Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 26 Issue 1 Version 1.0 Year 2026
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Integrating Machine Learning into Business Management Systems: The Rbox+ API

By Antonios Konomos & Spiros Chountasis

*Abstract-* This paper presents an innovative software interface for the utilization of widely used machine learning algorithms in a unified Python/R programming environment. This study makes two contributions. First, a more comprehensive and specialized architecture is made available for integrating machine learning into enterprise information systems. Second, a novel software model, Rbox+, is proposed to execute machine learning algorithms by jointly leveraging the capabilities of the Python and R programming languages through an Application Programming Interface (API). The proposed API is tested and evaluated using a publicly available benchmark dataset for regression analysis (Car-sales dataset, available on Kaggle), applying multiple machine learning models and comparative performance metrics. The obtained results demonstrate improved computational efficiency and scalability, with the execution of multiple models completed within a short processing time on standard hardware. Unlike conventional machine learning APIs or isolated ERP analytics tools, Rbox+ enables transparent, language-independent execution and validation of machine learning models while exposing the underlying source code. The proposed approach supports practical applications in enterprise analytics, reproducible research, and machine learning education, enhancing interoperability between ERP systems, analytics platforms, and statistical programming environments.

*Keywords: software interface, software applications, computer technologies, machine learning.*

*GJCST-C Classification: LCC Code: QA76.9.A43*

INTEGRATINGMACHINELEARNINGINTOBUSINESSMANAGEMENTSYSTEMSTHERBOXAPI

# Integrating Machine Learning into Business Management Systems: The Rbox+ API

Antonios Konomos [α] & Spiros Chountasis [σ]

*Abstract-* This paper presents an innovative software interface for the utilization of widely used machine learning algorithms in a unified Python/R programming environment. This study makes two contributions. First, a more comprehensive and specialized architecture is made available for integrating machine learning into enterprise information systems. Second, a novel software model, Rbox+, is proposed to execute machine learning algorithms by jointly leveraging the capabilities of the Python and R programming languages through an Application Programming Interface (API). The proposed API is tested and evaluated using a publicly available benchmark dataset for regression analysis (Car-sales dataset, available on Kaggle), applying multiple machine learning models and comparative performance metrics. The obtained results demonstrate improved computational efficiency and scalability, with the execution of multiple models completed within a short processing time on standard hardware. Unlike conventional machine learning APIs or isolated ERP analytics tools, Rbox+ enables transparent, language-independent execution and validation of machine learning models while exposing the underlying source code. The proposed approach supports practical applications in enterprise analytics, reproducible research, and machine learning education, enhancing interoperability between ERP systems, analytics platforms, and statistical programming environments.

*Keywords: software interface, software applications, computer technologies, machine learning.*

## I. Introduction

### *a) ML Overview and Challenges*

Machine Learning (ML) is the process of creating software applications that enables a system to gain new knowledge from training data rather than through explicit programming. It employs several algorithms that learn from data iteratively to enhance, characterize, and predict outcomes. More accurate models based on the training data can then be created as the algorithms ingest training data. ML can be categorized into supervised, unsupervised learning and reinforcement learning. Many effective ML applications, from information-filtering systems to data-mining techniques, have been developed in recent years.

*Author α: Intrasoft International SA, Athens, Greece.*
*e-mail: akonomos@gmail.com*
*Author σ: Independent Power Transmission Operator, Athens, Greece.*
*e-mail: schountasis@admie.gr*

### *b) Python and R Ecosystems*

The frameworks and algorithms that are the key components of this field have also undergone substantial modifications. Today's technological advancements have updated forms of software engineering and computer architecture [7,15,31]. Furthermore, large volume of data management stresses the need for more efficient visualization and presentation tools [9,19]. Leading open-source programming languages like Python and R, are at the forefront of requisite quantitative computer programs [8,14,22]. Both programming languages have their own distinctive characteristics that encompass a wide range of ML algorithms, statistical controls, and strong 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 unexpected, it is challenging to anticipate in advance the effort of training a model. The model training data set, learning parameter settings, and random variables all affect convergence [11,27].

### *c) MLaaS and Cloud Deployment*

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 [16,18]. 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 [35].

### *d) The Rbox+ Initiative*

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+), which is responsible for sending data to the R or Python programming environments, requesting 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 [29] 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 is intended to further develop an interface for integrating Python/R with the SAP S/4 Hana ERP system. SAP S/4 Hana ERP is an ERP business suite based on the SAP HANA in-memory database [4].

### *e) Rbox+ Features*

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 the ML algorithms if needed, as their validation becomes more critical than ever before.

### *f) Organization of the Paper*

The outline of this paper is organized as follows. Section 2 presents the Rbox+ interface and its utilization, emphasizing the implementation of ML algorithms. Section 3 introduces and analyzes the software architecture of the proposed system from a Web service perspective. Section 4 presents simulation results and code implementation based on a use-case scenario involving multiple ML regression algorithms. Finally, Section 5 concludes the paper with a discussion and an outlook on future work.

## II. The Python/R Integration for the Rbox+ Utilization

### *a) Rbox+Overview*

Machine learning data processing and visualization tools are rarely embedded directly into information systems that are commonly used for managing an organization's core operations [1]. 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 [34]. Despite its widespread adoption, SAP S/4 Hana lacks native advanced analytics, machine learning execution, and data visualization capabilities. Instead, these needs are typically addressed through separate analytical platforms such as SAP BI or SAP Analytics Cloud [30].

Rbox+ is designed to bridge this gap by providing a lightweight and flexible Web API that enables enterprise systems to directly invoke machine learning and statistical algorithms implemented in Python and R. 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 the perspective of system integration, Rbox+ simplifies the interaction 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.

### *b) Integration with Python and R*

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 [24]. From an integrator's perspective, Rbox+ simplifies the integration process by minimizing the required implementation effort. The integration consists of a single step, which involves preparing and executing a direct service call through Rbox+'s dispatcher method. This design simplicity and flexibility make Rbox+ highly user-friendly.

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 machine learning 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.

### *c) Output and Reproducibility via Jupyter Notebook*

From an end-user perspective, Rbox+ generates output in the form of 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 [13]. 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 final 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 [10,21].

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 [23].

Furthermore, the 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 ease of use and standardized output format make Rbox+ applicable to both introductory ML coursework and advanced analytical research.

In summary, 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.

### *d) Machine Learning Process in Rbox+*

A wide range of companies, including Google Cloud AI [20], Amazon Web Services (AWS) AI [2], IBM Watson [17], and Microsoft Azure [25], provide cloud-based machine learning 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 present several challenges, particularly when ML applications must be integrated into larger software systems. One approach to making ML more powerful, practical, and interoperable is the development of 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 through the introduction of Rbox+, a web-service-based interface that enables researchers and developers to perform statistical analyses and execute ML algorithms in the R and Python environments. Rbox+ provides a simplified and uniform access layer for ML execution, facilitating integration and comparative analysis across platforms. Tables 1 and 2 present the Rbox+ templates developed using the Python and R programming language.

*Table 1:* ML Algorithms in Python

| Template | ML Task | Options / Hyperparameters |
|---|---|---|
| Decision Trees | Classification/Regression | Descriptives<br>Correlation matrix<br>k-fold Cross Validation |
| Advanced Trees | Classification/Regression | Descriptives<br>Correlation matrix<br>Bagging<br>Random Forest<br>Extremely Randomized Trees<br>k-fold Cross Validation |
| Boosting Trees | Classification/Regression | Descriptives<br>Correlation matrix<br>AdaBoost<br>XG Boost<br>Light GBM<br>k-fold Cross Validation |
| Gaussian Mixture | Clustering | Descriptives<br>Correlation matrix |
| Mini Batch K. means | Clustering | Descriptives<br>Correlation matrix |

| | | |
|---|---|---|
| Naive Bayes | Classification | Descriptives<br>Correlation matrix<br>k-fold Cross Validation |
| Stochastic Gradient Descent | Classification/Regression | Descriptives<br>Correlation matrix<br>k-fold Cross Validation |
| Support Vector Machines | Classification/Regression | Descriptives<br>Correlation matrix<br>Linear kernel<br>RBF kernel<br>Sigmoid kernel<br>Polynomial kernel<br>k-fold Cross Validation |

*Table 2:* ML & Statistical Algorithms in R

| Template | ML Task | Options/Hyperparameters |
|---|---|---|
| k-means | Clustering | Descriptives<br>Silhouette method<br>Gap statistic method<br>Anova |
| K-prototypes | Clustering | Descriptives<br>Silhouette method<br>Anova<br>Chi-square tests |
| Hierarchical Clustering | Clustering | Descriptives<br>Anova |
| Partitioning Around Medoids (PAM) | Clustering | Descriptives<br>Silhouette method<br>Gap statistic method<br>Anova<br>Chi-square tests |
| Clustering Large Applications (CLARA) | Clustering | Descriptives<br>Silhouette method<br>Gap statistic method<br>Anova<br>Chi-square tests |
| K-Nearest Neighbors | Classification/ Regression | Descriptives<br>Correlation matrix<br>Elbow method<br>ROC curve |
| Linear Regression | Regression | Descriptives<br>Pearson's Correlation<br>R.square value<br>Residuals distribution<br>NCV test<br>Spread Level Plot |
| Linear Multiple Regression | Regression | Descriptives<br>Correlation matrix<br>Collinearity test<br>K-fold cross-validation<br>Stepwise Regression |
| Logistic Regression | Classification | Descriptives<br>Correlation matrix<br>Collinearity test<br>Hosmer test (Goodness of fit)<br>Predictive ability<br>ROC curve |
| Multinomial Logistic Regression | Classification | Descriptives<br>Correlation matrix<br>Evaluate Collinearity<br>Stepwise Regression |

| | | |
|---|---|---|
| Principal Component Analysis (PCA) | Dimensionality Reduction | Descriptives<br>Correlation matrix<br>Correlations Vars – PCs |
| PCA mix | Dimensionality Reduction | Descriptives<br>Plots |
| Random Forest | Classification / Regression | Descriptives<br>Correlation matrix |
| t-Test | Association | Descriptives<br>Kolmogorov-Smirnov<br>Levene's test<br>Mann-Whitney U<br>Boxplots<br>Histogram |
| Paired t-Test | Association | Descriptives<br>Pearson's Correlation<br>Kolmogorov-Smirnov<br>Mann-Whitney<br>Histograms |
| One-way Anova | Association | Descriptives<br>Tukey HSD (honest significance test)<br>Levene's test (homoskedasticity test)<br>Diagnostic plots<br>Kruskal Wallis |
| Two-way Anova | Association | Descriptives<br>Tukey HSD<br>Pairwise t-tests<br>Levene's test<br>Normality test |
| $x^2$ test | Association | Descriptives<br>Crosstabs<br>Balloon plot<br>Barplot |

### *e) The Rbox+ Learning Aspect*

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 machine learning algorithms and their execution context, while also supporting the development of Python/R programming skills.

Data analysis in a machine learning 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 [32]. 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 [26]. Within this framework, Rbox+ provides an accessible platform for exploring complex statistical and machine learning techniques, supporting the learning process of its users.

## III. System Architecture

### *a) The Three-Tier Design*

This section describes the System Architecture, which is defined independently of the implementation details. The Rbox+ API operates on a web server that is connected to both the R and Python programming environments and is implemented using the PHP programming language. The machine learning methods included in Rbox+ are accessible to any application capable of handling HTTP requests. The design of Rbox+ follows a conventional three-tier architecture [24]. More specifically, it is based on a client-server model consisting of three distinct processing layers, as illustrated in *Figure 1:*

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

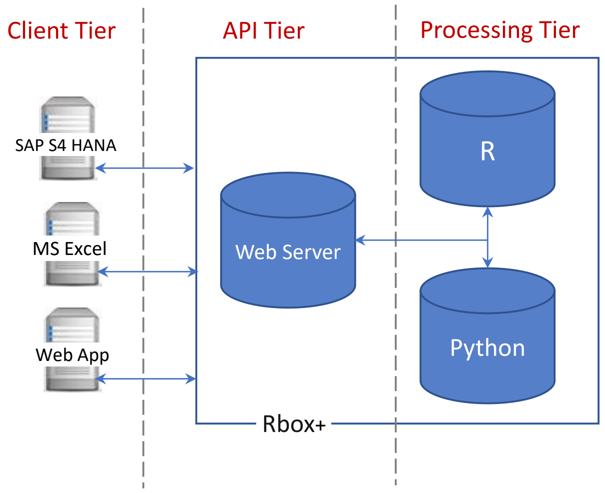


*Figure 1:* Rbox+ Three-tier Architecture

*b) Request JSON Structure*

All three tiers (Processing, API, and Client tier), operate in accordance with the previously proposed Rbox API [24] and are briefly described in the following section. The API tier functions as a web server that handles HTTP requests via POST method. Each request carries a JSON structure (Table 1) containing information about the requested machine learning algorithm (implemented in Python/R), as well as the dataset to which the model will be applied.

*Table 3:* JSON Structure

| Variable | Description | JSON sample for t-Test |
|---|---|---|
| template | ML / Statistical test | {"template": "P.NaiveBayes", "numvars": 2, "numopts": 6, "options": [1,1,1,1,1,1], "variables": [ {"id": "X0", "name": "VEHICLE_TYPE", "datatype": "factor", "values": ["Car", "Passenger",…]}, {"id": "X1", "name": "PRICE", "datatype": "numeric", "values": ["21.50", "28.40", …]}]} |
| numvars | Variables counter | |
| numopts | Options counter | |
| options | Options on/off | |
| variables | Variables list | |
| id | $x_1 \ldots x_n$ | |
| name | Variable name | |
| datatype | Variable type | |
| values | Value list | |

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 machine learning outputs are presented in Jupyter Notebook format [29], allowing end users to view both the generated Python/R source code and the corresponding computational results within a single HTML document.

The JSON element template specifies the requested machine learning technique or statistical test. As shown in Table 1, the options parameter includes on/off switches that control the hyperparameters of the ML algorithm or enable additional tests associated with 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 $x_0$ to $x_n$. The numvars parameter specifies the total number of variables, while numopts indicates the number of available on/off switches within the options array.

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 request.

## IV. Simulation Results and Code Implementation

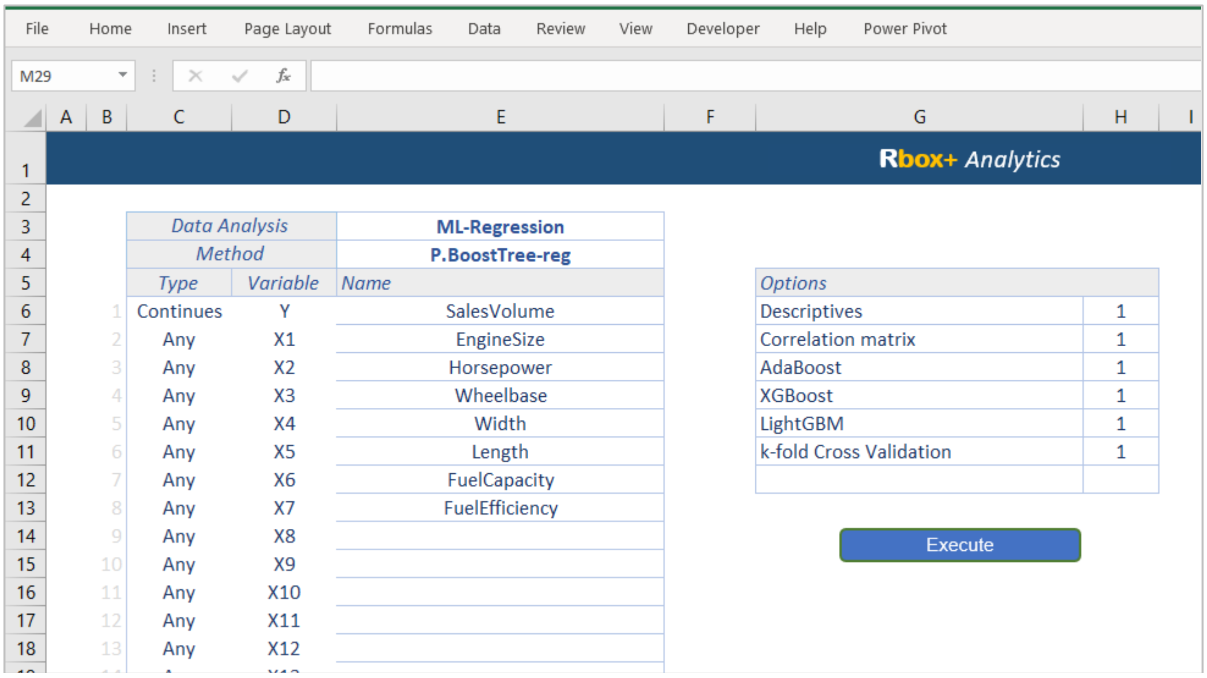
*a) Sample Dataset and use-case Scenario*

In this section, 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 data set 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 machine learning 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 $R^2$ and adjusted $R^2$ metrics [3].

*b) Model Selection*

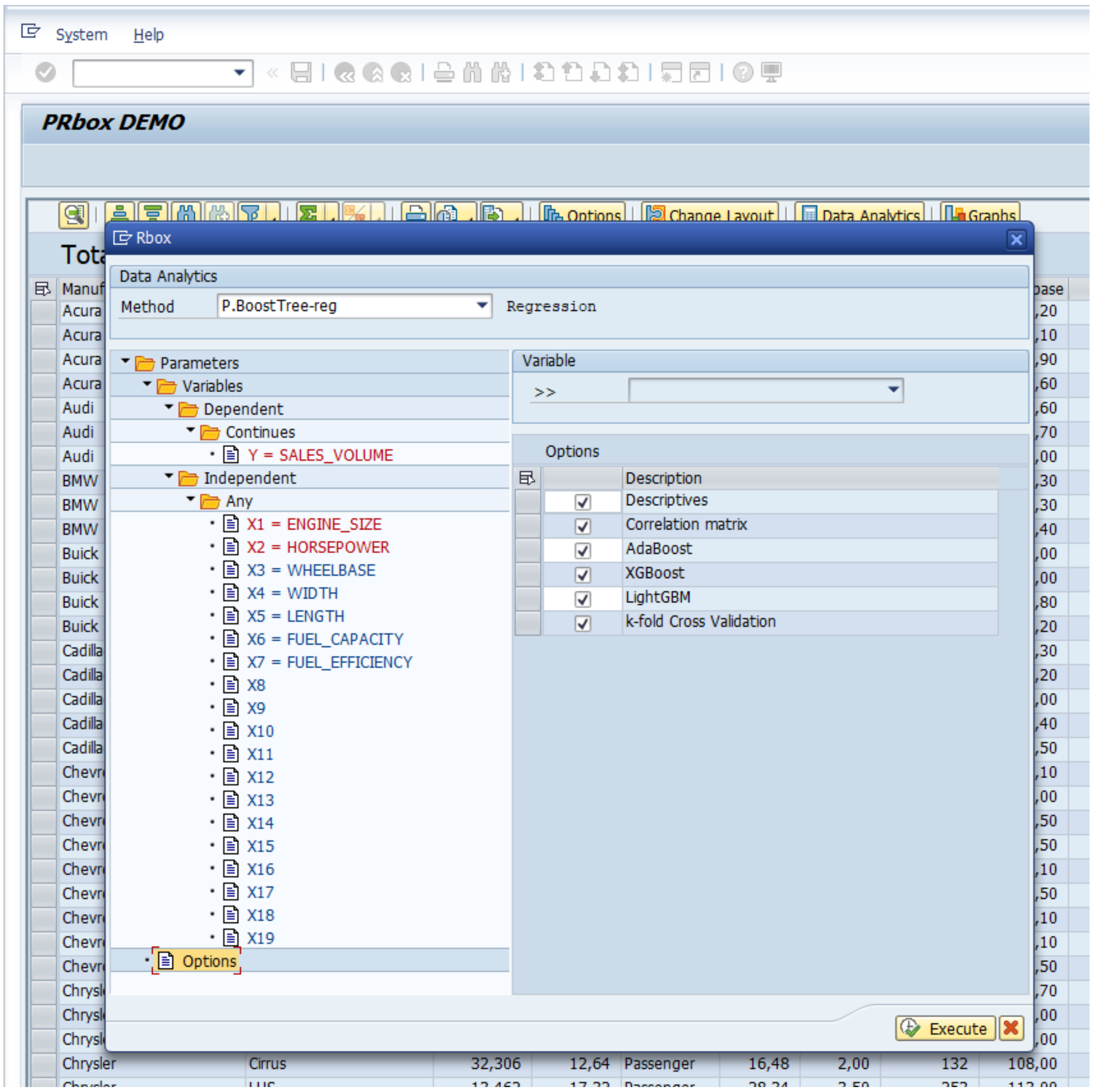
After importing the dataset in the Rbox+ Excel Client, the researcher can select the Data Analysis model (Regression, Classification, Clustering or Dimensionality Reduction) and the desired ML method. At this step, it is important to specify the response variable as well as the independent variables.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



| Data Analysis | | ML-Regression |
|---|---|---|
| Method | | P.BoostTree-reg |
| Type | Variable | Name |
| Continues | Y | SalesVolume |
| Any | X1 | EngineSize |
| Any | X2 | Horsepower |
| Any | X3 | Wheelbase |
| Any | X4 | Width |
| Any | X5 | Length |
| Any | X6 | FuelCapacity |
| Any | X7 | FuelEfficiency |
| Any | X8 | |
| Any | X9 | |
| Any | X10 | |
| Any | X11 | |
| Any | X12 | |

| Options | |
|---|---|
| Descriptives | 1 |
| Correlation matrix | 1 |
| AdaBoost | 1 |
| XGBoost | 1 |
| LightGBM | 1 |
| k-fold Cross Validation | 1 |

*Figure 2:* Rbox+ Excel Client



Data Analytics

Method: P.BoostTree-reg — Regression

- Parameters
  - Variables
    - Dependent
      - Continues
        - Y = SALES_VOLUME
    - Independent
      - Any
        - X1 = ENGINE_SIZE
        - X2 = HORSEPOWER
        - X3 = WHEELBASE
        - X4 = WIDTH
        - X5 = LENGTH
        - X6 = FUEL_CAPACITY
        - X7 = FUEL_EFFICIENCY
        - X8
        - X9
        - X10
        - X11
        - X12
        - X13
        - X14
        - X15
        - X16
        - X17
        - X18
        - X19
- Options

Variable: >>

Options

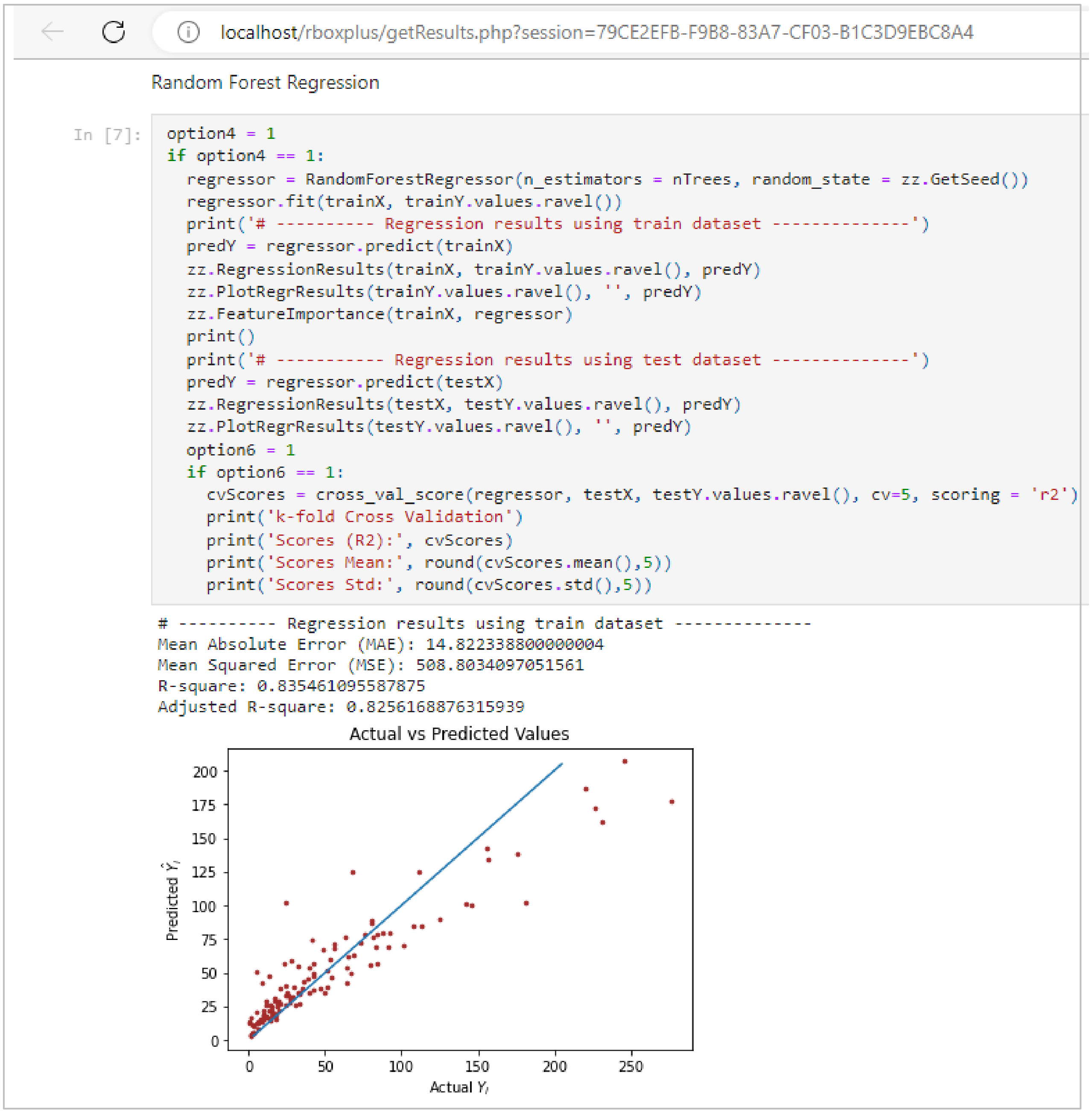
| | Description |
|---|---|
| ✓ | Descriptives |
| ✓ | Correlation matrix |
| ✓ | AdaBoost |
| ✓ | XGBoost |
| ✓ | LightGBM |
| ✓ | k-fold Cross Validation |

Chrysler Cirrus 32,306 12,64 Passenger 16,48 2,00 132 108,00

*Figure 3:* Rbox+SAP S4 Hana Client

## c) *Results Page*

The calculation outcomes are presented in Jupyter Notebook format within the user's web browser, as illustrated in Figures 4 and 5. 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.



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

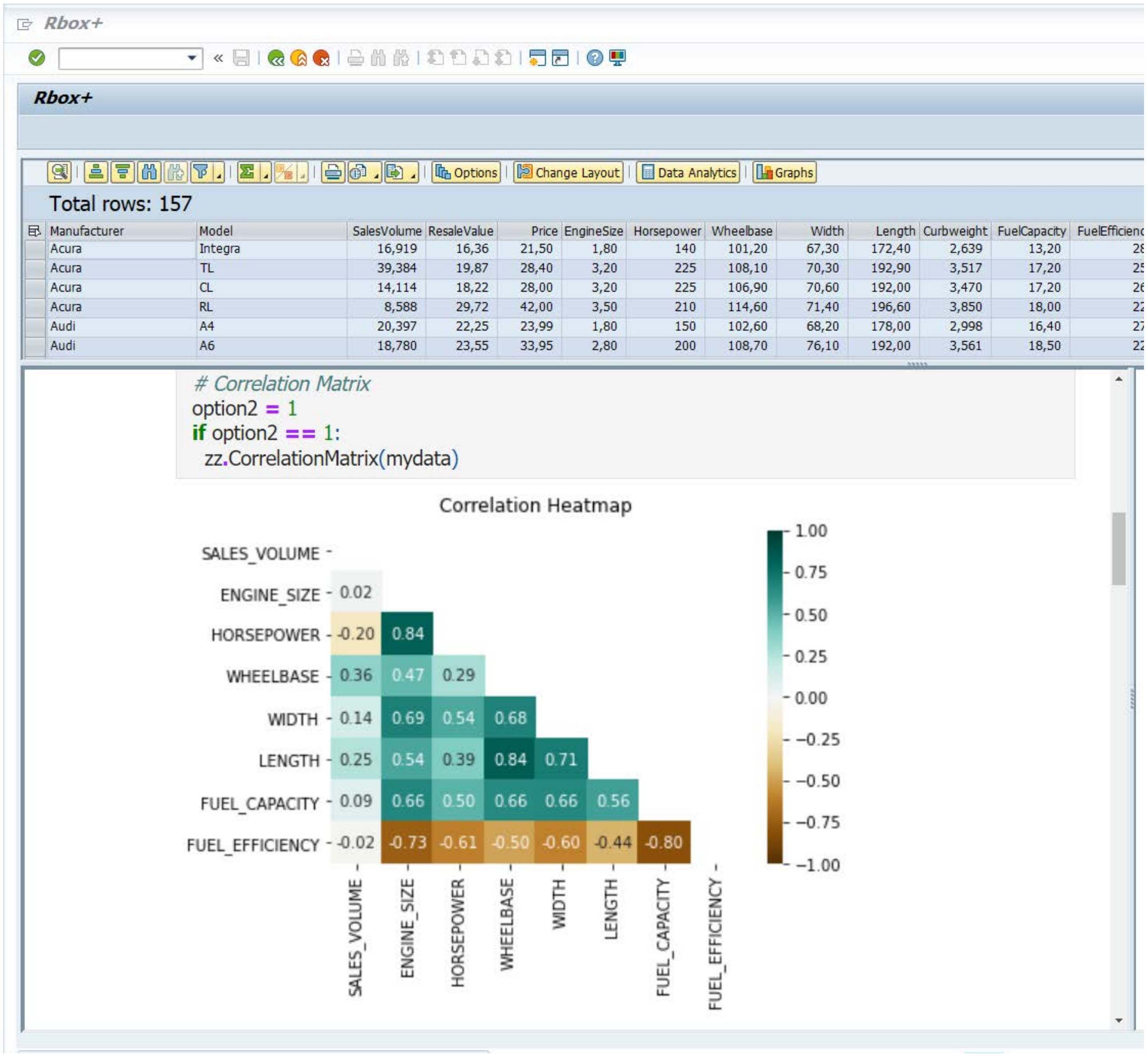
Random Forest Regression

```
In [7]: option4 = 1
        if option4 == 1:
          regressor = RandomForestRegressor(n_estimators = nTrees, random_state = zz.GetSeed())
          regressor.fit(trainX, trainY.values.ravel())
          print('# ---------- Regression results using train dataset --------------')
          predY = regressor.predict(trainX)
          zz.RegressionResults(trainX, trainY.values.ravel(), predY)
          zz.PlotRegrResults(trainY.values.ravel(), '', predY)
          zz.FeatureImportance(trainX, regressor)
          print()
          print('# ----------- Regression results using test dataset --------------')
          predY = regressor.predict(testX)
          zz.RegressionResults(testX, testY.values.ravel(), predY)
          zz.PlotRegrResults(testY.values.ravel(), '', predY)
          option6 = 1
          if option6 == 1:
            cvScores = cross_val_score(regressor, testX, testY.values.ravel(), cv=5, scoring = 'r2')
            print('k-fold Cross Validation')
            print('Scores (R2):', cvScores)
            print('Scores Mean:', round(cvScores.mean(),5))
            print('Scores Std:', round(cvScores.std(),5))
```

```
# ---------- Regression results using train dataset --------------
Mean Absolute Error (MAE): 14.822338800000004
Mean Squared Error (MSE): 508.8034097051561
R-square: 0.835461095587875
Adjusted R-square: 0.8256168876315939
```

*Figure 4:* Rbox + Results: Jupyter Notebook

Global Journal of Computer Science and Technology ( C ) XXVI Issue I Version I Year 2026


Rbox+

Rbox+

Total rows: 157

| Manufacturer | Model | SalesVolume | ResaleValue | Price | EngineSize | Horsepower | Wheelbase | Width | Length | Curbweight | FuelCapacity | FuelEfficienc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Acura | Integra | 16,919 | 16,36 | 21,50 | 1,80 | 140 | 101,20 | 67,30 | 172,40 | 2,639 | 13,20 | 28 |
| Acura | TL | 39,384 | 19,87 | 28,40 | 3,20 | 225 | 108,10 | 70,30 | 192,90 | 3,517 | 17,20 | 25 |
| Acura | CL | 14,114 | 18,22 | 28,00 | 3,20 | 225 | 106,90 | 70,60 | 192,00 | 3,470 | 17,20 | 26 |
| Acura | RL | 8,588 | 29,72 | 42,00 | 3,50 | 210 | 114,60 | 71,40 | 196,60 | 3,850 | 18,00 | 22 |
| Audi | A4 | 20,397 | 22,25 | 23,99 | 1,80 | 150 | 102,60 | 68,20 | 178,00 | 2,998 | 16,40 | 27 |
| Audi | A6 | 18,780 | 23,55 | 33,95 | 2,80 | 200 | 108,70 | 76,10 | 192,00 | 3,561 | 18,50 | 22 |

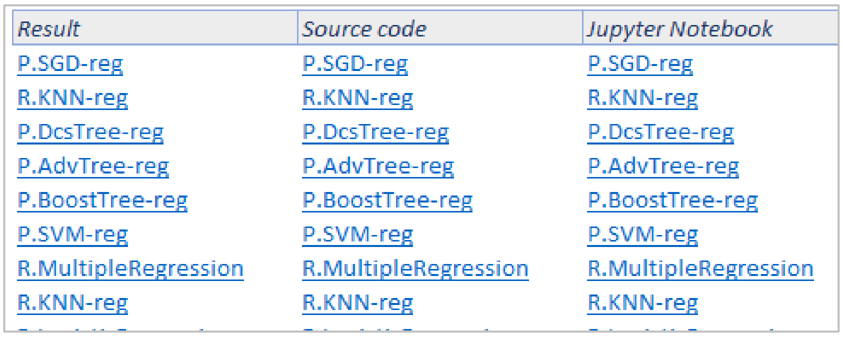
```
# Correlation Matrix
option2 = 1
if option2 == 1:
 zz.CorrelationMatrix(mydata)
```

Correlation Heatmap

|  | SALES_VOLUME | ENGINE_SIZE | HORSEPOWER | WHEELBASE | WIDTH | LENGTH | FUEL_CAPACITY | FUEL_EFFICIENCY |
|---|---|---|---|---|---|---|---|---|
| SALES_VOLUME |  |  |  |  |  |  |  |  |
| ENGINE_SIZE | 0.02 |  |  |  |  |  |  |  |
| HORSEPOWER | -0.20 | 0.84 |  |  |  |  |  |  |
| WHEELBASE | 0.36 | 0.47 | 0.29 |  |  |  |  |  |
| WIDTH | 0.14 | 0.69 | 0.54 | 0.68 |  |  |  |  |
| LENGTH | 0.25 | 0.54 | 0.39 | 0.84 | 0.71 |  |  |  |
| FUEL_CAPACITY | 0.09 | 0.66 | 0.50 | 0.66 | 0.66 | 0.56 |  |  |
| FUEL_EFFICIENCY | -0.02 | -0.73 | -0.61 | -0.50 | -0.60 | -0.44 | -0.80 |  |

*Figure 5:* Rbox + SAP S4 Hana output

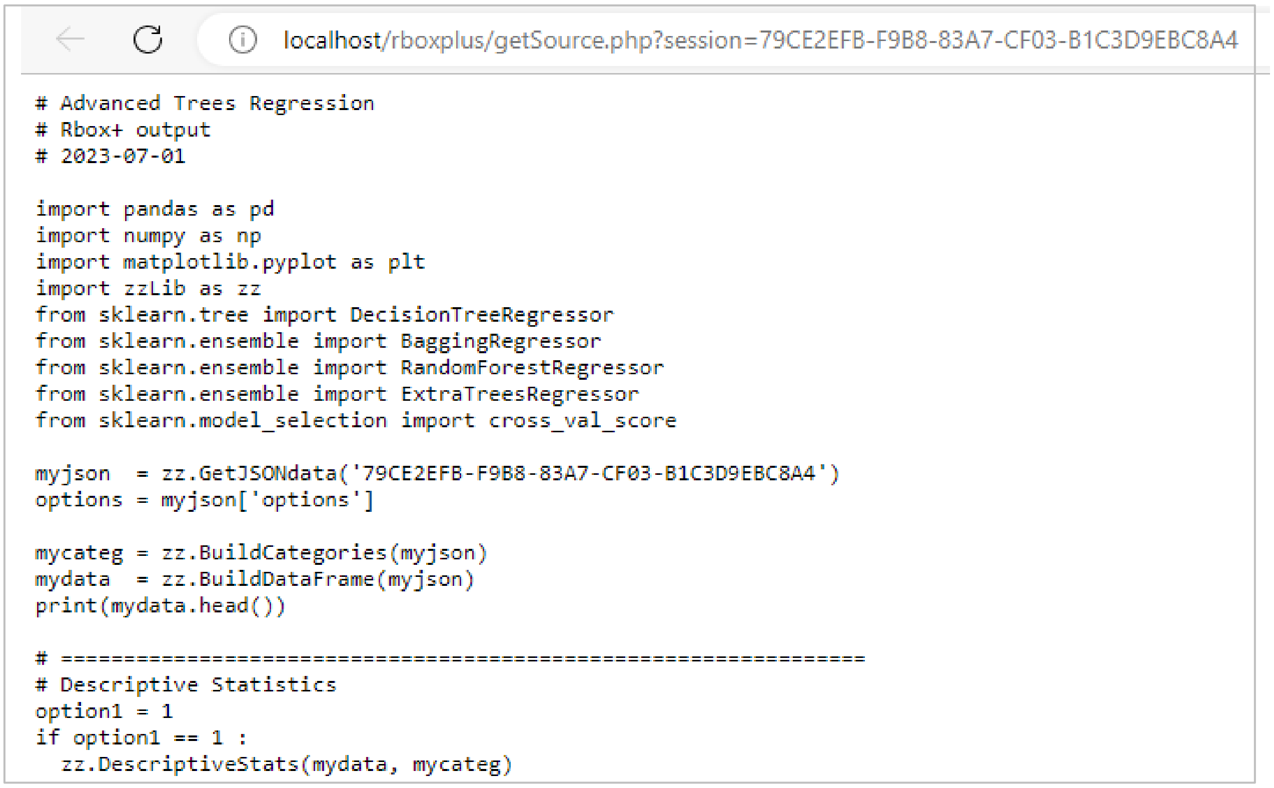
In the Excel worksheet, a table is displayed that contains hyperlinks to the results page, the generated source code, and the corresponding Jupyter Notebook file. Through these links, the analyst can access the complete history of the performed computations and revisit or reassess the associated results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

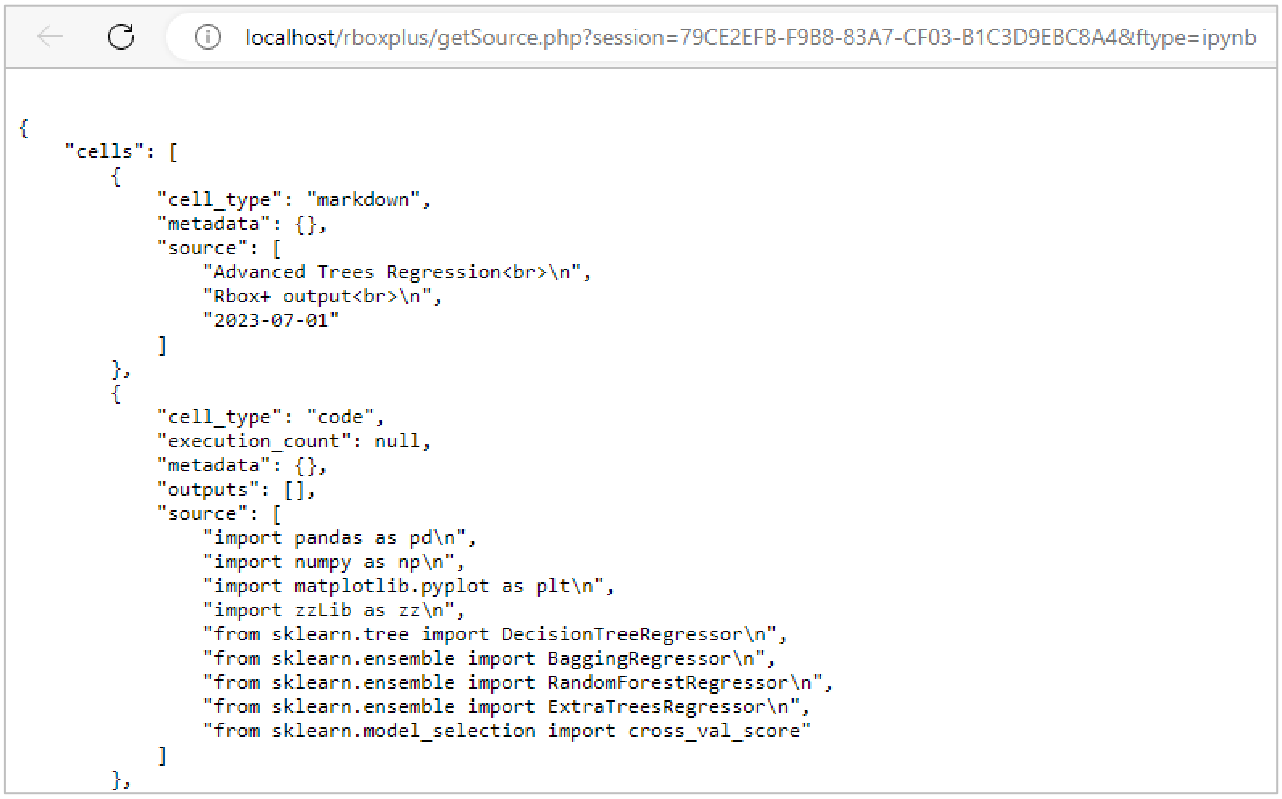
*Figure 6:* Rbox + API hyperlinks

The users can export the entire Python or R source code, as depicted in Figures 7 and 8, and subsequently 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

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

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import zzLib 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)
```

*Figure 7:* The Source Code


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 zzLib 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"
            ]
        },
```

*Figure 8:* Jupyter Notebook (.ipynb) file

*d) 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.

*Table 4:* ML Regression Algorithms and their Results

| ML Regression | Method | MSE | $R^2$ | Adj.$R^2$ |
|---|---|---|---|---|
| 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 |
| SGD | 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 |

Seven discrete ML regression algorithms were evaluated, some with multiple variations, resulting in a total of 13 distinct result sets. 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.$R^2$= 0.965) among the tested algorithms.

*e) Findings and Key Take aways*

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 [33]. Because every ML model carries inherent assumptions and potential biases [6], 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 [36].

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 [5]; 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 [28].

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 [12].

*f) Limitations and 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 necessitate 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 cloud infrastructure integration to enhance the system's flexibility and deployment readiness must also be considered.

## V. Conclusions

This paper introduces Rbox+, an innovative API designed as a comprehensive framework for the integration of machine learning 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, thirteen regression models were applied to a moderate-sized dataset, completing processing tasks on standard consumer hardware and 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 generated source code and delivering results in Jupyter Notebook format, which fosters greater trust in ML-driven decision support. Third, the practical applicability of the system 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.

The identified limitations and future research directions for Rbox+ focus on four primary areas: scalability, data governance, security protocols, and functional expansion.

In conclusion, Rbox+ offers a transparent, flexible, and extensible approach to integrating machine learning into enterprise systems. By combining interoperability with reproducibility, the framework helps bridge the gap between advanced ML techniques and operational enterprise analytics, acting in parallel as an educational and learning tool.

*Conflict of Interest Statement*

Spiros Chountas and Antonios Konomos declare no conflicts of interest.

## References Références Referencias

1. O. Ali, A. Nassif, L. Capretz: *Business intelligence solutions in healthcare a case study: Transforming OLTP system to BI solution*, IEEE 2013 Third International Conference on Communications and Information Technology (ICCIT), (2013), 209-214.
2. Amazon, *Amazon Artificial Intelligence Service*, https://aws.amazon.com/machine-learning/ai-services, 2020.
3. A. L. Arias, P. H. Westfall, *Understanding Regression Analysis: A Conditional Distribution Approach*, United States: CRC Press, 2020.
4. D. Bardhan, A. Baumgartl, N. Choi, M. Dudgeon, *SAP S/4 HANA An Introduction*, SAP Press, Waldorf, 2021.
5. M. M. Bejani, M. Ghatee: *A systematic review on overfitting control in shallow and deep neural networks*, Artificial Intelligence Review, 54(8), (2021), 6391-6438.
6. A. Bellet, A. Habrard: *Robustness and generalization for metric learing, Neurocomputing,* 151(1), (2015), 259-267.
7. E. Borgia: *The Internet of Things vision: Key features, applications and open issues*, Computer Communications, 54(2014), 1-31.
8. P. Bruce, A. Bruce, P. Gedeck, *Practical Statistics for Data Scientists: 50+ Essential Concepts using R and Python,* O'Reilly, Sevastopol 2020.
9. L. Cao: *Data Science: A Comprehensive Overview*, ACM Computing Surveys, 50(2017), 1-42.
10. A. Cardoso, J. Leitão, C. Teixeira: *using the Jupyter notebook as a tool to support the teaching and learning processes in engineering courses*, The Challenges of the Digital Transformation in Education: Proceedings of the 21st International Conference on Interactive Collaborative Learning (ICL2018), Springer, 2, (2019), 227-236.
11. Y. Chen, Z. Yuan, B. Chen: *Process optimization with consideration of uncertainties – An overview*, Chinese Journal of Chemical Engineering, 26(8), (2018), 1700-1706.
12. V. Cox: *Translating Statistics to Make Decisions: A Guide for the Non-Statistician*, Apress, California 2017.
13. J. Hao, T. K. Ho: *Machine learning made easy: a review of scikit-learn package in python programming language*, Journal of Educational and Behavioral Statistics, 44(3), (2019), 348-361.
14. T. Haslwanter, *An Introduction to Statistics with Python,* Springer, Linz, 2016.
15. B. Hazen, C. Boone, J. Ezell, D. Jones-Farme: *Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and application*, International Journal of Production Economics, 154(2014), 72-80.

16. J. C. Huang, K. M. Ko, M. H. Shu, B. M. Hsu: *Application and comparison of several machine learning algorithms and their integration models in regression problems*, Neural Computing and Applications, 32, (2020), 5461-5469.
17. IBM, *IBM Watson*, Online document https://www.ibm.com/14icros, 2020.
18. R. Ihaka, R. Gentleman: *A Language for Data Analysis and Graphics*, Journal of Computational and Graphical Statistics, 5(1996), 299-314.
19. A. Gandomi, M. Haider: *Beyond the hype: Big data concepts, methods, and analytics*, International Journal of Information Management, 35(2015) 137 – 144.
20. Google, *Google Cloud AI*, Online document https://cloud.google.com/ products/ai, 2020.
21. B. E. Granger, F. Pérez: *Jupyter: Thinking and storytelling with code and data*, Computing in Science & Engineering, 23(2), (2021), 7-14.
22. N. Jukic, A. Sharma, J. Nestorov B.: *Augmenting Data Warehouses with Big Data*, Information Systems Management, 32(2015), 200-209.
23. M. B. Kery, M. Radensky, M. Arya, B. E. John, B. A. Myers: *The story in the notebook: Exploratory data science using a literate programming tool*, Proceedings of the 2018 CHI conference on human factors in computing systems, (2018), 1-11.
24. A. Konomos, S. Chountasis: *Rbox: A web API for software integration with the R programming language*, Computer Applications in Engineering Education, DOI: 10.1002/cae.2262, (2023).
25. Microsoft, *Microsoft Azure Cognitive Services*, Online document https://azure.microsoft.com/en-us/services/cognitive-services, 2020.
26. S. A. Naghibi, H. R. Pourghasemi: *A comparative assessment between three machine learning models and their performance comparison by bivariate and multivariate statistical methods in groundwater potential mapping*, Water resources management, 29(2015), 5217-5236.
27. A. Neumann, N. Laranjeiro, J Bernardino: *An Analysis of Public REST Web Service APIs*, IEEE Transactions on Services Computing, 14(4) (2021), 957-970.
28. B. Pavlyshenko, u*sing stacking approaches for machine learning models*, IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (2018), 255-258
29. Project Jupyter, *Jupyter Documentation*, https://docs.jupyter.org/en/latest/
30. L. Richardson, S. Ruby, *RESTful Web Services*, O'Reilly Media, Cambridge, 2008.
31. W. Richert, L. P. Coelho, *Building Machine Learning Systems with Python*, Packt Publishing, 2013.
32. P. Y. Simard, S. Amershi, D. M. Chickering, A. E. Pelton, S. Ghorashi, C. Meek, G. Ramos, J. Suh, J. Verwey, M. Wang, J. Wernsing, *Machine teaching: A new paradigm for building machine learning systems,* (2017), arXiv preprint arXiv:1707.06742,
33. A. Singh, N. Thakur, A. Sharma: *A review of supervised machine learning algorithms*, 3rd International conference on computing for sustainable global development (INDIACom), (2016), 1310-1315.
34. H. Tanuwidjaja, R. Choi, S. Baek, K. Kim: *Privacy-preserving deep learning on machine learning as a service – A comprehensive survey*, International Journal of Computer Science and Network Security, 21(6) (2020), 137-42.
35. L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, B. Feng: *VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service,* IEEE Transactions on Parallel and Distributed Systems, 30(10) (2021), 2524-2540.
36. A. Zien, N. Krämer, S. Sonnen burg, G. Rätsch: The feature importance ranking measure, Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD Bled, Slovenia, September 7-11, 2009, Proceedings, (2009) 694-709.