Model interpretability (preview)

Article • 10/03/2022 • 7 minutes to read

This article describes methods you can use for model interpretability in Azure Machine Learning.

> **Important**
>
> With the release of the Responsible AI dashboard, which includes model interpretability, we recommend that you migrate to the new experience, because the older SDK v1 preview model interpretability dashboard will no longer be actively maintained.

Why model interpretability is important to model debugging

When you're using machine learning models in ways that affect people's lives, it's critically important to understand what influences the behavior of models. Interpretability helps answer questions in scenarios such as:

- Model debugging: Why did my model make this mistake? How can I improve my model?
- Human-AI collaboration: How can I understand and trust the model's decisions?
- Regulatory compliance: Does my model satisfy legal requirements?

The interpretability component of the Responsible AI dashboard contributes to the "diagnose" stage of the model lifecycle workflow by generating human-understandable descriptions of the predictions of a machine learning model. It provides multiple views into a model's behavior:

- Global explanations: For example, what features affect the overall behavior of a loan allocation model?
- Local explanations: For example, why was a customer's loan application approved or rejected?

You can also observe model explanations for a selected cohort as a subgroup of data

points. This approach is valuable when, for example, you're assessing fairness in model predictions for individuals in a particular demographic group. The **Local explanation** tab of this component also represents a full data visualization, which is great for general eyeballing of the data and looking at differences between correct and incorrect predictions of each cohort.

The capabilities of this component are founded by the InterpretML package, which generates model explanations.

Use interpretability when you need to:

- Determine how trustworthy your AI system's predictions are by understanding what features are most important for the predictions.
- Approach the debugging of your model by understanding it first and identifying whether the model is using healthy features or merely false correlations.
- Uncover potential sources of unfairness by understanding whether the model is basing predictions on sensitive features or on features that are highly correlated with them.
- Build user trust in your model's decisions by generating local explanations to illustrate their outcomes.
- Complete a regulatory audit of an AI system to validate models and monitor the impact of model decisions on humans.

How to interpret your model

In machine learning, *features* are the data fields you use to predict a target data point. For example, to predict credit risk, you might use data fields for age, account size, and account age. Here, age, account size, and account age are features. Feature importance tells you how each data field affects the model's predictions. For example, although you might use age heavily in the prediction, account size and account age might not affect the prediction values significantly. Through this process, data scientists can explain resulting predictions in ways that give stakeholders visibility into the model's most important features.

By using the classes and methods in the Responsible AI dashboard and by using SDK v2 and CLI v2, you can:

- Explain model prediction by generating feature-importance values for the entire

model (global explanation) or individual data points (local explanation).
- Achieve model interpretability on real-world datasets at scale.
- Use an interactive visualization dashboard to discover patterns in your data and its explanations at training time.

By using the classes and methods in the SDK v1, you can:

- Explain model prediction by generating feature-importance values for the entire model or individual data points.
- Achieve model interpretability on real-world datasets at scale during training and inference.
- Use an interactive visualization dashboard to discover patterns in your data and its explanations at training time.

> **Note**
>
> Model interpretability classes are made available through the SDK v1 package. For more information, see **Install SDK packages for Azure Machine Learning** and **azureml.interpret**.

Supported model interpretability techniques

The Responsible AI dashboard and `azureml-interpret` use the interpretability techniques that were developed in Interpret-Community , an open-source Python package for training interpretable models and helping to explain opaque-box AI systems. Opaque-box models are those for which we have no information about their internal workings.

Interpret-Community serves as the host for the following supported explainers, and currently supports the interpretability techniques presented in the next sections.

Supported in Responsible AI dashboard in Python SDK v2 and CLI v2

Interpretability technique	Description	Type

Interpretability technique	Description	Type
Mimic Explainer (Global Surrogate) + SHAP tree	Mimic Explainer is based on the idea of training global surrogate models to mimic opaque-box models. A global surrogate model is an intrinsically interpretable model that's trained to approximate the predictions of any opaque-box model as accurately as possible. Data scientists can interpret the surrogate model to draw conclusions about the opaque-box model. The Responsible AI dashboard uses LightGBM (LGBMExplainableModel), paired with the SHAP (SHapley Additive exPlanations) Tree Explainer, which is a specific explainer to trees and ensembles of trees. The combination of LightGBM and SHAP tree provides model-agnostic global and local explanations of your machine learning models.	Model-agnostic

Supported in Python SDK v1

Interpretability technique	Description	Type
SHAP Tree Explainer	The SHAP Tree Explainer, which focuses on a polynomial, time-fast, SHAP value-estimation algorithm that's specific to *trees and ensembles of trees*.	Model-specific
SHAP Deep Explainer	Based on the explanation from SHAP, Deep Explainer is a "high-speed approximation algorithm for SHAP values in deep learning models that builds on a connection with DeepLIFT described in the SHAP NIPS paper . *TensorFlow* models and *Keras* models using the TensorFlow back end are supported (there's also preliminary support for PyTorch)."	Model-specific
SHAP Linear Explainer	The SHAP Linear Explainer computes SHAP values for a *linear model*, optionally accounting for inter-feature correlations.	Model-specific
SHAP Kernel Explainer	The SHAP Kernel Explainer uses a specially weighted local linear regression to estimate SHAP values for *any model*.	Model-agnostic
Mimic Explainer (Global Surrogate)	Mimic Explainer is based on the idea of training global surrogate models to mimic opaque-box models. A global surrogate model is an intrinsically interpretable model that's trained to approximate the predictions of *any opaque-box model* as accurately as possible. Data scientists can interpret the surrogate model to draw	Model-agnostic

Interpretability technique	Description	Type
	conclusions about the opaque-box model. You can use one of the following interpretable models as your surrogate model: LightGBM (LGBMExplainableModel), Linear Regression (LinearExplainableModel), Stochastic Gradient Descent explainable model (SGDExplainableModel), or Decision Tree (DecisionTreeExplainableModel).	
Permutation Feature Importance Explainer	Permutation Feature Importance (PFI) is a technique used to explain classification and regression models that's inspired by Breiman's Random Forests paper (see section 10). At a high level, the way it works is by randomly shuffling data one feature at a time for the entire dataset and calculating how much the performance metric of interest changes. The larger the change, the more important that feature is. PFI can explain the overall behavior of *any underlying model* but doesn't explain individual predictions.	Model-agnostic

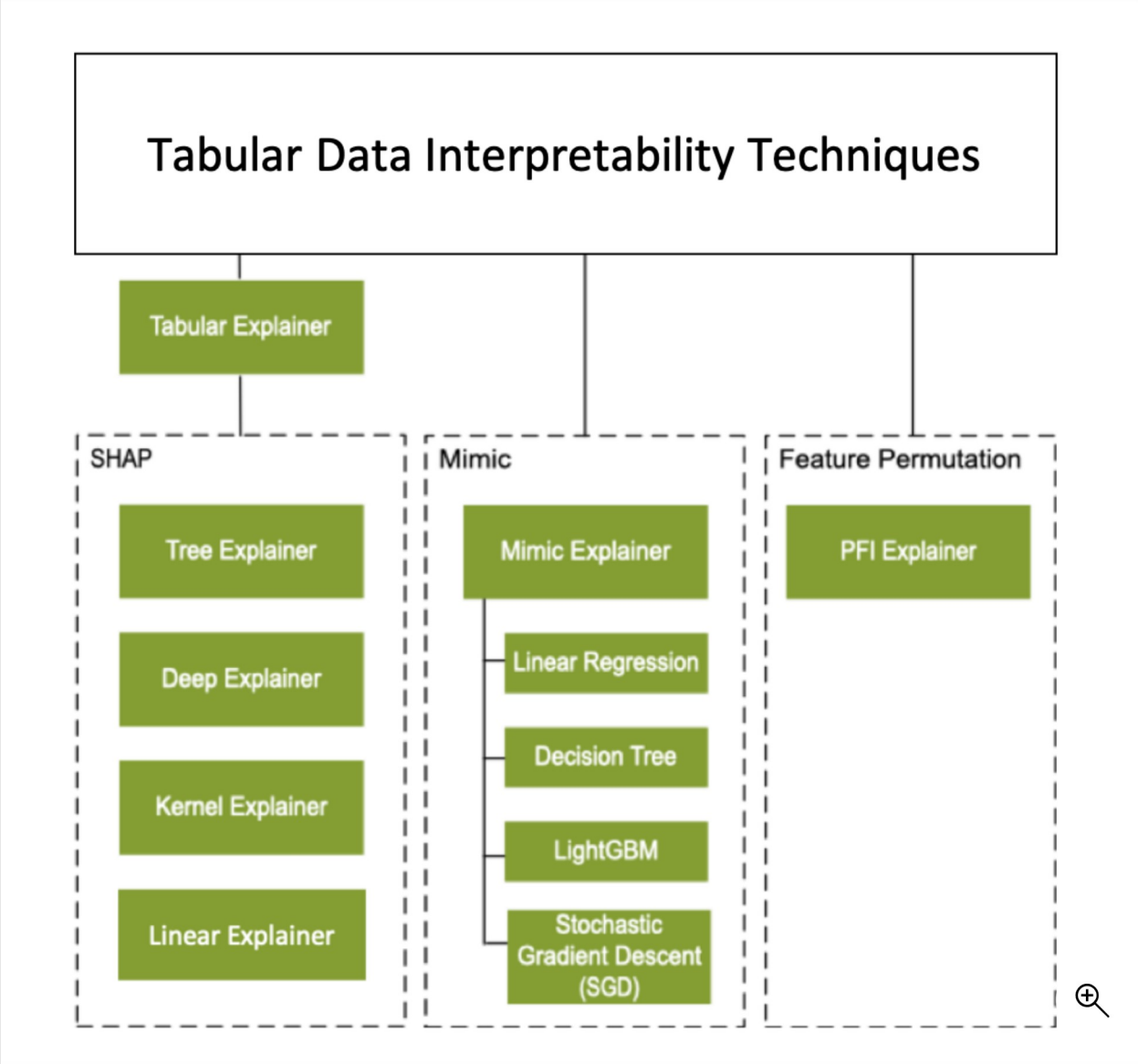
Besides the interpretability techniques described above, we support another SHAP-based explainer, called Tabular Explainer. Depending on the model, Tabular Explainer uses one of the supported SHAP explainers:

- Tree Explainer for all tree-based models
- Deep Explainer for deep neural network (DNN) models
- Linear Explainer for linear models
- Kernel Explainer for all other models

Tabular Explainer has also made significant feature and performance enhancements over the direct SHAP explainers:

- **Summarization of the initialization dataset**: When speed of explanation is most important, we summarize the initialization dataset and generate a small set of representative samples. This approach speeds up the generation of overall and individual feature importance values.
- **Sampling the evaluation data set**: If you pass in a large set of evaluation samples but don't actually need all of them to be evaluated, you can set the sampling parameter to `true` to speed up the calculation of overall model explanations.

The following diagram shows the current structure of supported explainers:



Supported machine learning models

The `azureml.interpret` package of the SDK supports models that are trained with the following dataset formats:

- `numpy.array`
- `pandas.DataFrame`
- `iml.datatypes.DenseData`
- `scipy.sparse.csr_matrix`

The explanation functions accept both models and pipelines as input. If a model is provided, it must implement the prediction function `predict` or `predict_proba` that conforms to the Scikit convention. If your model doesn't support this, you can wrap it in

a function that generates the same outcome as `predict` or `predict_proba` in Scikit and use that wrapper function with the selected explainer.

If you provide a pipeline, the explanation function assumes that the running pipeline script returns a prediction. When you use this wrapping technique, `azureml.interpret` can support models that are trained via PyTorch, TensorFlow, and Keras deep learning frameworks as well as classic machine learning models.

Local and remote compute target

The `azureml.interpret` package is designed to work with both local and remote compute targets. If you run the package locally, the SDK functions won't contact any Azure services.

You can run the explanation remotely on Azure Machine Learning Compute and log the explanation info into the Azure Machine Learning Run History Service. After this information is logged, reports and visualizations from the explanation are readily available on Azure Machine Learning studio for analysis.

Next steps

- Learn how to generate the Responsible AI dashboard via CLI v2 and SDK v2 or the Azure Machine Learning studio UI.
- Explore the supported interpretability visualizations of the Responsible AI dashboard.
- Learn how to generate a Responsible AI scorecard based on the insights observed in the Responsible AI dashboard.
- Learn how to enable interpretability for automated machine learning models.