

Tailored IoT & BigData Sandboxes and Testbeds for Smart,
Autonomous and Personalized Services in the European
Finance and Insurance Services Ecosystem



D5.7 – Library of ML/DL Algorithms - I

Revision Number	3.0
Task Reference	T5.4
Lead Beneficiary	FBK
Responsible	Bruno Lepri
Partners	AGRO, ATOS, ATOS-IT, BOUN, CP, CTAG, FBK, FTS, GFT, GLA, JRC, JSI, NUIG, ORT, PRIVE, RB, UBI, UPRC, WEA
Deliverable Type	Report (R)
Dissemination Level	Public (PU)
Due Date	2020-11-30
Delivered Date	2020-11-30
Internal Reviewers	GLA, ENG

HORIZON 2020 - ICT-11-2018



This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 856632

Quality Assurance	INNOV
Acceptance	WP Leader Accepted and/or Coordinator Accepted
EC Project Officer	Pierre-Paul Sondag
Programme	HORIZON 2020 - ICT-11-2018
	This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 856632

Contributing Partners

Partner Acronym	Role ¹	Name Surname ²
FBK	Lead Beneficiary	
AGRO	Contributor	
ATOS	Contributor	
BOUN	Contributor	
CP	Contributor	
ENG	Contributor	
FBK	Contributor	
GFT	Contributor	
GLA	Contributor	
INNOV	Contributor	
ISPRINT	Contributor	
JRC	Contributor	
NUIG	Contributor	
PRIVE	Contributor	
RB	Contributor	
UBI	Contributor	
UPCR	Contributor	

¹ Lead Beneficiary, Contributor, Internal Reviewer, Quality Assurance

² Can be left void

D5.7 - Library of ML/DL Algorithms - I

WEA	Contributor	
ABI	Contributor	
ENG	Internal Reviewer	
GLA	Internal Reviewer	

Revision History

Version	Date	Partner(s)	Description
0.1	2020-10-11	FBK	ToC Version
0.2	2020-10-12	FBK, JRC, INNOV	Contributions on Sections 1, 4.2
0.3	2020-10-13	ATOS	Contributions on Sections 3.1, 4.10
0.4	2020-11-02	FBK, BOUN	Contributions on Sections 3.3, 4.8
0.5	2020-11-04	ISPRINT	Contribution on Section 4.11
0.6	2020-11-05	FBK, GFT, WEA	Contributions on Sections 2, 3.2, 4.12
0.61	2020-11-06	FBK, GFT	Update contribution on Section 3.2
0.7	2020-11-10	UPCR	Contribution on Section 4.5
0.8	2020-11-12	CP, ENG	Contributions on Sections 4.6, 4.9
0.9	2020-11-13	GFT	Contribution on Section 4.1
0.10	2020-11-16	AGRO, PRIVE, FBK	Contributions on Sections 4.13, 4.4, 5
0.101	2020-11-17	FBK, AGRO	Update Contribution on Section 4.13
0.11	2020-11-20	JSI	Update Contribution on Section 4.7
0.12	2020-11-21	ABI	Contribution on Section 4.14
1.0	2020-11-22	FBK	First Version for Internal Review
1.1	2020-11-23	GLA	Internal Review
1.2	2020-11-26	ENG	Internal Review
2.0	2020-11-27	FBK	Version for Quality Assurance
3.0	2020-11-30	FBK	Version for Submission

Executive Summary

Deliverable D5.7 is the first of a series of three deliverables planned within the scope of Task 5.4 of the INFINITECH project. One of the main objectives of the task is to firstly identify state-of-the-art and innovative machine learning solutions to practical challenges within the sphere of the FinTech and InsuranceTech revolution. As a second objective, the task aims at implementing and integrating, as ML services, ready-to-use solutions of potential interest for a wide spectrum of financial institutions and companies.

Specifically, deliverable D5.7 aims at landscaping the more convenient machine learning tools that can be leveraged to address the variety of different challenges FinTech and InsuranceTech want to overcome. In particular, deliverable 5.7 focuses on both general purpose solutions as well as pilot-specific ones. The first ones are included as top-down tools, originated and developed to respond to broad questions and needs. They include tools from two strands, i.e. popular state of the art ML tools used in the INFINITECH project, as well as ML frameworks developed by the project's partners. The second ones are described as bottom-up solutions, being raised from real particular needs of the pilots of the project and their business and technological partners.

The structure of the deliverable is designed to enable the reader to easily navigate and retrieve the information of interest. Among the three main sections (Sections 2, 3, and 4), Section 2 is dedicated to the presentation of state-of-the-art machine learning libraries. Particular attention is devoted to ML/DL Python libraries, being Python the most widely adopted programming language in the ML/DL community as well as in the INFINITECH pilots. In this section, ML tools are presented with the intent of presenting the most commonly used libraries, and to inform the pilots and the potential future users of INFINITECH solutions on the state of the current technological development.

Complementarily, Sections 3 and 4 are uniquely devoted to present the framework in which these tools (among others) are used. Specifically, Section 3 presents five different top-down solutions developed and proposed by five of the project partners (ATOS, Engineering, GFT-Italy, FBK, and the University of Glasgow (GLA)). Section 4 presents the ML related tasks established and currently under development by most of the pilots and the guiding machine learning frameworks they plan to test and develop. Altogether, the concept behind this deliverable and, more precisely, behind these two sections is to bridge the generic and pilot-specific solutions. The vision developed here recognizes to this process the particularly relevant role of stabilizing the adopted methodologies, from both the top-down and bottom-up contributions, and of building a solid and shared ground of best practices.

This deliverable can thus provide a ready-to-use roadmap for the pilots, companies and research institutions involved in the project to better connect and synchronize in the first stage of the INFINITECH three-year program. The developers of top-down solutions can benefit from the participation and partnership with pilots by testing and expanding the applicability and deployability of their general purposes. Conversely, the pilots can profit from the wider and more informed perspective described and catalogued in this deliverable to more effectively develop and deploy their task-specific solutions.

Table of Contents

List of Figures	9
List of Tables	9
List of Abbreviations	10
1 Introduction	13
1.1 Objectives of the Deliverable	13
1.2 Insights from other Tasks and Deliverables	14
1.3 Structure	15
2 State of the art of existing machine learning libraries	16
2.1 Scikit-learn	17
2.2 TensorFlow and Keras	17
2.3 PyTorch	18
2.4 Other programming tools and machine learning libraries	19
3 INFINITECH general-purpose Machine Learning solutions	20
3.1 EASIER.AI: An Artificial Intelligence Orchestration Platform	20
3.2 ALIDA: microservice-based platform for composition, deployment, optimisation, execution, and monitoring of Big Data analytics workflows	21
3.3 Rulex: An Explainable AI Reasoner	23
3.4 AI-driven psychometric engine for financial behaviours	25
3.5 BetaRecSys - Deep Learning for Recommendation	27
4 INFINITECH pilots' Machine Learning tasks	30
4.1 Pilot 1 - Invoices processing platform for a more sustainable banking industry	31
4.2 Pilot 2 - Real-time risk assessment in investment banking	32
4.4 Pilot 4 - Personalized portfolio management - mechanisms for an AI-based portfolio construction	34
4.5 Pilot 5B - Business Financial Management (BFM) tools delivering smart business advice	36
4.6 Pilot 6 - Personalized closed-loop investment portfolio management for retail customers	38
4.7 Pilot 8 - Platform for Anti Money Laundering (AML) supervision	40
4.8 Pilot 9 - Analyzing Blockchain transaction graphs for fraudulent activities	43
4.9 Pilot 10 - Real-time cyber-security analytics on financial transactions' Big Data	44
4.10 Pilot 11 - Personalized insurance products based on IoT connected vehicles	45
4.11 Pilot 12 - Real-world data for novel health-insurance products	46
4.12 Pilot 13 - Alternative/automated insurance risk selection - product recommendation for Small and Medium Enterprises (SMEs)	47
4.13 Pilot 14 - Big Data and IoT for the agricultural insurance industry	48
4.14 Pilot 15 - Open Inter-Banking	49
5 Conclusions	51

Appendix A: Literature **52**

List of Figures

Figure 1 - EASIER.AI Hybrid Platform	20
Figure 2 - ALIDA platform flows	22
Figure 3 - Example of workflow in the Rulex platform	24
Figure 4 - BetaRecSys Architecture	29
Figure 5 - Schematic of the pilots' workflow	31
Figure 6 - Schematics of portfolio selection based on fitness score	36
Figure 7 - StreamStory pipeline	42

List of Tables

Table 1 - Mainstream state-of-the-art ML/DL libraries	17
Table 2 - Set of behavioural indicators extracted using the Psychometric Engine	27
Table 3 - Example of multi-factor aggregation for fitness computation of customer portfolios	35
Table 4 - Technologies, tools, and libraries to be used in pilot 5B	37

List of Abbreviations

ADWIN	Adaptive Windowing
AFAC	Allocation Factor
AI	Artificial Intelligence
AML	Anti-Money Laundering
ANOVA	ANalysis Of VAriance
API	Application Programming Interface
AUC	Area Under Curve
BDA	Big Data Analytics
BFM	Business Financial Management
BLEU	Bilingual Evaluation Understudy
BSD	Berkeley Software Development
CFT	Combating the Financing of Terrorism
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRM	Customer Relationship Management
CRISP-DM	Cross Industry Standard Process for Data Mining
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
DNNs	Dense Neural Networks
DPI	Dots Per Inch
EO	Earth Observation
ES	Expected Shortfall
FAIR	Facebook Artificial Intelligence Research
FAPAR	Fraction of Absorbed Photosynthetically Active Radiation
FinTech	Financial Technology
FIU	Financial Intelligence Unit

Forex	Foreign Exchange
FVC	Fractional Vegetation Cover
GDPR	General Data Protection Regulation
GNDVI	Green Normalized Difference Vegetation Index
GPU	Graphics Processing Unit
InsuranceTech	Insurance Technology
IoT	Internet Of Things
JSON	JavaScript Object Notation
KeLP	Kernel-based Learning Platform
kNN	K-nearest Neighbor
KPIs	Key Performance Indicators
KYC	Know Your Customer
LAI	Leaf Area Index
LLM	Logic Learning Machine
LMR	Learning for Mismatch Removal
LOOCV	Leave One Out Cross Validation
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAP	Mean Average Precision
MCARI	Modified Chlorophyll Absorption Ratio Index
METEOR	Metric for Evaluation of Translation with Explicit ORdering
Mifid	Markets in Financial Instruments Directive
ML	Machine Learning
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
NDCG	Normalized Discounted Cumulative Gain
NDVI	Normalized Difference Vegetation Index
NLP	Natural Language Processing
NPL	Non Performing Loan

OCR	Optical Character Recognition
PCA	Principal Component Analysis
R2	R Squared
RBF	Radial Basis Function
RDD	Resilient Distributed Dataset
REIP	Red Edge Inflection Point
RMSE	Root Mean Squared Error
RWD	Real Wheel Drive
SCDF	Spring Cloud Data Flow
SHAP	SHapley Additive exPlanation
SME	Small and Medium Enterprise
SMOTE	Synthetic Minority Oversampling Technique
SNAP	Stanford Network Analysis Project
SRF	Sharpe Ratio Factor
SUMO	Simulation of Urban MObility
SVM	Support Vector Machine
TB	Terabyte
Tf-idf	Term frequency-inverse document frequency
tps	transactions per second
TPU	Tensor Processing Unit
TSVD	Truncated Singular Value Decomposition
UI	User Interface
VaR	Value at Risk
VAT	Value Added Tax
XAI	eXplainable Artificial Intelligence
XML	eXtensible Markup Language

1 Introduction

The current deliverable is the first one of a series of three deliverables whose aim is to describe the activities conducted in Task T5.4 of the INFINITECH project. The main objectives of this task are the identification, implementation and integration, as ML services, of state-of-art as well as innovative Machine Learning (ML) solutions. These solutions may be generic-purposes or specifically designed to support the ML tasks and challenges identified by the INFINITECH pilots. This first deliverable provides a landscape of the existing most adopted ML libraries and tools, focusing the attention also to libraries and tools useful for Deep Learning (DL) tasks. Then, the deliverable describes five generic-purpose ML solutions that were used or developed by ATOS, Engineering, GFT Italy, FBK, and University of Glasgow (GLA).

More specifically, we first introduce ATOS' EASIER.AI, an Artificial Intelligence orchestration platform, which can help researchers and practitioners to seamlessly test, implement and deploy ML flows and models (see Section 3.1). This platform is currently planned to be used in Pilot 11 "Personalized insurance products based on IoT connected vehicles".

Second (see Section 3.2), we present the ALIDA platform, a microservice-based platform born with the scope of facilitating Big Data Analytics. Its flows help the construction, deployment, optimization, execution, and real-time monitoring of work processes such as real-time cyber-security analysis for financial transactions. Pilot 10 (see section 4.9) has planned to leverage over this tool to deploy its service.

The third generic-purpose solution is the Rulex platform, a tool developed by Rulex company and used in several financial tasks by GFT Italy (see Section 3.3). A unique characteristic of this tool is that it grants the explainability of data-driven decisions, thus avoiding black-box Artificial Intelligence (AI) processes. To this end, Rulex platform adopts a set of innovative algorithms for eXplainable Artificial Intelligence (XAI), that combine domain knowledge from experts with intelligible rules to provide the most effective and transparent decisions.

In the fourth solution, we describe an ML-driven psychometric engine (see Section 3.4), developed by FBK during the first year of INFINITECH, that extracts a rich and diverse set of behavioural features from credit card transactions and produces as outcome the classifications of psychological traits and dispositions useful for building customers' profiles for banks and other financial institutions.

Finally, we introduce BetaRecSys (see Section 3.5) that is a novel open source framework, developed by University of Glasgow (GLA), implementing a wide-range of state-of-the-art ML/DL-driven recommendation algorithms. This framework is currently planned to be used in Pilot 6 "Personalized closed-loop investment portfolio management for retail customers".

After the presentation of four top-down solutions that are already planned to be used by pilots or can be evaluated as useful assets for them, we move to a detailed analysis of the needs and plans of the pilots regarding their ML tasks (see Section 4).

1.1 Objectives of the Deliverable

The main goal of task T5.4 ("Library of ML/DL Algorithms for Financial/Insurance Services") is to describe a diverse set of ML/DL algorithms that are suitable and potentially transformative for a large number of applications in the finance and insurance domain. This set/library of algorithms

will include both conventional state-of-the-art ML/DL algorithms and innovative ones developed by the partners during the task. This set/library will also include the algorithms that will be exploited within the projects' pilots. Similarly, it will contain simpler algorithms that FinTech/InsuranceTech companies could use for training, testing, and experimenting with data.

This overall goal encompasses, in this first deliverable, the following specific objectives:

- Collect, study and document the most adopted state-of-the-art ML and DL libraries and tools. As previously said, the INFINITECH Task T5.4 aims at providing ML/DL services and tools for satisfying the needs of the end-users of the identified INFINITECH pilots and of all the stakeholders of the project. The first step for doing this is documenting for the pilots the most adopted ML/DL algorithms and tools to facilitate their use by the pilots.
- Develop innovative ML/DL solutions and services. One of the most important goals of this task is the development of novel ML/DL solutions that can be disruptive for a large number of tasks and challenges in the financial and insurance domain. In this deliverable, we are reporting the novel solutions developed by ATOS, Engineering, FBK, and University of Glasgow as well as the product developed by Rulex and used by GFT Italy. This objective will be pursued in the following months of the project further connecting these novel solutions to pilots' tasks.
- Map, study and support the ML/DL needs and planned solutions of the pilots. This is another very important goal of the task T5.4 since the development activities of the task are focused on the pilots, as it is the case for the INFINITECH project in more general terms. This first deliverable studies the planned ML/DL solutions devised by each pilot and reports in detail the needed ML/DL tools and libraries.

1.2 Insights from other Tasks and Deliverables

Deliverable D5.7 is released in the scope of WP5 "Data Analytics Enablers for Financial and Insurance Services" activities and documents the initial outcomes of the work performed within the context of T5.4 "Library of ML/DL Algorithms for Financial/Insurance Services". The task is connected with the outcomes of WP2 "Vision and Specifications for Autonomous, Intelligent and Personalized Services" in which the overall requirements of the entire INFINITECH platform are defined. More precisely, this deliverable, in particular in Section 4 - which is dedicated to the pilots' ML/DL needs and tasks - is influenced by the collected user stories of pilots' project and the extracted user requirements (Task 2.1). Furthermore, the work reported in this deliverable is connected tightly with the work performed in T5.1 and T5.5 of WP5. Indeed, the goal of T5.1 is to collect the data that are required for the training of the ML/DL algorithms. These include the algorithms that will be evaluated and used in the scope of the project's pilots. The sources of data are multiple: as a main source, data are provided by the financial and insurance organizations of the INFINITECH consortium, from open source and alternative data, and from data obtained by IoT devices (e.g., usage-based insurance pilots). Additionally, synthetic datasets produced are produced to facilitate the training and sharing of the data outside the financial institutions of the consortium.

Regarding T5.5, this task is currently implementing OpenAPIs for accessing the novel ML/DL algorithms specifically developed for the INFINITECH project. Currently, the ML-driven

psychometric engine is under discussion as an example of a service on which we may implement and test the OpenAPIs under development in T5.5.

1.3 Structure

The structure of D5.7 is organized to be easily associated with the objectives described in Section 1.1:

- Section 2 documents the most widely adopted state-of-the-art ML and DL libraries. In particular, we describe (i) Scikit-learn, that is the state-of-the-art library for most ML algorithms, as well as (ii) TensorFlow and (iii) PyTorch, that are the most well-established DL libraries and are developed respectively by AI divisions of Google and Facebook.
- Section 3 introduces five ML solutions that are the result of the top-down efforts of five project partners, namely ATOS, Engineering, Rulex and GFT Italy, FBK, and GLA. The idea of this section is to present ML tools that can be adopted by the INFINITECH pilots in the next months to deal with their ML tasks.
- Section 4 maps and describes in detail the ML/DL tasks of the INFINITECH pilots. There we focus the attention on the data, the algorithms, the evaluation metrics, and the tools/libraries that the pilots are using or planning to use.

2 State of the art of existing machine learning libraries

Despite being on the cutting edge of worldwide research efforts, Machine Learning (ML) and Deep Learning (DL) are already counting on different well-established technologies and software tools. After an initial phase of the proliferation of tools mainly designed for pure research tasks, ML/DL technologies have spread in the industrial world and conquered many market areas. This step has seen a corresponding change of paradigm in technologies, moving from almost hand-crafted, research-oriented tools, to well-engineered, frequently-maintained, and reliable packages.

In the meantime, Python is steadily present in the top-three most popular languages in multiple rankings, and it has emerged as the undisputed first choice for ML/DL, and in general for Data Science product development.

This prominence is well recognized also in the finance and insurance world. As an example, the top Finance curricula in Business Schools worldwide already use Python as a language of choice in their training in ML and Data Analysis.

The potentially disruptive effect of ML/DL technologies in the financial world is well recognized, see e.g. the reports by the Financial Stability Board [21] and the Alan Turing Institute [10]. This potential is an additional factor that requires the use of reliable, well-tested, and well-engineered solutions.

For these reasons, in this deliverable, we restrict our attention to the mainstream Python ML/DL libraries, that as we will see in the next sections are also the most adopted or planned to be adopted by the INFINITECH pilots. Their usage addresses specific fintech needs. Within the different sections of the deliverable we will encounter relevant examples such as the one presented in section 3 by the BetaRecSys framework (which uses PyTorch in the context of recommendation system tasks) or the ones proposed by the pilots as bottom-up solutions (e.g. pilot 5b, 9, and 13 which uses Scikit-learn, TensorFlow, and Pytorch in the framework of classification, recommendation, and encoding [autoencoders] tasks). Given the relevance of these libraries in building both top-down and bottom-up solutions, we find it particularly relevant to briefly present and discuss these tools. In this section we explore the main motivations for their wide success and adoption in different fields of applications. In particular, we describe (i) Scikit-learn, that is the state of the art library for most ML algorithms, as well as (ii) TensorFlow and (iii) PyTorch, that are the most well-established DL libraries and are developed respectively by AI divisions of Google and Facebook.

We summarize in Table 1 some of their features, and we describe them in more detail in the following subsections. For the sake of completeness, the section concludes with a brief reference to other proposed solutions.

Table 1 – Mainstream state-of-the-art ML/DL libraries

Name	Developer	Initial release	Current stable release	Supported OS	License	Source Language	Web site
Scikit-learn	INRIA	June 2007	Aug 2020	Linux, macOS, Windows	New BSD License	Python, Cython, C, C++	scikit-learn.org
TensorFlow	Google Brain	November 2015	Sept 2020	Linux, macOS, Windows, Android, JavaScript	Apache License 2.0	Python, C++, CUDA	tensorflow.org
PyTorch	Facebook' AI Research lab (FAIR)	September 2016	Oct 2020	Linux, macOS, Windows	BSD	Python, C++, CUDA	pytorch.org

2.1 Scikit-learn

Scikit-learn is currently developed by the French Institute for Research in Computer Science and Automation in Rocquencourt (INRIA). New releases are published on a 3-month cycle and are led by INRIA based on the contribution of a large user community, and with the financial support of a large consortium of universities, and tech, financial, and banking companies.

The library is developed in Python, Cython, C, and C++ and it is designed to interoperate with the NumPy and SciPy libraries for numerical and scientific computing.

Scikit-learn is the de-facto standard for machine learning development in the Python programming language, and it was the second ML/DL project on GitHub in 2018 [17].

The library contains implementations of most machine learning algorithms for classification (e.g., Support Vector Machines [13], Nearest Neighbours [26], Random Forests [9]), regression (e.g., Support Vector Regression, Nearest Neighbours, Random Forests [29]), clustering (e.g., K-means, spectral clustering, mean-shift). Moreover, it offers tools for data processing and hyperparameter tuning, such as dimensionality reduction (e.g., K-means [35], feature selection [25], non-negative matrix factorization [40]), model selection (e.g., grid and random search [7], cross-validation [2]), and preprocessing (e.g., data scaling and normalization, feature extraction [3]).

All algorithms are implemented with a common interface that allows a user to easily implement extensions and custom algorithms.

2.2 TensorFlow and Keras

TensorFlow was developed by Google Brain, which produced the first release in November 2015, and then the TensorFlow version 1.0.0 that was publicly released in February 2017. A major release of TensorFlow 2.0 is available since September 2019, and the current version is 2.3.1.

The library is implemented in Python, C++, and CUDA³ and is based on a dataflow on directed graphs that enables forward passes and backward chaining.

TensorFlow was the first DL library that managed the training of very deep neural networks, and it has been a large driver of the first record-breaking results obtained by DL in image recognition tasks. As per 2018, it was the first machine learning project on GitHub in terms of contributions [17].

TensorFlow implements operations that allow one to define, train, and deploy deep neural networks. It contains implementations of standard and state-of-the-art layers that may be composed into custom architectures, to which is possible to assign losses and apply optimization algorithms, either readily implemented or customized.

The library runs on CPUs, but it is also designed for taking full advantage of GPUs via CUDA to obtain substantial acceleration. Google also develops Tensor Processing Units (TPUs) that are application-specific integrated circuits built specifically for training and development via TensorFlow and designed for high-volume low-precision computations. Access to TPU computation is available through Google's cloud computing infrastructure.

Keras, instead, is a library designed for fast experimentation with deep neural networks in a user-friendly way. It is also mainly developed by Google engineers. The library uses TensorFlow as a backend, even if it was initially supporting different libraries.

Keras offers a higher-level implementation of several commonly used building blocks and architectures, and it simplifies the writing of code for deep neural networks training and deployment.

2.3 PyTorch

PyTorch is developed primarily by FAIR (Facebook's AI Research lab) and it was initially released in September 2016, later including the project Convolutional Architecture for Fast Feature Embedding (Caffe2) that was also maintained by Facebook. The library is based on Torch, which was a Lua library for tensor computation and deep learning released in October 2002, and developed until February 2017. The current release of PyTorch 1.7.0 was published in October 2020.

The library is implemented in Python, C++, and CUDA and it works around an Autograd module that implements automatic differentiation of tensors.

PyTorch provides NumPy-like tensor computing with acceleration via GPUs, and it permits to develop deep neural networks built on the automatic differentiation system. The library offers state-of-the-art architectures that may be defined via layers in the nn module, that simplifies the creation of computational graphs and the computation of gradients when designing deep architectures. Moreover, the optim module offers several optimization algorithms that are ready to use.

PyTorch is usually recognized as easier to use than TensorFlow, and it is rapidly gaining popularity, especially in research [27], while TensorFlow is still dominant in production systems [18].

³ Compute Unified Device Architecture, the API to program Nvidia' GPU, <https://developer.nvidia.com/cuda-zone>.

2.4 Other programming tools and machine learning libraries

Other ML/DL libraries have been developed, such as Theano⁴ and Pylearn2⁵, but most of them have lost users and popularity in favour of PyTorch and TensorFlow. These were developed mainly for research purposes, and have been discontinued as soon as stable libraries developed by companies have been established.

Moreover, whilst Python and the libraries discussed previously in this section are undoubtedly the most diffused ones, other solutions are still adopted and are emerging.

One of these solutions is represented by the R programming language. This language is sometimes preferred for its richness and completeness in statistical tools. Several machine learning libraries are available within its framework. An important example of its usage and its power is reported within the scopes of pilot 14 in section 4.13.

From a different perspective, it is also of particular interest to mention the increasing diffusion of the Resilient Distributed Dataset (RDD) abstraction and the Apache Spark framework. The exponential growth of the available data in multiple research and development contexts have made relevant the importance of distributed computing. Apache Spark (and the associated APIs, e.g., PySpark) facilitates both the frequent query of data (particularly useful in exploratory and interactive analysis) and the implementation of iterative algorithms, such as the ones involved during the training of machine learning algorithms.

⁴ <https://github.com/Theano>

⁵ <https://github.com/lisa-lab/pylearn2>

3 INFINITECH general-purpose Machine Learning solutions

In this Section, we describe five general-purpose ML solutions that were identified, developed or refined in the first months of the project. While ALIDA and Rulex are already existing tools, which are identified as top-down assets by pilots for task-specific solutions, EASIER.AI, the psychometric-engine, and BetaRecSys are tools which are entirely and/or partially developed exclusively within the scopes of the project. These five solutions are already planned to be used by pilots (e.g EASIER.AI will be used in Pilot 11 for improving the behavioural profiling of drivers, ALIDA will be used in Pilot 10 to facilitate real-time Big Data Analytics, BetaRecSys will be adopted by Pilot 6 for building recommendations of products) or ready to be used by pilots. These solutions are general-purpose, but suitable for deployment and use in ML-based FinTech and InsuranceTech use cases. The INFINITECH pilots will provide tangible showcase of their use.

3.1 EASIER.AI: An Artificial Intelligence Orchestration Platform

EASIER.AI⁶ is a hybrid platform that may operate with cloud and edge nodes, thus allowing the efficient use of resources as well as the allocation of the best options for each deployment. This Artificial Intelligence (AI) orchestration platform is a tool that can help researchers and practitioners to test, implement and deploy ML models and flows seamlessly.

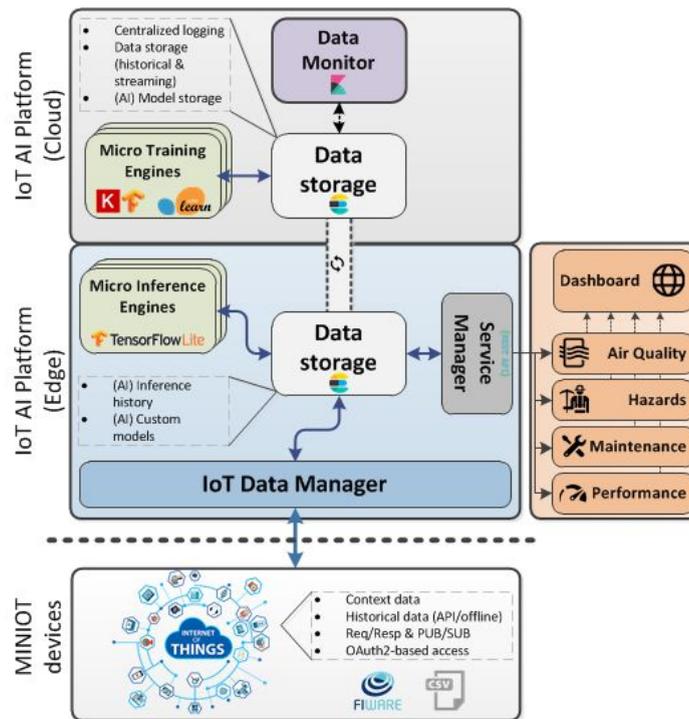


Figure 1 -EASIER.AI Hybrid Platform

EASIER.AI runs on top of a Kubernetes cluster. This means that all applications need to be containerized on Docker images. Kubernetes follows the microservices paradigm and its usage enables scalability for all services.

⁶ The platform will be developed by ATOS (<https://atos.net/en/artificial-intelligence>)

EASIER.AI offers multiple functionalities and in particular:

- **Data science capabilities:** Pre-configured Jupyter notebooks are provided for easy data analysis and engineering.
- **Data storage:** The platform provides mechanisms to store and retrieve datasets.
- **Model training:** The platform EASIER.AI provides the means to easily train and test the developed models.
- **Model storage:** Once researchers and practitioners have developed and trained their models, it is easy to store and version them on the platform. By saving relevant metadata, these models can be shared with other users of the platform, allowing a collaborative workflow.
- **Service deployment:** The developed services can be deployed on the Kubernetes cluster, providing a usable API for consuming the models. Edge or cloud nodes in the cluster can be chosen for deployment, depending on the use case.

In the context of the INFINITECH project, EASIER.AI will be initially used in Pilot 11 to provide orchestration and scalability support. The platform will provide an easy way to manage all the components of the Artificial Intelligence (AI) services: (i) dataset analysis, (ii) data storage, (iii) model training, and (iv) service deployment.

3.2 ALIDA: microservice-based platform for composition, deployment, optimisation, execution, and monitoring of Big Data analytics workflows

ALIDA⁷ asset, a microservice based platform for data management and composition, deployment, optimisation, execution and monitoring of big data analytics data workflow (covering ingestion, preparation, analysis and visualization), which has been developed by ENG in previous and ongoing research activities. The main functionalities of ALIDA are:

- Streaming and Batch data workflow processing
- Catalogue of Big Data Analytics (BDA) services (covering ingestion, preparation analysis, visualization) for various data analytics scenarios
- Graphical editor for building data data workflow
- Data pipeline deployment by means of modern resource orchestrators such as Kubernetes.
- Workflow execution monitoring.
- Data visualization customization through a set of graphs and query editor
- Graphical User Interface to easily create and redistribute new custom BDA services.

ALIDA presents a microservice architecture, where microservices are deployed in containers, whose management is largely simplified by Kubernetes, a container orchestrator which automates the deployment, management, scaling, and networking of containers.

Basically, Figure 2 shows three flows: service registration (blue flow), workflow design and execution (red flow) and visualization (orange flow).

⁷ <https://home.alidalab.it/>

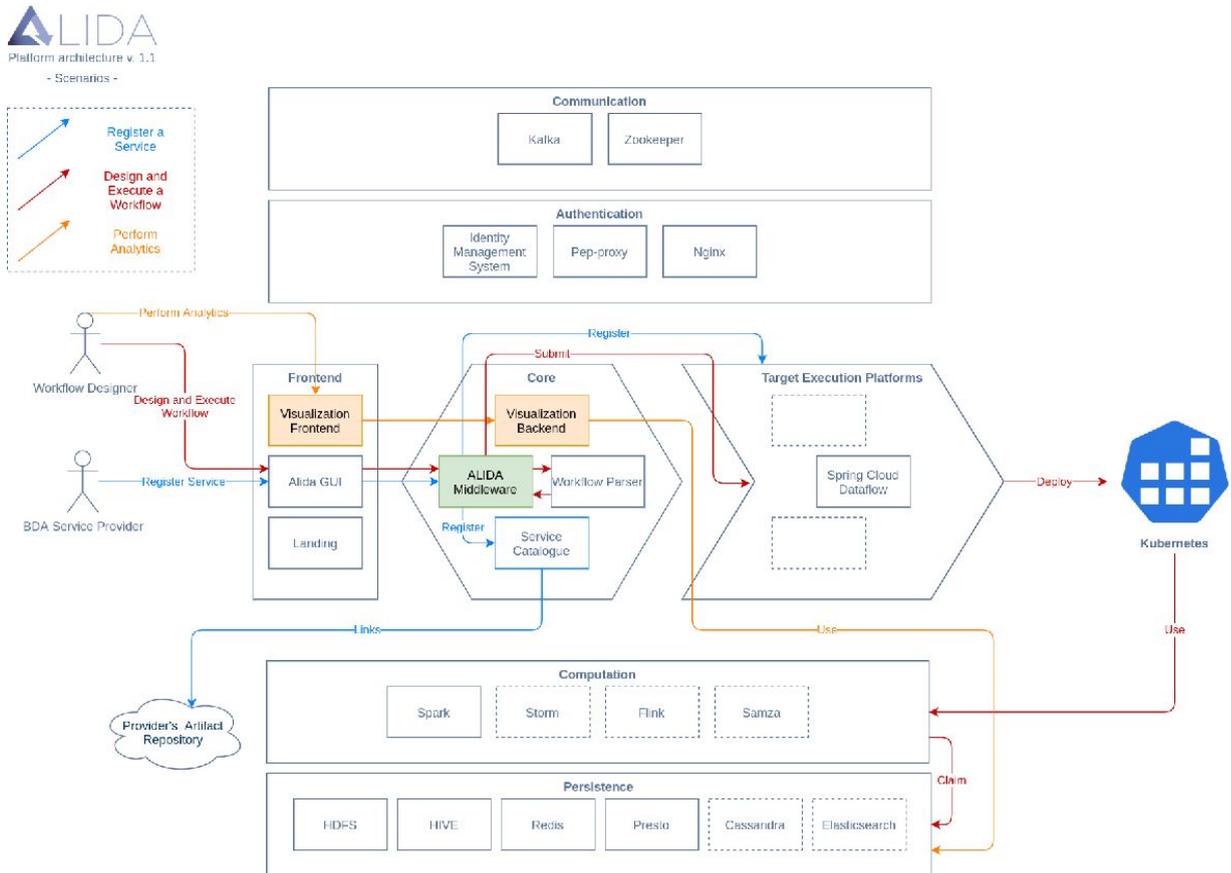


Figure 2 - ALIDA platform flows

The key flow is the workflow design and execution: by means of the GUI user designs a workflow and executes it. In this case, the core components of ALIDA submit the request of execution and deployment of the workflow to the candidate orchestrator. Currently just Spring Cloud Data Flow⁸ is adopted as pipeline orchestrator in ALIDA.

Spring Cloud Data Flow (SCDF) is a Microservice based Streaming and Batch data processor that can make use of a variety of container orchestrators.

In the ALIDA platform, the SCDF engine instance uses an implementation of the Spring Cloud Deployer for Kubernetes. Then, Kubernetes uses the computation resources available into the platform and the persistence layer to manage data storage and workloads.

The platform currently supports several frameworks such as Spark, H2O, Flink, mainly used for BDA service development and registration. Other frameworks may be deployed, and relevant machine learning libraries available for Python can be used.

It is worth to notice that BDA services may or may not be implemented working on these frameworks. The only requirement that the BDA services must match is that they have to be implemented as a spring boot application shipped within a docker image that can optionally contain a python application.

⁸ <https://spring.io/projects/spring-cloud-dataflow>

In ALIDA, most of the BDA services developed are based on Spark, so they are able to process a large volume of data, in a distributed environment. Spark is used for both batch and streaming applications thanks to Spark Streaming.

With regard to the persistence layer, Hive and Redis play several roles. Hive is used both to access the data resulting from the execution of the workflows through the visualization component, and to ensure that SPARK can write and read the datasets stored into HDFS.

Redis is used by ALIDA to store some properties related to the platform and to serve the visualization component with the aim to create graphs and visualizations with real time updating capabilities. Last but not least, Kafka is adopted on various fronts: the platform, the core component of ALIDA, uses it to transmit and update the properties during the workflow execution phases. SCDF itself exploits it in the management of streams pipelines.

ALIDA is cloud native software, this means that it can be seamlessly deployed both in an on-premise environment and on the cloud environments provisioned by the widely known providers such as Microsoft Azure, Amazon AWS and Google Cloud Platform. In the context of the INFINITECH project, ALIDA will be used in Pilot 10 to facilitate the development of real-time BDA service in the context of Cyber-Security for financial transactions.

3.3 Rulex: An Explainable AI Reasoner

In this Section, we introduce the Rulex⁹ platform, a product developed by Rulex company and used together with GFT Italy in several financial services such as onboarding evaluation, fraud detection, explainable credit rating, non-performing loan management, etc.

In simple words, the main idea of the Rulex platform is to support the daily decisions needed to drive business processes, such as deciding if a transaction is fraudulent or not, approving a credit, evaluating a loan request etc. In order to achieve this, Rulex platform integrates the needed knowledge to predict optimal decisions and to further improve current decision processes.

The main characteristics of Rulex are (i) a unified environment providing all the tools needed to manage data-driven decisions with simple integration into the customers' business processes, and (ii) high flexibility permitting to adapt the tool to the customers' specific business processes.

More specifically, business experts can use this tool without any requirement for programming skills, while data analysts and machine learning experts may have access to more advanced features (e.g., writing code in R or Python). Additionally, the Rulex platform permits the explainability of the data-driven decisions by leveraging a set of innovative algorithms for explainable Artificial Intelligence (XAI). These algorithms produce results that can be understood by humans. In this way, they contrast with the concept of the "black box" in ML/DL where even the algorithms' developers cannot explain why the AI arrived at a specific decision/outcome [16].

In particular, the Rulex platform uses Logic Learning Machine (LLM), an efficient implementation of the Switching Neural Networks [34] framework (based on the usage of Boolean algebra), to combine knowledge from domain experts with intelligible rules.

A typical Rulex workflow is usually composed of the following three phases:

⁹ <https://www.rulex.ai/>

- **Data import and data preparation:** Powerful import tasks allow you to import data from different sources (e.g., databases, spreadsheets, JSON/XML, etc.) on Rulex, to perform preprocessing operations (e.g., join, merge, pivot, filter, etc.). There are several visualization techniques available to help a user in understanding his/her data.
- **Model creation and validation:** Rulex provides a wide range of tasks to perform advanced analyses and make data-driven decisions. More precisely, the platform offers its proprietary white-box approach, Logic Learning Machine, that provides understandable and GDPR compliant by design “if-then” rules. Besides, the tool also offers neural networks and other black-box algorithms, Decision Trees, association rule tasks, regression, clustering and optimization tasks.
- **Model implementation:** Once the model is created and verified, it can be deployed in production environments to make decisions on new data. Different architectures may be chosen (both on-premises and in the cloud). Numerous features, such as scheduling, priority management, modules, macros, alerting, versioning, etc. make decision management in production easier.

Below, we provide an example of a workflow: It imports some datasets, makes some elaborations, and provides a prediction about the customers of a bank using the Logic Learning Machine.

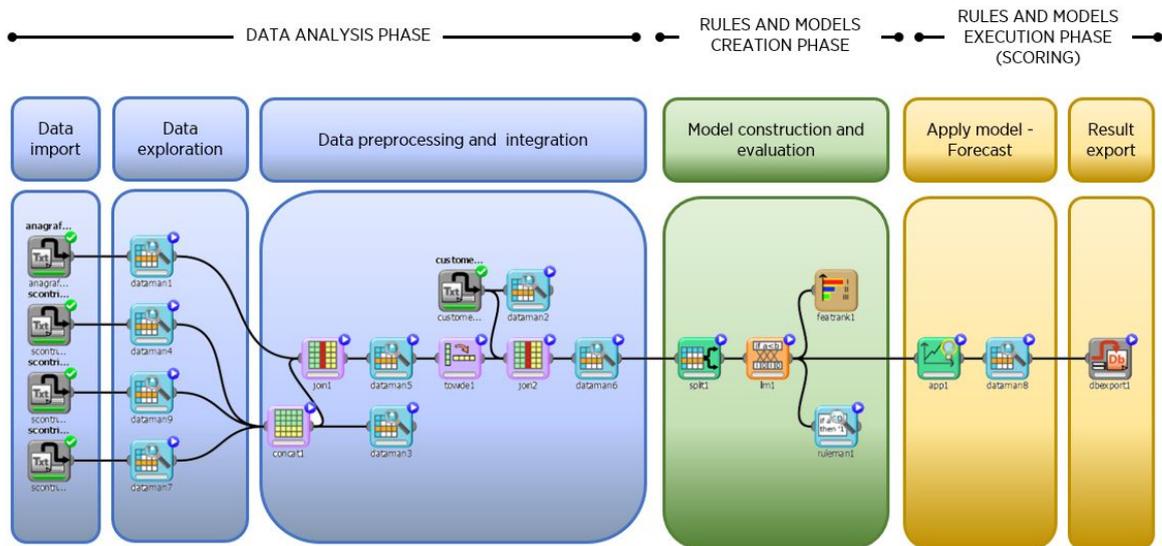


Figure 3 -Example of workflow in the Rulex platform

As previously mentioned, the Rulex platform was applied to different financial tasks in projects with some of the leading European banking groups. Some examples (the list is not exhaustive) are the following ones:

- **Onboarding evaluation:** The Rulex platform can combine data from traditional approaches with additional transactional data to calculate the financial risk of a customer without a credit line or a credit history. A characteristic of Rulex is its ability to clearly explain the reason for a decision with explicit “if-then” rules.

- Fraud detection: Rulex can determine the type of fraud, estimate its probability, and suggest checks for further analyses on preventing frauds. It is worth to notice that, using this platform, the knowledge of business experts can be combined with the rules generated by the ML approaches.
- Non-Performing Loan management: In this scenario, Rulex can be used to create different workflows to deal with different phases in Non-Performing Loan (NPL) management. These workflows can then identify the best strategy or the best way to minimize losses.
- Profiling up-selling and cross-selling: Rulex can implement a workflow to identify clients who are most likely to buy a new product using data from previous selling campaigns. In particular, the usage of an LLM approach makes it possible to check the predicted results and identify the key attributes.
- Explainable credit rating: In compliance with the GDPR, banks must be able to explain how decisions are made. Rulex uses LLM to analyse or produce rating assignments and to identify the set of rules which can explain each credit rating.

3.4 AI-driven psychometric engine for financial behaviours

The automatic assessment of psychological traits from digital footprints allowed researchers and practitioners to build up profiles and characterizations of customers at an unprecedented scale. One type of digital footprint, that is universal across the world and has received relatively little attention to date, is spending behaviour. With around 80% of adults in high-income economies using a debit or a credit card, people's spending has become increasingly digitized, making it possible to capture consumer choices. To characterize the spending behaviour of a customer, we can extract several behavioural features from credit card transaction data. In this Section, we describe a psychometric engine built by FBK, during the first months of INFINITECH, to extract four categories of behavioural features, grouped according to the type of spending behaviour they capture: (i) overall spending behaviour, (ii) temporal spending behaviour, (iii) category-related spending behaviour, and (iv) customer category profile. In the forthcoming paragraphs, we present in detail the four categories of behavioural features.

(i) Overall spending behaviour. These features are computed over the entire duration of a credit card transaction dataset. In particular, the engine has defined summary statistics of customers' spending behaviour such as the total number of transactions, the total amount a customer had spent over the entire period, and the average amount per transaction spent by each customer. Moreover, to measure the relative variability of a customer spending behaviour, the engine has used the coefficient of variation, $cv = \sigma / \mu$, defined as the ratio of the standard deviation of the number of transactions (σ) to the average amount of transactions (μ).

(ii) Temporal spending behaviour. The engine can analyse the temporal spending behaviour at different granularities using different time windows like one month, 10-days intervals and day of the week, to take into account the seasonal differences in spending behaviour. For example, the 10-days intervals can help to account for differences in spending which are due to when a customer receives his/her salary. For each customer and time unit, we can measure the temporal patterns of spending behaviour calculating (i) the variability of the spending amount, (ii) the persistence of the spending patterns, and (iii) the presence of bursty spending behaviour.

- Variability of spending amount. To study the variability in the spending amount of each customer, the engine may look at the fraction of amount spent in a particular period (day,

10-days, month) and use this information to calculate the variability of spending amount, computed as the standard deviation of the spending distribution for each customer. This results in measures of monthly (σ_M), 10-days interval (σ_P) and daily variability (σ_D).

- Persistence of spending amount. To evaluate the consistency in the amount a customer spends at monthly and a weekly observation level, the engine computes the persistence of the fraction of spending as the average of the cosine similarity between adjacent intervals.
- Bursty dynamics in spending patterns. Bursty dynamics are defined as the heterogeneous property of time series having short-time periods of intense activities alternating with long-time periods of low-frequency activities. They allow us to measure the intensity of spending activities over different periods. To compute the burstiness of the spending patterns, the engine first computes the inter-event times as the time difference between two adjacent credit card transactions. Then, a burstiness parameter B is computed. When the burstiness parameter B is -1, the purchasing pattern of customers is completely stable. If it is $B = 0$, the spending behaviour of the customer is random. Finally, $B = 1$ indicates extreme and unpredicted spikes in spending behaviour.

(iii) Category-related spending behaviour. The engine can create metrics based on the categories of purchases made by each customer. Since the total amount of credit card transactions can be biased towards high-value categories (e.g., the purchase of a car), the engine bases its metrics on the total number of transactions to measure the frequency of purchasing activities in different categories. We can compute metrics such as:

- Number of spending categories. The number of distinct categories in which a customer purchased during the entire period of the dataset.
- Diversity of spending categories. The diversity of the purchases made by each customer by looking at the diversity of categories D_{cat} using an entropy-based measure where a low value of D_{cat} indicates that the customer expenses were mostly made in a few categories. On the other hand, a high value of D_{cat} means that a customer equally distributed his/her expenses in all the categories in which he/she purchased.
- Persistence of spending categories. This metric measures the consistency in customers' purchasing categories over time. It is defined as the average of the cosine similarity between every two adjacent months.
- Category turnover. To evaluate a customer's consistency in spending over time, the engine calculates the turnover in spending categories as the average Jaccard similarity of spending categories in two consecutive months.

(iv) Spending category profile. The spending category profile reflects the relative number of transactions made in each of the spending categories present in the dataset.

Table 2 – Set of behavioural indicators extracted using the Psychometric Engine

Type	Feature	Description
Overall	n_{tot}	Number of transactions (log)
Overall	a_{tot}	Total amount of transactions (log)
Overall	a_{avg}	Average amount per transaction (log)
Overall	cv	Relative spending variability
Temporal	σ_M	Spending amount variability (monthly)
Temporal	σ_P	Spending amount variability (10-days interval)
Temporal	σ_D	Spending amount variability (daily)
Temporal	$persistence_M$	Persistence of spending amount (month)
Temporal	$persistence_W$	Persistence of spending amount (week)
Temporal	B_{tot}	Transactional burstiness
Temporal	B_{daily}	Daily burstiness
Category related	N_c	Number of unique categories
Category related	D_{cat}	Diversity of purchased categories
Category related	$C_{persistence}$	Persistence of purchased categories
Category related	$C_{turnover}^3$	Top 3 category turnover
Category related	$C_{turnover}^5$	Top 5 category turnover
Category related	$C_{turnover}^{all}$	Category turnover
Category profile	C_k	Fraction of expenses (number) in a category k

These kinds of features can be important and useful in several problems related to time-series and transactional data. The engine already uses these features to predict the personality of customers. In a three-class classifier predicting the personality of an individual as high/middle/low, the engine was able to reach a F1 score of 0.423 (baseline: a random classifier obtaining a F1 score of 0.333) for a trait like Materialism for a small dataset of 1000 individuals. This approach could be used to improve “Know Your Customer” (KYC) tasks, psychologically-informed advertising strategies for specific products as well as personality-based spending management apps and credit scoring approaches.

Currently, the FBK team is working on finalizing this engine as a microservice that can be invoked through an Application Programming Interface (API).

3.5 BetaRecSys - Deep Learning for Recommendation

As previously mentioned, BetaRecSys¹⁰ is a new open-source framework, developed by GLA, implementing a wide-range of state-of-the-art DL/ML recommendation algorithms. This framework was originally developed as part of the EC H2020 BigDataStack project¹¹, and now is under refinement to be adopted by Pilot 6. In the remainder of this section, we summarize the motivation that drove the development of BetaRecSys and its main functionalities.

Motivation and technology gap: Rapid progress has been made over the last 5 years, significantly advancing the state-of-the-art in recommendation tasks, largely driven by new DL models. However, partially as a result of this intensive and rapid progress, there are now many

¹⁰ <https://beta-recsys.readthedocs.io/en/latest>

¹¹ <https://bigdatastack.eu/>

barriers-to-entry in the recommendation field for new researchers and industry practitioners, as well as increasing challenges when comparing different works from the literature. In particular, current challenges include:

- Algorithm implementations are fragmented across different code repositories and often do not function out-of-the-box.
- Reported performances across works are often not comparable even when reported on the same dataset and with the same metrics, due to different data pre-processing and splitting techniques being employed.
- The usage of different evaluation toolkits is problematic as subtle differences in metric implementations lead to different reported performances.
- There is highly inconsistent use of model tuning strategies (most notably the omission of baseline tuning), leading to unfair comparisons.

Indeed, recent works [19, 38, 37, 46] have demonstrated that, without properly tuning model hyper-parameters, existing state-of-the-art deep learning baselines cannot consistently outperform non-neural linear ranking models. Hence, there is a clear need for platforms that provide a common recommendation methodology and pipeline, enabling model comparison across datasets, metrics, and baselines in a sound and fair manner. To date, a range of platforms have been proposed and developed, including Spotlight, Microsoft-Recommendors, DeepRec [46], OpenRec [45] and Cornac¹². However, while these works have greatly aided in making the field more accessible, they are also to blame for some of the challenges summarized earlier, as these platforms are not strongly “opinionated” about the recommendation methodology and pipeline. For instance, the popular Microsoft-Recommendors platform provides no common framework, resulting in each contained model implementing their own data preparation process. By leaving these aspects at the user's discretion and providing little in the way of guidance/best practices to follow, means that poor and/or inconsistent methodological choices are sadly commonplace. Hence, we proposed a new platform named BetaRecSys, developed by GLA, which is a unified platform for building, evaluating and tuning automatic recommender systems.

BetaRecSys architecture: BetaRecSys provides an end-to-end workflow for researchers and practitioners to build their new models or use the built-in models, thus extending the popular Pytorch format. It also provides a standardized way to configure model training and evaluate the resultant models under a unified framework that is fully compatible with INFINITECH. Figure 4 provides an overview of the platform architecture, which highlights the major components of the platform. We summarize each of the four main components of the framework (i.e., Prepare Data, Build Models, Tune & Train Models, and Evaluate Models) below.

¹² <https://cornac.preferred.ai/>

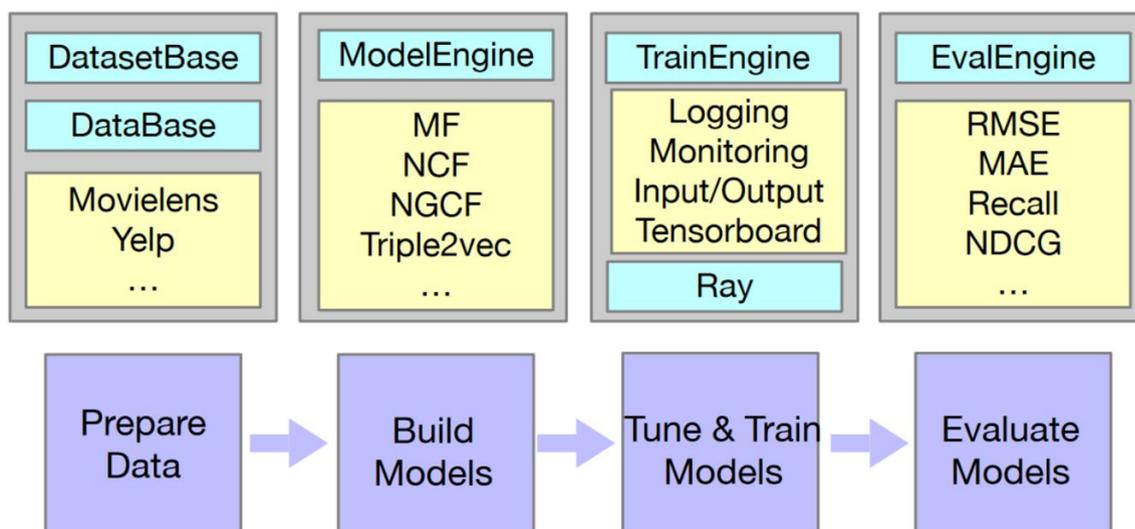


Figure 4 - BetaRecSys Architecture

Prepare Data: To make the workflow efficient, BetaRecSys implements two key reusable components for preparing training data for different recommender models, namely **DatasetBase** and **DataBase**. The **DatasetBase** component provides unified interfaces for processing the raw dataset into interactions and splitting it using common strategies (e.g., leave-one-out, random split or temporal split) into training/validation/testing sets. Meanwhile the **DataBase** provides the tools to further convert the resultant data sets into usable data structures (e.g., tensors with <user, item, rating> or <user, positive_item, negative_item(s)> structures), depending on the requirements/supported features of the target model.

Build Models: BetaRecSys framework provides a model engine (i.e., represented by the class **ModelEngine**) for conveniently building recommender models, in PyTorch (see Section 2.3) in a unified manner. In particular, it provides a unified implementation for saving and loading models, specifying the computing device, the optimizer and the loss (e.g., Bayesian Personalized Ranking loss [36] or Binary Cross Entropy loss [28]) to use during training. Out-of-the-box, 9 recommendation models are provided, including classic baselines like Matrix Factorization [37], as well as more advanced neural models such as Neural Collaborative Filtering [28], Neural Graph Collaborative Filtering [44] and Triple2Vec [23].¹³

Train & Tune Models: The **TrainEngine** component provides unified mechanisms to manage the end-to-end training process. This encompasses: (i) loading the configuration, (ii) loading the data, training each epoch, (iii) calculating validation performance, (iv) checkpointing models, (v) testing early stopping criteria; and (vi) calculating the final test performance. The **TrainEngine** also supports monitoring/visualizing the training progress in real time, including resource consumption and training metrics (such as the training loss and evaluation performance on both the validation and testing sets) of a deployed model via TensorBoard (i.e., a suite of visualization

¹³See <https://beta-recsys.readthedocs.io/en/latest/notes/models.html> for full list of supported models

tools helping the understanding, debug, and optimization of TensorFlow programs). It can also expose these real-time metrics to a Prometheus¹⁴ time-series data store via an in-built Prometheus exporter, thus enabling programmatic access to the training state. To support easier and faster hyperparameter tuning for each model, we also integrate the Ray framework, which is a Python library for model training at scale. This enables the distribution of model training/tuning across multiple GPUs and/or compute nodes.

Evaluate Performance: Three categories of commonly used evaluation metrics for recommender systems are included in this platform, namely rating metrics, ranking metrics and classification metrics. For rating metrics, we use Root Mean Square Error (RMSE), R Squared (R2) and Mean Average Error (MAE) to measure the effectiveness. For ranking metrics, we include Recall, Precision, Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) to measure performance of ranking lists. Model evaluation using built-in classification metrics like Area-Under-Curve (AUC) and Logistic loss are also supported. To accelerate the evaluation process, the metric implementations are multi-threaded.

In summary, BetaRecSys is a new open source framework for performing recommendation tasks that was developed within BigDataStack to support next generation recommendation technologies. The framework has been released as open source and is actively being built upon in INFINITECH for financial product recommendation in Pilot 6¹⁵. Moreover, BetaRecSys is suitable for the implementation of other recommendation-related use cases in the digital finance sector, such as recommendations in trading, investing, advising and risk assessment applications.

4 INFINITECH pilots' Machine Learning tasks

In this Section, we describe in detail

- the ML/DL tasks (classification vs. regression tasks, ranking tasks, unsupervised/semi-supervised/supervised approaches, etc.) that pilots are designing,
- the data they are using or planning to use (see also deliverable D5.13 to have more details on real-world and synthetic data identified and collected by INFINITECH pilots),
- the algorithms they are using or they are planning to use (Random Forest, Decision Trees, Support Vector Machines, Deep Neural Networks, etc.),
- the evaluation metrics (Accuracy, F1-score, Precision, Recall, RMSE, etc.), and the libraries and tools (e.g. Scikit-learn, TensorFlow, PyTorch, etc.).

In general, the workflow of each pilot follows a three-step process (see Figure 5). At first, the gathering, preprocessing and preparation of a reliable and clean data batch. As a second step, the study and construction of task-specific AI pipelines that use state-of-the-art ML/DL algorithms. As a third and final step, the development and deployment of innovative financial services.

It is worth noting that two pilots (i.e., Pilot 3 and Pilot 7) are currently in a reorganization phase due to the change of some partners and thus while they are not discussed in the current deliverable but, will be included in a future deliverable. This process is in-line with the INFINITECH

¹⁴ <https://prometheus.io/>

¹⁵ <https://github.com/beta-team/beta-recsys>

Reference Architecture (INFINITECH-RA) and mainstream processes for data mining such as the CRISP-DM (Cross Industry Standard Process for Data Mining).

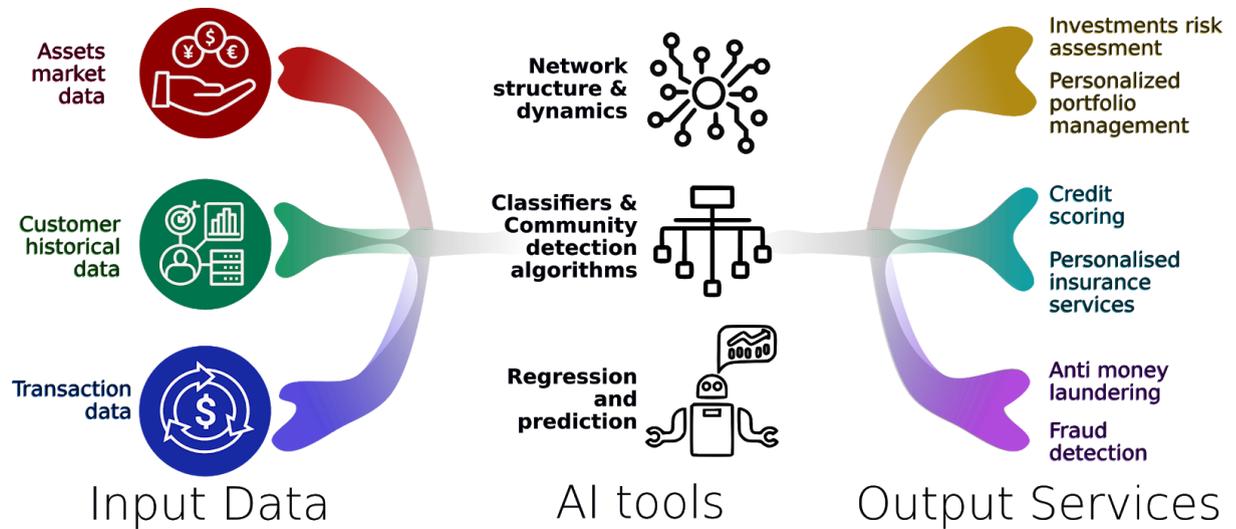


Figure 5 - Schematic of the pilots' workflow

4.1 Pilot 1 - Invoices processing platform for a more sustainable banking industry

The main objectives of Pilot 1 are to develop, integrate and deploy a data-intensive system to extract information from notary invoices, in order to: (i) establish the sustainability index of each notary based on the number of physical copies that are issued; (ii) provide to financial institutions the information (properly indexed) about the documents that are finally generated by the notarial services required by the bank, and (iii) promote notarial services from those with the higher sustainability score.

The pilot uses 32000 invoice documents from 3000 different notaries as well as the TableBank dataset, namely a table benchmark for image-based table detection and recognition (i.e., 500000 documents)

In terms of technical results, the following components will be developed, integrated, deployed and operated: (i) a batch processing architecture to process notary invoices; (ii) a Computer Vision (CV) system to identify and extract tables, (iii) a Machine Learning (ML) system to extract tables, and (iv) a visual console to show results and the sustainability score.

The pilot's success will be measured on the solution of real-life challenges as the speed up of the invoice payment process, the savings in terms of money due to fast error detection, etc.

The ML tasks comprise three main steps:

1. Conversion of the document in image format to a digital format with text recognized as characters -- Optical Character Recognition (OCR) -- together with the relative position of the characters.

2. Recognition of the different entities relevant to the auditing process, like invoice location, billable concept, quantity, attributes and price, etc.
3. Association of the different entities forming the concept of interest: for a given billable concept which is the corresponding attribute, quantity, etc

Optical Character Recognition (OCR): OCR is usually performed using standard open source packages like Tesseract¹⁶ that use a combination of ML algorithms. In the case of our project we are also using Keras-ocr¹⁷, which is an open-source package that uses Convolutional Neural Networks (CNN) with deep layers for the OCR, together with other CNNs for character recognition. In the framework of this project, the Keras-ocr package has been forked, different elements of the network modified and re-trained.

Entity Recognition: Entity recognition is performed by a ML pipeline composed of different elements. Features are extracted using Natural Language Processing (NLP) algorithms (tokenization, lemmatization, TF-IDF algorithm). Additional features are created. The result is vectorized and the resulting vectors are reduced in dimensionality by means of a Truncated Singular Vector Decomposition (TSVD). The resulting vectors are fed to a Dense Neural Network (DNN) of several layers.

Concept Association: Concept association is also implemented as a Dense Neural Network (DNN) of several layers that takes as input the input vectors and the result of the previous step Entity Recognition.

Evaluation Metrics: Due to the availability of a well-defined ground truth, and because of the unbalanced nature of the task, the performance metrics used for extraction and concept association are the combination of Precision and Recall. For the OCR standard metrics like the character recognition ratio could be used, but has been found to be impractical. For the association, Accuracy (due to the balanced nature of candidates) has been found to be the most practical.

4.2 Pilot 2 - Real-time risk assessment in investment banking

This pilot has the ambition of implementing a real-time risk assessment and monitoring procedure for two standard risk metrics – Value-at-Risk (VaR) [31] and Expected Shortfall (ES) [1]. Both metrics can be applied for measuring various types of risk, such as the market risk of portfolios of assets. The pilot implements both risk metrics for estimating market risk and it allows updates with changing market prices and/or changes in the bank's portfolio in (near) real-time. To this end, the pilot leverages real-time signals of assets that comprise a Forex (Foreign Exchange) portfolio. In addition, it implements the evaluation of what-if-scenarios allowing pre-trade analysis, i.e. estimating changes in risk measures before a new trading position is entered.

The innovation of the pilot is two-fold: First of all, it adopts real-time analytics over Forex portfolios to empower traders to take accurate decisions on-line. Second, it illustrates different

¹⁶ [https://en.wikipedia.org/wiki/Tesseract_\(software\)](https://en.wikipedia.org/wiki/Tesseract_(software))

¹⁷ <https://keras-ocr.readthedocs.io/en/latest/>

ways for calculating mainstream risk parameters (i.e., VaR), including methods based on scientific/statistical computing (e.g., Monte Carlo simulations) and machine learning approaches.

The pilot uses price data for the most liquid Forex, Stocks, Stock Indices and Derivatives. The initial focus of the pilot is on Forex assets, yet its extension to other types of portfolios and securities will be straightforward.

Additional data will be considered at the late stages of pilot deployment for stress testing and performance evaluation at the system level.

To calculate risk metrics (such as VaR) and to conduct pre-trade analysis, the pilot will explore two different approaches:

Scientific/Statistical Computing Approach: This approach will be based on a purely statistical approach for computing the VaR, following some common assumptions about the distribution of the prices of the different securities (e.g., Gaussian distribution). Specifically, three different methods for VaR calculation will be pursued:

- Calculation based on historical approaches that assume that the history of prices will repeat itself.
- Calculation taking into account the volatility of the portfolio within a given time window, i.e. based on the calculation of the variance-covariance matrix of the portfolio in a given time window of prices.
- Calculation leveraging Monte Carlo simulations, i.e. simulations of the value of the portfolio based on random parameters about its ultimate value in-line with known methods for financial markets modelling [MacKenzie, 2006].

Machine Learning Approaches: ML models will be explored to conduct post-trade analysis. They will aim at determining the maximum value/price of specific securities that traders should accept before closing the position. Specifically, a trader may not hold a position above or below a certain threshold. To calculate this threshold and its impact on the VaR of the portfolio, different classifiers will be tested such as (i) Nearest Neighbor, (ii) Linear Regression, (iii) Random Forest [9] and Long Short Term Memory (LSTM) [30] neural networks. The latter will be used to identify if the prices' changes are below a targeted quantile or not. Metrics such as the Mean Squared Error (MSE), the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) will be calculated and used to compare the different classifiers.

It is worth stating that the exact ML algorithms that will be explored are still under investigation, as the initial pilot prototype leverages a pure statistical/scientific computing approach.

The data analytics tasks implementation will be based on Python libraries (see Section 2), namely:

- Python Pandas for data transformations and data manipulation in general.
- NumPy and SciPy for mathematical calculations, numerical analysis and scientific computing in Python.

- Scikit-learn (see Section 2.1) for implementing different ML classifiers (e.g., Linear Regression, Random Forest, etc.).
- Python Keras (see Section 2.2) for Implementing LSTM solutions.

4.4 Pilot 4 - Personalized portfolio management - mechanisms for an AI-based portfolio construction

The main goal of this pilot is to develop and adapt within a wealth management platform, called “Privé Managers”, an optimization algorithm (also called “Privé Optimizer AIGO”), as well as improving and expanding its capabilities as an AI engine to aid investment propositions for retail clients.

The AI-based portfolio construction will enable advisors and/or end user customers to use the existing wealth management platform and make use of its risk-profiling and investment proposal capabilities, starting from customers’ risk-awareness. AIGO allows for a variety of use cases that cater to the needs of financial advisors, end-clients and financial firms alike.

The innovative AIGO genetic algorithm can be used for proposing investments, and evaluating them given an easy-to-use, personalizable set of criteria, in the form of “fitness factors”. These “fitness factors” will be used to generate “health” scores, which are used to define the “fittest” investments. Both quantitative and qualitative historical data will be leveraged for personalizing the portfolio construction and investment proposals, rendering the journey clear from a regulatory perspective.

The highlights of this methodology lay in four different characteristics:

- Flexibility in the fitness factors used to define the objectives of the optimization process: Users (i.e., financial advisors) can design and assign weights to those factors that are important to investors. It gives a place to the users to integrate their market views and also a feature “frozen” to keep the holdings an investor wants to stay untouched. With the flexibility of this methodology, the optimizer can provide a highly customized solution depending on the clients’ needs.
- Convergence speed to reach a recommended portfolio: Given that there are multiple angles that need to be optimized and there is a huge product universe to select from, the optimizer can provide a solution with an average 5.4 second response time.
- Compatibility with other optimization methods and other strategies: For example, a traditional mean-variance optimization can be defined by a Sharpe Ratio Factor (SRF), by which the optimizer will recommend a portfolio with the highest sharpe ratio, or a strategy such as a product diversification can be defined by a product diversification allocation factor (AFAC), by which the optimizer can recommend a portfolio according to the product diversification strategy.
- Multi-asset support: This methodology can work with different asset classes and product types including stocks, bonds, funds and structured products.

The typical steps of using Prive Optimizer are:

1. Provide current portfolio holdings details;

2. Specify their preference including account currency, preferred currency(ies), fitness factors, weights, etc.;
3. Specify product universe (a product universe is a pool of assets which the optimizer can select from);
4. Send the request through Prive Optimizer API or UI;
5. Prive Optimizer generates and returns the recommended portfolio through the developed algorithm.

Below we provide the details of the patented optimization algorithm:

1. After receiving the inputs, the 1st generation of portfolios is generated;
 - 1.1. There will be 256 portfolios in every generation;
 - 1.2. A random generation process is used;
 - 1.3. The first portfolio of the first generation will be the input portfolio (from the data input);
2. The assets and their weights within the portfolio are randomly assigned;
3. The Fitness Score, which is the weighted aggregate of different factors, is calculated for all portfolios. The portfolios are then ranked according to their scores;
4. Portfolios with fitness score in the top 50% range will be selected, the bottom 50% will be disposed of;
5. The surviving 50% (128) portfolios will be treated as parents to generate 128 offsprings; Each child is generated by two adjacent parents, for example, the 1st portfolio will be combined with the 2nd one, the 2nd one with the 3rd one and so on. The 128-th portfolio will pair with a randomly created new portfolio;
6. Mutation is performed by introducing assets, not from existing portfolios;
7. Step 2 to step 6 will be repeated for 100 times;
8. The portfolio with the highest fitness score is selected as the final output.

Table 3 – Example of multi-factor aggregation for fitness computation of customer portfolios

	Weighted Fitness Factor 1	Weighted Fitness Factor 2	Weighted Fitness Factor 3	...	Fitness-Score (Weighted Sum of Factors)
Portfolio 1	0.13	0.29	0.14		0.88
Portfolio 2	0.10	0.07	0.03		0.24
Portfolio 3	0.34	0.16	0.21		0.90
...					
Portfolio 254	0.01	0.01	0.00		0.05
Portfolio 255	0.03	0.06	0.05		0.20
Portfolio 256	0.29	0.18	0.31		0.83

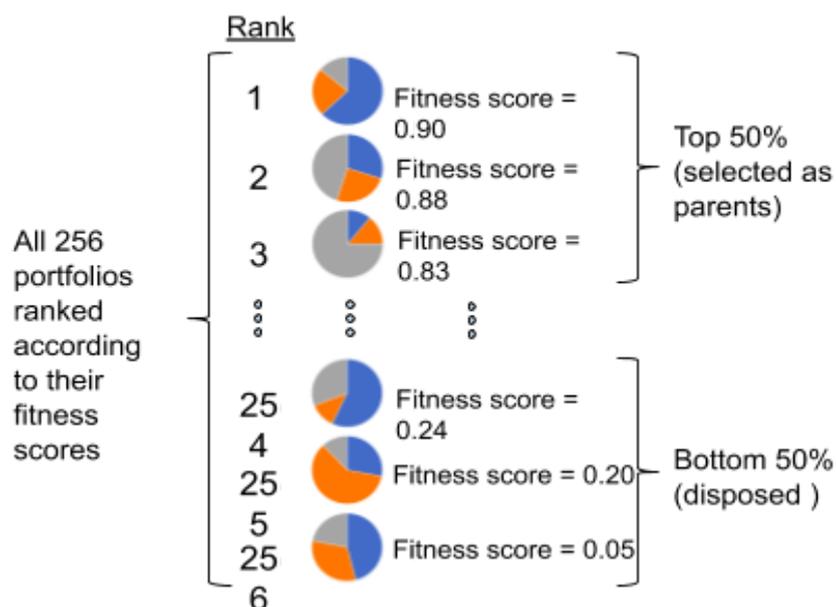


Figure 6 - Schematics of portfolio selection based on fitness score

4.5 Pilot 5B - Business Financial Management (BFM) tools delivering smart business advice

This pilot aims at building AI-powered Business Financial Management (BFM) tools that assist Small and Medium Enterprises (SMEs) clients of the Bank of Cyprus (BOC) in managing their financial health by providing assistance with activities in the area of cash flow management, continuous spending/cost analysis, budgeting, revenue review, or Value Added Tax (VAT) provisioning. BFM tools unlock the potential of AI and harness the power of data to generate personalized actionable business insights. More precisely, the BFM tools aim at driving the SME digital adoption rate as well as paving the ground for reducing credit risk, lowering the amount of Non-Performing Loans (NPLs) and for vital needed improved/streamlined SME lending.

The system will integrate multiple BFM tools, such as cash flows, health status, benchmarking, budgeting, into a global recommender unit. The recommendation system will be based on data from three main pillar sources: (i) Market data, (ii) Bank data (e.g., customer data), and (iii) SMEs data (e.g., SMEs taxation data).

This stream of data will be then used as input of the ML algorithms at the core of the pilot analytics. In particular, the ML concept constitutes one of the innovative aspects of the pilot's approach. It leverages multiple data sources to generate actionable insights on key BFM aspects. The tasks on which ML will be used to construct the recommender system are the following ones: (i) cash flow prediction, (ii) budget prediction, (iii) Key Performance Indicators (KPIs) monitoring, (iv) taxation monitoring, (v) taxation categorization, (vi) invoices processing, and (vii) benchmarking.

The recommender system will merge and process insights to transform them into “recommended actions”, which will construct the output of the coupled data-analytics system.

Many of the components above will be interconnected to provide holistic business information, alerts and recommendations. Table 5 summarizes the technologies, tools and libraries used to develop the data analytics’ and ML components.

Table 4 – Technologies, tools, and libraries to be used in pilot 5B

Technologies, tools and programming languages	Minimum version required	Function
Python	3.6	Main programming language
Docker	18.09.7	A platform for docker-container building/running/distributing
Pandas	0.25.2	Library required for the data exploration and manipulation
NumPy, SciPy	-	Statistical algorithms and numerical manipulation
Statsmodel	-	Time-series analysis and forecasting
Scikit-learn		Development of ML models and model testing
Keras, PyTorch, TensorFlow	-	Deep-Learning platforms utilized for the recommender engine
Apache Kafka	-	Open-source stream-processing software platform
Apache Spark	-	Open-source distributed general-purpose cluster-computing framework
GluonTS	-	Probabilistic time-series modelling framework for cash flow forecasting

ML algorithms and technologies will be utilized for most of the pilot’s analytical components. In more detail, the Transaction Categorization Engine, a key component for the development of the rest of the tools, follows a hybrid approach combining rule-based categorization to label a high percentage of SME transactions. The transactions which are not categorized through the rule-based approach are then fed into an ML model. Both Classification (supervised) and Clustering (unsupervised) approaches are tested aiming at higher Accuracy, Precision, Recall and F1-Score. For the classification approach Gradient Boosting algorithms (Tree-based model) and Deep Neural Networks are used, while for the clustering approach, both DBSCAN and K-Means models are utilized. When the final model is presented, more focus will be given on the Explainable AI part, where LMR (Learning for Mismatch Removal) techniques and/or SHAP (SHapley Additive exPlanation) values calculation will be used. More precisely, SHAP is a

game-theoretic approach used to explain the output of any ML model that connects optimal credit allocation with local explanations [42].

As some of the components to be developed have prerequisites and are still not mature, a more detailed list of the ML algorithms used in the pilot through its completion will be available at a later development phase and discussed in the second deliverable of task T5.4.

4.6 Pilot 6 - Personalized closed-loop investment portfolio management for retail customers

Pilot 6 focuses on providing personalized investment recommendations for the retail customers of the National Bank of Greece (NBG). NBG will leverage large customer datasets and large volumes of customer-related alternative data sources (e.g., social media, news feeds, on-line information) to make the process of providing investment recommendations to the retail customer more targeted, automated, effective, as well as context-aware (i.e., tailored to the state of the market).

More specifically, the pilot will use data related to investment products' clients from the NBG data warehouse: (i) Customer Relationship Management (CRM) data, (ii) deposit account transactions, (iii) cards transactions, (iv) investment related transactions, and (v) Mifid (Markets in Financial Instruments Directive) questionnaires data. These different sources of data will be used in order to feed a ML-driven Customer Investment Risk Appetite Profile algorithm. As outcome the algorithm will divide all customers in specific profile clusters based on the investment and banking behaviour.

Based on the (i) customer risk profile clusters and (ii) the relative mapping of financial instruments vs. customer risk profile data provided from the bank, we will feed the ML-driven Personalized Investment Recommendation algorithm, in order to provide personalized recommendations for each customer on the more suitable and profitable financial instruments.

In order to enhance the results of the Personalized Investment Recommendation algorithm the recommendations will take into account data based on news and social media leveraging sentiment analysis, for each of the financial instruments that the algorithm will select and for each customer, thus also providing an expected suitability score.

In summary, the pilot aims at utilizing ML tools and algorithms for the development of two main components:

- a Customer Investment Risk Appetite Profiling service
- a Personalized Investment Recommender service.

The final objective aims to provide an application for NBG that will allow account officers to provide more personalized recommendations for specific portions of the clients that are interested in investing in various financial instruments available from the bank.

The pilot is split into a few main parts as follows:

The first service of the pilot is the generation of the Customer Investment Risk Appetite Profile algorithm, using transactional data. This prediction will be an important input information for the second service of the pilot which is the Recommendation algorithm.

The first service will try to predict whether any customer, either with or without investment activity, would belong to one of 4 possible risk appetite profiles. The 4 target classes for the first service are: conservative; income; balanced; developmental.

In general pilot's implementation uses for the ML part the following Python modules: Scikit-learn; Pandas; NumPy; imblearn (for target class balancing).

For the first service, we start with a data aggregation step. Indeed, the data extracted are at per-transaction level, however we need to predict the risk appetite of customers. Thus, we build suitable Knowledge Performance Indicators (KPIs) that project/transform the transactional information into customer data. Our KPIs include:

- Basic aggregations (e.g., sum, counts, medians, standard deviations, etc.);
- time windowed aggregations (i.e., 3, 6, 9 and 12 months);
- aggregations per value of important nominal attributes, aggregations per user;
- percentages;
- quality attributes like salaries and pensions, spending ratios, etc.

The second step consists in data cleaning and more precisely in (i) handling extreme and missing values, and (ii) removing non-useful attributes.

Then, we have a feature engineering task consisting of the following steps:

- Discretize semi-empty attributes so that all information is used;
- standardize all numeric features before use, so that relative magnitudes don't influence models;
- apply one-hot encoding on nominal values;
- test various feature transformation techniques provided by Scikit-learn, like PCA (Principal Component Analysis), TruncatedSVD, etc.

Finally, we have a model inference and hyperparameter tuning task consisting of the following steps:

- The evaluation process of the models uses both cross-validation and a simple Percentage-Split testing. In order to oversample the training data the usage of imblearn's SMOTE technique [12] is selected, so that to have balanced classes in the training set, as long as enough data to train the models on;
- Test various feature selection techniques provided by Scikit-learn (see Section 2.1), such as SelectKBest & Chi2, SelectKBest & ANOVA, etc.;
- Tune, using grids, the parameters of various Scikit-learn models such as Logistic Regression, Random Forest, Decision Trees, AdaBoost, Neural Network Multi Layer Perceptron (MLP);
- Provide some demo predictions on the processed customer test data.

It is of note that the 1st service has a multi-class classification problem to handle. However, many of Scikit-learn's popular techniques and functions do not handle the multi-class classification problem. During the inference stage, these techniques handled the multiclass problem either by using algorithms that handle this task (e.g., Logistic Regression and MLP), either by using `sklearn.multiclass.OneVsRestClassifier`, which is a meta-classifier that uses a simple classifier and applies the one-vs-all strategy, which consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. Each instance belongs to the predicted class by the classifier with the smallest distance value.

In order to evaluate our models, we may look at the generic picture of and compare the following metrics on the test set:

- Precision and Recall per class;
- Overall Accuracy;
- One-vs-One ROC AUC scores (macro) & (weighted by prevalence);
- One-vs-Rest ROC AUC scores (macro) & (weighted by prevalence);
- Cohen's Kappa score.

Since it has applied the 1st machine learning cycle on the data, testing out the results on various models is performed, but not yet decided on a most suitable one. All tested out models provide evaluation metrics of a similar range of values, which implies to try out a 2nd machine learning cycle with emphasis on better exploitation of the input data, with a goal of improved results and finding a clear "winner" among models. For now we see the tendency of sklearn's Multilayer Perceptron classifier (sklearn.neural_network.MLPClassifier) to give better results than the other tested-out sklearn classifiers.

For the second service, the Personalized Investment Recommender, Pilot 6 aims at trying out various approaches such as:

- "Surprise"¹⁸ Python module: Surprise is a Python module for building and analyzing recommender systems that deal with explicit rating data;
- Beta-RecSys project: Beta-RecSys - described in Section 3.4 above - is an open source project for building, evaluating, and tuning automated recommender systems;
- Custom-made DL recommendation models using TensorFlow 2.

However the experiments for the 2nd service have not yet started; thus a final approach cannot be chosen.

As some of the components to be developed have prerequisites and are still not mature, a more detailed list of the ML algorithms used in the pilot through its completion will be available at the next version of the deliverable.

4.7 Pilot 8 - Platform for Anti Money Laundering (AML) supervision

This pilot has the ambition of developing a platform that will improve the effectiveness of the existing supervisory activities in the area of Anti-Money Laundering/Combating the Financing of Terrorism (AML/CTF) by processing large quantities of data owned by the Bank of Slovenia and other competent authorities (e.g., Financial Intelligence Unit (FIU)).

More precisely, this platform is based on three components, (i) an anomaly detection and prediction component, (ii) a pattern discovery and matching component, and (iii) a StreamStory component. Below, we provide details on these three components.

Anomaly detection and prediction component:

Anomaly detection is a task often addressed in the field of AI, but usually not well-defined. An anomaly refers to a previously unseen or rare uncharacteristic event. In practice, an anomaly can refer to an event that has been seen previously, but is defined as undesirable within the system (e.g., fraud, money laundering, etc.). In general, we could define an anomaly as "an outlier" data

¹⁸ <http://surpriselib.com/>

point which does not follow the collective common pattern of the majority of the data points and hence can be easily separated or distinguished from the rest of the data [6]. In the INFINITECH project we are dealing with data from the financial domain: mainly transactions of various types and interactions between different entities. Time series can describe behaviour of various entities on different levels, for example time series of transactions of particular entities (which are more or less similar), or on the level of one particular financial institution, or on a set of similar entities. Within a time-series an anomaly can be defined as a data point that does not follow the expected trends or seasonality characteristics. To detect anomalies, one must first define what type of anomalies he/she has in mind. The biggest challenge in anomaly detection is the definition of what we regard as an anomaly within the selected INFINITECH use cases. Therefore, we will mainly be focused on unsupervised methodologies. Thus, in order to detect and identify anomalies we will test a range of various approaches:

- Clustering based approaches (k-nearest neighbor algorithm (k-NN) [4], hierarchical clustering [39]);
- statistical profiling (n-sigma/percentile filter);
- concept drift detection (Drift Detection Method [5], Adaptive Windowing (ADWIN) [8]);
- predictive confidence based on:
 - o baseline models;
 - o isolation forest [32];
 - o ML predictive models.

The architecture of the anomaly detection and prediction component is straightforward: The component would anticipate feature vectors relevant for a particular use-case and would yield anomaly alerts based on those data.

Internally the anomaly detection and prediction component could implement several anomaly detection approaches. However, the findings on real INFINITECH use-case data will drive the choice of the more adequate approaches. As libraries and tools used or developed, the Pilot 8 will leverage the following ones for this component:

- Qminer (NodeJS) [24];
- Scikit-learn (Python).

Pattern discovery and matching component:

Pattern analysis provides two main functionalities: (i) pattern matching and (ii) discovery. The component will provide support for detection of complex patterns in a graph data representation. A pattern is defined as a sequence of events with a given length or specific graph paths or sequences. In pattern matching analysis we will focus on detecting relevant (already identified) patterns. In this analysis, specifically in the financial domain, there are two main challenges: either we need to reduce the data points from the initial time series or the data patterns are too sparse, hence we have the opposite problem. Usually appropriate data preparation needs to be performed in the data preparation phase, but in case of sparse data, there is no easy solution. During the development of the pattern discovery and matching component, Pilot 8 will research and develop novel approaches to detect rare patterns in sparse data space. As libraries and tools, the pilot will leverage the following ones for this component:

- Qminer (NodeJS) [24];
- SNAP (Stanford Network Analysis Project)¹⁹.

StreamStory Component:

StreamStory (Stopar et al., 2019) is a component for the analysis of multivariate time series on multiple scales. It computes and visualizes a hierarchical hidden Markov model (HMMM) [22] which captures the qualitative behaviour of the systems’ dynamics, where the system is described with a group of timeseries. StreamStory is mostly used for multivariate data visualization, time series visualization, cluster visualization and multi-scale visualization. It enables analyses on the intersection of two or more data categories. In each category, we focus on techniques able to visualize large datasets. Despite each of these categories offering a wide array of tools and techniques, the visualization of large, multivariate time series is still an understudied area. Let’s assume that one system is described with a series of time-series data. StreamStory enables analysts to visualize and compute regularities between different time series and therefore enable analysts to create new knowledge about system’s behavior.

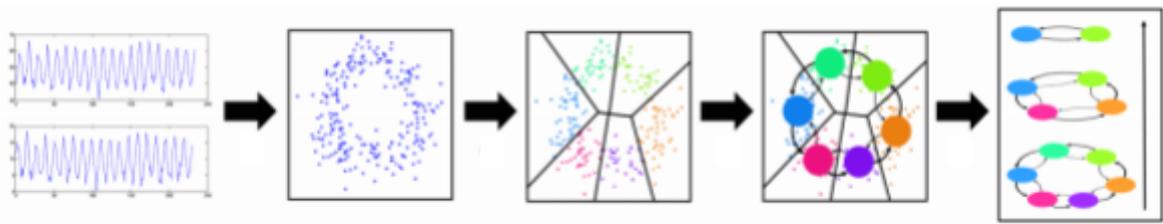


Figure 7 -StreamStory pipeline

Figure 7 depicts the StreamStory pipeline:

- First of all, the multivariate time series is represented as a point cloud. We show two noisy approximately periodic signals, mapping to 2D space;
- The space is then partitioned and the typical states of the system are identified using a clustering algorithm. System states correspond to intervals of the phase of the signal and represent partitions;
- Each partition is then translated into a Markov chain model;
- The Markov chain model is simplified by aggregating states, giving a multi-scale view of the model.

In this manner a novel abstraction for multivariate time series is presented and it is based on multi-scale continuous time Markov chains extracted from data using ML techniques. The data is shown in a qualitative manner - via a hierarchy of directed graphs - allowing users to find suitable scales to interpret the data. The system requires minimal user input and, in addition to automatically constructing the representation, it provides numerous auxiliary tools to aid the interpretation. As libraries and tools, the pilot will leverage for this component Stream Story [41] and Qminer (NodeJS) [24].

¹⁹ <http://snap.stanford.edu/index.html>

4.8 Pilot 9 - Analyzing Blockchain transaction graphs for fraudulent activities

This pilot aims at developing and deploying a scalable and high-performance system dedicated to the analysis of the blockchain transaction graph. In particular, the system investigates whether customers' blockchain account transactions can be traced to fraudulent activities or accounts. Indeed, blockchain cryptocurrencies and tokenized assets, that are obtained fraudulently, can go through various transfers on the blockchain and end up as stable coins (e.g., USD, EUR, TRY tokens) in different jurisdictions. As a result, a company accepting stable coins may be paid by using coins that can be traced to addresses involved in fraudulent activities. Holding stable coins, that originated from fraudulent or sanctioned addresses, can be risky for the company. Hence, the construction of a massive blockchain transaction graph and its analysis are necessary activities to detect frauds.

It is worth noting that transaction graphs are big and growing: for example, Bitcoin transactions are currently 527 million and Ethereum transactions are 700 million. Moreover, transactions-per-second (tps) is low on public blockchains, namely 7 tps for Bitcoin and 15 tps for Ethereum (i.e., Ethereum performance is expected to increase in future releases). Other blockchains such as Hyperledger reported achieving 3500 tps in a cloud environment.

To carry out the analyses, the pilot developed will explore two approaches:

1) Graph Algorithms Approach: Here, parallel graph traversal-based algorithms will be used to trace transactions from blacklisted addresses to customers' addresses. A directed subgraph that contains paths from blacklisted nodes to customers' addresses will be computed. Such a graph can be huge, especially, if addresses of crypto-currency exchanges appear in between the paths from blacklisted nodes to customers' nodes. Hence, in order to discover important addresses such as those of the exchanges, a parallel PageRank algorithm can be used.

Block explorer sites such as Etherscan.io also have a directory of exchanges and hence these will be explored to extract exchange addresses. After a directed subgraph is extracted, a score will also be assigned to addresses that indicate the degree to which an address is related to blacklisted addresses. Currently, a simple strategy that computes the shortest path distance from blacklisted addresses is used.

Since our system aims at building a scalable solution, we cannot assume that the transaction graph will fit in one node of a computer system. Therefore, the transaction graph is partitioned in parallel using ParMetis²⁰ (i.e., a Message Passing Interface-based parallel library that implements a variety of algorithms for partitioning unstructured graphs) during the initial graph construction phase and the parallel traversal operations are carried out on the partitioned graph.

2) Machine Learning Approach: As mentioned above, a graph traversal based approach can give us subgraphs that trace graph nodes (i.e., blockchain addresses) to blacklisted addresses and, for each address, a simple score can be calculated based on path lengths. Moreover, we also plan to investigate the effectiveness of other intelligent strategies that take into account other features such as the amounts of money received and spent, the times of transactions, licitness and illicitness of addresses (if available) and aggregated features based on the adjacent addresses that are a few hops away from each node.

²⁰ <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>

Since a scalable parallel graph analysis infrastructure that constructs transaction graphs and that operates on an HPC cluster is already built, parallel feature extractions can be performed fast. Thus, once subgraphs containing customer addresses from the massive transaction graph are extracted, Python Scikit-learn (Section 2.1) and PyTorch (Section 2.3) machine learning softwares will be used to further analyze the subgraphs. We will explore K-Means, Support Vector Machines [14], Naive Bayes [11], Logistic Regression, Random Forest [9, 27], Artificial Neural Networks (Multilayer Perceptron), and an ensemble method based on the majority voting of these mentioned classifiers. A comparison between the best performing classifiers will also be included. These classifiers are available in both of the aforementioned software packages.

4.9 Pilot 10 - Real-time cyber-security analytics on financial transactions' Big Data

The pilot aims at improving significantly the detection rate of malicious events (i.e., frauds attempts) and enabling the identification of security-related anomalies while they are occurring by the analysis in real-time of the financial transactions of a home and mobile banking system. This approach allows proactive and prompt interventions on potential security threats.

More specifically, the pilot will develop a tool based on ML techniques applied to real-time, financial transaction data-streams focused on adaptive detection for malicious transactions leveraging on established big-data analytics practices. The analysis of vast amounts of data will help to define relevant cyber-risk ratings metrics and allow us to implement adaptive security measures and controls, based on real cyber-security postures.

The final dataset will be a generated, realistic dataset that is consistent with the real data present in the data operations environment. The synthetic dataset will be created by Poste Italiane starting from randomly generated personal data with no correlation with real people.

The use case involves a lambda architecture approach (consisting of a batch and streaming workflow) that will be implemented by means of the adoption of ALIDA. In a nutshell, ALIDA is a microservice based platform, developed by ENG (Engineering), for composition, deployment, execution and monitoring of workflows of BDA services; it is entirely developed with open source technologies.

Machine Learning (ML) approaches

Techniques such as K-Means will first be used to group the different data points. Moreover, the data, before being processed by an unsupervised algorithm, will be properly prepared. Data preparation includes: transformation of characteristics from categorical to numerical, data normalization and standardization. A study of the identified clusters will then be carried out by Poste Italiane through data visualization and exploration tools. This step will include representation of data on different charts and tables containing statistical information (count, mean, standard deviations, etc.) of data grouped by cluster.

Once the dataset has been labelled, a supervised algorithm such as Random Forest [9] will be used to train a ML model which will then be used in a real-time context to label new incoming data. Depending on the result label, it will be possible to identify whether a given transaction is identifiable to a possible fraudulent action or not. The batch workflow can be scheduled in order

to produce an updated ML model based on the data collected, thus updating the model used for the streaming workflow.

The clustering step is used as input for a classification algorithm that can apply all the classification metrics such as Accuracy, Precision, Recall, F1-score, etc. In this way, the performance of the supervised method can then serve as a surrogate for the performance of the unsupervised learner.

For this pilot, the Apache Spark²¹ framework will be used to process large amounts of data. Apache Spark provides a set of supervised and unsupervised algorithms across its scalable ML library and support for data preparation. Other libraries could be considered, such as Scikit-Learn to perform data analysis and machine learning operations, and TensorFlow/Keras to implement artificial neural networks.

Spark supports different programming languages. Pyspark, the Python API written in Python to support Apache Spark, will be used for the implementation of use case tasks.

4.10 Pilot 11 - Personalized insurance products based on IoT connected vehicles

This pilot aims at improving the driving insurances' profiling by making use of connected vehicles' generated data while driving. More specifically, the pilot will generate 2 services:

- **Fraud detection:** This service will develop a Driver Profile model to search for differences in driving patterns in order to detect frauds on insurance use or vehicle thefts.
- **Pay as you Drive:** This service will process the connected vehicles' data to classify driver's performance in order to better adjust the price of their insurance policy.

For the pilot, two different machine learning approaches will be developed. The first, for the Fraud Detection service, will consist of a clustering algorithm, which allows differentiating multiple routes taken by the same car and to infer if a different driver is using the vehicle. Techniques such as K-Means and Mean-Shift will be used to group different data points.

For the second service, Pay as you Drive, classification algorithms will need to be used. For this problem, models such as K-Nearest Neighbours or Support Vector Machine (SVM) can be used, allowing for drivers to be classified in an efficient and powerful way.

The data used for these purposes will mainly come from connected vehicles provided by CTAG (Galician Automotive Technology Centre)²². These vehicles can provide real-time data while being on-road, such as the current speed, the steering angle or the instant acceleration. Additional to this data, weather information from the city of Vigo and its surroundings will be collected as well, in order to match the weather conditions to the driving style. CTAG will also be able to provide alerts info from the Vigo surroundings, in order to infer driving incidences and consider them for the driver's profile. To collect great amounts of data for analysis and training, the SUMO (Simulation of Urban MObility) simulation package [33] will be used to provide simulated routes in different cities.

For this pilot, various AI and data science libraries will be used.

²¹ <https://spark.apache.org/>

²² <https://ctag.com/en/>

- Scikit-learn: This library gives developers the ability to perform data analysis and simple Machine Learning operations (Section 2.1).
- NumPy: Provides functions to work on arrays and perform computations on them.
- Pandas: Allows performing data manipulations on the datasets.
- TensorFlow/Keras: High-level Artificial Intelligence library, provides methods to easily implement, train and serve Machine Learning models (Section 2.2).
- EASIER.AI: This pilot will use the aforementioned EASIER.AI platform (Section 3.1) to train and orchestrate its services.

4.11 Pilot 12 - Real-world data for novel health-insurance products

The objective of this pilot is to demonstrate how real-world data can be utilized by insurance companies to develop novel products for the benefit of the insurer and of the customers. To achieve this goal, Healthentia, the eclinical platform, will be used as a real-world data capturing infrastructure to collect data from IoT devices (activity trackers) in conjunction with users' reports. The data will be ingested into the pilot's testbed via INFINITECH's Data Collection component (see INFINITECH deliverable D5.13), to be anonymised and stored in the LeanXcale database.

The pilot will then be offering the following two services to its end-users, namely the health insurance companies:

- Risk assessment: The lifestyle of each client will be scored, allowing the end-users to offer personalised pricing for their insurance plan.
- Fraud detection: The behaviour of clients will be analysed to detect fraudulent usage of the activity trackers or untruthful replies to questionnaires.

For the pilot, different data sources will be used:

- Data coming from the users of the Healthentia app (~750 kB/user/week);
- synthetic data from the Innovation Sprint's RWD (Rear Wheel Drive) Simulator (~750 kB/user/week);
- insurance company data (minor volume of quasi-static data).

Machine Learning (ML) approaches

The methodology for providing the risk assessment service is straightforward. Classifiers will be trained to yield if the lifestyle of the client is improving or worsening. Since the pilot does not expect an abundance of data, Random Forest classifiers will most probably be used. On the other hand, the simulated data will allow training of Neural Network classifiers as well. Should the RWD Simulator data be found realistic enough, it might also be used for training the final classifier.

Fraud detection will be based on the clustering of behaviours for outlier detection. Both traditional approaches will be investigated (K-Means and Gaussian Mixture Models), but also deep learning ones using autoencoders.

The ML/DL libraries to be used to offer this functionality are Scikit-learn (Section 2.1) and TensorFlow/Keras (Section 2.2). The data and numerical processing libraries are Pandas and NumPy.

4.12 Pilot 13 - Alternative/automated insurance risk selection - product recommendation for Small and Medium Enterprises (SMEs)

The main objective of Pilot 13 is to implement a data analysis platform applying ML technologies to better predict the insurance needs of Small and Medium Enterprises (SMEs). In this context, the platform will generate a risk map of the SMEs based on their daily activities and will predict how the risk will vary over time. Therefore, the pilot will design and implement a service that effectively monitors the current risks of SMEs, as well as their risk variance in the future, in order to improve the control of the accident rate, the renewal of insurance policies and offer personalised insurance coverage.

To this end, real data will be leveraged that will be obtained from a variety of sources such as the web, social media profiles, official registers, opinion platforms, business directories, e-commerce platforms, and more. The dataset will be composed of data originating from 50000 European SMEs (i.e., structured and unstructured data in image format and text format). The estimated data volume is expected to be around 1 TB of data.

The innovation in the project is threefold: (i) the use of machine learning techniques to predict the needs of SMEs concerning their insurance, (ii) the use of alternative data, traditionally not considered by insurance companies, which come from open sources, and finally (iii) data with periodic extractions and not only at the time of contracting the product, as is currently the case

As already mentioned, the use of the data is one of the differentiating factors since the pilot will leverage data from open and non-traditional online sources such as a dataset on SMEs geolocation information and characteristics, a dataset on reviews and opinions about SMEs, etc. Doing this, the pilot owners manage to accelerate processes, improve efficiency and be able to offer products to the SME clients of the insurance companies according to their risk profile without any previous interaction with the company.

To calculate metrics (such as VaR [31]) and to conduct pre-trade analysis, the pilot will explore two different approaches. More precisely, the output data will be classified into two groups:

- Direct, raw data, whose utility is basic for data cleaning and autocompletion in the subscription processes and in the automation of the same. In this type of data techniques semantic analysis will be applied, tagging and search for coincidences and chain similarity algorithm
- Processed data based on algorithms designed based on ML to determine risk levels, propensity to purchase such insurance, and insurance package recommendations To do this we will use ML approaches:

The ML techniques planned to be used and already used by Wenalyze are the following ones:

Supervised learning: Linear regression, namely a statistical technique used to study the relationship between variables. A small percentage of the data is extracted for subsequent verification of the results, to which the relationship obtained will be applied.

Unsupervised learning: Clustering or grouping objects by similarity, in groups or sets so that members of the same group have similar characteristics.

Reinforcement Learning: This is a subfield of machine learning that teaches an agent how to choose an action from his or her space of action, within a particular environment, to maximise rewards over time.

It is worth noticing that the exact ML algorithms that will be explored are still under investigation, as the initial pilot prototype leverages a pure statistical/scientific computing approach.

4.13 Pilot 14 - Big Data and IoT for the agricultural insurance industry

The objective of the pilot is to define, structure and test specific services for the agricultural insurance sector in order to better protect agricultural assets by evaluating risks in a data-driven way and to improve the business process of agricultural insurers and clients (farmers). The services tested will be: (i) a mapping of risks related to agriculture in predefined markets; (ii) the prediction and assessment of weather and climate risk probability; and (iii) a damage assessment calculator for insurance companies.

In particular, the pilot aims at identifying areas where crop productivity and catastrophe probability is high based on intelligent risk mapping; at creating additional datasets with high predictive value for improving underwriting of agricultural risks concerning weather and climate risk probability, and finally at improving the damage assessment and claims handling procedures for the insurance industry to increase the efficiency of calculating indemnity pay-outs.

Specifically, the formulation of risk metrics and damage calculation are developed as two different services. One is focused on drought and the consequent prediction of the afflicted crop production (barley, wheat, and cereals), while the second is aimed at estimating the damage caused by hailstorms on wheat, maize, and soybean production.

These services both leverage on an extensive use of Earth Observation (EO) data. More in detail,

- The “drought service” makes use of the EO zonal statistics leveraging on the Normalized Difference Vegetation Index (NDVI) and its anomalies.
- The “hailstorm service” works with Change Detection DPI (specifically, the difference percentage) of EO features. It involves both biophysical parameters (such as the Fraction of Absorbed Photosynthetically Active Radiation (FAPAR), the Leaf Area Index (LAI), and the Fractional Vegetation Cover (FVC)) as well as multiple vegetation indices (such as NDVI, Green NDVI (GNDVI), Red Edge Inflection Point (REIP) index, Modified Chlorophyll Absorption Ratio Index (MCARI)).

Albeit the two services come as two different tasks and, potentially, as separate tools, the ML pipeline, adopted and implemented within the scopes of the pilot, is shared.

The ML framework consists of five principal steps. The first, a preliminary analysis of the data which furnish a first understanding of the relations in play. Second, a detailed analysis focused on outlier events. Specifically, this step will make use of Principal Component Analysis (PCA) and Leave One Out Cross Validation (LOOCV) as well as residual and influence plots. The third step consists of a feature selection in preparation for the training of Support Vector Machines models. This step is performed on different classes of features: all variables, only variables which are highly correlated with the predictor, highly and mildly correlated variables, selection by filter. The fourth step is dedicated to the training of the model. The models are chosen within the SVM and Random Forest (RF) models. The training of each model the dataset is partitioned into training,

validation, and prediction subsets. On these subsets the models are evaluated through LOOCV (due to the limited number of samples).

In particular, the evaluation part is performed in the fifth step of the ML pipeline. Regression models use as performance metrics the Root Mean Square Error of Calibration (RMSEC), Coefficient of Determination for Calibration (R^2_{Cali}), Root Mean Square Error of Validation (RMSEV), Coefficient of Determination for Validation (R^2_{Vali}), Residual Prediction Deviation (RPD), Residual Prediction Interquartile Range (RPIQ). Classification models use as performance metrics Precision, Recall, and F1 score for in class evaluation and Overall Accuracy, and Choen's Kappa for an overall evaluation.

The pilot takes advantage of the Orfeo Toolbox and several ML libraries and tools within the R environment. In particular, the main R-libraries used are (among others): readxl, sj labelled, magrittr, dplyr, Hmisc, xtable, corrplot, ggplot2, reshape2, caret, e1071, Metrics, MLmetrics.

4.14 Pilot 15 - Open Inter-Banking

The main objective of Pilot 15 is to promote the development of a common use case, which will involve different banks through a shared research approach in order to develop, integrate and deploy a data-intensive system to extract key concepts from internal operational documents and to create a common business glossary.

The pilot will use a set of documents from different banks and the solution will be based on ML and Natural Language Processing (NLP) approaches. The task will also involve resources such as controlled vocabularies, ontologies and process modelling databases developed and shared across banks federated through ABI Lab across years. These models make available both concept descriptions as well as textual definitions. NLP here will be applied to enable the learning of actionable representations of texts and definitions (i.e., vector or tensor based encodings) in order to support textual inferences. Examples of such semantic inferences are text classification, for metadata creation, compliance checking based on text contents as well as clustering for the discovery of banking specific phenomena and trends in the texts.

The ML task will be applied to the language processing functions in order to optimize/specialize them to the banking texts as well as to document-driven decision making. It thus will consist of:

- Supervised classification and regression tasks, where the target classification expresses the semantics related to the detection of concepts of interest in the banking prose, or to the assessment of some complex properties in documents or definitions (e.g., association between ontological concepts and some text fragments). Main algorithms adopted here are neural networks of the recurrent family, such as LSTM networks [30], transformer based approaches [43] as well as Support Vector Machines [13] and, in particular, kernel machines [14].
- Regression will be also targeted whereas properties correspond to quantitative variables (e.g., risk levels, or degrees of compliance).
- Unsupervised learning in the form of large scale text processing will be devoted to optimize linguistic resources in the so called pre-training stages. We plan here to apply distributional semantic models based onto the recent progresses in the encoding-decoding embedding architecture known as BERT [15], where transformer-based algorithms are used to embed linguistic knowledge in the basic

resources (lexicons, word form modeling, language modeling and text inference) directly reusable as early parametrization of supervised fine-tuning learning stages.

According to the above perspective, Pilot 15 is planning to use the following ML algorithms, evaluation metrics, and ML libraries and tools:

- Algorithms: Support Vector Machines [14], LSTMs [30], text-driven Transformers [43], and BERT [15].
- Evaluation metrics will range from the Precision and Recall scores, typical of Information Retrieval tasks, to more textual oriented measures such as BLEU (known as bilingual evaluation understudy) or METEOR (Metric for Evaluation of Translation with Explicit ORDERing) metrics that act on specific text structures (sentences as well as paragraphs) and are oriented to capture rewriting quality (such as in Machine Translation)
- Machine Learning libraries and tools. The pilot will adopt solutions that strictly depend on realistic operational scenarios, but will also explore applicability issues in the area of semantic business analytics for banking. In this perspective traditional (mainstream) frameworks will be reused such as the PyTorch implementation of neural machines (BERT, Transformers, LSTMs) as a rather reusable learning framework. However, the impact of simpler frameworks, more suitable for deployment of large scale flexible and more maintainable solutions will be studied. We will rely on KeLP²³, a learning framework specifically designed for large scale kernel based text textual inference Support Vector Machines, as it provides a very large library of Java-based components for off-line, and on-line learning, and it includes specific support in the treatment of linguistic data. More technical details on this framework are discussed in Filice et al. [20].

Finally, the pilot will adopt a standard workflow comprising of the following phases:

1. Design and analysis of task-specific AI pipelines;
2. Design of the experimental set-up with emphasis on individual tasks, and their dependencies;
3. Selection and preparation of reliable and clean data batches for the individual tasks;
4. Large scale experimentation of learning approaches for the optimization of the different individual tasks;
5. Development and deployment of individual services within an integrated solution.

²³ <http://www.kelp-ml.org/>

5 Conclusions

This deliverable has been developed around the three main objectives that have been outlined in Section 1.1.

As the first objective, this deliverable has collected and presented the overview of state-of-the-art frameworks and libraries of AI algorithms and technologies. Existing libraries, containing pre-developed and tested algorithms, have been discussed in Section 2 with a particular interest in presenting the more stable and regularly supported ones. In particular, the importance of relying on continuously updated and supported libraries has been highlighted as a crucial step to ensure algorithmic stability and reliability, as well as compatibility with most recent data streams and APIs.

The second objective that has been pursued is the showcase of innovative ML/DL solutions that have been developed, also in collaboration with the pilots.

Section 3 was devoted to present some general-purpose cutting-edge machine learning solutions applied to financial data. Five solutions were illustrated as general finance case-studies and as candidate solutions for potential task specific challenges. Some of these solutions are being adopted in the pilots of the INFINITECH project.

As a final objective, this deliverable includes an extensive survey of the ML/DL needs and solutions planned by the pilots, which represent the solid core and powerful vision of Task 5.4. Each pilot contributing to Task 5.4 was presented in Section 4, where a collection of applications to real-world fintech problems is being addressed. Section 4 is indeed the place where top-down solutions are potentially tested on a broader range of case-studies and the place where bottom-up ML solutions emerge and rise from specific applications. From this comprehensive overview it emerged that a variety of tasks, such as personalization of financial products, risk assessment and fraud detection, and automatic processing of large amounts of documents, are addressed by the pilots with ML/DL. Moreover, different ML/DL approaches are adopted, such as k-means clustering, support vector machines, random forests, and neural networks, to name a few. All these tasks can be successfully tackled with the libraries discussed in Section 2.

The final aim of this series of deliverables is to collect successful implementations, document their workflows, and provide a ready-to-use application environment as a powerful tool for innovative financial solutions. In the second deliverable of this Task, we will propose a more concise and more technical description of the results obtained by the pilots bottom-up implementations. In the third deliverable we will be presenting an accessible and open coded version of these implementations with ready to use synthetic examples.

Appendix A: Literature

- [1] Acerbi, C., & Tasche, D. (2002). Expected Shortfall: a natural coherent alternative to Value at Risk. *Economic notes*, 31(2), 379-388. *Economic notes*
- [2] Allen, D. M. (2012). The Relationship Between Variable Selection and Data Augmentation and a Method for Prediction. *Technometrics*, 16(1), 125-127. doi/abs/10.1080/00401706.1974.10489157
- [3] Alpaydin, E. (2014). *Introduction to Machine Learning* (3rd ed.). MIT Press.
- [4] Altman, N.S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3), 175-185
- [5] Barros, R., & Santos, S. (2018). A large-scale comparison of concept drift detectors. *Information Sciences*, 451-452, 348-370
- [6] Bhattacharya, A. (2020) *Effective Approaches for Time Series Anomaly Detection*. towardsdatascience.com.
- [7] Bergstra, J., & Bengio, Y. (2012) Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(1), 281-305.
- [8] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining, SDM '07*, SIAM, pp. 443–448
- [9] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32. Springer Link. doi.org/10.1023/A:1010933404324
- [10] Buchanan, B. G. (2019). *Artificial Intelligence in finance*. The Alan Turing Institute, 00(00), 3-49. 0.5281/zenodo.2626454
- [11] Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ICML '06: Proceedings of the 23rd International Conference on Machine learning*, 161-168. ACM Digital Library. doi.org/10.1145/1143844.1143865
- [12] Chawla, N. V., Herrera, F., Garcia, S., & Fernandez, A. (2018) SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61, 863--905.
- [13] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297. Springer. doi.org/10.1007/BF00994018
- [14] Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press. doi.org/10.1017/CBO9780511801389
- [15] Devlin, J., Chang, M.-W., Lee K., Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4171--4186.
- [16] Edwards, L., & Veale, M. (2017). Slave to the Algorithm? Why a 'Right to an Explanation' Is Probably Not the Remedy You Are Looking For. *Duke Law & Technology Review*, 16(18), 1-67. SSRN. dx.doi.org/10.2139/ssrn.2972855
- [17] Elliott, T. (2019). *The State of the Octoverse: machine learning*. GitHub. Retrieved 11 10, 2020, from <https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/>

- [18] Exxact Corporation. (2020). PyTorch vs TensorFlow in 2020: What You Should Know About These Frameworks. Exxact.
<https://blog.exxactcorp.com/pytorch-vs-tensorflow-in-2020-what-you-should-know-about-the-se-frameworks/>
- [19] Ferrari Dacrema, M., Cremonesi, P., & Jannach, D. (2019). Are we really making much progress? A worrying analysis of recent neural recommendation approaches. RecSys '19: Proceedings of the 13th ACM Conference on Recommender Systems, 101-109. ACM Digital Library. <https://doi.org/10.1145/3298689.3347058>
- [20] Filice, S., Castellucci, G., Da San Martino, G., Moschitti, A., Croce, D., & Basili, R. (2018). KELP: a Kernel-based Learning Platform. *Journal of Machine Learning Research* 18(191):1-5
- [21] Financial Stability Board (2017) , Artificial intelligence and machine learning in financial services, www.fsb.org/wp-content/uploads/PO111117.pdf
- [22] Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32, 41-62
- [23] Fionda, V., & Pirró, G. (2019). triple2Vec: Learning Triple Embeddings from Knowledge Graphs. arxiv Preprint. arXiv. <https://arxiv.org/abs/1905.11691>
- [24] Fortuna, B., Rupnik, J., Brank, J., Fortuna, C., Jovanoski, V., Karlovcec, M., Kazic, B., Kenda, K., Leban, G., Muhic, A., Novak, B., Novlian, J., Papler, M., Rei, L., Sovdat, B., Stopar, L., Grobelnik, M., & Mladenic, D. (2014). QMiner: Data Analytics Platform for Processing Streams of Structured and Unstructured Data. In Proceedings of the Software Engineering for Machine Learning Workshop at Neural Information Processing Systems (NIPS 2014)
- [25] Gareth, J., Witten, D., Hastie, T., & Robert Tibshirani. (2013). An Introduction to Statistical Learning. Springer-Verlag New York. 10.1007/978-1-4614-7138-7
- [26] Goldberger, J., Hinton, G., Roweis, S., & Salakhutdinov, R. R. (2004). Neighbourhood Components Analysis. *Advances in neural information processing systems*, 17(1), 513-520. <https://proceedings.neurips.cc/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf>
- [27] He, H. (2019, 10 10). The State of Machine Learning Frameworks in 2019. The Gradient.
<https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/>
- [28] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural Collaborative Filtering. WWW '17: Proceedings of the 26th International Conference on World Wide Web, 173-182. ACM Digital Library. <https://doi.org/10.1145/3038912.3052569>
- [29] Ho, T. K. (1995). Random decision forests (Proceedings of 3rd International Conference on Document Analysis and Recognition ed.). IEEE. doi.org/10.1109/ICDAR.1995.598994
- [30] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8). ACM Digital Library. doi.org/10.1162/neco.1997.9.8.1735
- [31] Holton, G. A. (2014). Value-at-Risk: Theory and Practice (2nd ed.). London : Academic Press.
- [32] Liu, F.T., Ting, K.M., & Zhou, Z.-H. (2008). Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422
- [33] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y. P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., & Wiessner, E. (2018). Microscopic Traffic Simulation using

- SUMO. 21st International Conference on Intelligent Transportation Systems (ITSC), 2575-2582. IEEE Xplore. /doi.org/10.1109/ITSC.2018.8569938
- [34] Muselli, M. (2006). Springer Berlin Heidelberg. In *Neural Nets* (pp. 23-30). Springer Berlin Heidelberg. doi.org/10.1007/11731177_4
- [35] Pelleg, D., & Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 5(1), 277-281. ACM Digital Library. doi.org/10.1145/312129.312248
- [36] Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. *UAI '09: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 452-461. ACM Digital Library. doi/10.5555/1795114.1795167
- [37] Rendle, S., Krichene, W., Zhang, L., & Anderson, J. (2020). Neural Collaborative Filtering vs. Matrix Factorization Revisited. *RecSys '20: Fourteenth ACM Conference on Recommender Systems*, 240-248. ACM DigitalLibrary. https://doi.org/10.1145/3383313.3412488
- [38] Rendle, S., Zhang, L., & Koren, Y. (2019). On the difficulty of evaluating baselines: A study on recommender systems. *arXiv Preprint, Information Retrieval*. arXiv. https://arxiv.org/abs/1905.01395
- [39] Rokach, L., & Maimon, O. (2005). Clustering methods. *Data mining and knowledge discovery handbook*. Springer US, 321-352.
- [40] Sra, S., & Dhillon, I. (2005). Generalized Nonnegative Matrix Approximations with Bregman Divergences. *Advances in Neural Information Processing Systems*, 18(1), 283-290.
- [41] Stopar, L., Skraba, P., Grobelnik, M., & Mladenic, D. (2019). StreamStory: Exploring multivariate time series on multiple scales. *IEEE Transactions on Visualization and Computer Graphics*, 25(4), pp. 1788-1802
- [42] Strumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3), 647-665. Springer Link. doi.org/10.1007/s10115-013-0679-x
- [43] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, 30, pp. 5998--6008.
- [44] Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural Graph Collaborative Filtering. *SIGIR'19: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 165-174. ACM digital Library. https://doi.org/10.1145/3331184.3331267
- [45] Yang, L., Bagdasaryan, E., Gruenstein, J., Hsieh, C.-K., & Estrin, D. (2018). Openrec: A modular framework for extensible and adaptable recommendation algorithms. *WSDM '18: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 664-672. ACM Digital Library. https://doi.org/10.1145/3159652.3159681
- [46] Zeng, S., Tay, Y., Yao, L., Wu, B., & Sun, A. (2019). Deeprec: An open-source toolkit for deep learning based recommendation. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 6581-6583. *dblp computer science bibliography*. https://doi.org/10.24963/ijcai.2019/963