Open-source library for data mining techniques

Deliverable No. 1.4 | October 2023

i4Driving
integrated 4D driver modelling under uncertainty

Deliverable No. 1.4 Open-source library for data minig techniques

Project Acronym	Grant Agreement #	Project Title	Deliverable Reference #	Deliverable Title
I4Driving	101076165	Integrated 4D Driver Modelling under Uncertainty	1.4	Open-source library for data mining techniques

AUTHORS

Giulia Vannucci	Institute for Cognitive Sciences and Technologies (Cnr-ISTC)
Andrea Saltelli	Institute for Cognitive Sciences and Technologies (Cnr-ISTC)
Sara Sbaragli	Institute for Cognitive Sciences and Technologies (Cnr-ISTC)
Roberta Siciliano	University of Naples Federico II
Antonio D'Ambrosio	University of Naples Federico II
Michele Staiano	University of Naples Federico II

INTERNAL REVIEWERS

Vincenzo Punzo	University of Naples Federico II

DISSEMINATION LEVEL

X	P	PUBLIC
	C	CONFIDENTIAL

Funded by the European Union

This project has received funding from the European Union's Horizon Europe programme, under grant agreement No 101076165.

Disclaimer

Version History

Revision	Date	Authors	Organisaton	Description
0.1	30.08.2023	Giulia Vannucci, Andrea Saltelli	CNR	First draft
0.2	04.09.2023	Roberta Siciliano, Antonio D'Ambrosio, Michele Staiano	UNINA	Integration
0.3	06.09.2023	Giulia Vannucci	CNR	Revision
0.4	13.10.2023	Ali Nadi Najafabadi	TUD	First review
0.5	19.10.2023	Giulia Vannucci, Sara Sbaragli, Andrea Saltelli, Michele Staiano	CNR, UNINA	Revision
V1.0	24.10.2023	Lucia Schlemmer, Maria Rodrigues, Arnaud Burgess	PAN	Final review

Executive Summary

The major scientific challenge of the **i4Driving Project** is a human driver model that captures the relevant behavioural mechanisms for safety assessment. For the Driving Behavioural Analysis, Data Mining and Machine Learning ought to be considered. The main purpose of this deliverable is to suggest and describe a list of available open-sources programming languages and libraries that are useful to implement data mining techniques in the context of the Driving Behavioural Analysis. The present deliverable contains a brief description of two of the most widely used open-source software in the field of data science, statistics and analytics, as well as Python and R programming languages, alongside with pointer to available resources. The features that permit interfacing this software with one another are also described. After a review of data mining learning techniques, articulated into Supervised learning, Unsupervised learning, Semi-supervised learning and Reinforcement learning, some of the most useful libraries for these tasks are discussed. Moreover, the interoperability of these tools with traffic simulation software is described. The present deliverable is integrated by a GitHub repository[1] that contains proper linkages to all resources described here. The GitHub will be a living entity and will be enriched with cases that are developed during the i4Driving project.

[1] https://github.com/Giuliana87/i4Driving-OpenSourceLibaryDataMining

Contents

List of Tables

1. Introduction

This report describes a selection of data mining techniques, which are also linked on GitHub repository "i4Driving-OpenSourceLibaryDataMining[2].

While the research in data mining techniques - and thus machine learning - is constantly evolving, the software the libraries and other resources discussed in this report are a good selection out of the vast amount of available material. The materials were chosen based on their usability, specifically focusing on widely used and updated programs and libraries. The primary focus was on R and Python, the most popular environments for machine learning. These were selected due to their extensive user communities, ensuring a constant flow of updated packages. These packages are thoroughly reviewed and tested in official and community repositories, providing the confidence that they can be utilised effectively. Additionally, the repository includes codes developed exclusively for the i4Driving project. As new analyses are conducted, the repository expands, incorporating new content over time.

The report is structured as follows:

- In Section 2, two of the most widely used open-source software, Python and R programming languages, are described, focusing on their capability to interface with each other;
- In Section 3 some of the most useful libraries for data mining techniques are outlined;
- In Section 4, the interoperability with traffic simulation software is briefly described; and
- In Section 5, the conclusions are drawn.

2. Open-source Software

Data Mining is the study of collecting, processing, and analysing large volumes of data to discover meaningful patterns, trends, correlations, or insights. It implies analysing data patterns in large batches of data using one or more software. The choice of the software to be used is strongly linked to several issues: better efficiency in organising data, better computational performance, technical choices linked to the machine where the software is to run (i.e. the operating system or the physical memory of the available computer), the need to work online or offline, the accessibility to a diverse range of algorithms to be used on that software. This section describes how to install two of the most popular open-source software available, along with the most widely text editor and IDEs used, and how to install libraries from the relative software.

For the needs of scientific computation in the framework of i4Driving, the main approach to developing and deploying computational tools, especially oriented to the task of statistical data analysis, relies on scripting languages. Python and R are the most common choices to this end. Additionally the two languages can be interfaced with one another, and their environments can be set up to effectively exchange data structures. This can be achieved by means of several Integrated Development Environments (IDE) packages. Thus, Python and R programming languages are presented in this deliverable as synergistic instead of antagonistic tools.

[2] https://github.com/Giuliana87/i4Driving-OpenSourceLibaryDataMining

2.1 Python

Python (https://www.python.org) is a high-level, general-purpose programming language. It is a multi-paradigm programming language, where object-oriented programming and structured programming are fully supported.

2.1.1 Installing Python

The latest Python source distribution is always available from https://www.python.org/downloads/. The latest development sources can be obtained at https://github.com/python/cpython/. The standard documentation for the current stable version of Python is available at https://docs.python.org/3/.

2.1.2 Text editors and integrated development environments (IDEs) available

Using an Integrated Development Environment (IDE) or text editor is not an absolute requirement, but it is often recommended. IDEs are coding tools that make writing, debugging, and testing the code easier. Many provide helpful features like code completion, syntax highlighting, debugging tools, variable explorers, visualisation tools and many other features. Here, are some of the most widely used IDEs and Text editor for Python are described, whose versions and releases are summarised in Table 1.

- **Jupyter Notebook / JupyterLab (https://jupyter.org):** Jupyter Notebook and its updated version JupyterLab are interactive environments (both released under a modified BSD license) widely used for data science and scientific computing not only in Python, since Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R. They allow you to combine code, visualisations, and narrative text in a single document.
- **Spyder (https://www.spyder-ide.org):** Spyder is an IDE (released under the MIT License) specifically designed for scientific computing with Python. It offers features like an integrated IPython console, variable explorer, and support for data visualisation libraries. It integrates with a number of prominent packages in the scientific Python stack.
- **Atom (https://github.com/atom/atom):** Atom is a customisable and free text editor (released under the MIT License) developed by GitHub with support for plug-ins written in JavaScript, and embedded Git Control. It can be extended using packages to support Python development.

It is worth mentioning two more very common IDEs, even if they are not fully open source and free projects, since it is possible to exploit some FLOSS versions of them:

- **PyCharm (https://www.jetbrains.com/pycharm/):** PyCharm is a powerful commercial IDE used for programming in Python. It offers code analysis, debugging, integrated testing, and support for web development frameworks, scientific libraries, and more. An academic free license is available to qualified users. Moreover, the Community edition is Free and built on open-source code released under Apache License 2.0.
- **Visual Studio Code (VS Code, https://code.visualstudio.com):** VS Code is a lightweight and highly customisable code editor by Microsoft. It has a strong Python extension ecosystem, providing features like debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality. Microsoft's VS Code source code is open source (MIT-licensed), but the product available for download (Visual Studio Code) is

licensed under a not-FLOSS license and contains telemetry/tracking. VSCodium (https://vscodium.com/) is a community-driven, freely-licensed binary distribution of Microsoft's editor VS Code (released under the MIT License) and obtaining its binaries guarantees that telemetry is disabled.

Table 1 Python IDEs summary

IDE	Url	Last release to date	Open-Source	Free version available
Jupyter Notebook/ JupyterLab	https://jupyter.org	7.0.5	Yes	Yes
Spyder	https://www.spyder-ide.org	5.4.5	Yes	Yes
Atom	https://github.com/atom/atom	1.61.0-beta0	Yes	Yes
PyCharm	https://www.jetbrains.com/pycharm/	2023.2.2	Yes (CE is released under Apache License 2.0)	Yes
Visual Studio Code	https://code.visualstudio.com	1.83	Yes (released under the MIT License)	Yes

2.1.3 Installing Python packages

The installation of packages in Python can be done by opening the Command Prompt or terminal and using the *pip install* command followed by the package name. The detailed instructions can be found at https://packaging.python.org/installing/

There are also package and environment management systems alternatives to pip (or its last version pip3) mainly based on conda (released under BSD 3-Clause License) or derived from it. The most common platform for scientific computation is Anaconda (https://www.anaconda.com/download) and it is highly recommended to resort to it for smoothly setting up a Data Science workstation.

2.2 R Software

R (https://www.r-project.org) is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. It provides a wide variety of statistical techniques (i.e. linear and nonlinear modelling, statistical tests, time-series analysis, classification, clustering, and machine learning algorithms) and graphical techniques, and is highly extensible. R has gained immense popularity in the fields of statistics, data analysis, and data visualisation due to its rich set of built-in functions and packages tailored for these tasks. R provides an Open Source route to participation in that activity.

2.2.1 Installing R

The latest R source distribution is available from CRAN Mirrors at https://cran.r-project.org/mirrors.html, where one must choose the closest location and download the desired version of R.

2.2.2 Text editors and IDEs available

As mentioned for Python, the usage of a text editor or an IDE for R is suitable. Some of the most widely used IDEs and Text editor for R are listed below, for which versions and release are summarised in Table 2.

- **RStudio (https://posit.co/download/rstudio-desktop/):** RStudio is a widely used and feature-rich IDE (released under the AGPL v3 license) designed specifically for R programming. It provides an intuitive interface for writing R code, managing packages, and creating visualisations. It is worth mentioning that by free registration at posit.cloud it is possible to use at no cost R by a cloud instance of RStudio with a limited set of resources – that can be enhanced on subscription at need. Posit (formerly RStudio) also offers a commercial desktop license for RStudio, called RStudio Desktop Pro, and a commercial license for Posit Workbench (previously RStudio Workbench/RStudio Server Pro).
- **Jupyter Notebook / JupyterLab:** as mentioned for Python, Jupyter Notebook and JupyterLab (both released under a modified BSD license) support R kernels, enabling you to create interactive documents with R code and visualisations.

As in the case of Python, some IDEs exist that are well suited and can be used for free even if not released as open-source software:

- **Visual Studio Code (VS Code):** as mentioned for Python, VS Code, as well as VSCodium, has extensions that allow you to work with R. It provides features like syntax highlighting, debugging, and integrated terminal support for R.
- **Tinn-R (https://tinn-r.org/en/):** Tinn-R is an editor/word processor ASCII/UNICODE generic for the Windows operating system, very well integrated into the R, with R specific features including syntax highlighting, code completion, and integration with R's help system. The Tinn-R program remains free for use in the education sector at any level.

Table 2 R IDEs summary

IDE	Url	Last release to date	Open-Source	Free version available
Rstudio	https://posit.co/download/rstudio-desktop/	2023.09.0+463. pro11	Yes	Yes
Jupyter Notebook/ JupyterLab	https://jupyter.org	7.0.5	Yes	Yes
Visual Studio Code	https://code.visualstudio.com	1.83	Yes (released under the	Yes

			MIT License)	
Tinn-R	https://tinn-r.org/en/	8.2.2.1	No	Yes (educational sector)

2.2.3 Installing R packages

In R, the installation of packages is straightforward and can be done using the *install.packages()* function. More detailed instructions can be found at https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages.

2.3 Interfacing Python and R

There are two options to exploit the best from both languages: calling R code from a Python script or invoking Python code from within the R environment.

When Python is preferred as the main language, e.g. since it has good capabilities for data manipulation and integration with existing code for side tasks, it is useful to perform data mining analyses by running R code from within Python processes (via the `rpy2` Python library, GNU General Public License Version 2, https://rpy2.github.io/doc/latest/html/introduction.html). This offers the following advantages:

1. Access to a vast number of statistical algorithms and packages: the R community has developed a vast number of packages for statistical analysis and data science. By using `rpy2`, Python users can tap into this extensive library and leverage the well-tested algorithms and methodologies implemented in R.
2. Seamless integration with Python: `rpy2` provides a seamless interface between Python and R, allowing users to easily move between the two languages. This enables 'Pythonistas' to take advantage of R's statistical capabilities while still being able to utilise the wide range of Python libraries for tasks such as data manipulation, visualisation, and web development.
3. Compatibility with existing R code and workflows: For users who already have code or workflows written in R, `rpy2` allows them to reuse and integrate their R code into Python processes. This eliminates the need to rewrite or replicate existing R code, saving time and effort.
4. Flexibility and versatility: By combining the strengths of both Python and R, users can leverage the best features of each language for their analysis or data science tasks. Python provides a powerful and expressive programming language, while R offers a rich ecosystem of statistical functions and packages. This combination of languages gives users the flexibility to choose the best tool for each part of their workflow.
5. Support and community: The R community has a long history and a large user base, which means that there is a wealth of resources and support available by R coders and users for many data mining tasks. By using `rpy2`, Python users can tap into this community and benefit from the knowledge and expertise of R users.

Overall, `rpy2` provides a powerful and convenient way for Python users to harness the capabilities of R's statistical packages and algorithms, while still enjoying the flexibility and functionality of the Python ecosystem.

On the other hand, if R is the customary environment for the analysis, the R package `reticulate` offers the possibility of seamlessly integrating Python code chunks in an R notebook or markdown file managed by a suitable IDE (like RStudio).

3. Libraries for data mining techniques

A crucial step of every type of analysis is the efficient manipulation of data and a good visualisation of the results. Here some useful libraries for these steps are outlined.

In Python, for data manipulation `Pandas, NumPy and scipy` are some very important libraries for data manipulation and data analysis. They provide data structures and functions needed to effectively manipulate and analyse structured data. For data visualisation, `ggplot` (`ggpy`) is the counterpart of R library `ggplot2`, and it provides a high-level interface for creating complex visualisations. `Plotly` is an interactive data visualisation library that allows to create interactive web-based charts and dashboards and it supports a wide range of chart types, including scatter plots, bar charts, line charts, 3D plots, and more. `Matplotlib` is one of the most widely used 2D plotting libraries for Python. It provides a simple and flexible way to create static, publication-quality plots and figures and it offers a wide range of customisation options for creating line plots, bar plots, histograms, and more. `Seaborn` is built on top of `Matplotlib` and provides a high-level interface for creating informative and attractive statistical visualisations. It simplifies the process of creating complex statistical plots, such as heatmaps, violin plots, and pair plots, by providing easy-to-use functions. Finally, `Bokeh` is a Python library for creating interactive, web-ready visualisations. It is particularly well-suited for building interactive dashboards, web applications, and real-time visualisations.

In R, packages `dplyr` and `data.table` are excellent for data manipulation, the latter is strongly efficient in handling large datasets. For data visualisation, `ggplot2` it is the most widely used package for plotting and visualising data. It provides helpful commands to create complex plots from data in a data frame and a more programmatic interface for specifying what variables to plot, how they are displayed, and general visual properties.

There are several important open-source libraries for data mining techniques for Python and R. In next sections the methods most used for the three categories of data mining techniques are described: supervised learning, unsupervised learning, and reinforcement learning. Furthermore, it could be also mentioned that semi-supervised learning is a further paradigm somewhere between unsupervised and supervised learning. In fact, most semi-supervised learning strategies are based on extending either unsupervised or supervised learning to include additional information typical of the other learning paradigm. After a brief introduction of these macro-categories, we listed some of the most widely used libraries within these techniques.

For a comprehensive description of the statistical models and machine learning methods mentioned in this section, the reader is referred to the reading of Anderson et al. (1958), Agresti (2015), McCullagh, P. (2019), Hastie, Tibshirani et al. (2021, 2023), Saltelli et al. (2004, 2008), Sutton and Barto (2018), Zhu and Goldberg (2009).

3.1 Supervised learning

In supervised learning, the algorithm is provided with input data (the set of explanatory variables, also named features in data science parlance) and corresponding target outputs (the set of response variables), allowing it to learn how to map inputs to outputs. When the target is numerical, it becomes a regression problem while when the target is categorical with a set of response classes, it is a classification problem. The goals of the models of supervised learning are to identify the best model

capable of representing how the target variable is related to the explanatory variables and to make accurate predictions. Among the parametric supervised learning models, it is worthy to recall linear regression, logistic regression and discriminant analysis. With high dimensional data, recall LASSO, RIDGE, ELASTIC-NET Regression, support vector regression with linear kernel, or dimension reduction methods such as Principal Component Regression or Partial Least Squares Regression. Among nonparametric models we recall Regression Splines, Smoothing Splines, and Local Regression. Among algorithms based on decision trees the most popular are Classification and Regression Trees, Boosting, Random Forests and Bayesian Additive Regression Trees (BART). Finally, we recall Deep Learning with Neural Networks. Additionally, we suggest some library for Sensitivity Analysis.

3.1.1 Python libraries for supervised learning

- `statsmodels`: it is a library for statistical modelling and hypothesis testing, which can be useful for supervised learning when interpretability and inference are important. It provides a complement to `scipy` for statistical computations including descriptive statistics and estimation and inference for statistical models.
- `scikit-learn`: this is one of the most popular libraries for machine learning in Python. It covers a wide range of supervised learning algorithms, including linear and logistic regression, penalised regression, PCA, decision trees, random forests, support vector machines, and more. It also provides various tools for model fitting, data pre-processing, model selection, model evaluation, and many other utilities.
- `XGBoost`: short for "Extreme Gradient Boosting" it is a powerful machine learning algorithm for boosting models known for its speed and performance.
- `LightGBM`: it is another gradient-boosting framework that is designed to be efficient and scalable. It's particularly useful for large datasets and is known for its speed and accuracy.
- `CatBoost`: a gradient boosting library that is capable of handling categorical features effectively without extensive preprocessing.
- `TensorFlow`: it is an open-source deep learning library that provides a flexible framework for building and training various neural network models. It's particularly well-suited for tasks involving neural networks and deep learning.
- `PyTorch`: it is another popular deep learning library. It's known for its dynamic computation graph and is widely used for research in machine learning and artificial intelligence.
- `SAlib`: it is a library containing implementations of commonly used sensitivity analysis methods, including Sobol, Morris, and FAST methods, useful in systems modelling to calculate the effects of model inputs or exogenous factors on outputs of interest.

3.1.2 R libraries for supervised learning

- `stats` package is one of the core packages in R and comes pre-installed with R itself. It provides a wide range of statistical functions, including basic statistical tests, linear and nonlinear modelling, regression, and much more.
- `kknn`: it implements k-nearest neighbours (k-NN) algorithms, which is particularly useful for classification tasks based on similarity metrics.
- `glmnet`: it is particularly popular for fitting Lasso and Ridge regression models, and it offers efficient implementations of these regularisation techniques.
- `rpart` and `tree`: these two packages are the most widely used for fitting classification and regression trees.

- `party`: it implements recursive partitioning methods, including the C4.5 algorithm and conditional inference trees, which can be used for decision tree-based models.
- `caret`: it is a versatile and powerful tool for training, tuning, and evaluating machine learning models. It provides a unified interface for working with a wide range of algorithms, preprocessing techniques, and model evaluation methods. It supports parallel processing, which can significantly speed up the training and tuning process.
- `randomForest`: it implements the random forest algorithm, an ensemble learning method that's effective for classification and regression tasks. It's known for its robustness and ability to handle high-dimensional data.
- `ranger`: it is a fast and efficient implementation of random forest algorithm.
- `xgboost`: it offers an implementation of the extreme gradient boosting algorithm, which is highly efficient and performs well on a variety of machine learning tasks.
- `gbm`: The gbm (Generalised Boosted Regression Models) package provides an implementation of gradient boosting, a machine learning technique for regression and classification tasks.
- `nnet`: it is used for training and evaluating neural networks. While more specialised deep learning frameworks are available (i.e. TensorFlow, Keras of Python), nnet offers a simpler approach for basic neural network tasks.
- `e1071`: it provides a collection of functions for support vector machines (SVMs), as well as other machine learning algorithms for classification and regression.
- `sensobol`: it is a package that provides several functions to conduct variance-based uncertainty and sensitivity analysis, from the estimation of sensitivity indices to the visual representation of the results.

3.2 Unsupervised learning

Algorithms of unsupervised learning have the objective to identify patterns and structures in data and the relationship among the features without a target variable to be predicted. Common tasks in unsupervised learning include: Association rules mining, clustering – where data points are grouped based on similarities, and dimensionality reduction – where the algorithm reduces the complexity of the data while preserving important features. Among dimensionality reduction methods, the most widely used methods are the Principal Component Analysis, Correspondence Analysis and Multiple Correspondence Analysis, Non-linear factorial methods, and Multidimensional Scaling Methods.

3.2.1 Python libraries for unsupervised learning

- `scikit-learn`: as already mentioned above, this is one of the most popular libraries for machine learning in Python. Among clustering algorithms, if offers from `sklearn.cluster`: `KMeans`, `AgglomerativeClustering` for hierarchical clustering, and `DBSCAN` for density-based clustering. Among dimensionality reduction techniques, it offers `PCA` from `sklearn.decomposition` for Principal Component Analysis (PCA), `t-SNE` from `sklearn.manifold`, t-Distributed Stochastic Neighbor Embedding that is useful for visualising high-dimensional data by reducing it to a lower-dimensional space. If the work involves multi-label classification tasks, `Scikit-multilearn` provides options like MLkNN, which combines multi-label k-nearest neighbors and dimensionality reduction.
- `Apache Spark`: it is a fast and general-purpose cluster computing system that also includes libraries for machine learning (`MLlib`) and graph processing (`GraphX`). It's well-suited for big

data processing, Exploratory Data Analysis (EDA), feature extraction and Machine Learning models.

- `PyCaret`: it is a low-code machine learning library that automates machine learning workflows. It is an end-to-end machine learning and model management tool that exponentially speeds up the experiment cycle and makes you more productive. Compared with the other open-source machine learning libraries, PyCaret is an alternate low-code library that can be used to replace hundreds of lines of code with a few lines only, which makes experiments exponentially fast and efficient. This repository works as a wrapper around several machine learning libraries and frameworks, such as `scikit-learn` for unsupervised and supervised learning, `XGBoost`, `LightGBM` and `CatBoost` for boosting algorithms, `Optuna` for an automatic hyperparameter optimisation software framework, and others. While it's a high-level machine learning library, PyCaret also provides support for various clustering algorithms, making it easier to experiment with different techniques.
- `Yellowbrick`: it is a suite of visual analysis and diagnostic tools designed to facilitate machine learning with `scikit-learn`. It focuses on visualisation tools to understand the effect of dimensionality reduction and tools to help visualise, i.e., clustering results.
- `Prince`: this library specialises in factor analysis and dimensionality reduction, offering implementations of various techniques, including PCA and correspondence analysis. It provides efficient implementations, using a `scikit-learn` API.

3.2.2 R libraries for unsupervised learning

- `arules`: it is focused on association rule mining, a technique that identifies interesting relationships or associations between items in a dataset. This package provides tools needed to create and manipulate input data sets for the mining algorithms and for analysing the resulting item sets and generating association rules.
- `cluster`: it provides various clustering algorithms, including hierarchical clustering, K-Means clustering, and more advanced techniques like PAM (Partitioning Around Medoids) and CLARA (Clustering Large Applications).
- `dbscan`: it provides an implementation of the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, which is effective for identifying clusters of varying shapes and sizes.
- `mclust`: it is focused on model-based clustering, and it offers tools for fitting Gaussian Mixture Models to the data, allowing the identification of clusters with different shapes and orientations.
- `clustermesh`: it offers tools for hierarchical clustering using dendrograms, including the possibility of creating clustered heatmaps and visualising the results.
- `dendextend`: it is a versatile package for extending the functionality of dendrograms, providing additional options for manipulating and visualising hierarchical clustering results.
- `NbClust`: it is designed to determine the number of clusters in a dataset, and it offers a comprehensive suite of indices for cluster validation
- `factoextra`: it is a helpful tool for extracting and visualising information from various multivariate analysis methods. This package provides functions to visualise the results of clustering algorithms, including hierarchical clustering and K-Means, and includes plotting cluster assignments, dendrograms, and more. It offers tools to visualise the outcomes of dimensionality reduction techniques like PCA (Principal Component Analysis) and t-SNE (t-

Distributed Stochastic Neighbor Embedding), to visualise the factor analysis results, including factor loadings and factor correlations, and it provides heatmap functions that can display clustering results and PCA outcomes.
- `FactoMineR`: this comprehensive package offers a range of multivariate analysis techniques, including Principal Component Analysis (PCA), Multiple Correspondence Analysis (MCA), and Factor Analysis of Mixed Data (FAMD).

3.3 Semi-supervised learning

Semi-supervised learning approach falls between supervised learning (where training data is labelled) and unsupervised learning (where there are no labels). In semi-supervised learning there is a small amount of data where the target labels are provided. This labelled data is used to train the model initially, whereas most of your data remains unlabelled, meaning there are no corresponding target labels for these instances. The algorithm aims to leverage both the labelled and unlabelled data to build a more accurate model. Therefore, it considers information contained in the unlabelled data, which can improve the model's generalisation and predictive performance. Semi-supervised learning is particularly useful when obtaining labelled data is expensive or time-consuming. By utilising unlabelled data, it can help improve the performance of models, especially in cases where labelled data is scarce. Techniques in semi-supervised learning often involve methods like self-training, co-training, label propagation, and consistency-based learning to make predictions on unlabelled instances.

3.3.1 Python libraries for semi-supervised learning

- `Scikit-learn`: this package can be used also for semi-supervised tasks with the semi_supervised module, and it also provides Label Propagation and Label Spreading algorithms, which are popular semi-supervised techniques for propagating labels in graph-structured data. Moreover, the library is extended by `Scikit-multilearn` to support multi-label classification tasks, which can be useful in some semi-supervised scenarios.
- `TensorFlow`: it provides support for building custom semi-supervised models using its high-level API, Keras.
- `PyTorch`: it allows to create custom semi-supervised learning models using its flexible framework.
- `D3M` (Data Driven Discovery of Models): it is a framework for building and evaluating machine learning pipelines, and it includes support for semi-supervised learning tasks.
- `Deep Graph Library` (DGL): this library is specifically designed for deep learning on graphs. It includes support for semi-supervised learning on graph data.
- `ModAL`: it is specialised in active learning, which can be used for semi-supervised learning by iteratively selecting the most informative unlabelled data points for labelling.
- `Snorkel`: it is a system for building and managing training data and creating labelling functions to assist in semi-supervised learning.

3.3.2 R libraries for semi-supervised learning

- `caret`: it supports various semi-supervised learning techniques and can be used with other packages for specific algorithms.
- `RSSL` (R for Semi-Supervised Learning): it is a package designed specifically for semi-supervised learning in R. It provides functions for different semi-supervised algorithms.

- `SSL` (Semi-Supervised Learning): it is another package focused on semi-supervised learning. It includes a variety of algorithms and functions for semi-supervised classification.
- `ConSSEL` (Consistency-based Semi-Supervised Learning): it is a collection of consistency-based semi-supervised learning algorithms and tools.
- `tmle`: it includes functions for performing semi-supervised learning in a targeted and efficient manner.
- `bootSSL`: it combines bootstrapping with semi-supervised learning techniques to provide more robust results.
- `RSSLTools`: it offers tools and utilities for various semi-supervised learning algorithms.
- `coresvm`: it implements semi-supervised support vector machines.
- `SemiParSampleSel`: it is specifically designed for semi-parametric model selection in a semi-supervised learning context.
- `RSSLadj`: it provides functions for performing semi-supervised learning with graph-based algorithms.

3.4 Reinforcement learning

Reinforcement Learning (RL) is a subfield of machine learning that focuses on training agents to make a sequence of decisions in an environment to maximise a cumulative reward. It is inspired by behavioural psychology and aims to simulate the learning process observed in humans and animals, where they learn by interacting with their surroundings and receiving feedback. In RL, an agent interacts with an environment and learns how to take actions that lead to the most favourable target. The agent's learning process involves exploring various actions and their consequences, gradually refining its policy to make better decisions. Reinforcement Learning has found applications in a wide range of fields, such as robotics, autonomous systems, game playing, recommendation systems, finance, and healthcare, where sequential decision-making is crucial. It's a powerful tool for training agents to adapt and make decisions in complex and dynamic environments.

3.4.1 Python libraries for reinforcement learning

- `OpenAI Gym`: it is a popular and user-friendly library that provides a wide range of environments for testing and developing RL algorithms.
- `Stable Baselines`: Built on top of OpenAI Gym, it is a set of high-quality implementations of common reinforcement learning algorithms. It simplifies the process of experimenting with and applying RL algorithms.
- `TensorFlow`: already mentioned for supervised learning, it has support for deep reinforcement learning. It provides flexibility for building custom neural network architectures and has libraries like TensorFlow Agents (TF-Agents) for RL-specific tasks.
- `PyTorch`: it is another popular deep learning library that is widely used in RL research. It offers dynamic computation graphs, which are advantageous for implementing various RL algorithms.
- `RLlib` (Reinforcement Learning Library): it provides a high-level, scalable framework for reinforcement learning. It offers a range of RL algorithms and is particularly suitable for distributed and parallel training.
- `Dopamine`: it is a research framework for developing and testing reinforcement learning algorithms and it includes a variety of pre-implemented algorithms and experimental setups.

- `Keras-RL`: it simplifies the development and testing of RL algorithms by providing implementations of common algorithms and helpful utilities.
- `Ray`: it is a general-purpose distributed computing framework that includes RLlib, as mentioned earlier. It's particularly useful for large-scale training and distributed reinforcement learning.

3.4.2 R libraries for reinforcement learning

- `RLearn`: it provides implementations of common RL algorithms, including Q-learning and SARSA.
- `Keras` and `TensorFlow` in R: on can use the R interface for Keras and TensorFlow to implement reinforcement learning algorithms.
- `ReinforcementLearning`: it is an interface to the popular OpenAI Gym environment. It allows to work with Gym environments and perform reinforcement learning experiments.
- `RLAgents`: it provides a flexible and extensible framework for reinforcement learning. It supports a range of RL algorithms and includes tools for creating custom environments.
- `CausalInferenceRL`: it combines causal inference methods with reinforcement learning. It's particularly useful for applications where causal reasoning is essential, i.e. healthcare.
- `MLR`: while not specifically designed for reinforcement learning, it is a versatile toolkit for various machine learning tasks, including reinforcement learning.
- `RLtools`: it provides a set of tools and utilities for reinforcement learning, including different types of agents and environments.

4. Means for interoperability with simulation software

Software for traffic simulation used during the i4Driving project could be easily interfaced to the code deployed for data mining through the open-source solutions so far described.

Althoff et. al (2017) developed CommonRoad (composable benchmarks for motion planning on roads), that is a benchmarking suite for motion planning on roads and autonomous driving algorithms. The framework provides realistic traffic scenarios and various tools for researchers and developers working on autonomous vehicles. It is developed as an open-source project in Python, allowing researchers to use it as a benchmark for testing and comparing their autonomous driving algorithms. By establishing common benchmarks, it promotes collaboration in the autonomous driving community, which is crucial for advancing the technology and ensuring its safety and reliability. As part of the CommonRoad framework, software tools are available that monitor traffic rule compliance and evaluate criticality metrics of CommonRoad traffic scenarios, making the framework a valuable tool for data mining projects.

Furthermore, from the very simple interfacing of data mining via an interoperable file format (e.g. as straightforward as using csv files for data exchange) to the tighter integration made possible by exploiting statistical procedures' call from within the simulation software.

Interfacing to Python software is really a straightforward task (see for a wide set of languages the official documentation: https://wiki.python.org/moin/IntegratingPythonWithOtherLanguages).

About R code, it can directly be invoked from within Python processes by running an R script as a subprocess (https://docs.python.org/3/library/subprocess.html#module-subprocess) and collecting its output or by exploiting suitable libraries, as already discussed (see `rpy2` in section 2.3).

For software written in Java, Renjin (https://www.renjin.org/) is an open-source library (released under GNU GENERAL PUBLIC LICENSE Ver. 2) that easily adds to Java projects the capability to spawn an R process allowing zero-overhead data sharing between R and Java.

5. Conclusions

This deliverable comprehensively outlines open-source software and libraries essential for data mining techniques. Python and R emerge as the foremost contenders in the realms of data science, statistics, and analytics. Their selection in this document is predicated on their widespread adoption and utility.

Both languages exhibit distinctive strengths and limitations, contingent upon specific use cases and individual preferences. Notably, Python and R are lauded for their versatility, enabling tasks encompassing machine learning and deep learning. Their interoperability, both within each other and with a multitude of programming languages, underscores their adaptability. Additionally, the robust backing from expansive online communities ensures a wealth of resources, including comprehensive documentation, tutorials, and support.

Moreover, these languages play a pivotal role in facilitating reproducible research, a cornerstone in scientific contexts. The diverse array of libraries and frameworks available within both Python and R is noteworthy. This report delves into the most extensively used library for data mining, recognising the vast selection of packages for similar tasks. Importantly, the field is dynamic, witnessing the development and release of new packages practically every day. Hence, it remains imperative for users to stay current with these advancements to harness the latest tools for their data mining pursuits.

6. References

Althoff, M., Koschi, M., & Manzinger, S. (2017, June). *CommonRoad: Composable benchmarks for motion planning on roads. In 2017 IEEE Intelligent Vehicles Symposium (IV)* (pp. 719-726). IEEE.

Anderson, T. W., Anderson, T. W., Anderson, T. W., & Anderson, T. W. (1958). *An introduction to multivariate statistical analysis* (Vol. 2, pp. 1468-1482). New York: Wiley.

Agresti, A. (2015). *Foundations of linear and generalized linear models*. John Wiley & Sons.

McCullagh, P. (2019). Generalized linear models. Routledge.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Second Ed., Springer. New York.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning*. Second Ed., Springer, New York, NY.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning - with Applications in Python.* Springer Cham.

Saltelli, A., et al. (2004). *Sensitivity analysis in practice: a guide to assessing scientific models*. Vol. 1. New York: Wiley.

Saltelli, A., et al. (2008). *Global sensitivity analysis: the primer*. John Wiley & Sons.

Zhu, X., Goldberg, A.B. (2009). Overview of Semi-Supervised Learning. In: Introduction to Semi-Supervised Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer, Cham.