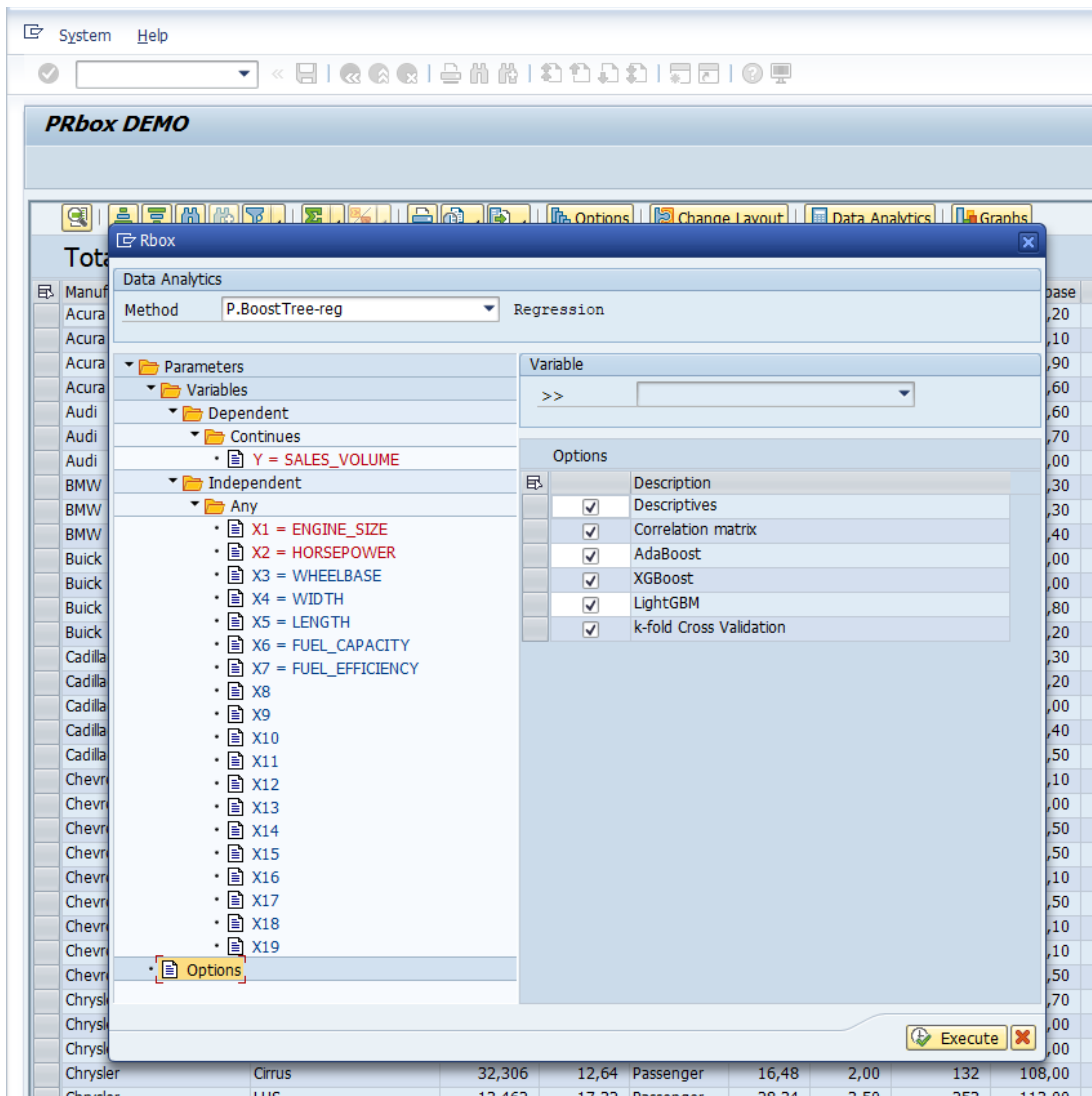


Fig. 3 Rbox+ SAP S4Hana Client

4.3 Results

The calculation outcomes are presented in Jupyter Notebook format within the user’s web browser, as illustrated in Fig.4 Both the source code and the extracted results appear on the same page, allowing the analyst to review the outcomes and compare the goodness of fit across different models.

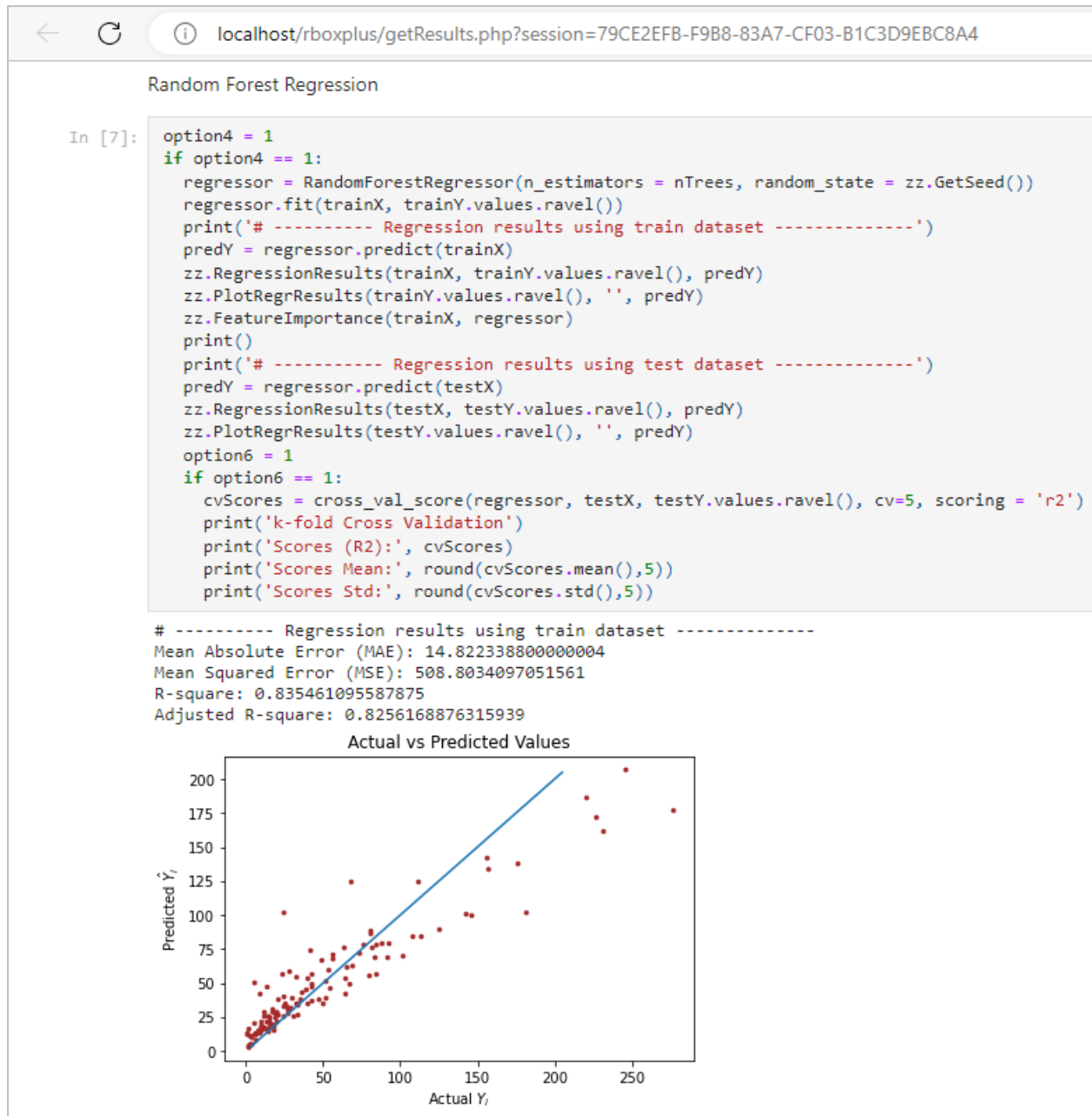


Fig. 4 Rbox+ results: Jupyter Notebook

Accordingly, Fig. 5 shows the code and the extracted results of a correlation matrix in the SAP S/4HANA output format.

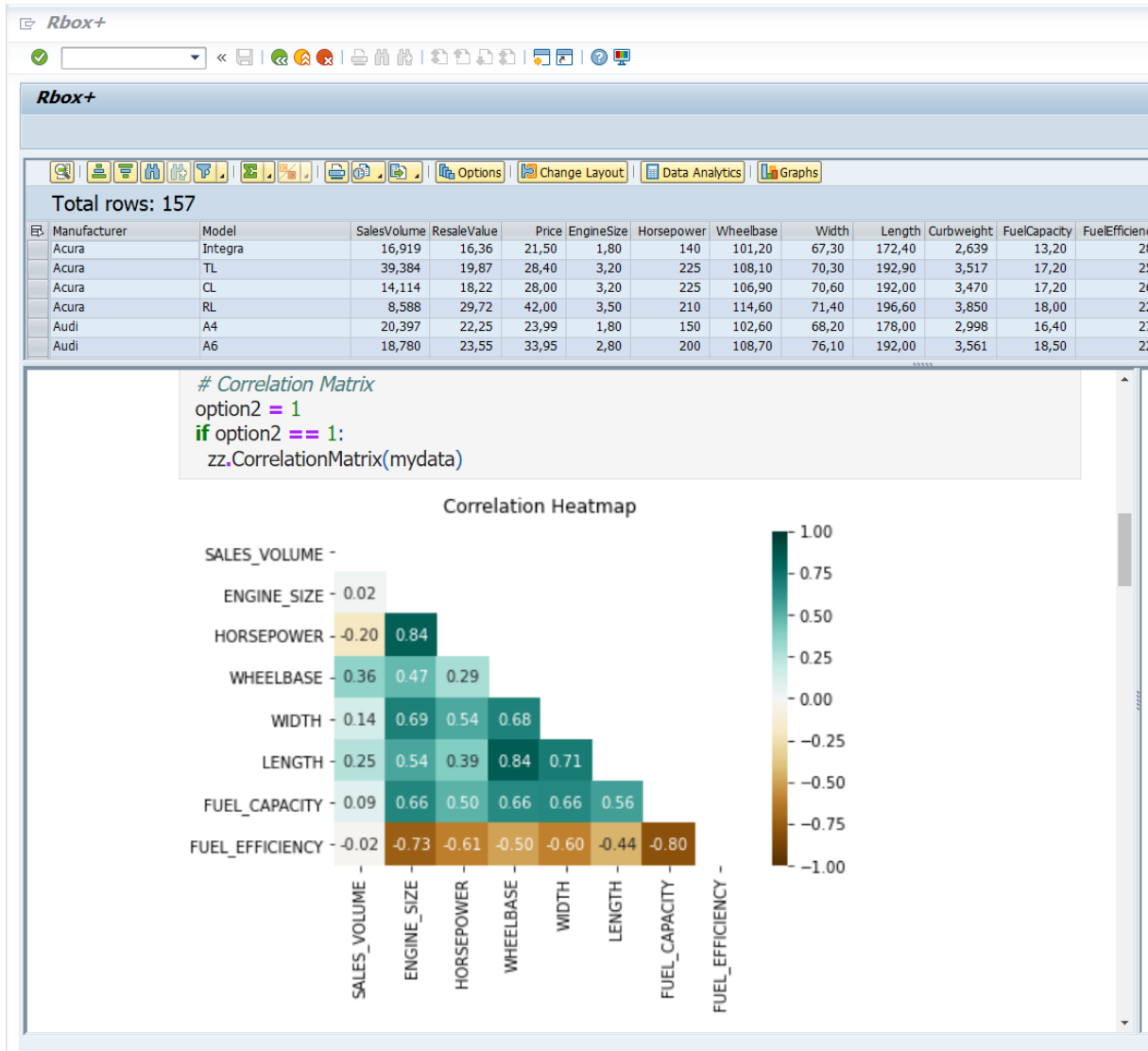


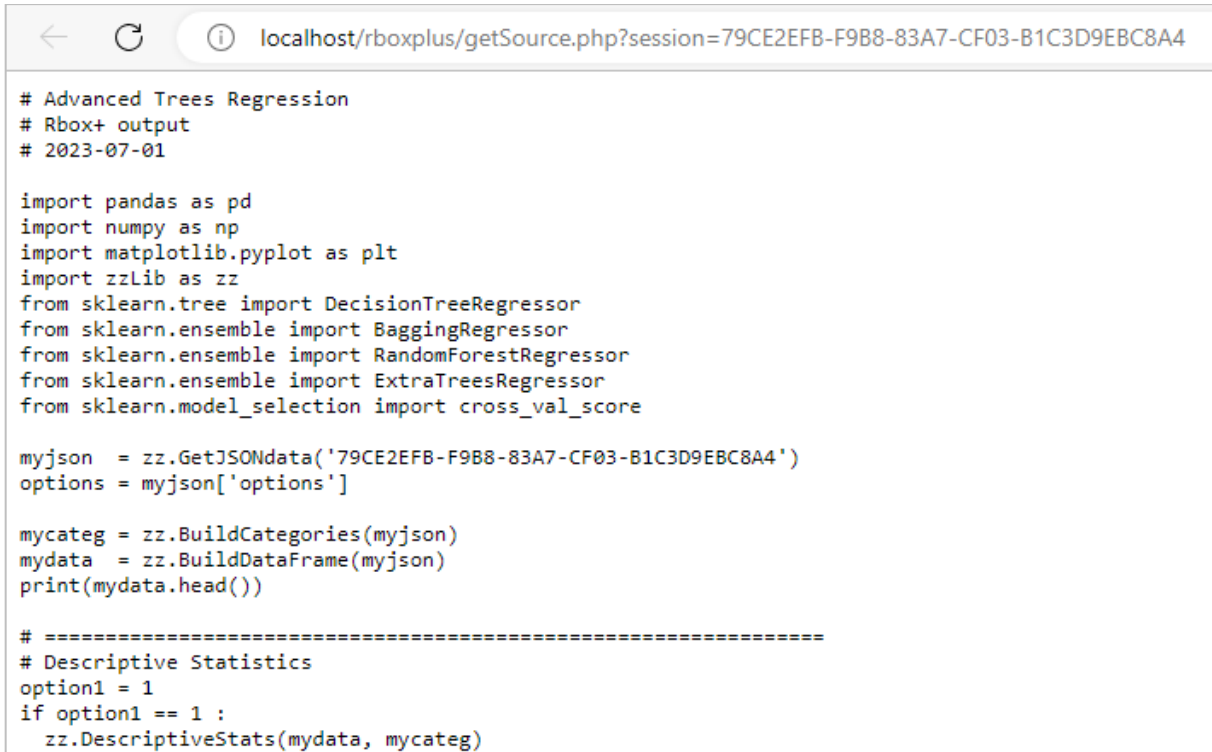
Fig. 5 Rbox+ SAP S4Hana output

In the Excel worksheet, a table is displayed that contains hyperlinks to the results page, the generated source code, and the corresponding Jupyter Notebook file. Through these links, the analyst can access the complete history of the performed computations and revisit or reassess the associated results at any time, as illustrated in Fig. 6.

Result	Source code	Jupyter Notebook
P.SGD-reg	P.SGD-reg	P.SGD-reg
R.KNN-reg	R.KNN-reg	R.KNN-reg
P.DcsTree-reg	P.DcsTree-reg	P.DcsTree-reg
P.AdvTree-reg	P.AdvTree-reg	P.AdvTree-reg
P.BoostTree-reg	P.BoostTree-reg	P.BoostTree-reg
P.SVM-reg	P.SVM-reg	P.SVM-reg
R.MultipleRegression	R.MultipleRegression	R.MultipleRegression
R.KNN-reg	R.KNN-reg	R.KNN-reg

Fig. 6 Rbox+ API hyperlinks

The users can export the entire Python (or R) source code, as depicted in Figs 7 and 8.



```
localhost/rboxplus/getSource.php?session=79CE2EFB-F9B8-83A7-CF03-B1C3D9EBC8A4

# Advanced Trees Regression
# Rbox+ output
# 2023-07-01

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import zlib as zz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.model_selection import cross_val_score

myjson = zz.GetJSONdata('79CE2EFB-F9B8-83A7-CF03-B1C3D9EBC8A4')
options = myjson['options']

mycateg = zz.BuildCategories(myjson)
mydata = zz.BuildDataFrame(myjson)
print(mydata.head())

# =====
# Descriptive Statistics
option1 = 1
if option1 == 1 :
    zz.DescriptiveStats(mydata, mycateg)
```

Fig. 7 Python Source Code

Subsequently, they edit and process it in their preferred development environment (e.g., Jupyter Notebook, VS Code, Eclipse).



```
localhost/rboxplus/getSource.php?session=79CE2EFB-F9B8-83A7-CF03-B1C3D9EBC8A4&ftype=ipynb

{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "Advanced Trees Regression<br>\n",
        "Rbox+ output<br>\n",
        "2023-07-01"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {},
      "outputs": [],
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "import zlib as zz\n",
        "from sklearn.tree import DecisionTreeRegressor\n",
        "from sklearn.ensemble import BaggingRegressor\n",
        "from sklearn.ensemble import RandomForestRegressor\n",
        "from sklearn.ensemble import ExtraTreesRegressor\n",
        "from sklearn.model_selection import cross_val_score"
      ]
    }
  ]
}
```

Fig. 8 Jupyter Notebook .ipynb File

4.4 Performance Evaluation

For the purposes of the selected use-case scenario, a wide variety of ML regression algorithms were executed, and the results are presented in Table 4. Seven discrete ML regression algorithms were evaluated, some with multiple variations, resulting in a total of 13 distinct result sets.

Table 4 - ML Regression Algorithms and their Results

ML Regression	Method	MSE	R2	Adj.R2
Support Vector Machines	Linear kernel	0.804	0.195	0.147
	Polynomial kernel	0.830	0.169	0.120
	RBF kernel (Gaussian Kernel Radial Basis)	0.767	0.232	0.186
Boosting Trees	AdaBoost (Adaptive Boosting)	270.738	0.912	0.907
	XGBoost (eXtreme Gradient Boosting)	101.074	0.967	0.965
	LightGBM (Light Gradient Boosting Machine)	1678.109	0.457	0.424
Advanced Trees	Bagging Regression	507.963	0.835	0.825
	Random Forest	508.803	0.835	0.825
	ExtraTrees (Extremely Randomized Trees)	100.603	0.967	0.964
KNN	K-Nearest Neighbors	1.073	0.067	-0.216
SDG	Stochastic Gradient Descent	0.742	0.257	0.213
Decision Tree		1559.749	0.495	0.465
Multiple Regression		56.010	0.331	0.291

The predictive performance of each model can be assessed using the R^2 and Adjusted R^2 metrics. Notably, the execution of all 13 models on a dataset with 8 variables and 157 observations required less than two minutes on a conventional commercial laptop equipped with an Intel Core i5 processor and 8 GB of RAM. The analyst can readily identify the best-fit model by comparing the extracted results and determining the next steps. For the Car-sales dataset, the XGBoost (eXtreme Gradient Boosting) regression model produced the best fit (Adj.R2 = 0.965) among the tested algorithms.

Exploring datasets with Rbox+ across multiple ML models provides several advantages. It enables researchers to identify the model that best fits a given dataset, supporting informed decision-making while offering insight into the relative strengths and limitations of each approach. Because every ML model carries inherent assumptions and potential biases, evaluating diverse algorithms helps assess the robustness of results and ensures that observed patterns are consistent across methods, thereby enhancing the generalizability of findings. Additionally, models may assign varying levels of importance to features within the dataset, and comparing feature rankings across algorithms allows researchers to identify the most influential variables and gain insight into underlying data dynamics. Employing multiple models also aids in detecting overfitting, a common challenge in which models perform well on training data but fail to generalize to unseen data; this detection informs necessary adjustments or regularization strategies. Furthermore, combining models through ensemble or stacking techniques can improve predictive performance by mitigating individual model weaknesses and leveraging complementary strengths. Finally, each ML model embodies unique assumptions, and exploring a range of algorithms provides alternative perspectives that can reveal hidden patterns, thereby enrich the depth of analysis and contribute to a more comprehensive understanding of the data.

4.5 Comparative Analysis: Rbox+ vs. Existing ML Platforms

To clearly demonstrate the originality of the proposed Rbox+ interface, it is essential to compare it against the prevailing landscape of Machine Learning tools. As reviewed in recent literature (Konomos, 2026), existing tools generally fall into two categories: "No-Code/Low-Code" educational platforms and "Black-Box" enterprise analytics suites.

Accessibility vs. Depth

Educational tools like Teachable Machine, KMINE or WEKA excel at lowering the barrier to entry but often "hide" the underlying code to reduce cognitive load (Carney et al., 2020; Jain et al., 2022; Nidhya & Shah, 2023). In contrast, while Google Colab provides full code access, it requires significant prior programming knowledge (Nelson & Hoover, 2020; Ferreira et al., 2024). Rbox+ bridges this gap by providing a "dispatcher" mechanism that allows enterprise users to trigger complex models via simple JSON requests while simultaneously generating a full Python/R Jupyter Notebook.

Transparency and Reproducibility

A critical limitation of proprietary platforms (e.g., SAP Analytics Cloud or PowerBI) is the "Black Box" nature of their automated insights. While these tools provide accurate predictions, the exact mathematical formulations and data transformations are often inaccessible to end users (Ashtari & Eydgahi, 2017; Avelino & Santos, 2023; Li & Wu, 2022). The Rbox+ ensures that every result is accompanied by its source code. This fulfills the requirement for reproducible research, a feature often missing in enterprise-grade solutions.

Language Interoperability

Most existing platforms force a choice between Python and R. Even integrated environments like Anaconda or Jupyter require manual switching of kernels or specific library configurations (like rpy2), which can be unstable in production (Costa, 2023; Larose & Larose, 2019). The Rbox+ supports Python and R simultaneously within a single unified web service.

System Integration Effort

Conventional ML implementations in ERP environments such as S/4HANA often require significant infrastructure changes or specialized system architectures (Sajja, 2025; Bussu, 2024). The Rbox+ uses a standardized Web API to decouple the ML logic from the enterprise host. This allows legacy systems or even basic spreadsheet applications (such as Excel) to perform predictive analytics without installing Python or R libraries locally.

4.6 Future Research

While Rbox+ offers significant advantages, certain constraints remain. Current testing has focused on moderate-sized datasets; consequently, further research is required to assess scalability regarding high-frequency data streams and "Big Data" contexts. Additionally, the complexity of security protocols and access control in highly regulated environments necessitates deeper investigation. To address these gaps, future development could transition the "open box" framework toward more sophisticated capabilities, including automated hyperparameter tuning and distributed processing to manage large-scale data. A planned emphasis on deeper integration with cloud infrastructure to enhance the system's flexibility and deployment readiness must also be considered.

5.0 Conclusions

In conclusion, Rbox+ offers a transparent, flexible, and extensible approach to integrating ML into enterprise systems. By combining interoperability with reproducibility, the framework helps bridge the gap between advanced ML techniques and operational enterprise analytics, serving as both an educational tool and a learning resource. This paper introduces Rbox+, an innovative API that serves as a comprehensive framework for integrating ML systems. By bridging the gap between enterprise information systems and the Python/R programming environments, Rbox+ streamlines the analytics design cycle, minimizes implementation complexity, and enhances the operational utility of ML algorithms. The framework provides a unified, language-independent interface that allows complex statistical analyses to be executed seamlessly within existing corporate infrastructures.

Empirical evaluation indicates that Rbox+ efficiently executes multiple ML models within a single analytical workflow. In a demonstrated use case, 13 regression models were applied to a moderate-sized dataset, completing processing tasks on standard consumer hardware, thereby confirming the practical feasibility of the approach. Furthermore, the framework's ability to directly compare model performance using established metrics facilitates more informed model selection and validation.

The primary contributions of this study are fourfold. First, Rbox+ establishes a unified Web API that provides a generic and extensible interface, enabling enterprise systems to invoke ML algorithms without requiring native support for statistical languages. Second, the framework promotes transparency and reproducibility by exposing the generated source code and delivering results in Jupyter Notebook format, thereby fostering greater trust in ML-driven decision support. Third, the system's practical applicability is demonstrated through successful integration with SAP S/4Hana and Microsoft Excel. Finally, Rbox+ supports comparative ML evaluation by enabling the parallel execution of multiple algorithms, allowing analysts to efficiently assess predictive performance and robustness.

References

- [1] Ashtari, S. and Eydgahi, A., (2017). Student perceptions of cloud applications effectiveness in higher education. *Journal of Computational Science*, 23, 173-180. <https://doi.org/10.1016/j.jocs.2016.12.007>
- [2] Avelino, G., & Santos, P. (2023). Low-code and No-code Technologies Adoption: A Gray Literature Review. *Brazilian Symposium on Information Systems*. <https://doi.org/10.1145/3592813.3592929>
- [3] Bejani, M. M., & Ghatee, M. (2021). A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, 54(8), 6391–6438. <https://doi.org/10.1007/s10462-021-09975-1>.
- [4] Birnbaum, H. G., Greenberg, P. E., & Springerlink (2017). *Decision Making in a World of Comparative Effectiveness Research: A Practical Guide*. Springer Singapore.
- [5] Botchkarev, A. (2018). Evaluating Performance of Regression Machine Learning Models Using Multiple Error Metrics in Azure Machine Learning Studio. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3177507>.
- [6] Brunton, S. L., & Kutz, J. N. (2022). *Data-driven science and engineering: machine learning, dynamical systems, and control*. Cambridge University Press.
- [7] Bussu, V. R. R. (2024). Unlocking Business Potential: Artificial Intelligence and Machine Learning Capabilities in SAP S/4HANA. *International Journal of Innovative Science and Research Technology*. <https://doi.org/10.38124/ijisrt/ijisrt24mar644>
- [8] Carney, M., Webster, B., Alvarado, I., Phillips, K., Howell, N., Griffith, J., Jongejan, J., Pitaru, A. and Chen, A., (2020). Teachable machine: Approachable Web-based tool for exploring machine.

- [9] Chen, L.P. (2021). Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. *Technometrics*, 63(2), 272–273. <https://doi.org/10.1080/00401706.2021.1904738>.
- [10] Chourasiya, S., & Jain, S. (2019). A Study Review on Supervised Machine Learning Algorithms. *International Journal of Computer Science and Engineering*, 6(8), 16–20. <https://doi.org/10.14445/23488387/ijcse-v6i8p104>.
- [11] Costa, H. (2023). For me, the choice was R or Python. <https://doi.org/10.59350/83vew-1de35>
- [12] Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C., & Mishra, D. (2022). RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. *Applied Sciences*, 12(9), 4369. <https://doi.org/10.3390/app12094369>.
- [13] Ferreira, R., Canesche, M., Jamieson, P., Neto, O.P.V. and Nacif, J.A., (2024). Examples and tutorials on using Google Colab and Gradio to create online interactive student-learning modules. *Computer Applications in Engineering Education*, 32(4), p.e22729. <https://doi.org/10.1002/cae.22729>
- [14] Gandomi, A., & Haider, M. (2015). Beyond the Hype: Big Data Concepts, Methods, and Analytics. *International Journal of Information Management*, 35(2), 137–144. <https://doi.org/10.1016/j.ijinfomgt.2014.10.007>.
- [15] Google (2020), Google Cloud AI, Online document <https://cloud.google.com/products/ai>.
- [16] Granger, B. E., & Perez, F. (2021). Jupyter: Thinking and Storytelling with Code and Data. *Computing in Science & Engineering*, 23(2), 7–14. <https://doi.org/10.1109/mcse.2021.3059263>.
- [17] Hamid, K., Muhammad Ibrar, Amir Mohammad Delshadi, Hussain, M., Muhammad Waseem Iqbal, Hameed, A., & Noor, M. (2024). ML-based Meta-Model Usability Evaluation of Mobile Medical Apps. *International Journal of Advanced Computer Science and Applications*, 15(1). <https://doi.org/10.14569/ijacsa.2024.0150104>.
- [18] Hao, J. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. *Journal of Educational and Behavioral Statistics*, 44(3), 107699861983224. <https://doi.org/10.3102/1076998619832248>.
- [19] Hou, S., Gao, J., & Wang, C. (2023). Optimization Framework for Crowd-Sourced Delivery Services with the Consideration of Shippers' Acceptance Uncertainties. *IEEE Transactions on Intelligent Transportation Systems*, 24(1), 684–693. <https://doi.org/10.1109/tits.2022.3215512>.
- [20] Hu, X., Chen, C. L. P., & Zhang, T. (2025). Dynamic Broad Metric Learning. *IEEE Transactions on Artificial Intelligence*, 6(10), 2603–2617. <https://doi.org/10.1109/tai.2025.3553832>.
- [21] IBM (2020), IBM Watson, Online document <https://www.ibm.com/161icross>.
- [22] Jain, A., Somwanshi, D., Joshi, K. and Bhatt, S.S., (2022). A review: data mining classification techniques. In 2022 3rd International conference on intelligent engineering and management (ICIEM) 636-642. <https://doi.org/10.1109/ICIEM54221.2022.9853036>
- [23] Jiang, T., Chen, G., Wang, H., & Song, M. (2025). A Temporal Property Graph Data Model Compatible with Static Graphs and its Temporal Graph Query Language. *International Journal of Sensor Networks*, 1(1). <https://doi.org/10.1504/ijnsnet.2025.10074270>.
- [24] King, T. (2021). A Practical Guide to Manufacturing Variances in SAP S/4HANA. Espresso Tutorials GmbH.
- [25] Konomos, A. (2026). Mind the Gap: A Review of Machine Learning Tools in Educational Settings. *Global Journal of Business and Integral Security*, 9(1)
- [26] Konomos, A., & Chountasis, S. (2023). Rbox: A web API for software integration with the R programming language. *Computer Applications in Engineering Education*, <https://doi.org/10.1002/cae.22621>.
- [27] Lalaoui, I. L., El Haji, E., & Kounaidi, M. (2025). The Evolution and Challenges of Real-Time Big Data: A Review. *Computer Sciences & Mathematics Forum*, 10(1), 11. <https://doi.org/10.3390/cmsf2025010011>.
- [28] Larose, C.D. and Larose, D.T., (2019). *Data science using Python and R*. John Wiley & Sons.
- [29] Li, L. and Wu, Z. W. (2022) How Can No/Low Code Platforms Help End-Users Develop ML Applications? - A Systematic Review, 338–356. https://doi.org/10.1007/978-3-031-21707-4_25

- [30] Microsoft (2020), Microsoft Azure Cognitive Services, Online document <https://azure.microsoft.com/en-us/services/cognitive-services>.
- [31] Muhammad Imran Sarwar Abbas, Q., Tahir Alyas, Alzahrani, A., Alghamdi, T., & Yazed Alsaawy. (2023). Digital Transformation of Public Sector Governance with IT Service Management–A Pilot Study. *IEEE Access*, 11, 6490–6512. <https://doi.org/10.1109/access.2023.3237550>.
- [32] Nelson, M.J. and Hoover, A.K., (2020). Notes on using Google Colaboratory in AI education. In *Proceedings of the 2020 ACM conference on innovation and Technology in Computer Science Education* pp. 533-534.
- [33] Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An Analysis of Public REST Web Service APIs. *IEEE Transactions on Services Computing*, 1-1.
- [34] Nidhya, M.S. and Shah, P.K., (2023). Acquiring Knowledge by Performing Classification and Clustering of Datasets using WEKA: Intelligence through MLP, RF, DT, and RepTree. In *2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI)* 833-837. <https://doi.org/10.1109/ICCSAI59793.2023.10421077>
- [35] Parkash, V. (2023). Fourier Series Approximation of Square and Sawtooth Waves Using Python Script in Jupyter Notebook. *International Journal of Science and Research (IJSR)*, 12(3), 1834–1840. <https://doi.org/10.21275/sr23326170631>.
- [36] Rajput, A., & Tiwari, S. (2023). A Review on Privacy Preserving Using Machine Learning and Deep Learning Techniques. *International Journal for Research in Applied Science and Engineering Technology*, 11(3), 1785–1790. <https://doi.org/10.22214/ijraset.2023.49781>.
- [37] Sajja, J. W. (2025). AI-Powered Hyperautomation in SAP S/4HANA Migration: Transforming ERP Transitions. *European Journal of Computer Science and Information Technology*, 13(32), 88–115. <https://doi.org/10.37745/ejcsit.2013/vol13n3288115>
- [38] Shu, X., & Ye, Y. (2022). Knowledge Discovery: Methods from data mining and machine learning. *Social Science Research*, 110, 102817. <https://doi.org/10.1016/j.ssresearch.2022.102817>.
- [39] Song, Y. (2021). Comparison of Several Machine Learning Algorithms for Prediction. *Journal of Physics: Conference Series*, 1952(2), 022065. <https://doi.org/10.1088/1742-6596/1952/2/022065>.
- [40] Westfall, P. H., & Arias, A. L. (2020). *Understanding Regression Analysis*. CRC Press.
- [41] Wufka, M., Canonico, M. (2026). Edge Computing, Internet of Things and Cloud-to-Edge Continuum. In: *Introduction to Cloud Computing*. Springer, Cham. https://doi.org/10.1007/978-3-032-07151-4_9.
- [42] Yang, X., Guo, Y., Dong, M., & Xue, J.-H. (2024). Toward Certified Robustness of Distance Metric Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(3), 3834–3844. <https://doi.org/10.1109/tnnls.2022.3199902>.
- [43] Zhao, L., Wang, Q., Wang, C., Li, Q., Shen, C., & Feng, B. (2021). VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service. *IEEE Transactions on Parallel and Distributed Systems*, 32(10), 2524–2540. <https://doi.org/10.1109/tpds.2021.3068195>.