Machine Learning Explainability as a Service: Service Description and Economics

Paolo Fantozzi[1][0000−0002−5442−2239], Luigi Laura[2][0000−0001−6880−8477], and Maurizio Naldi[1][0000−0002−0903−398X]

[1] Dpt. of Law, Economics, Politics, and Modern Languages at LUMSA University, Via Pompeo Magno, 28, Rome, 00192, Italy, m.naldi@lumsa.it, p.fantozzi1@lumsa.it

[2] International Telematic University Uninettuno, Rome, Italy, luigi.laura@uninettunouniversity.net

Abstract. Explainability is a growing concern in many machine learning applications. Machine learning platforms now typically provide explanations accompanying their model output. However, this may be considered as just a first step towards defining explainability as a service in itself, which would allow users to get more control over the kind of explainability technique they wish to employ. In this paper, we first provide a survey of the current offer of machine learning platforms, observing their pricing models and the explainability features they possibly offer. In order to progress towards Explainability-as-a-Service (XaaS), we propose to base its definition on the REST paradigm, considering three major examples of explainability techniques, relying on either feature scoring, surrogate linear models, or internal state observation. We also show that XaaS is dependent on machine-learning model provisioning, and the two services are linked by a one-way essential complement relationship, where ML provisioning plays the role of the essential component and XaaS is the complement option. We also suggest that vertical integration is the natural arrangement for companies offering either service, given their mutual relationship.

Keywords: Explainability · XaaS · Pricing.

1 Introduction

Explainability is becoming an unavoidable companion to all machine learning (ML) models. In many fields, providing the output of a machine-learning task, though exhibiting impressive performances, is not enough. An explanation of how the result was obtained and what contributed most to those results is required by both domain experts and people affected by those results and is typically attained by identifying the most relevant features to determine the output of the ML model.

The literature on explainability techniques is now so large that it includes hundreds of papers. In Section 2, we provide a brief review of the most relevant ones. A recent survey dedicated to a single category of machine learning architectures, namely transformers, counted roughly 150 papers [9].

Though explainability is an additional task with respect to building a machine-learning model, its growing relevance has led many machine-learning platforms (e.g., Azure or AWS) to include explainability parameters when showing the resulting ML model. Though bundling explainability information adds value to ML model building, we observe that it is lacking at least in two respects. First, explainability is provided on as as-is basis, denying users the possibility to specify the explainability technique they wish. Second, it does not extract the full value of the information that is provided alongside ML model results.

We claim that explainability should be considered as a service in itself, which the user can ask for and pay accordingly. Though introducing explainability as a service has been invoked in the literature, no further actions have been taken to fulfil that wish. In this paper, we wish to mark a first step in that direction through the following contributions: a) we review the current panorama of machine-learning platforms, highlighting their side explainability features (Sec. 3); b) we provide an initial description of the service, adopting the REST (Representational State Transfer) paradigm (Sec. 4); c) we show that Explainability-as-a-Service can be considered to be in a one-way essential complement relationship with the ML model service, with consequences on its pricing (Sec. 5).

2 Literature Review

Explainability-as-a-Service (XaaS for short) relies on the adoption of explainability methods and can be seen as an accompanying feature of ML models. In this section, we briefly survey the literature on explainability and then examine the (scarce) literature on XaaS, highlighting the research gap.

There are many ways to classify the explainability methods. Here, we adopt the classification into model-agnostic and model-aware techniques proposed in [2], based on the relationship to the model to be explained. In model-agnostic techniques, the model to be explained is considered as a black box, i.e., as a set of input-output tuples, and the explanation is derived just by analysing these tuples. On the other hand, in the model-aware family, the architecture of the model to be explained is known and can be exploited to gain insight into the way the model result is arrived at.

2.1 Model-agnostic techniques

In most model-agnostic methods, explanations are sought by perturbing the input and measuring the differences in the output. A major technique is Local Interpretable Model-agnostic Explanation (LIME) [14], where linear models are derived as surrogates for the actual ML model. It provides a local explanation with no guarantee about its capabilities outside a small neighbourhood. Another method providing a surrogate model is SHAP (SHapley Additive exPlanations) [11], which employs game theory based on the notion of Shapley values [17].

2.2 Model-aware techniques

Model-aware techniques are largely employed when the ML model to be explained employs neural networks. Such model-aware methods observe, e.g., the activations returned by layers in the neural network or the gradients computed by the learning algorithm. Methods dedicated to attention-based architectures, e.g., transformers, are achieving greater visibility [9].

A major activation-based method is Layer-wise Relevance Propagation (LRP) [3], where all layers of the neural network use ReLU activation, like in convolutional architectures (CNN). Neuron activations represent the relevance of the neuron. Many subsequent works extended LRP beyond the ReLU assumption, e.g., [8]. Further variations of LRP are Partial-LRP, applicable to transformers [18], and that proposed for images in [6]. A different approach dedicated to CNN architectures to classify images was Class Activation Mapping (CAM) [19], where a global average pool layer is included just before the fully connected last layer. The activation given by each filter of the last convolutional layer is multiplied by the weights matrix of the last fully connected layer to form scores for each super-pixel (a patch) of the image. Super-pixels are then reconciled with the original image to form a saliency map over the original pixels. Variations of this method were presented in [12] (based on Single Value Decomposition, in [16] (Grad-CAM) and in [5] (Grad-CAM++). Attention Rollout and Attention Flow were both presented in [1], using the attention weights as a proxy for explaining the output. By accumulating the attention scores over the layers (from last to first), a score can be computed for each input feature.

We have just offered a brief survey of the large body of literature dealing with explainability. The notion of configuring it as a service has not been explored as largely. One of the first attempts concerned a very specific application field, i.e., the use of explainable AI for robot planning, where an automated planner is called to justify its plans based on the user's constraints [4]. XaaS has been invoked as an approach to address biases and satisfy users' requirements in [10], where, however, no specific solutions are suggested. Further support for the introduction of XaaS was given in [13], where the abbreviation XaaS was also introduced, in the context of finance applications of AI, with third-party providers offering specialized tools and services for AI explainability, in particular surrogate linear models (namely LIME and SHAP). Again, no specific solutions are suggested for the service. The notion of XaaS was again put forward in [15], with, however, no specific description of the service architecture. What we see missing in the literature is a precise definition of a service architecture for XaaS.

3 Machine Learning Platforms

In addition to developing their own ML solutions via coding on their machines, researchers and practitioners may also exploit the services offered by ML platforms. Those platforms are to be given the problem and a dataset and allow the user to select an algorithm and proceed with model training, validation and deployment. In this section, we provide a brief description of those platforms.

Most of these platforms target non-technical users, so they offer a simple user interface to upload data of many different types (images, text, videos, etc.), guiding the user through preprocessing the data and labelling them. Also, the platforms we examine here focus on supervised learning techniques because they are easier to understand for a non-technical user.

The major machine-learning platforms are:

- Google Cloud AutoML (https://cloud.google.com/automl)
- AWS SageMaker (https://aws.amazon.com/sagemaker), which offers both no-code solutions (JumpStart, Canvas, etc.) and tools to train a model by coding.
- Azure AutoML (https://azure.microsoft.com/solutions/automated-machine-learning)
- IBM Watson Studio (https://www.ibm.com/products/watson-studio)
- BigML (https://bigml.com/)
- ObviouslyAI (https://www.obviously.ai/)

All these platforms offer some preprocessing tools, usually guided by visualizations produced through automatic data exploration, and also labelling tools, with many identifiable similarities with dedicated tools (such as Label Studio[3]).

Even though all the platforms offer the possibility to train a model from scratch, from simple regressions to more complex models such as (sequential) neural networks, only some allow us to exploit the so-called *Foundation Models*. Since foundation models are most often closed-source, the availability of these models inside a platform means that there exists a deal between the owner of the platform and the companies owning the models. Depending on the model, they can be used either by fine-tuning on the user data, or with a few-shots approach.

The implication of these deals between companies is that some models are provided only by some platforms. For instance, the only platform to provide access to OpenAI models is Azure AutoML, because of the exclusive deal signed by OpenAI and Microsoft[4]. For this reason, the differences between platforms are not limited to their functionalities and user interface, but may extend to the set of pre-trained models available. Although most online machine learning platform providers try to train their own foundation models to be used in their platforms, there exists a differentiation between providers. Some providers use their own models as their crown jewels (e.g., Google and IBM), while others focus on models developed by other companies (e.g., Microsoft and Amazon).

A further difference among providers is related to the provider's cloud ecosystem itself. Indeed, while small providers (e.g., BigML and ObviouslyAI) do not have a complete cloud ecosystem, big providers (e.g., Google, Amazon, Microsoft and IBM) offer a full set of services related to machine learning, e.g., data storage, which is critical for data-intensive applications and also for training ML models. Small providers tend to integrate as many services as possible to increase their services' attractiveness. On the other hand, big providers wish to

[3] https://labelstud.io/

[4] https://blogs.microsoft.com/blog/2023/01/23/microsoftandopenaiextendpartnership/

turn customers away from competitors and have them switch to their services. They do so by providing integration with other providers just through favouring migration, leaving as a prerequisite that the data should be either uploaded directly to the ML service or integrated by using their data storage service.

All big providers offer an explainability service (among small providers, BigML offers explainability while ObviouslyAI does not). Furthermore, all big providers (which usually also offer PaaS services to train machine learning models) give the users the possibility to upload already-trained models built using standard frameworks (e.g., PyTorch or Tensorflow), so the platform can apply explainability even if models have been trained outside. Platforms tend to use well-known explainability methods, typically either model-agnostic post-hoc methods or post-hoc methods based on neural network architectures.

4 Explainability As A Service

In the previous sections, we examined current explainability solutions and ML platforms, where explainability is offered as a free companion to ML model building. We claim that explainability can now be offered as a separate service. A formal description of XaaS has not been provided yet. In this section, we wish to introduce a formal description of explainability as a web service, adopting REST (Representational State Transfer) as the tool of reference.

In order to describe the interactions between systems, we have chosen the OpenAPI standard instead of WSDL (Web Service Definition Language), due to its wide use in real-world applications and the availability of graphical tools to visualize and interact with APIs. Also, OpenAPI is mainly based on REST, which is the de-facto standard for web applications. Also, the stateless nature of REST makes users tend to build easier APIs that are also much easier to scale up because of the lack of critical sections intrinsic to the APIs themselves.

The platform should include a main service to run models for inference and the explainability service for the models loaded with the main service. In order to run a model for inference, we need a trained model in a well-known format and the amount of resources for the model. Ideally, the service should support all the file formats produced by the de-facto standard tools (just like PyTorch and TensorFlow), but it is not enough to be able to read and run the model file, since we also need the architecture used in the model (e.g., it is not easy to inspect a model to understand whether it is similar to an encoder-only transformer). We also need to size the hardware to run the model, which again is not obtainable by inspecting the model: the size of RAM needed to run a Mixture-of-experts model is not linearly dependent on the number of model parameters, because during inference only a fraction of the parameters are loaded into RAM. In Fig.1, we show the OpenAPI code to load the model.

Besides the main service access point, we should also provide at least three different access points to explainability services. Depending on the model type (or architecture) to be explained, we could apply a subset of methods, e.g., the following three classes (whose OpenAPI code is in Figures 2 through 4):

```
"/load_model/": {
    "post": {
        "summary": "Load Model",
        "operationId": "load_model_load_model__post",
        "parameters": [{
            "name": "model_file_url", "in": "query", "required": true,
            "schema": { "type": "string", "format": "uri", "minLength": 1}
        }, {
            "name": "model_type", "in": "query", "required": true,
            "schema": {"$ref": "#ArchitectureType"}
        }, {
            "name": "resources_to_allocate", "in": "query", "required": true,
            "schema": {"type": "string", "title": "Resources To Allocate"}
        }],
        "responses": {
            "200": {
                "description": "Successful Response",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#ModelAllocated"}
                    }
                }
            },
            "422": {
                "description": "Validation Error",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#HTTPValidationError"}
}}}}}},
```

Fig. 1. OpenAPI specification for load model API

Features' scores , including the methods providing an importance score for each input feature. A major method in this class is GradCAM, applicable to any CNN-based architecture and returning a matrix of scores to be projected on the input image to produce a saliency map, plottable as a heatmap showing the most important pixels in the input image. In order to apply this method, we first need to know the architecture used since the computation of the convolutional part is carried out separately from the classification (or regression) head mounted on the convolutional layers.

Surrogate model , including the methods providing a surrogate model (often a linear one) locally approximating the original model. The surrogate model obtained should roughly provide the same output as the original model, but its quality (its accuracy) degrades as we move away from the local neighbourhood for which the surrogate model was generated. A major method in this class is LIME, where the most important regressors in the linear model represent the most relevant features to explain the model. LIME is model-agnostic, so that we can dispense with the need to know the architecture of the ML model to be explained.

Model internal observation , including the methods that just plot the internal parameters of the models. For instance, for transformer-based architectures we could plot the attention weights learned in different layers of the model. The only computation performed in this case is the aggregation of multi-head attention weights. For this type of approach, we first need the model architecture and then check whether this architecture is supported by the observation methods provided by the platform.

```
"/model_inspect/": {
    "post": {
        "summary": "Model Inspect",
        "operationId": "model_inspect_model_inspect__post",
        "parameters": [{
            "name": "allocation_id", "in": "query", "required": true,
            "schema": {"type": "string"}
        }, {
            "name": "section_of_model_to_inspect", "in": "query", "required": true,
            "schema": {"type": "string"}
        }, {
            "name": "method", "in": "query", "required": false,
            "schema": {"type": "string", "default": "auto"}
        }],
        "requestBody": {
            "required": true,
            "content": {
                "application/json": {
                    "schema": {"$ref": "#Body_model_inspect_model_inspect__post"}
                }
            }
        },
        "responses": {
            "200": {
                "description": "Successful Response",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#SampleExplanationModelInspect"}
                    }
                }
            },
            "422": {
                "description": "Validation Error",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#HTTPValidationError"}
}}}}}},
```

Fig. 2. OpenAPI specification for model inspect API

```
"/features_scores/": {
    "post": {
        "summary": "Features Scores",
        "operationId": "features_scores_features_scores__post",
        "parameters": [{
            "name": "allocation_id", "in": "query", "required": true,
            "schema": {"type": "string"}
        }, {
            "name": "method", "in": "query", "required": false,
            "schema": {"type": "string", "default": "auto"}
        }],
        "requestBody": {
            "required": true,
            "content": {
                "application/json": {
                    "schema": {"$ref": "#Body_features_scores_features_scores__post"}
                }
            }
        },
        "responses": {
            "200": {
                "description": "Successful Response",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#SampleExplanationFeaturesScores"}
                    }
                }
            },
            "422": {
                "description": "Validation Error",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#HTTPValidationError"}
}}}}}},
```

Fig. 3. OpenAPI specification for features scores API

```
"/surrogate_model/": {
    "post": {
        "summary": "Surrogate Model",
        "operationId": "surrogate_model_surrogate_model__post",
        "parameters": [{
            "name": "allocation_id", "in": "query", "required": true,
            "schema": {"type": "string"}
        }, {
            "name": "method", "in": "query", "required": false,
            "schema": {"type": "string", "default": "auto"}
        }],
        "requestBody": {
            "required": true,
            "content": {
                "application/json": {
                    "schema": {"$ref": "#Body_surrogate_model_surrogate_model__post"}
                }
            }
        },
        "responses": {
            "200": {
                "description": "Successful Response",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#SampleExplanationFeaturesScores"}
                    }
                }
            },
            "422": {
                "description": "Validation Error",
                "content": {
                    "application/json": {
                        "schema": {"$ref": "#HTTPValidationError"}
}}}}}}
```

Fig. 4. OpenAPI specification for surrogate model API

5 Pricing models

None of the ML platforms in Section 3 offers explainability as a separate service from ML provisioning. Rather, it is offered as a built-in feature of ML models, which is not priced separately. In this section, we first review the pricing models adopted by platform providers for ML provisioning and then examine the characteristics of a pricing model for XaaS.

The pricing policies adopted by the platforms can be classified again into two categories based on the platform providers themselves: big providers offering cloud services and small providers more devoted to the product.

For big providers, the main cost is linked to the amount of resources used during model training and inference, with a pay-as-you-go plan where either a little or no cost at all is added. For example, Azure offers a pay-as-you-go scheme where computing capacity is metered by the second, with no long-term commitments or upfront payments, and increasing or decreasing consumption on demand. However, a savings plan can be subscribed, where customers commit to spend a fixed hourly amount for 1 or 3 years. Similarly, Amazon's Sagemaker offers an On-Demand Pricing scheme (metered by the hour) with neither minimum fees nor upfront commitments, as well as Savings Plans. In Google's AutoML, a price per node hour is proposed based on the specific operation, e.g., Training, Deployment and online prediction, or Batch prediction. Small providers instead apply pricing policies based on a monthly fee covering all the services offered by the platform. BigML offers a set of plans differing for the size of datasets supported and parallel tasks. ObviouslyAI offers a single plan with higher costs and limitations on the number of predictions provided.

Since XaaS would require a dedicated pricing policy, how can we attach a price to explainability? Here, we do not provide specific indications, but we can describe some constraints based on the nature of XaaS. First, we notice that you cannot get an explanation if you do not set an ML model first, so the demand for XaaS cannot be greater than that for ML. We also envisage that the price of XaaS should be lower than that of ML model provisioning. In addition, we can consider XaaS as a complement of ML (rather than a substitute), so that the demand and prices of ML model provisioning and XaaS should be related. Precisely, we expect the demand for XaaS to be negatively impacted by the price of ML model provisioning as well as by its own price. Since ML modelling can take place without explainability, but not the reverse, we can classify the relationship between the two services as a one-way essential complement, where ML provisioning is essential while XaaS is optional [7].

As to the market structure, we have a case of natural vertical integration, where the providers of ML and XaaS coincide. When the value of the option (XaaS) is smaller than the essential service (ML) and marginal costs are zero, the provider would choose to price the option at zero (what the providers are doing today) [7]. The joint monopolist would give XaaS away and earn all its profits by selling ML provisioning. Providers offering ML would, therefore, include XaaS in a bundle package and compete on the quality and prices of ML provisioning.

6 Conclusions

We have set the case for adding explainability as a service to ML model provisioning. Currently, explainability is provided as a side output of ML provisioning. However, providing XaaS would allow users to cast specific requests and get the explainability they wish. Our description of the service relying on the OpenAPI standard can be employed as a first step in that direction. We have sketched the service primitives for three types of XaaS output, showing the variety of service features that users can get with respect to the current offer where XaaS is offered on an as-is basis. However, ML provisioning and XaaS are actually not two totally separate services, their relationship being that of one-way essential complementarity. This may have some effect on pricing. Though listing XaaS as a separate service would allow the provider to set separate prices, under certain conditions, the best option for the provider may be to include XaaS in a bundle package with ML and practically offer XaaS for free. We expect to examine the impact of XaaS bundle offer on competition in future works.

References

1. Abnar, S., Zuidema, W.: Quantifying attention flow in transformers. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 4190–4197. (2020)

2. Adadi, A., Berrada, M.: Peeking inside the Black-Box: A survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On Pixel-Wise explanations for Non-Linear classifier decisions by Layer-Wise relevance propagation. PLoS One **10**(7), e0130140 (2015)
4. Cashmore, M., Collins, A., Krarup, B., Krivic, S., Magazzeni, D., Smith, D.: Towards explainable ai planning as a service. arXiv preprint arXiv:1908.05059 (2019)
5. Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-CAM++: Generalized Gradient-Based visual explanations for deep convolutional networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 839–847. IEEE (2018)
6. Chefer, H., Gur, S., Wolf, L.: Transformer interpretability beyond attention visualization. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 782–791 (June 2021)
7. Chen, M.K., Nalebuff, B.J.: One-way essential complements. Tech. Rep. CFDP 1588, Cowles Foundation for Research in Economics (2006)
8. Ding, Y., Liu, Y., Luan, H., Sun, M.: Visualizing and understanding neural machine translation. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1). pp. 1150–1159, Vancouver, Canada (2017)
9. Fantozzi, P., Naldi, M.: The explainability of transformers: Current status and directions. Computers **13**(4), 92 (Apr 2024)
10. Jovanović, M., Schmitz, M.: Explainability as a user requirement for artificial intelligence systems. Computer **55**(2), 90–94 (2022)
11. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 4768–4777. NIPS'17, Red Hook, NY, USA (2017)
12. Muhammad, M.B., Yeasin, M.: Eigen-CAM: Class activation map using principal components. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (July 2020)
13. Rane, N., Choudhary, S., Rane, J.: Explainable artificial intelligence (xai) approaches for transparency and accountability in financial decision-making. Available at SSRN 4640316 (2023)
14. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Proc. of the 22nd ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining. pp. 1135–1144, New York, NY, USA (2016)
15. Samiei, S., Baratalipour, N., Yadav, P., Roy, A., He, D.: Addressing stability in classifier explanations. In: 2021 IEEE International Conference on Big Data (Big Data). pp. 1920–1927. IEEE (2021)
16. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: 2017 IEEE Intl Conf on Computer Vision (ICCV). pp. 618–626 (2017).
17. Shapley, L.S.: A value for n-person games. In: Harold William Kuhn, A.W.T. (ed.) Contributions to the Theory of Games (AM-28), Volume II, pp. 307–318. Princeton University Press (1953)
18. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing Multi-Head Self-Attention: Specialized heads do the heavy lifting, the rest can be pruned. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 5797–5808, Florence, Italy (2019)
19. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (June 2016)