Partial Dependence and Individual Conditional Expectation plots

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 8)

Unknown directive type "currentmodule".

```
.. currentmodule:: sklearn.inspection
```

Partial dependence plots (PDP) and individual conditional expectation (ICE) plots can be used to visualize and analyze interaction between the target response [1] and a set of input features of interest.

Both PDPs [H2009] and ICEs [G2015] assume that the input features of interest are independent from the complement features, and this assumption is often violated in practice. Thus, in the case of correlated features, we will create absurd data points to compute the PDP/ICE [M2019].

Partial dependence plots

Partial dependence plots (PDP) show the dependence between the target response and a set of input features of interest, marginalizing over the values of all other input features (the 'complement' features). Intuitively, we can interpret the partial dependence as the expected target response as a function of the input features of interest.

Due to the limits of human perception the size of the set of input feature of interest must be small (usually, one or two) thus the input features of interest are usually chosen among the most important features.

The figure below shows two one-way and one two-way partial dependence plots for the California housing dataset, with a :class:`HistGradientBoostingRegressor <sklearn.ensemble.HistGradientBoostingRegressor>`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 32); *backlink*

Unknown interpreted text role "class".

One-way PDPs tell us about the interaction between the target response and an input feature of interest feature (e.g. linear, non-linear). The left plot in the above figure shows the effect of the average occupancy on the median house price; we can clearly see a linear relationship among them when the average occupancy is inferior to 3 persons. Similarly, we could analyze the effect of the house age on the median house price (middle plot). Thus, these interpretations are marginal, considering a feature at a time.

PDPs with two input features of interest show the interactions among the two features. For example, the two-variable PDP in the above figure shows the dependence of median house price on joint values of house age and average occupants per household. We can clearly see an interaction between the two features: for an average occupancy greater than two, the house price is nearly independent of the house age, whereas for values less than 2 there is a strong dependence on age.

The :mod:`sklearn.inspection` module provides a convenience function :func:`~PartialDependenceDisplay.from_estimator` to create one-way and two-way partial dependence plots. In the below example we show how to create a grid of partial dependence plots: two one-way PDPs for the features `0` and `1` and a two-way PDP between the two features:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 57); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 57); *backlink*

Unknown interpreted text role "func".

```
>>> from sklearn.datasets import make_hastie_10_2
>>> from sklearn.ensemble import GradientBoostingClassifier
```

```
>>> from sklearn.inspection import PartialDependenceDisplay

>>> X, y = make_hastie_10_2(random_state=0)
>>> clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
...     max_depth=1, random_state=0).fit(X, y)
>>> features = [0, 1, (0, 1)]
>>> PartialDependenceDisplay.from_estimator(clf, X, features)
<...>
```

You can access the newly created figure and Axes objects using `plt.gcf()` and `plt.gca()`.

For multi-class classification, you need to set the class label for which the PDPs should be created via the `target` argument:

```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> mc_clf = GradientBoostingClassifier(n_estimators=10,
...     max_depth=1).fit(iris.data, iris.target)
>>> features = [3, 2, (3, 2)]
>>> PartialDependenceDisplay.from_estimator(mc_clf, X, features, target=0)
<...>
```

The same parameter `target` is used to specify the target in multi-output regression settings.

If you need the raw values of the partial dependence function rather than the plots, you can use the :func:`sklearn.inspection.partial_dependence` function:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 91);** ***backlink***
>
> Unknown interpreted text role "func".

```
>>> from sklearn.inspection import partial_dependence

>>> results = partial_dependence(clf, X, [0])
>>> results["average"]
array([[ 2.466...,  2.466..., ...
>>> results["values"]
[array([-1.624..., -1.592..., ...
```

The values at which the partial dependence should be evaluated are directly generated from `X`. For 2-way partial dependence, a 2D-grid of values is generated. The `values` field returned by :func:`sklearn.inspection.partial_dependence` gives the actual values used in the grid for each input feature of interest. They also correspond to the axis of the plots.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 103);** ***backlink***
>
> Unknown interpreted text role "func".

Individual conditional expectation (ICE) plot

Similar to a PDP, an individual conditional expectation (ICE) plot shows the dependence between the target function and an input feature of interest. However, unlike a PDP, which shows the average effect of the input feature, an ICE plot visualizes the dependence of the prediction on a feature for each sample separately with one line per sample. Due to the limits of human perception, only one input feature of interest is supported for ICE plots.

The figures below show four ICE plots for the California housing dataset, with a :class:`HistGradientBoostingRegressor <sklearn.ensemble.HistGradientBoostingRegressor>`. The second figure plots the corresponding PD line overlaid on ICE lines.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 123);** ***backlink***
>
> Unknown interpreted text role "class".

While the PDPs are good at showing the average effect of the target features, they can obscure a heterogeneous relationship created by interactions. When interactions are present the ICE plot will provide many more insights. For example, we could observe a linear relationship between the median income and the house price in the PD line. However, the ICE lines show that there are some exceptions, where the house price remains constant in some ranges of the median income.

The :mod:`sklearn.inspection` module's :meth:`PartialDependenceDisplay.from_estimator` convenience function can be used to create ICE plots by setting `kind='individual'`. In the example below, we show how to create a grid of ICE plots:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 141); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 141); *backlink*

Unknown interpreted text role "meth".

```
>>> from sklearn.datasets import make_hastie_10_2
>>> from sklearn.ensemble import GradientBoostingClassifier
>>> from sklearn.inspection import PartialDependenceDisplay
```

```
>>> X, y = make_hastie_10_2(random_state=0)
>>> clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
...     max_depth=1, random_state=0).fit(X, y)
>>> features = [0, 1]
>>> PartialDependenceDisplay.from_estimator(clf, X, features,
...     kind='individual')
<...>
```

In ICE plots it might not be easy to see the average effect of the input feature of interest. Hence, it is recommended to use ICE plots alongside PDPs. They can be plotted together with `kind='both'`.

```
>>> PartialDependenceDisplay.from_estimator(clf, X, features,
...     kind='both')
<...>
```

If there are too many lines in an ICE plot, it can be difficult to see differences between individual samples and interpret the model. Centering the ICE at the first value on the x-axis, produces centered Individual Conditional Expectation (cICE) plots [G2015]. This puts emphasis on the divergence of individual conditional expectations from the mean line, thus making it easier to explore heterogeneous relationships. cICE plots can be plotted by setting *centered=True*:

```
>>> PartialDependenceDisplay.from_estimator(clf, X, features,
...     kind='both', centered=True)
<...>
```

Mathematical Definition

Let X_S be the set of input features of interest (i.e. the *features* parameter) and let X_C be its complement.

The partial dependence of the response f at a point x_S is defined as:

$$pd_{X_S}(x_S) \overset{def}{=} \mathbb{E}_{X_C}[f(x_S, X_C)]$$

$$= \int f(x_S, x_C) p(x_C) dx_C,$$

where $f(x_S, x_C)$ is the response function (:term:`predict`, :term:`predict_proba` or :term:`decision_function`) for a given sample whose values are defined by x_S for the features in X_S, and by x_C for the features in X_C. Note that x_S and x_C may be tuples.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 193); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 193); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 193); *backlink*

Unknown interpreted text role "term".

Computing this integral for various values of x_S produces a PDP plot as above. An ICE line is defined as a single $f(x_S, x_C^{(i)})$ evaluated at x_S.

Computation methods

There are two main methods to approximate the integral above, namely the 'brute' and 'recursion' methods. The *method* parameter controls which method to use.

The 'brute' method is a generic method that works with any estimator. Note that computing ICE plots is only supported with the 'brute' method. It approximates the above integral by computing an average over the data X:

$$pd_{X_S}(x_S) \approx \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n} f(x_S, x_C^{(i)}),$$

where $x_C^{(i)}$ is the value of the i-th sample for the features in X_C. For each value of x_S, this method requires a full pass over the dataset X which is computationally intensive.

Each of the $f(x_S, x_C^{(i)})$ corresponds to one ICE line evaluated at x_S. Computing this for multiple values of x_S, one obtains a full ICE line. As one can see, the average of the ICE lines correspond to the partial dependence line.

The 'recursion' method is faster than the 'brute' method, but it is only supported for PDP plots by some tree-based estimators. It is computed as follows. For a given point x_S, a weighted tree traversal is performed: if a split node involves an input feature of interest, the corresponding left or right branch is followed; otherwise both branches are followed, each branch being weighted by the fraction of training samples that entered that branch. Finally, the partial dependence is given by a weighted average of all the visited leaves values.

With the 'brute' method, the parameter X is used both for generating the grid of values x_S and the complement feature values x_C. However with the 'recursion' method, X is only used for the grid values: implicitly, the x_C values are those of the training data.

By default, the 'recursion' method is used for plotting PDPs on tree-based estimators that support it, and 'brute' is used for the rest.

Note

While both methods should be close in general, they might differ in some specific settings. The 'brute' method assumes the existence of the data points $(x_S, x_C^{(i)})$. When the features are correlated, such artificial samples may have a very low probability mass. The 'brute' and 'recursion' methods will likely disagree regarding the value of the partial dependence, because they will treat these unlikely samples differently. Remember, however, that the primary assumption for interpreting PDPs is that the features should be independent.

Examples:

- :ref:`sphx_glr_auto_examples_inspection_plot_partial_dependence.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst, line 260); *backlink*

Unknown interpreted text role "ref".

Footnotes

[1] For classification, the target response may be the probability of a class (the positive class for binary classification), or the decision function.

References

[H2009] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning, Second Edition, Section 10.13.2, Springer, 2009.

[M2019] C. Molnar, Interpretable Machine Learning, Section 5.1, 2019.

[G2015] (1,2) `arxiv:` A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation" Journal of Computational and Graphical Statistics, 24(1): 44-65, Springer, 2015. <1309.6392>`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\modules\(scikit-learn-main)(doc)(modules)partial_dependence.rst`, **line 280**); ***backlink***

Unknown interpreted text role "arxiv".