# Risk Prediction as a Service: a DSS architecture promoting interoperability and collaboration

Stefano Mariani[†] and Franco Zambonelli[†] – Akos Tenyi[*§], Isaac Cano[*§] and Josep Roca[*§]

[†]*Department of Sciences and Methods for Engineering*
*Università di Modena e Reggio Emilia, Reggio Emilia, Italy*
*Email: {stefano.mariani, franco.zambonelli}@unimore.it*

[*]*Institut d'Investigacions Biomèdiques August Pi i Sunyer, Hospital Clinic de Barcelona*
*Universitat de Barcelona, Barcelona, Spain*
[§]*Center for Biomedical Network Research in Respiratory Diseases*
*Madrid, Spain*
*Email: tenyi.akos@gmail.com, {ISCANO, JROCA}@clinic.cat*

***Abstract*—Clinical research and practice are rapidly changing mostly due to Information and Communication Technology, especially, as Machine Learning (ML) offers great potential for predictive and personalised medicine. Nevertheless, barriers are still existing for widespread adoption of ML tools, as highlighted by studies from the European Union. In this paper, we propose an architecture for a Decision Support System assisting clinicians in assessing health risk of patients by delivering "Risk Prediction as a Service". By leveraging standard web technologies as well as the PMML and PFA formats for exchange of trained models, we achieve ubiquitous access to predictions, ease of deployment, and seamless interoperability, while promoting collaboration.**

***Keywords*-risk prediction; PMML; PFA; machine learning; decision support system; CONNECARE**

## I. Introduction

The landscape of both clinical research and clinical practice is rapidly and substantially changing mostly due to the fundamental role played by Information and Communication Technology (ICT) in *(a)* facilitating access to and exploitation of an enormous amount of patient-related information, and *(b)* providing increasingly reliable and precise computational and Machine Learning (ML) algorithms for predictive modelling [1]–[3]. Indeed, in this novel scenario, the paradigm shift towards predictive and personalised medicine is triggering a whole new set of *computational requirements* in terms of predictive modelling for decision making.

The need for advancing computational tools, supporting for instance risk assessment for screening and stratification, arises when recognising that, while rule-based systems for clinical management are accepted in the current clinical practice, exploitation of predictive modelling tools for clinical decision support is still far from reaching maturity. In fact, a European Union study on Big Data in Public Health, Telemedicine, and Healthcare [4] highlighted many opportunities as well as *barriers* for improving current clinical practice. Among the many, the following areas of improvement are those explicitly targeted by the work presented in this paper:

- **Standards and protocols.** As we adopt the Predictive Model Markup Language and the Portable Format for Analytics [5] standard formats specifically conceived for *seamless exchange of ML models*—ultimately enabling collaboration.
- **Technological development.** As we leverage state-of-art "Machine Learning as a Service" approaches grounded in the most modern technological stacks and best practices stemming from the work in service-oriented architectures—to enable *ubiquitous access and systems interoperability*.
- **Data analytics.** As we promote collaboration between data scientists regardless of the programming language and ML software framework preferred, and among data scientists and clinicians—ultimately enabling *separation of concerns* in integrated care practice.

Accordingly, in the context of the CONNECARE[1] European project for personalised integrated care, we designed a Decision Support System (DSS) for health risk prediction whose main goal is to provide *ubiquitous access* to and *seamless deployment* of prediction models, as well as to promote *collaboration* and *interoperability* between data science teams and, possibly, clinicians.

## II. Background

*Health risk assessment* is a relevant component in the strategic adoption of integrated care because of its impact

This work has been supported by the CONNECARE (Personalised Connected Care for Complex Chronic Patients) project (EU H2020-RIA) under contract no. 689802.

[1]http://www.connecare.eu/

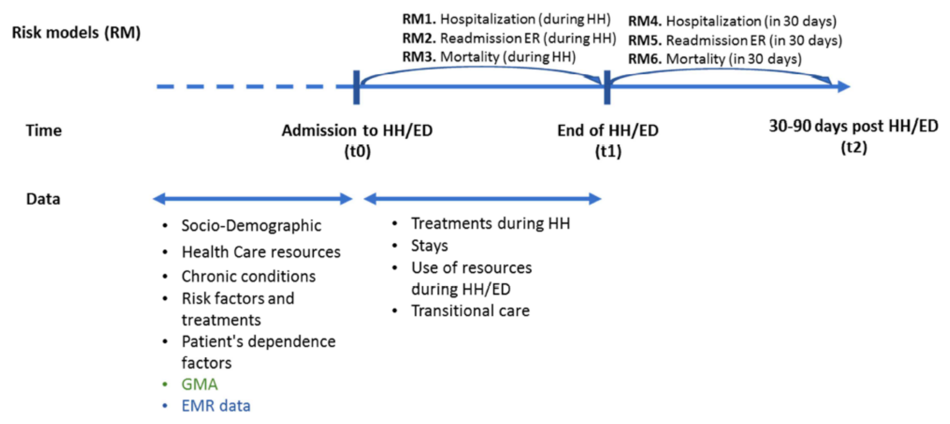
Risk models (RM)
RM1. Hospitalization (during HH)
RM2. Readmission ER (during HH)
RM3. Mortality (during HH)
RM4. Hospitalization (in 30 days)
RM5. Readmission ER (in 30 days)
RM6. Mortality (in 30 days)
Time
Admission to HH/ED (t0)
End of HH/ED (t1)
30-90 days post HH/ED (t2)
Data
Socio-Demographic
Health Care resources
Chronic conditions
Risk factors and treatments
Patient's dependence factors
GMA
EMR data
Treatments during HH
Stays
Use of resources during HH/ED
Transitional care

Figure 1. Development of predictive modelling within CONNECARE ("ER" stands for "Emergency cases" [of early readmission]).

on the design of healthcare services ("population-based" health risk assessment), as well as for enhanced clinical management ("patient-based" health risk assessment). Patient management exclusively based on clinical criteria (professional training, knowledge, instinct, and experience) or combined with rule-based clinical management (thresholds for certain parameters defining pre-established decision criteria) constitute most of current practice. In contrast, the use of predictive modelling tools for clinical decision support (predictive modelling establishing relationships between sets of variables and outcomes generated using statistical or ML tools) is still in its infancy, despite seemingly a natural step towards personalisation of care.

In CONNECARE, one of the objectives is to develop ICT tools enabling *creation, exchange, and deployment of predictive risk models* that can help, for instance, in service selection of home hospitalisation for complex chronic patients. In particular, current focus is on validating predictive modelling for Home Hospitalisation (HH) / Early Discharge (ED)—as summarised in Figure 1:

- during HH / ED (t0), to identify *(a)* risk of early readmission after hospital discharge and *(b)* mortality, to stratify patients in order to optimise care (Risk Models 1-3)
- after HH / ED (t1), to identify risk of *(a)* early readmission after hospital discharge and *(b)* mortality, to stratify patients for transitional care purposes (Risk Models 4-6)

The models built in such a way should be made available for inspection and usage across clinical sites, and regardless of the technological stack exploited for their creation.

## A. MLaaS

In the healthcare domain there is growing momentum around the "*X*-as-a-Service" paradigm, especially in the form of "Machine Learning as a Service" (MLaaS) [6], already applied, for instance, to readmission score prediction [7]. Nevertheless, MLaaS as is may be unsuitable to unravel its full potential, mostly due to healthcare peculiar restrictions regarding *data disclosure*: even data anonymised in compliance with regulations is not completely immune to malicious attacks [8].

Many well established platforms for MLaaS are available as provided by big players such as IBM, Google, Microsoft, and Amazon, and they are extensively exploited by both the industry and the academia for good reasons—among the many: performance, reliability, comprehensiveness of ML models supported. Nevertheless, as all Cloud-based approaches they may suffer of issues related to privacy and disclosure of data, particularly relevant for the healthcare domain. Here, in fact, clinicians and data scientists may be less keen to share sensible training data with third-parties, even if they are well recognised and worth of trust. Also, not every data science team nor clinic can afford them, either from a cost perspective or due to shortage of the required technical skills—working with RStudio is quite different than interacting with a Cloud-hosted MLaaS platform.

Besides training, MLaaS can also be used for effective *deployment of models* to production environments once they have been trained, although this is a relatively less explored practice that only recently started getting attention—see, for instance, IBM MAX [9]. Here, many conceptual and technical issues do not have a widely acknowledged solution, yet. For instance, models versioning, fault-tolerance and availability, *heterogeneity of languages and ML frameworks* used, and *diversity of skillsets and goals* between data scientists, clinicians, and managers are just a few to mention.

## B. PMML and PFA

The Predictive Model Markup Language (PMML) and Portable Format for Analytics (PFA) standards have been precisely conceived with the goal of promoting *seamless exchange and deployment* of ML models *(a)* without bringing along the burden of computationally expensive training

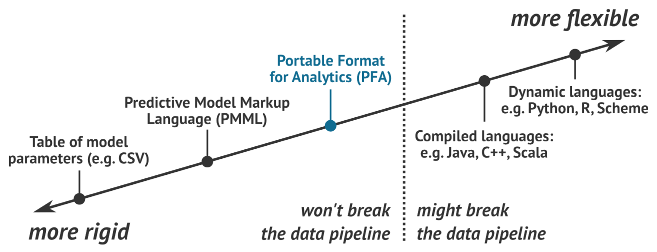
more flexible
Portable Format for Analytics (PFA)
Predictive Model Markup Language (PMML)
Table of model parameters (e.g. CSV)
Dynamic languages: e.g. Python, R, Scheme
Compiled languages: e.g. Java, C++, Scala
more rigid
won't break the data pipeline
might break the data pipeline

Figure 2. The data pipeline in ML (image taken from http://dmg.org/pfa/docs/motivation/).

sessions and data access, *(b)* without breaking the data pipeline (see Figure 2), and *(c) across ML / statistical frameworks and languages*: models are built once, and then "run" everywhere.

Both PMML and PFA are standard representation formats, the former based on XML, the latter on JSON, meant to support an operational and machine-readable description of any kind of (already trained) prediction model, such as regression models, classifiers, clustering methods, etc.

The idea behind these standards is hence perfectly aligned with the goals defined in the introduction:

- enable independent development / deployment of predictive models
- ensure flexibility of the production environment to easily integrate novel models
- allow for the development of models in different programming languages and frameworks, such as `caret` for R and `scikit-learn` for Python, regardless of the language used in the production environment—for instance, as in our case, Java

PMML and PFA are currently implemented in a handful of open source software packages readily available for usage in the most common programming languages. For instance:

- **jpmml-r** is a Java library for converting models saved in R native format to PMML
- **Aurelius** is a R package for converting models saved in R native format to PFA
- **jpmml-sklearn** is a Java library for converting models saved in Python native format to PMML
- **Titus** is a Python package for converting models saved in Python native format to PFA
- **jpmml-evaluator** is a Java library for executing PMML models
- **Hadrian** is a Java library for executing PFA models

Nevertheless, any ML framework usually includes programming constructs to create brand new models, not only to re-use existing, state-of-art ones, thus there will always be the possibility that a given model is not (yet) supported by the aforementioned libraries and packages: for instance, jpmml-r is limited to the R packages listed in its homepage [10].

## III. The CONNECARE solution

Within the CONNECARE project three are the clinical sites to consider for deployment of ICT solutions supporting and promoting integrated care, hence, the relevance of providing a flexible DSS architecture allowing for seamless integration with and greater interoperability across heterogeneous technological environments is well recognised.

Accordingly, our "Risk prediction as a Service" solution has been conceived and designed with three precise motivating use cases in mind—depicted in Figure 3.

*Use case A:* A handful of data science teams want to join efforts for advancing the state of art in creating ML models for clinical risk assessment and prediction. They are geographically distributed, hence collaboration should happen remotely, over the network. They belong to different organisations, thus sharing data for training models would require a lot of paperwork and may even be forbidden by respective administrations. They also have different technical preferences and skills about the programming language and ML framework of choice: for instance, some prefer working with Python and its scikit-learn package, while others are well trained with the R environment and its caret package. Under these premises, their only hope is to locally build prediction models, then share them in a technology-agnostic format allowing seamless integration and deployment with whatever technological environment, while retaining descriptive information such as performance measures and input (training) data schema.

*Use case B:* The clinical staff of a healthcare organisation wants to experiment with predictive risk assessment for its first time, as a feasibility analysis whose success may pave the way towards rolling out a local trial. As such, on the one hand it lacks the necessary technical resources and skills, while on the other would like to avoid investments in new staff or MLaaS solutions without a guaranteed successful outcome. A solution could be to collaborate with another organisation developing prediction models so as to to get those already trained models and apply them to the data available locally. Once a suitable model with desired performance is found, further investments can be considered.

*Use case C:* Two healthcare organisations would like to asses transferability of the prediction models they built with the assistance of a data science team each. Nevertheless, the lack of reciprocal trust hinders data sharing agreements, and the difference in technical skills is an obstacle for interoperability and deployment of the models developed in one site to the other. A solution letting the two organisations exchange not training data and operational instructions about how to re-build the ML pipeline, but the already built prediction models in a data format suitable to be used in whatever programming language and ML framework would do the trick.

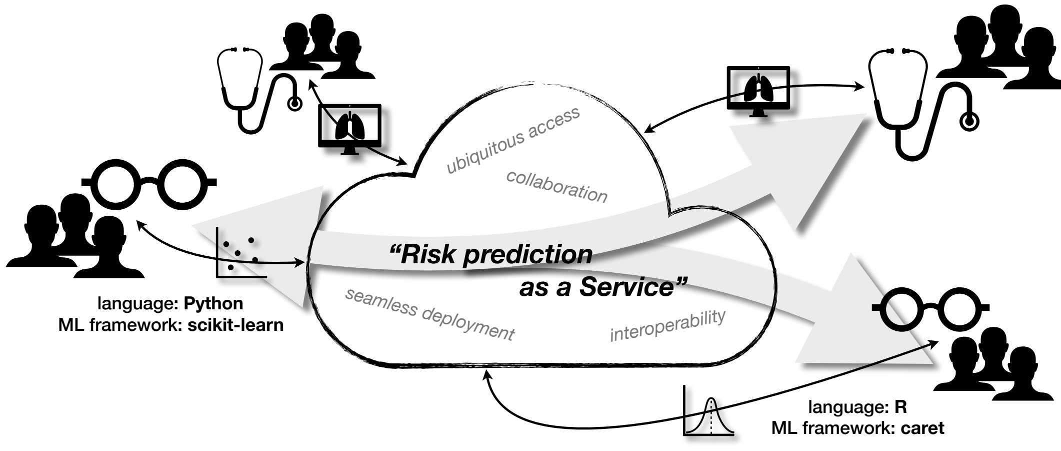

Figure 3. Targeted use cases: interoperability between ML frameworks and seamless deployment.

### A. Functionalities

The DSS works in two modalities, each with its own set of functionalities and a few common ones meant to fully support the aforementioned use cases.

*Plugin mode:* The DSS is able to absorb already trained risk prediction models and apply them on new data samples. In particular, the operations provided to either data scientists or clinicians are:

- *upload model* to plug into the DSS service a new risk prediction model anytime, from anywhere—through a single operation (file upload), to be quick and easy
- *list models* to get a list of the currently available models with information on their purpose (e.g. 30 days mortality risk), health indicators exploited for the prediction (e.g. age, BMI), and possibly metadata regarding their nature (e.g. random forest) and applicability (e.g. schema of input data required, prediction target)
- *inspect model* to observe inner parameters (e.g. equation of a regression line, precision, recall) and metadata (date of latest update, provider of the model, reference publication if available)
- *download model* to get any of the available models anytime, from anywhere, in either native format or PMML / PFA—again, through a single operation
- *delete model* to remove a model from the DSS. It is worth noting that safety measures against accidental or malicious attempts to delete models are in place (e.g. check if model has been last used long ago, ask consensus to model provider)
- *apply model* to apply whichever model available to whatever data sample compatible, getting as result its predictions, anytime, from anywhere

*Learning mode:* The DSS is also able to create its own risk prediction models as well as to re-train them. In particular, the operations provided are:

- *train / test model* to (re)train or test any of the models supported by the DSS[2] with data contextually supplied (through file upload)
- *save model* to "freeze" a model in time and save it for later usage. This means that the model will be actually replicated in two instances: one copy would not be further trained with (possibly) incoming data streams, the other one will continue online training (see "set online learning" common functionality)
- *discard model* to stop training a model and delete it

*Common functionalities:* Regardless of the mode of operation, the following settings are available:

- *set local / global* to restrict or not scope and applicability of a model:
  - local models can be trained with and applied to data coming from the same provider, solely
  - global models can be trained with any compatible data, regardless of its provenance
- *set online* to restrict or not its learning capabilities:
  - online models continues to be trained on the data samples subject of predictions
  - offline models are "frozen" as soon as the training process terminates

### B. Non-functional properties

The leading contribution we intend to deliver is not as much in the functionalities described above, which although comprehensive are rather standard, as in the set of non-functional properties that the DSS promotes.

*Trust:* The DSS promotes trust in the prediction models it makes available by two means: *(a)* models are *transparently* available for inspection (see "inspection" functionality in previous section), hence crucial information such as provider organisation, schema of data used for training, and performance measures are made observable by anyone, anytime,

[2]See "Learning Service" in section "Architecture".

from anywhere; *(b)* the plugin operation mode actually promotes a workflow in which data scientists build a model, clinicians (locally) validate them, and then – hence, only when they are trusted – upload it to the DSS system which makes it available at a large scale.

*Language / framework orthogonality:* The DSS seamlessly operates with models built in any language and with any ML / statistical framework. Many of the most widely established software frameworks represent prediction models in a custom format, often different also from framework to framework even if the programming language is the same. To enable technical interoperability and collaboration among data science teams, the DSS *automatically* translates models to/from PMML / PFA, as described in next section.

*Technical interoperability:* The DSS follows technical standards to be seamlessly available to any software client as RESTful web service, regardless of its implementation language. For instance, XML / JSON data formats and REST APIs guarantee interoperability with the vast majority of programming languages.

*Accessibility:* The DSS functionalities are meant to be accessible from clinicians' workstations, and seamlessly integrated with the case management systems there available—through the service-oriented RESTful layer. It is worth emphasising that the DSS is meant to be a "behind the scenes" service accessible (visible) through the user interface of the system that users are used to work with.

*Transferability:* Factors such as *(a)* license binding constraints, *(b)* proprietary software; *(c)* lack of availability for inspection, and *(d)* rigidity of computational algorithms are limiting factors for transferability of most of the existing population-based risk assessment tools. The DSS relies on open-source software with adequate licensing *(a,b)*, is open to run-time inspection of working parameters *(c)*, and is flexible enough to allow for both design-time (extensibility) and run-time (configurability) modification *(d)*.

*Privacy and security:* Issues related to privacy, security of data transfer, as well as risks associated with decisions are traditionally brought along by the very essence of the healthcare domain. It is thus of foremost importance that any DSS is developed with the necessary privacy-preserving and security-compliant methodologies and technologies. With respect to this, it is worth to highlight that, once again, the plugin operation mode actually promotes privacy and security by, for instance, allowing data scientists to not disclose the datasets used for training, hence avoiding altogether the top privacy and security concern[3].

## C. Architecture

The architecture of the DSS (depicted in Figure 4) features five logical modules interacting to deliver the functionalities and non-functional properties described above.

[3]Nevertheless, it is worth to remark that even models should be carefully exposed to prevent data leakage [8].

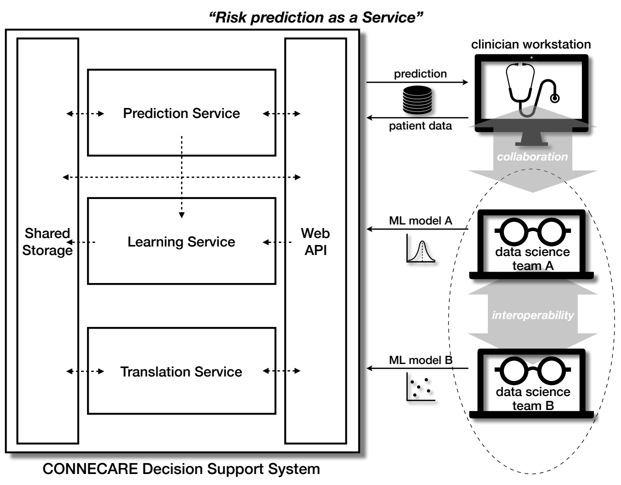

Figure 4. The DSS architecture.

*Web API:* Provides the RESTful endpoints to contact for exploiting the DSS functionalities. The module relies on third-party libraries providing all the building blocks for implementing secure, reliable, interoperable, and efficient RESTful web services—namely Jersey [11] and Jackson [12]. The module can:

- receive requests for upload, download, etc. of prediction models and training / testing data by the data scientists, possibly triggering the Translation Service and the Shared Storage layer
- receive requests for delivering predictions from the clinician, which are dispatched to the right Prediction Service, in turn interacting with the Shared Storage layer to retrieve models and (possibly) with the Learning Service for (re)training models

*Translation Service:* Provides translations between representation formats of prediction models. For instance, it is in charge of the *automatic* translation to PMML and PFA of models uploaded in R or Python formats. The module can:

- receive requests for translation between models representation format by the Web API module
- fetch models from the Shared Storage layer
- write translated models back to the Shared Storage

The model exploits jpmml-r for conversion from R to PMML, Aurelius for conversion from R to PFA, jpmml-sklearn for conversion from Python to PMML, and Titus for conversion from Python to PFA.

*Prediction Service:* Computes predictions based on the models fed with the supplied input data. The module can receive requests for making predictions from the Web API module, which triggers fetching the chosen model from the Shared Storage layer and (possibly) interacting with the Learning Service for (re)training models. The module exploits jpmml-evaluator and Hadrian for, respectively, making predictions out of PMML and PFA models.

*Learning Service:* This module is responsible for online learning, thus, for enabling the DSS to build its own risk prediction models. Based on the DSS configuration, this module may get feedback from the Prediction Service module (e.g. using predictions for training) or not. The model can:

- receive requests to perform training / testing by the Web API module, triggered by the data scientists
- receive requests to perform (re)training by the Prediction Service, which in turn may require access to the Shared Storage layer

The module relies on the Smile Java library for statistical analysis and ML, hence supports any of the algorithms and techniques there available.

*Shared Storage:* Stores and provision all the data and meta-data required for supporting DSS execution, there excluded the patient data—which is used for training, testing, or prediction, and then discarded. Also, the module provides fault-tolerance by persisting data and by automatically backing data up. As such, it can interact with any other module—as already described. The module relies on the Redis database system.

## IV. Conclusion

In this paper, we proposed a reference architecture for a Decision Support System enabling and promoting interoperability amongst technological environments, and collaboration between teams of data scientists and amongst clinical staff. In particular, we have taken as a reference the "Risk prediction as a Service" instantiation of the more general MLaaS paradigm, steadily gaining momentum in clinical practice. In this context, we motivated how leveraging state of art web technologies – such as service-orientation and RESTful architecture – and emerging standards for machine learning models representation and exchange – namely, PMML and PFA – could support practical use cases demanding interoperability and collaboration.

The proposed architecture is actually implemented in a ICT solution used within the CONNECARE European project, where it delivers predictions about hospitalisations, emergency re-admissions, and mortality, both during home hospitalisation and after discharge (30-90 days). The models currently used have been created by a partner with locally available data regarding 6000+ patients with 5+ years of follow-ups, which could not be disclosed to other partners. Hence, the DSS has been a precious asset to let clinical partners both *(a)* use and validate each other models and *(b)* join locally available datasets to cooperatively build prediction models, without the need for data sharing. Those models started generating predictions on new data samples (new patients enrolled in clinical trials) at the beginning of 2019, hence data collection for validation and evaluation purpose is still in place. Trials are set to end after summer, then assessment of the generated models will begin.

The project is set to end at the end of 2019, then, the DSS system here described will be released as open-source software free to use and modify by any interested developer, data scientist, clinical practitioner.

## References

[1] M. TB and D. AS, "The inevitable application of big data to health care," *JAMA*, vol. 309, no. 13, pp. 1351–1352, 2013. [Online]. Available: http://dx.doi.org/10.1001/jama.2013.393

[2] I. Cano, M. Lluch-Ariet, D. Gomez-Cabrero, D. Maier, S. Kalko, M. Cascante, J. Tegnér, F. Miralles, D. Herrera, and J. Roca, "Biomedical research in a digital health framework," *Journal of Translational Medicine*, vol. 12, no. 2, p. S10, Nov 2014. [Online]. Available: https://doi.org/10.1186/1479-5876-12-S2-S10

[3] J. Andreu-Perez, C. C. Y. Poon, R. D. Merrifield, S. T. C. Wong, and G. Z. Yang, "Big data for health," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1193–1208, July 2015.

[4] E. Commission, "Big data in public health, telemedicine and health," European Commission Publications Portal, December 2016. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/study-big-data-public-health-telemedicine-and-healthcare

[5] J. Pivarski, C. Bennett, and R. L. Grossman, "Deploying analytics with the portable format for analytics (pfa)," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 579–588. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939731

[6] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, "Mlaas: Machine learning as a service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec 2015, pp. 896–902.

[7] V. R. Rao, K. Zolfaghar, D. K. Hazel, V. M, S. B. Roy, and A. Teredesai, "Readmissions score as a service (RaaS)," 2014.

[8] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," *CoRR*, vol. abs/1609.02943, 2016. [Online]. Available: http://arxiv.org/abs/1609.02943

[9] IBM, "Model asset exchange." [Online]. Available: https://developer.ibm.com/exchanges/models/

[10] Openscoring Ltd, "Jpmml-r home page." [Online]. Available: https://github.com/jpmml/jpmml-r

[11] Oracle Corporation, "Jersey home page." [Online]. Available: https://jersey.github.io/

[12] FasterXML, LLC, "Jackson home page." [Online]. Available: https://github.com/FasterXML/jackson