Interpretable Machine Learning

Master ESA - University of Orléans

Christophe HURLIN

University of Orléans and IUF

Master Econométrie et Statistique Appliquée

Why interpretability matters?

Modern ML models such as random forests, boosting, and neural networks often achieve excellent predictive performance but are criticized as **black boxes**. Without interpretability, it is difficult to ensure transparency, accountability, and trust in model-based decisions.

Scope of interpretability

Interpretability in ML aims to make models and predictions understandable to humans. It is essential in domains such as credit scoring, risk management, taxation, policy evaluation, etc.

Variety of methods

- **Global methods:** variable importance, partial dependence plots.
- **Local methods:** LIME, SHAP values, counterfactual explanations.
- **Trade-off:** accuracy vs. transparency.

References

Main reference

Molnar, C (2022), Interpretable machine learning. A Guide for Making Black Box Models Explainable,. https://christophm.github.io/interpretable-ml-book/.

Other references

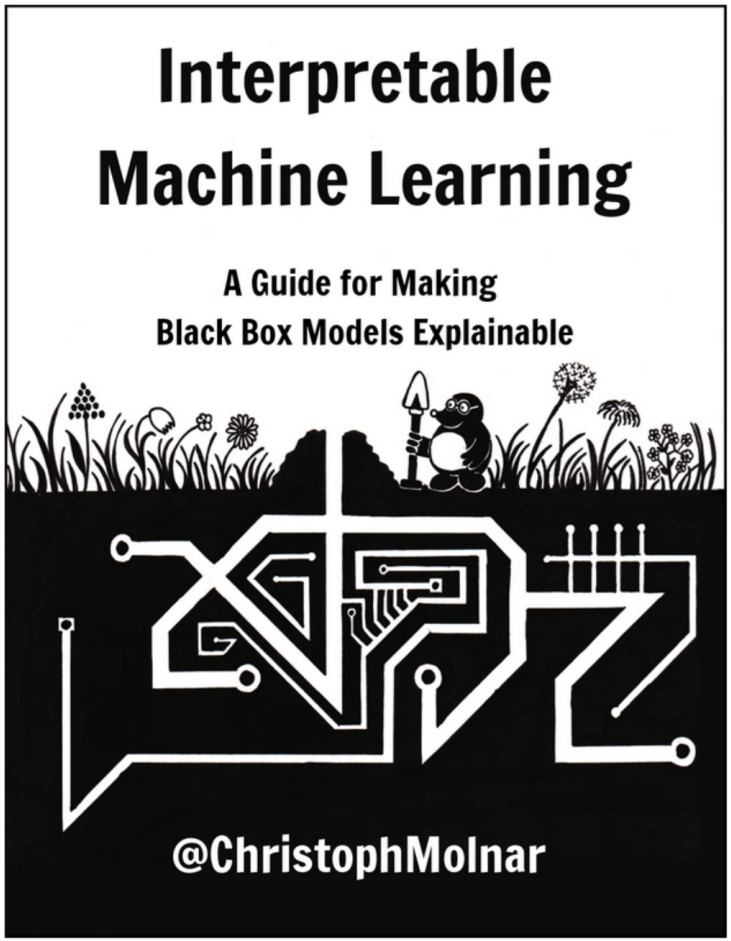
Lipton, Z.C. (2018), The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery, Queue, 16(3), 31–57.

Miller, T. (2019) Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence, 267, 1-38.

Molnar C., Casalicchio G. and Bischk B. (2020), Interpretable Machine Learning: A Brief History, State-of-the-Art and Challenge, arXiv:2010.09337v1.

Zhao Q. and T. Hastie, (2019). Causal Interpretations of Black-Box Models, Journal of Business & Economic Statistics, 272-281.

Main reference



References on the Fairness


Barocas, S., Hardt, M., and Narayanan, A. (2020). Fairness and Machine Learning. fairmlbook.org. http://www.fairmlbook.org.


Bartlett, R., Morse, A., Stanton, R., and Wallace, N. (2019). Consumer-lending discrimination in the Fintech era. Working Paper 25943, National Bureau of Economic Research.


Fuster, A., Plosser, M., Schnabl, P., and Vickery, J. (2019). The Role of Technology in Mortgage Lending. *The Review of Financial Studies*, 32(5):1854–1899.


Hurlin C., Pérignon C., and Saurin S. (2024), The Fairness of Credit Scoring Models, Management Science, .


Prince, A. and Schwarcz, D. (2020). Proxy discrimination in the age of artificial intelligence and big data. *Iowa Law Review*, 105:1257–1301.


Verma, S. and Rubin, J. (2018). Fairness definitions explained. In 2018 IEEE/ACM International Workshop on Software Fairness (FairWare), pages 1–7.

Applications with Python on Colab

Applications and exercises during the training will be done using Python via (Google) Colab.

Google Colab, short for Google Colaboratory, is a free cloud platform provided by Google that allows you to write and execute Python code in a browser environment.

It offers a Jupyter notebook hosting service that enables users to create and share documents containing live code, equations, visualizations, and narrative text.

`https://colab.research.google.com/`

Objectives of the Session

1. Understand why interpretability is important in economics and finance.
2. Learn the main concepts of interpretable machine learning.
3. Discover global post-hoc methods such as PDP and ALE.
4. Explore local post-hoc methods such as LIME and SHAP.
5. Introduce inherently interpretable models such as PLTR and GAM.
6. Discuss governance issues: transparency, fairness, stability, accountability.

Credit: iStock

Outline

Governance of AI in Finance

The need of a governance of AI in Finance

- AI in general and ML in particular are more and more often used in finance for many critical applications (ACPR, 2018).
- Examples: credit issuance, credit scoring (Hurlin and Pérignon, 2019), insurance pricing, fraud detection and Anti-Money Laundering (AML), etc.
- As these algorithms affect critical decisions both for banks and customers, there is a need for a specific **governance** and **regulation** for these models.


ACPR (2018), Artificial intelligence: challenges for the financial sector. Discussion papers publication, December 2018


Hurlin C. and Pérignon, C. (2019), Machine Learning, Nouvelles Données et Scoring de Crédit, Revue d'Economie Financière, 135, 21-50.

References on the Governance of AI in Finance

Many reports on the governance of AI in Finance


ACPR (2018), Artificial intelligence: challenges for the financial sector. Discussion papers publication, December 2018.


ACPR (2020). Governance of artificial intelligence in finance. Discussion papers publication, November, 2020. Main reference


Bracke, P., Datta, A., Jung, C., and Sen, S. (2019). Machine learning explainability in finance: an application to default risk analysis. Bank of England, Staff Working Paper No. 816.


EBA (2020). Report on big data and advanced analytics. European Banking Authority, January, 2020.


European Commission (2020). White paper on artificial intelligence: A european approach to excellence and trust. European Commission, February, 2020.

References on the Governance of AI in Finance

June 2020

Governance of Artificial Intelligence in Finance

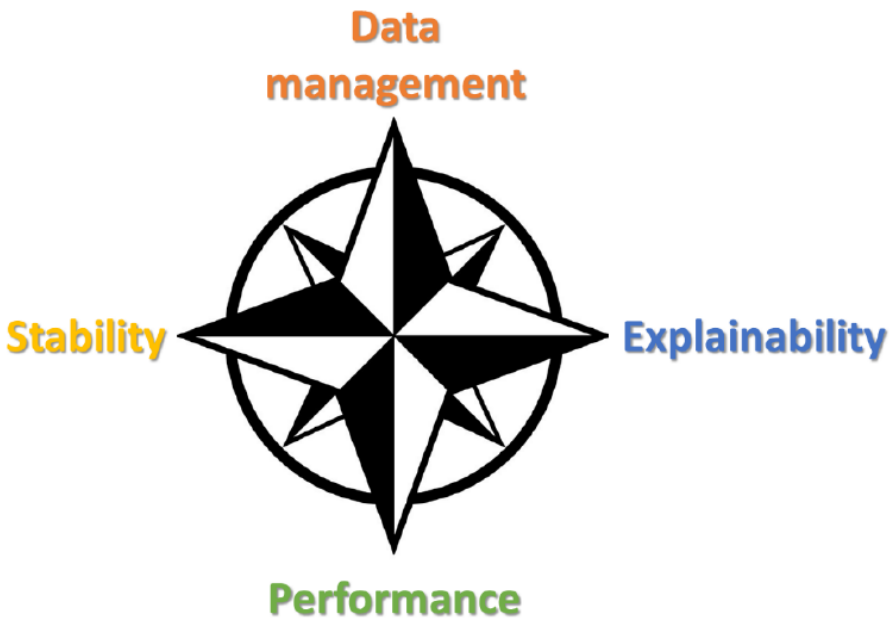
Discussion document

AUTHORS
Laurent Dupont, Olivier Fliche, Su Yang
Fintech-Innovation Hub, ACPR

ACPR
BANQUE DE FRANCE

Governance of AI in Finance

ACPR (2020) suggests four evaluation principles for AI algorithms and models:



Data Management

Principle 1: Data management

- All data processing should be as **thoroughly documented** as the other design stages of an AI algorithm (source code, performance of the resulting model, etc.).
- This documentation enables risk assessment in the areas of regulatory compliance and ethics, and the implementation of tools for detecting and mitigating undesired biases.
- The **data management** concerns each stage of the design and implementation process of an algorithm:
  1. Input data management: reference data, training data, and test (or evaluation) data.
  2. Pre-processing: operations performed on input data prior to the ML phase itself.
  3. Post-processing: operations performed on the output (i.e. predictions or decisions) of the model produced by the ML algorithm.

Data Management and Regulatory compliance

Regulatory Compliance

Regulatory **compliance** on data often includes requirements of two kinds:

- Compliance with regulation pertaining to data protection and individual privacy, starting with the **General Data Protection Regulation (GDPR)** in Europe, which sets rules on the collection, processing, and storage of personal data, and grants individuals specific rights such as access, rectification, and the right to be forgotten.
- Taking into account regulatory requirements specific to a domain or use case.

Example: Banking sector

Prudential norm BCBS 239 (*Principles for effective risk data aggregation and risk reporting*) on data completeness and data quality requirements in the banking sector.

Ethics and Fairness

Ethics and fairness

- Ethical issues lie at the core of the ever-increasing usage of AI in business processes which impact individuals and groups of people.
- Those issues include social and ethical concerns, and particularly questions of fairness raised by any automated or computer-aided decision process
- ACPR (2020) observes that bias detection and mitigation were at an early stage in the industry.
- The emphasis is put on internal validation of AI systems and on their regulatory compliance, not on the analysis of algorithmic fairness.

Hurlin C., Pérignon C., and Saurin S. (2024), The Fairness of Credit Scoring Models, Management Science.

AI act

EUROPEAN COMMISSION

Brussels, 21.4.2021

COM(2021) 206 final

2021/0106(COD)

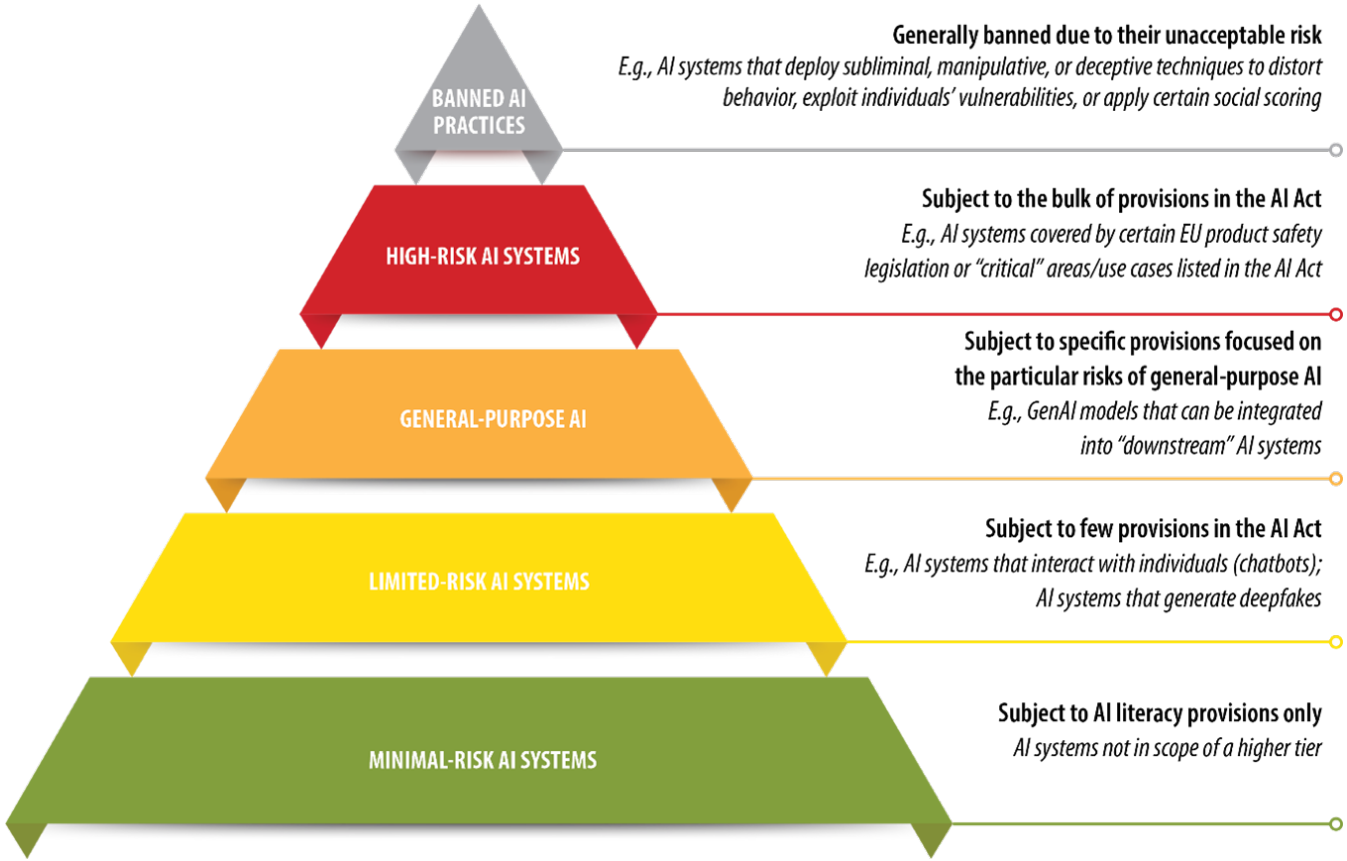
Proposal for a

REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL

LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS

{SEC(2021) 167 final} - {SWD(2021) 84 final} - {SWD(2021) 85 final}

AI act



Source: Nauwelaerts and Greaves (2024), Alston & Bird

AI act

A concern for the regulator

The recent AI act released by the **European Commission** states that:

> *"AI systems used to evaluate the credit score or creditworthiness of natural persons should be classified as **high-risk AI systems** [...] AI systems used for this purpose may lead to **discrimination** of persons or groups and perpetuate historical patterns of discrimination, for example based on racial or ethnic origins, disabilities, age, sexual orientation, or create new forms of discriminatory impacts"*

EC (2021). Laying Down Harmonized on Artificial Intelligence. Proposal for a Regulation of the European Parliament and of the Council, European Commission, April 2021.

Case Study: Apple Credit Card

Goldman Sachs

In November 2019, Apple launches its apple card. This credit card was operated by Goldman Sachs and it has been seen by some commentators as a potential blueprint for the future of banking.

Case Study: Apple Credit Card

Note : David Hansson (DHH), fondateur de Basecamp et Steve Wozniak, co-fondateur d'Apple

A few days after the launch, the Apple Card faced criticism from David Heinemeier Hansson, the designer of Basecamp, and Steve Wozniak, Apple's co-founder. They alleged gender bias in Goldman Sachs' credit-scoring algorithm, claiming that their wives were granted significantly lower credit limits despite having the same financial resources.

Case Study: Apple Credit Card

2019 : Enquête du New York State Department of Financial Services (NY-DFS).

NEW YORK STATE
DEPARTMENT of
FINANCIAL SERVICES

Within a few weeks, the New York State Department of Financial Services (NYDFS), the banking supervisor for New York, opened an investigation.

Case Study: Apple Credit Card

The investigation made headlines worldwide, with many outlets reporting that the Apple Card was "sexist" and discriminatory against women.

Case Study: Apple Credit Card

NEW YORK STATE
DEPARTMENT of
FINANCIAL SERVICES

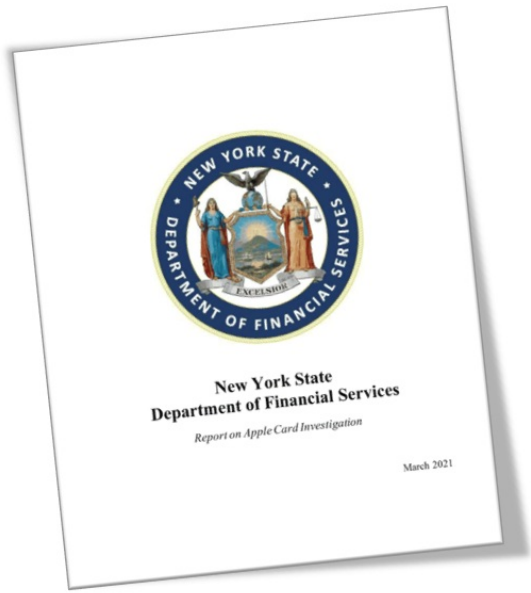
Press Release

March 23, 2021

DFS ISSUES FINDINGS ON THE APPLE CARD AND ITS UNDERWRITER GOLDMAN SACHS BANK

No Fair Lending Violations Found

More Broadly, Report Stresses Need for Modernizing Credit Scoring Models and Updating Anti-Discrimination Laws Governing Credit Access



Mars 2021 : rapport d'enquête du NY-DFS qui blanchit Goldman Sachs : il n'y a pas de discrimination de genre dans les algorithmes

After two years of investigation, the DFS has concluded in March 2021, that the Goldman Sachs credit scoring models did not discriminate based on gender. No fair lending violation was found.

Case Study: Apple Credit Card

The Apple Card case highlighted two key lessons for banks and FinTechs:

- **Algorithmic fairness** cannot be ignored.
- **Model evaluation** must go beyond accuracy metrics (AUC, Gini, classification rate) to include a fairness assessment, particularly for ML-based scoring models.

Sources of Algorithmic Discrimination

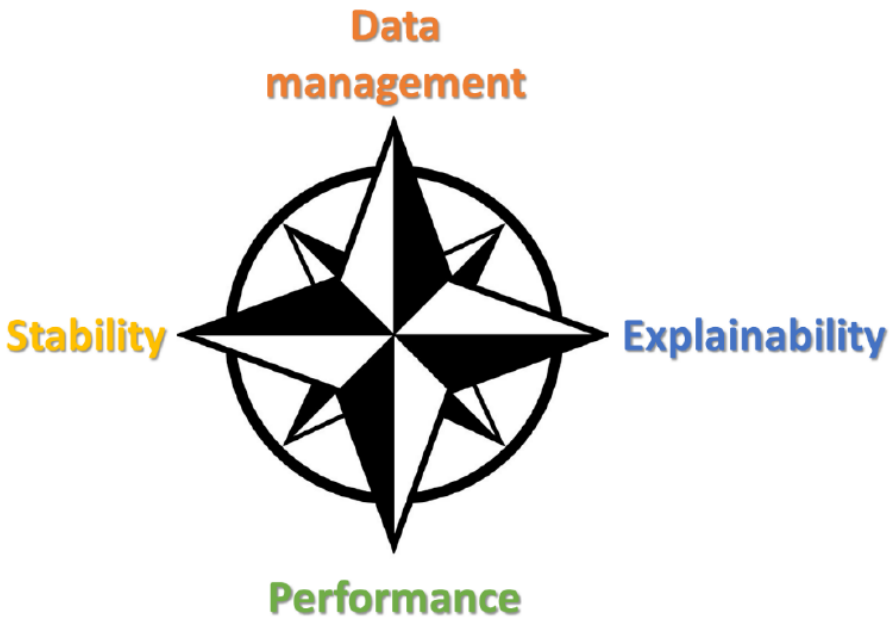
Sources of discrimination

- Credit scoring models are designed to differentiate between good and bad risks. But, this should not be done on the basis of protected attributes.
- Discrimination bias may come from the data (unlikely in the credit scoring context) or from the model.
- The scoring algorithm can learn from a dataset of actual defaults that occurred in the past. Two cases arise:
  1. Disparate treatment (direct discrimination) occurs when the algorithm explicitly uses the protected attribute.
  2. Disparate impact (indirect discrimination, proxy discrimination) occurs when the lender use facially neutral variables that are able, collectively, to synthetically reconstruct the protected attribute.

In practice, disparate impact is more likely to occur with highly non-linear, non-interpretable, and opaque scoring models, such as advanced ML algorithms.

Governance of AI in Finance

Principle 2: Performance



5

Model Performance

Principle 2: Performance

- Performance metrics of an ML algorithm must be carefully chosen, depending on the prediction task. For example, accuracy may be misleading in the presence of an imbalanced dataset, where alternative measures such as precision, recall, F1-score, or the AUC are more appropriate.
- The evaluation should also distinguish between in-sample fit and out-of-sample performance, using proper validation techniques (train/test split, cross-validation).
- The inherent trade-off between the algorithm's simplicity (interpretability, transparency) and its efficacy (predictive power, robustness) has to be taken into account.
- In practice, the choice of performance metrics reflects the economic or policy objective of the analysis, ensuring that the selected algorithm performs well on the dimensions that matter for decision-making.

Model Performance metrics

Performance metrics

Performance of an ML algorithm can typically be assessed using two types of metrics:

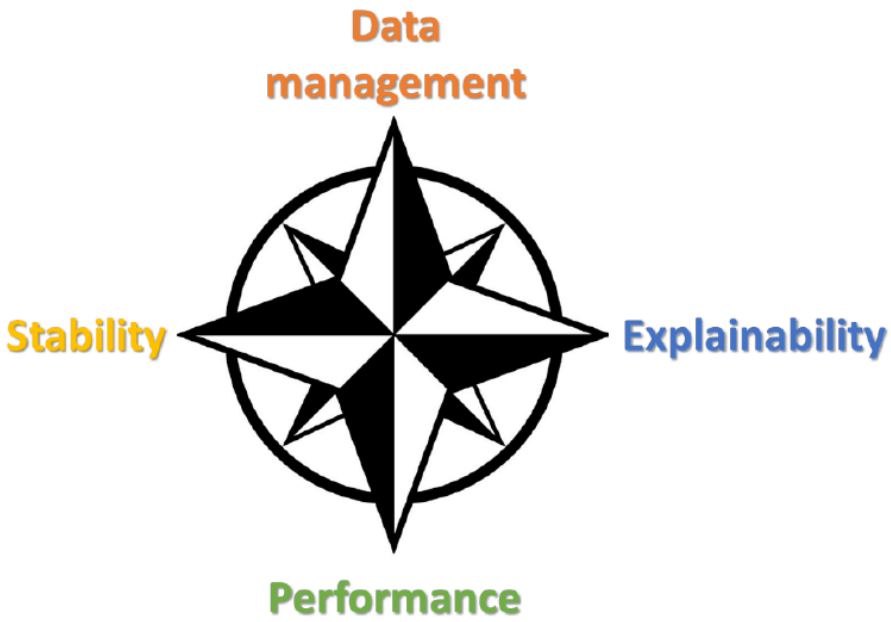
1. Predictive performance metrics (R^2, MSE, AUC, Gini, etc.) or **Key Risk Indicators** (KRI).
2. Business performance metrics or **Key Performance Indicators** (KPI).

A point of attention for KPI metrics is compliance compatibility. See Lessman et al. (2015) for a discussion on KPI metrics for credit scoring models.

Lessmann, S., Baesens, B., Seow, H.-V., and Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research, 247(1):124 –136.

Governance of AI in Finance

Principle 3: Stability



5

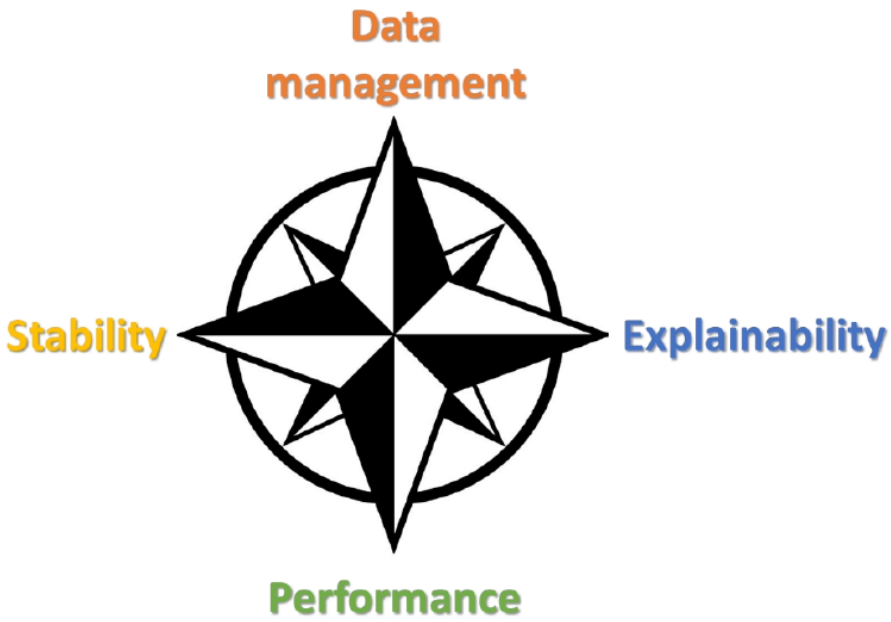
Stability

Principle 3: Stability

- Stability principle: ensuring that an ML algorithm's performance and its main operational characteristics are consistent over time.
- Potential instability sources which may affect AI algorithms deployed in the organization over time should be identified.
- Associated risks (operational risks and compliance risks) should be assessed and mitigation methods implemented.
- Three major sources of instability:
  1. **Temporal drift:** stability of an ML model implies absence of drift over time.
  2. **Generalization:** Can be understood as robustness, in the sense of generalization power when confronted with new data
  3. **Re-training:** Re-training an ML model, whether periodically or on a quasi-continuous basis, does not solve all instability issues. It may result at in non-reproducible decision between subsequent versions of the model.

Governance of AI in Finance

Principle 4: Explainability



5

Explainability and Interpretability

Principle 4: Explainability

The distinction between explainability and interpretability has been the subject of numerous debates, especially in the context of financial regulation and responsible AI.

- **Explainability** generally refers to the ability to provide a technical and objective account of an algorithm's internal functioning. It often relies on tools such as feature importance, sensitivity analysis, LIME or SHAP, and is crucial for auditing, compliance, and model validation.

- **Interpretability** is more closely related to the human understanding of the model's outputs. It involves presenting results in a way that is accessible to decision-makers,regulators, consumers, and other stakeholders impacted by the algorithm.

Both dimensions are essential in economics and finance: explainability ensures technical accountability, while interpretability guarantees trust and transparency in decision-making processes.

Transparency and Auditability

Related concepts

- **Transparency** refers to the degree to which the design and functioning of an ML algorithm can be made intelligible and accessible. It may involve access to documentation, training data, model architecture, and even source code. In the absence of such access, the algorithm is said to operate as a black box.

- **Auditability** denotes the practical feasibility of conducting a systematic evaluation of an algorithm. It requires sufficient documentation and access to logs, models, and results to assess key properties such as data governance, performance, robustness, fairness, and stability.

Transparency and auditability are complementary: transparency enables the necessary access to information, while auditability ensures the possibility of independent verification for regulatory compliance and risk management.

Governance of AI in Finance

Key concepts

1. Governance of AI in Finance
2. Data management
3. Regulatory compliance
4. Ethics and fairness
5. Performance
6. Stability
7. Explainability and interpretability

Outline

Interpretable Machine Learning: Definition

Definition: Interpretable Machine Learning

Interpretable Machine Learning (IML) refers to methods and models that make the internal mechanisms and predictions of machine learning systems understandable to humans, enabling users to trace, justify, and trust algorithmic decisions.

When Do We Need to Interpret

When we need to interpret (and when we do not) (1/2)

- In some cases, interpretability is not critical.
  - *e.g., Search engines predicting whether a document is relevant for a user. The cost of errors for users is low in such contexts.*
- In many other situations, interpretability is essential because mistakes are costly and user confidence in the model is key.
  - *e.g., loan applications, organ donor/recipient matching, job applications, parole decisions, university admissions.*

"The algorithm rejected you. Sorry, I do not have any additional information."

is not going to be an acceptable answer.

When Do We Need to Interpret

When we need to interpret (2/2)

- Interpretability may be required by law or regulation.
  - *e.g., GDPR's Chapter 11: Right to explanation; French Code for Public Health (Article L4001-3): algorithm developers must ensure that the functioning of the algorithm can be explained to users (Law #2021-1017, August 2, 2021).*
- To assess fairness: does the model produce unduly biased decisions? If challenged in court, can the company prove that the model is not discriminatory?
  - *e.g., Is the model racist or misogynistic?*
- To improve the model by identifying weaknesses.
  - *e.g., An image recognition system distinguishes between polar bears and dogs. If the model bases its decision on the background (ice vs. grass) rather than the animals' shape, this indicates a flaw.*

Whom to Explain?

Whom to Explain?

In the financial context, Bracke et al. (2019) identify at least six different types of stakeholders.

1. Developers, i.e. those developing or implementing an ML application.
2. 1^{st} line model checkers, i.e. those directly responsible for making sure model development is of sufficient quality.
3. Management responsible for the application.
4. 2^{nd} line model checkers, i.e. sta that, as part of a firm's control functions, independently check the quality of model development and deployment.
5. Conduct regulators that take an interest in deployed models being in line with conduct rules.
6. Prudential regulators that take an interest in deployed models being in line with prudential requirements.

Bracke, P., Datta, A., Jung, C., and Sen, S. (2019). Machine learning explainability in finance: an application to default risk analysis. Bank of England, Staff Working Paper No. 816.

Whom to Explain?

	Stakeholder interest					
	Developer	1st line model checking	Manage-ment	2nd line model checking	Conduct regulator	Prudential regulator
1) Which features mattered in individual predictions?	X				X	
2) What drove the actual predictions more generally?	X	X	X		X	
3) What are the differences between the ML model and a linear one?	X	X				
4) How does the ML model work?	X	X	X	X	X	X
5) How will the model perform under new states of the world? (that aren't captured in the training data)	X	X	X	X	X	X

Source: Bracke et al. (2019)

An Ideal Explanation

An ideal explanation should have the following properties:

- Accurate: it should describe as precisely as possible the case studied (for a local explanation) or the algorithm's behaviour (for a global explanation).
- Comprehensive: it should cover the entirety of motives and characteristics of the considered decision(s) or prediction(s).
- Comprehensible: it should not require excessive effort in order to be correctly understood by its intended recipients.
- Concise: it should be succinct enough to be grasped in a reasonable amount of time, in accordance with the time and productivity constraints of the encompassing process.
- Actionable: it should enable one or more actions by a human operator, such as overriding a prediction or decision.
- Robust: it should remain valid and useful even when input data are ever-changing and noisy.

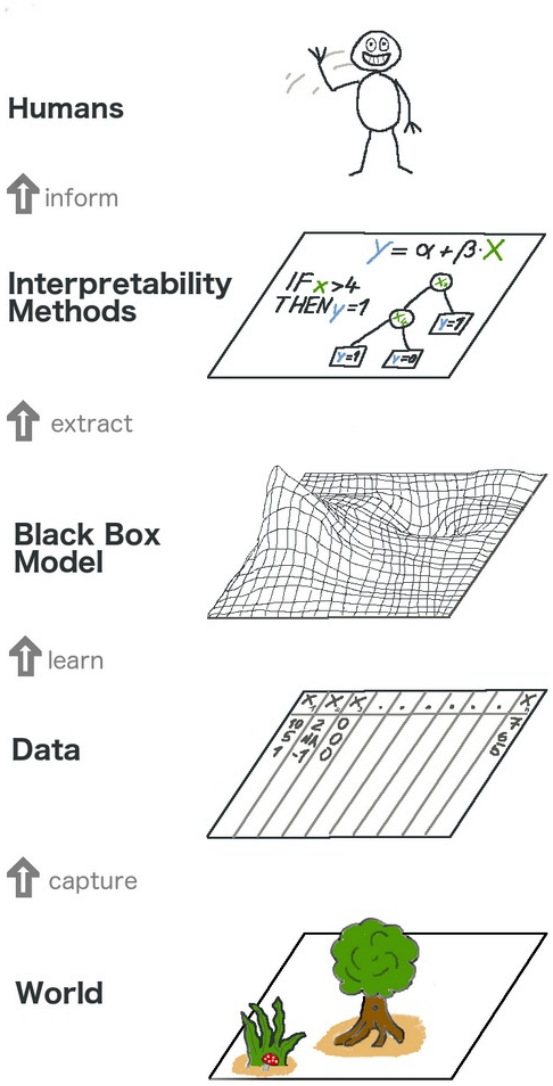
Taxonomy of Interpretability Methods

Taxonomy of interpretability methods (1/2)

We distinguish between:

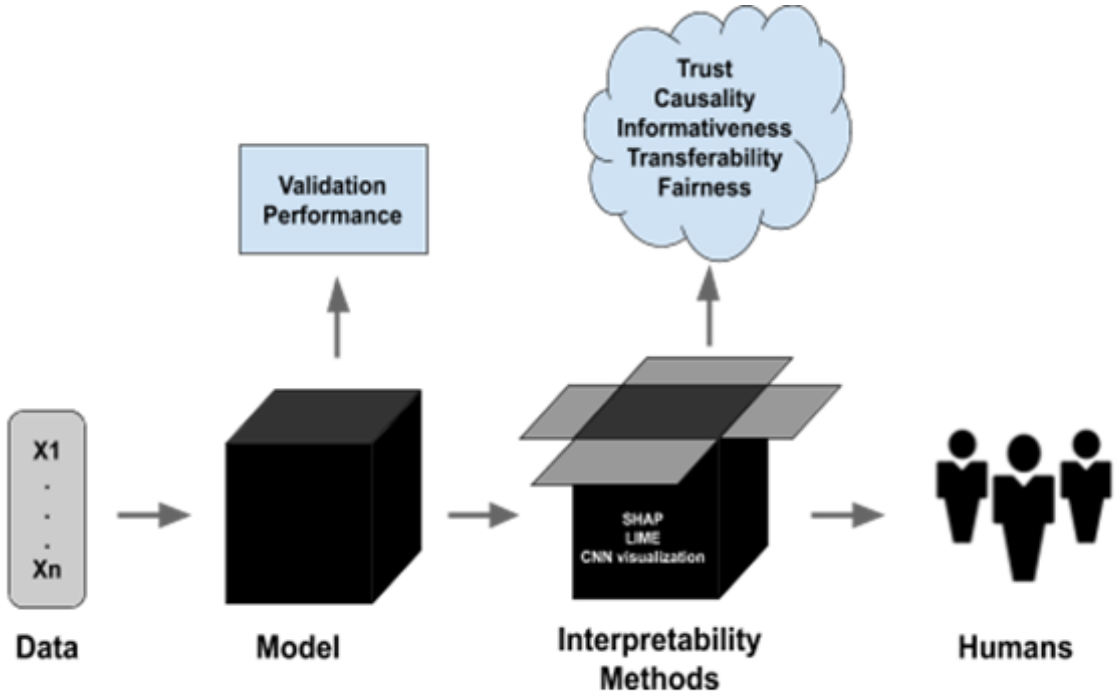
- Intrinsic interpretability refers to ML models that are considered interpretable due to their simple structure
  - *e.g., Decision tree, OLS, Logistic regression*
- Post-hoc interpretability refers to application of interpretation methods after model training.
  - *e.g., Neural Network, Random Forest, XGBoost*

Post-hoc Interpretability



Source: Molnar (2019)

Post-hoc Interpretability



Taxonomy of Interpretability Methods

Taxonomy of interpretability methods (2/3)

We distinguish between:

- Global Interpretability: Understanding a model
  - *For instance, in a given Artificial Neural Network, the variable with the strongest impact on the estimated house price is the lot size.*
- Local Interpretability: Understanding a decision for a particular instance
  - *For instance, why Mrs Smith's consumer loan application was rejected by an algorithmic lender?*

Explainability and Interpretability in Machine Learning

Taxonomy of interpretability methods (3/3)

We distinguish between:

- Model-specific interpretation tools are limited to specific model classes.
  - *Example : The interpretation of regression weights in a linear model is a model-specific interpretation, since by definition, the interpretation of intrinsically interpretable models is always model-specific.*
- Model-agnostic tools can be used on any ML model and are applied after the model has been trained (post hoc).
  - *These agnostic methods usually work by analyzing feature input and output pairs.*
  - *By definition, these methods cannot have access to model internals such as weights or structural information.*

Framework and Notations

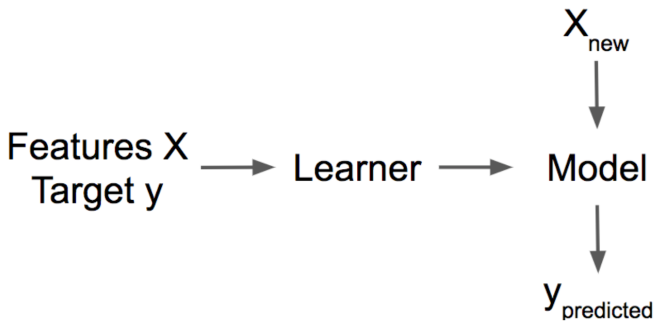
Some preliminary definitions and notations:

Algorithm: a well-defined sequence of rules or instructions that a machine follows to achieve a specific task or goal.

Machine Learning (ML): a collection of computational methods that enable computers to learn patterns from data and improve their predictions or decisions over time.

Learner (or ML algorithm): the procedure used to estimate a machine learning model from data.

Machine Learning Model: the output of the learning process, representing a function that maps inputs to predictions. The trained ML model is denoted by $\widehat{f}$ or $\widehat{f}(\cdot)$.



Framework and Notations

Dataset: a structured collection of data, typically represented as a table, from which the machine learns. It contains both the features and the target to be predicted.

Instance: a row in the dataset, also referred to as a (data) point, example, or observation. An instance consists of the feature values $x^{(i)}$ and, if available, the target outcome y_i.

Features: the input variables used for prediction or classification. A feature corresponds to a column in the dataset.

- The matrix of all features is denoted by $\mathbf{X}$, and the feature vector for a single instance i is $x^{(i)}$.
- The vector of a single feature j across all instances is denoted x_j, and the value of feature j for instance i is $x_j^{(i)}$.

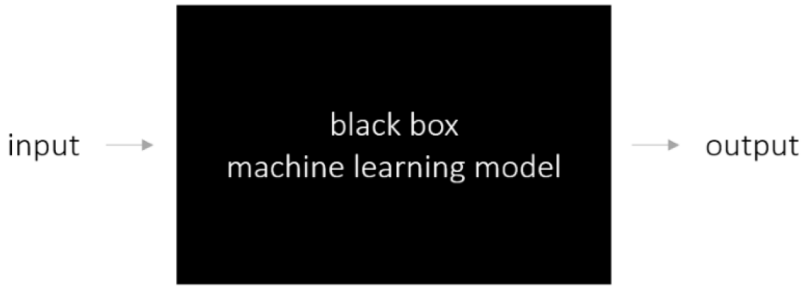
Target: the variable that the machine learns to predict, denoted by y (or y_i for a single instance).

Framework and Notations

Prediction: the output of the machine learning model, representing the estimated value of the target given the features. It is denoted by $\widehat{f}(x^{(i)})$ or $\widehat{y}$.

Black Box Model: a model whose internal decision-making process is not transparent or interpretable. In ML, the term usually refers to complex models where the parameters and their influence are difficult to understand.

White Box Model: a model that is inherently interpretable and whose inner workings can be directly understood.



Interpretable Machine Learning

Key Concepts

1. Black-box and white-box models.
2. Interpretable Machine Learning.
3. Intrinsic interpretability vs. post-hoc interpretability.
4. Model-specific vs. model-agnostic methods.
5. Local vs. global explanatory methods.
6. Counterfactual explanations.

Outline

Global Post hoc Interpretability

In this section, we introduce three model-agnostic interpretation methods:

1. **Global surrogate**
2. **Partial Dependence Plot (PDP)**
3. **Accumulated Local Effects (ALE)**

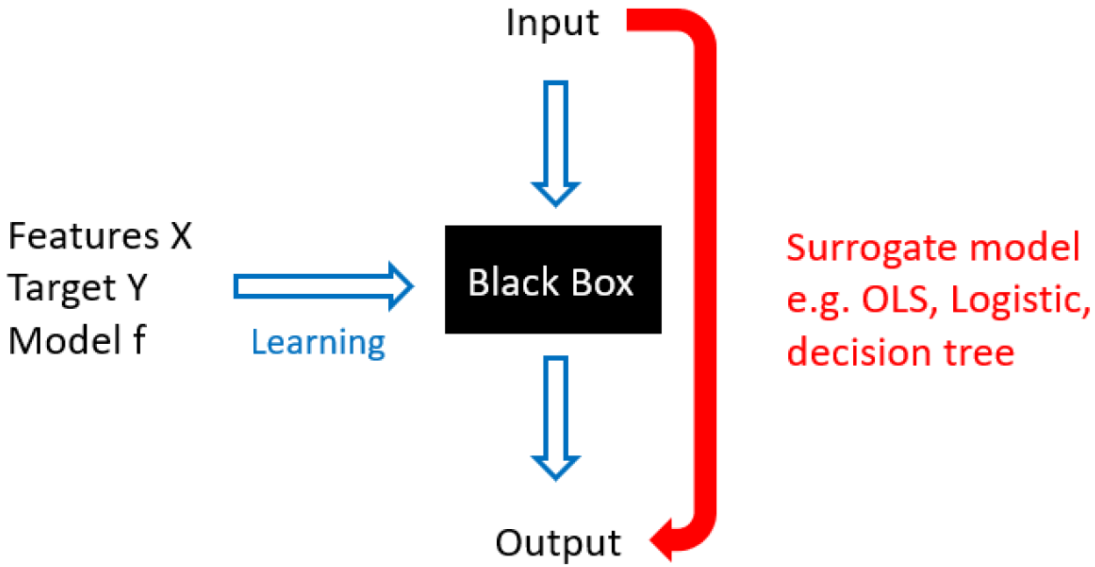
Global Surrogate Model

Global Surrogate Model: Definition

Definition: Global Surrogate Model

A **global surrogate model** is an interpretable model trained to approximate the overall behavior of a black-box model and provide insights into its predictions.

Consider a given opaque machine learning model that produces predictions $\hat{y}$.



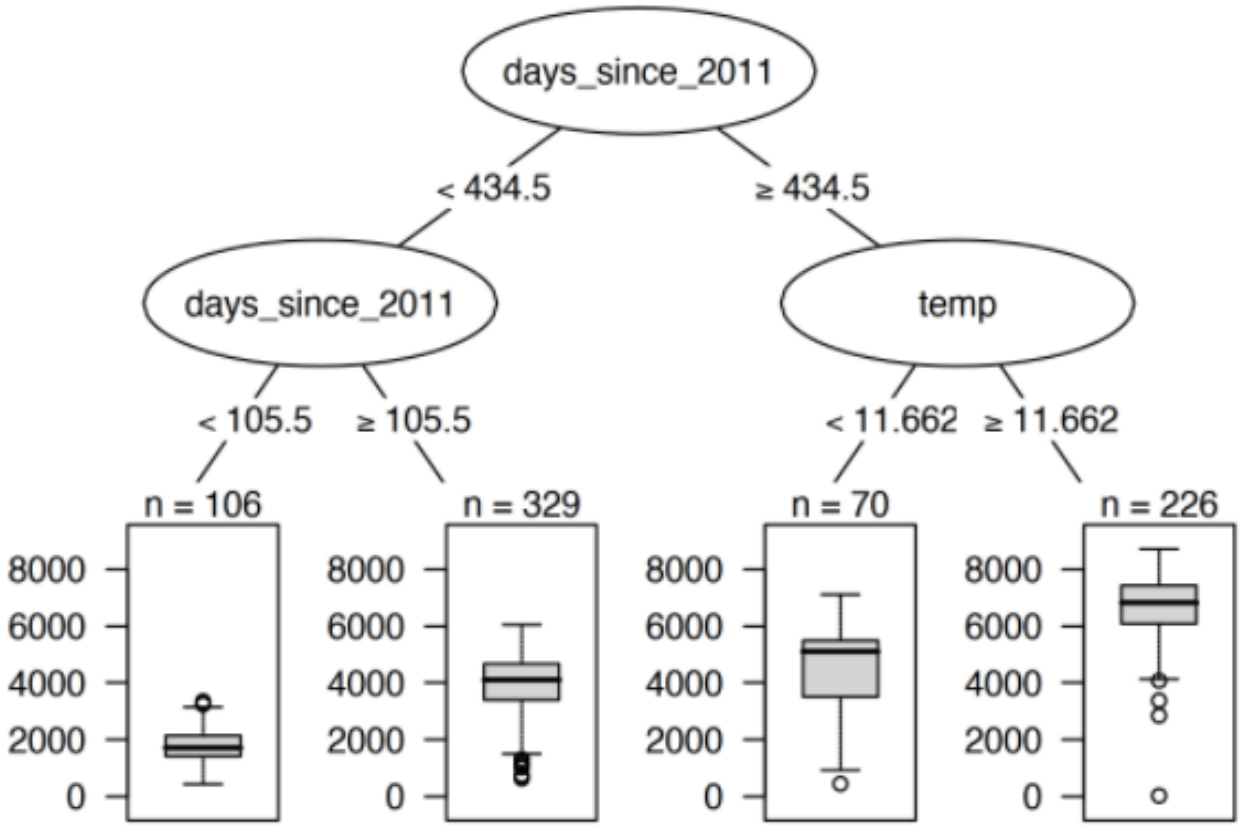
Build a Global Surrogate Model

How to build a global surrogate model?

1. Select a dataset X. This can be the same dataset that was used for training the black box model or a new dataset from the same distribution. You could even select a subset of the data or a grid of points, depending on your application.
2. For the selected dataset X, get the predictions of the black box model.
3. Select an interpretable model type (linear model, decision tree, ...).
4. Train the interpretable model on the dataset X and its predictions.
5. Measure how well the surrogate model replicates the predictions of the black box model (R^2, AUC, other metrics).
6. Interpret the surrogate model.

Global Surrogate Model: Illustration

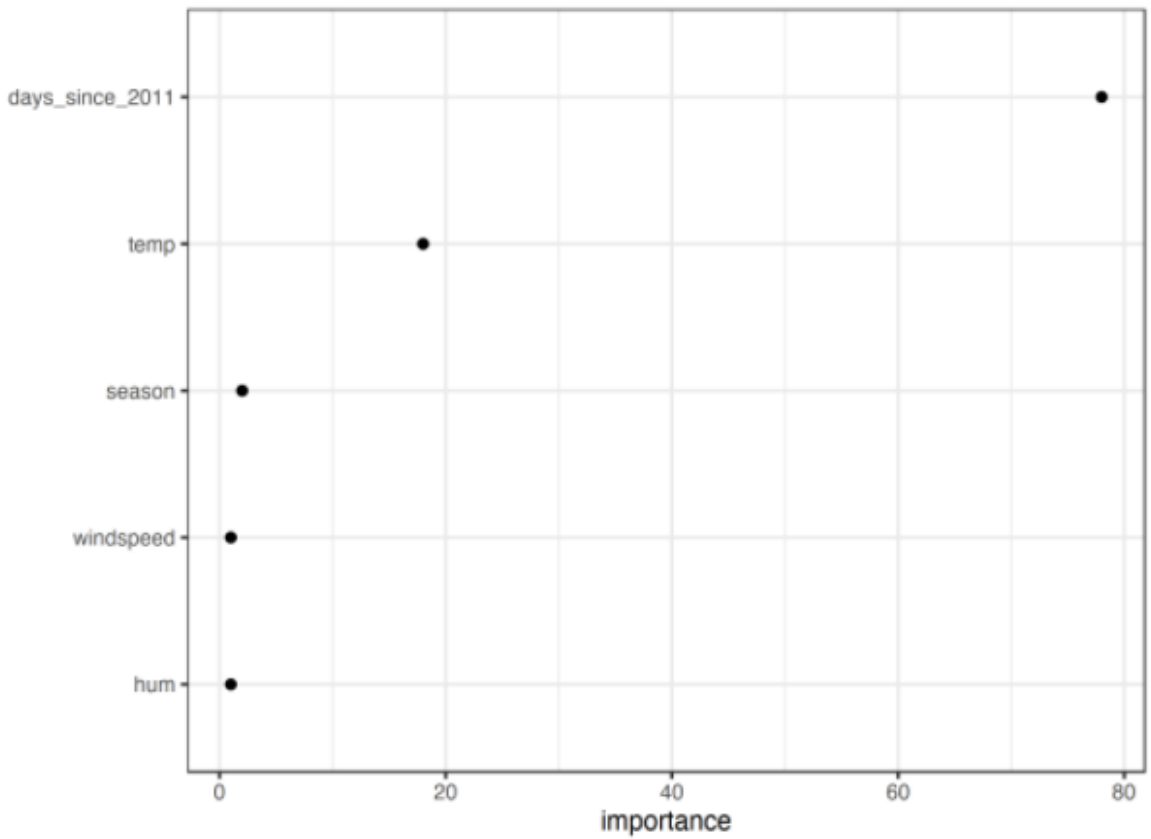
Decision Tree: bike rental data



Source: Molnar (2025)

Global Surrogate Model: Illustration

Decision Tree: bike rental data



Source: Molnar (2025)

Global Surrogate Model: Advantages and Limits

Advantages

1. The surrogate model method is **simple**, **flexible,** and **intuitive**.
2. However, it leads to conclusions about the model and not about the **data**, since the surrogate model never sees the real outcome.

Limits

1. It is not clear what the best **cut-off** for the R^2 is in order to be confident that the surrogate model is close enough to the black box model.
2. It could happen that the interpretable model is very close for one subset of the dataset, but widely divergent for another subset.

Partial Dependence Plot (PDP)

Partial Dependence Plot: Definition

Definition: Partial Dependence Plot

A **Partial Dependence Plot** (PDP) illustrates the marginal effect of one (or several) features on the predicted outcome of a machine learning model (Friedman, 2001).

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232.

Framework and Notations

Notations

- Denote by X_s the features for which the partial dependence function should be plotted and Ω_s the universe of its realization.
- Denote by x_s a realization of the random variable X_s.
- Denote by X_c the other features used in the machine learning model $\widehat{f}$.

Partial Dependence Function

Definition: Partial Dependence Function

The **partial dependence function** $pd_s(\cdot)$ is defined as the expectation of the model output $\widehat{f}(\cdot)$ over the marginal distribution of all variables other than X_s:

$$pd_s(x_s) = \mathbb{E}_{X_c}\left[\widehat{f}(x_s, X_c)\right] = \int \widehat{f}(x_s, x_c)\, \mathrm{d}\mathbb{P}(x_c), \qquad \forall x_s \in \Omega_s.$$

Remark: *The PDP differs from the conditional expectation $\mathbb{E}_{X_c|X_s}[\widehat{f}(x_s, X_c)]$, where the expectation is taken with respect to the conditional distribution of X_c given $X_s = x_s$. In practice, this means that PDPs implicitly assume that X_s is not correlated with the other features.*

Example of Partial Dependence Function

Example: Partial Dependence Function in a Linear Model

Consider the **linear model**

$$Y = \widehat{f}(X) + \varepsilon = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \varepsilon,$$

where $\varepsilon \sim$ i.i.d.$(0, \sigma^2)$. The **partial dependence function** associated with feature X_1 is

$$pd_1(x_1) = \beta_0 + \beta_1 x_1 + \beta_2 \mathbb{E}[X_2] + \ldots + \beta_p \mathbb{E}[X_p], \qquad \forall x_1 \in \mathbb{R}.$$

Example of Partial Dependence Function

Example: Partial Dependence Function in a Logit Model

Consider the **logit model**

$$\Pr(Y = 1 \mid X) = \widehat{f}(X) = \Lambda(\beta_0 + \beta_1 X_1 + \beta_2 X_2)\,,$$

where $\Lambda(z) = \dfrac{\exp(z)}{1 + \exp(z)}$ is the cdf of the logistic distribution. The **partial dependence function** associated with feature X_1 is

$$\begin{aligned} pd_1(x_1) &= \mathbb{E}_{X_2}[\Lambda(\beta_0 + \beta_1 x_1 + \beta_2 X_2)]\,, \qquad \forall x_1 \in \mathbb{R}, \\ &= \int_{-\infty}^{+\infty} \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)} g_2(x_2)\,\mathrm{d}x_2, \end{aligned}$$

where $g_2(\cdot)$ is the pdf of the marginal distribution of X_2.

Empirical Partial Dependence Function

Definition: Empirical Partial Dependence Function

In practice, the partial dependence function is **estimated** by averaging over the sample:

$$\widehat{pd}_s(x_s) = \frac{1}{n}\sum_{i=1}^{n}\widehat{f}(x_s, x_c^{(i)}), \qquad \forall x_s \in \Omega_s.$$

$$\begin{pmatrix} x_s & x_c^{(1)} \\ x_s & x_c^{(2)} \\ \vdots & \vdots \\ x_s & x_c^{(n)} \end{pmatrix}$$

Remark: x_s can take any value within the domain of X_s, not only the observed values in the dataset. For instance, if X_s corresponds to *age*, then x_s can take any value between $\min(\text{age})$ and $\max(\text{age})$.

Partial Dependence Plot: Definition

Definition: Partial Dependence Plot

A **Partial Dependence Plot** (PDP) displays the values of the estimated partial dependence function $\widehat{pd}_s(x_s)$ for a grid of selected values of the feature X_s. It represents the **average marginal effect** of X_s on the model prediction, obtained by averaging over the joint distribution of all other features.

- The PDP reveals whether the relationship between the target and a feature is linear, monotonic, or more complex.
- For a discussion of causal interpretations of PDP, see Zhao and Hastie (2019).

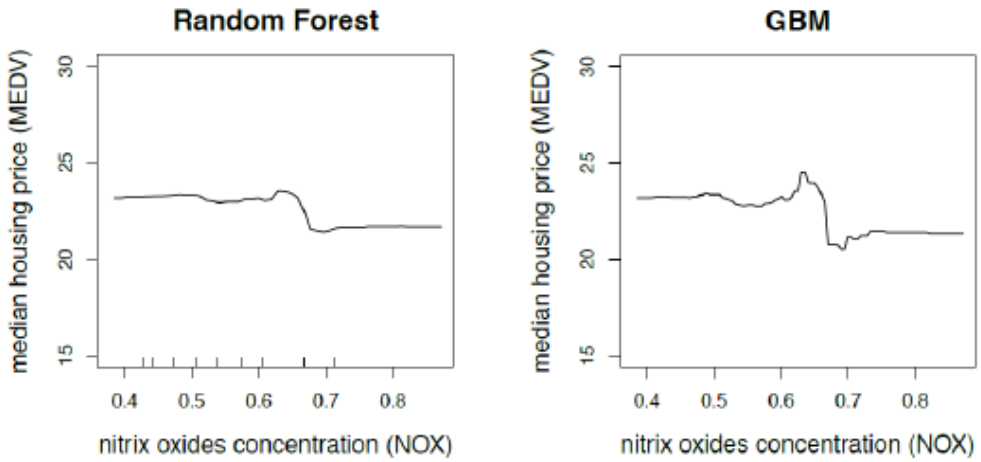
Zhao, Q. and Hastie, T. (2019). Causal Interpretations of Black-Box Models. *Journal of Business & Economic Statistics*, 37(2), 272–281.

Pseudo-algorithm

Pseudo-algorithm for Computing a PDP for a Continuous Feature

1. Select the feature X_s of interest.
2. Define a grid of values $x_s \in \{c_1, \ldots, c_k\} \subset \Omega_s$.
3. For each x_s in the grid:
    1. Replace the realizations of X_s in the dataset by x_s.
    2. Compute $\widehat{f}(x_s, x_c^{(i)})$ for each instance $i = 1, \ldots, n$.
    3. Average the predictions across all instances.
4. Plot the curve of $\widehat{pd}_s(x_s)$ against x_s.

Partial Dependence Plot: Continuous Feature and Regression Model



Example: PDP for a Continuous Feature in a Regression Model

Zhou and Hastie (2019) consider two predictive models for housing price: random forest and gradient boosting machine. PDPs suggest that the housing price is insensitive to air quality until it reaches certain pollution level around 0.65.

Partial Dependence Plot

Categorical Features

- For categorical features, the partial dependence is straightforward to compute.
- For each category, we obtain a PDP estimate by forcing all data instances to take that category value.

Categorical Feature

The estimated partial dependence function for a categorical feature $X_s \in \{c_1, \ldots, c_p\}$ is defined as

$$\widehat{pd}_s(c_j) = \frac{1}{n} \sum_{i=1}^{n} \widehat{f}(c_j, x_c^{(i)}), \qquad \forall j = 1, \ldots, p.$$

Partial Dependence Plot: Categorical Feature and Regression Model

Figure 1: Example of PDP for a categorical feature and a regression model

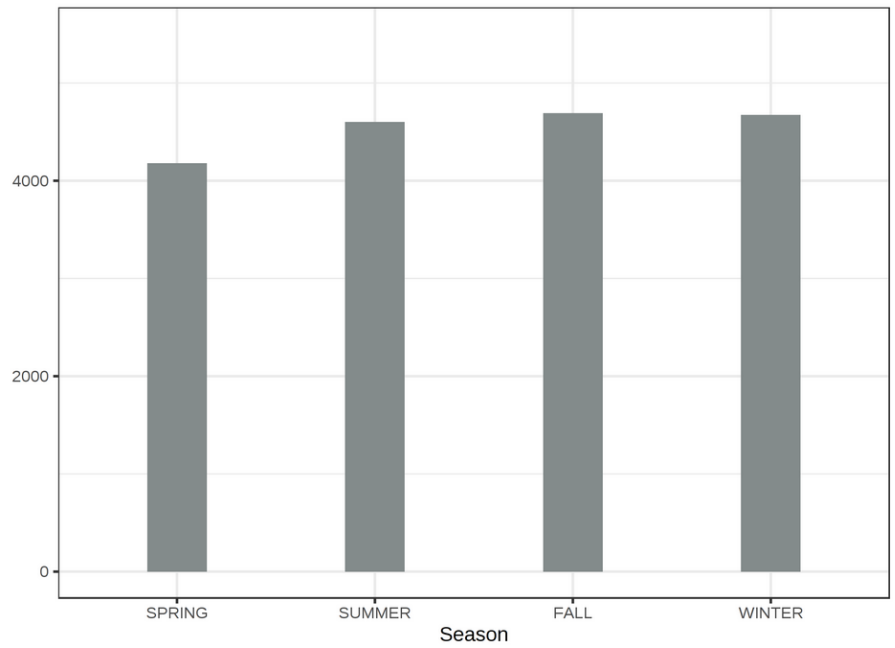


FIGURE 5.3: PDPs for the bike count prediction model and the season. Unexpectedly all seasons show similar effect on the model predictions, only for spring the model predicts fewer bicycle rentals.

Source: Molnar (2025)

Partial Dependence Plot for Classification Model

Classification model

When the target variable Y is **categorical**, the PD function displays the average marginal effect of the feature s on the **conditional probability** $\Pr\left(Y = 1 | x_s, x_c^{(i)}\right)$ as

$$pd_s(x_s) = \frac{1}{n}\sum_{i=1}^{n} \Pr\left(Y = 1 | x_s, x_c^{(i)}\right) \quad \forall x_s \in \Omega_s$$

Similarly, for a categorical feature $X_S \in \{c_1, \ldots, c_p\}$, we have

$$pd_s(c_j) = \frac{1}{n}\sum_{i=1}^{n} \Pr\left(Y = 1 | c_j, x_c^{(i)}\right) \quad \forall j = 1, \ldots, p$$

Partial Dependence Plot: Numerical Feature and Classification Model

Figure 2: Example of PDP for a numerical feature and a classification model

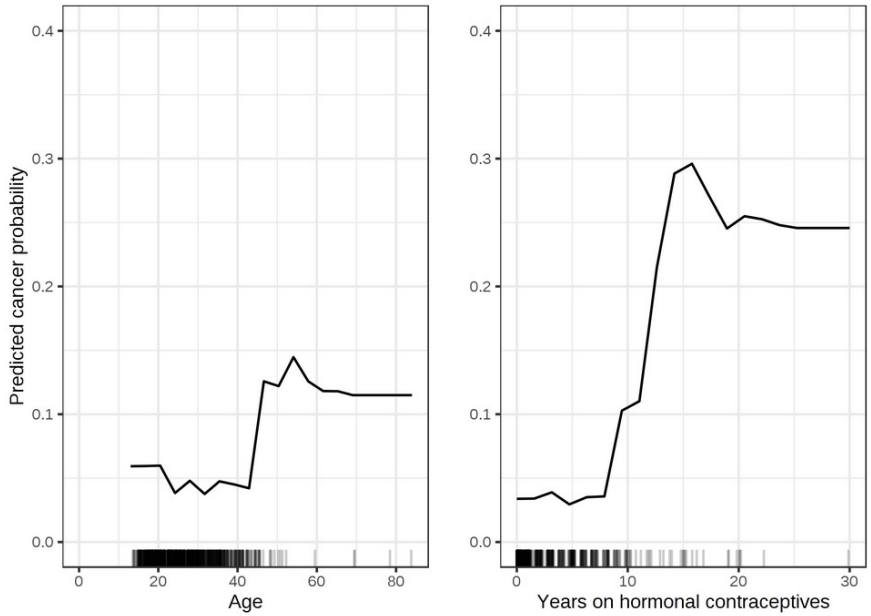


FIGURE 5.4: PDPs of cancer probability based on age and years with hormonal contraceptives. For age, the PDP shows that the probability is low until 40 and increases after. The more years on hormonal contraceptives the higher the predicted cancer risk, especially after 10 years. For both features not many data points with large values were available, so the PD estimates are less reliable in those regions.

Source: Molnar (2025)

Partial Dependence Plot: Advantages and Limits

Advantages

1. Interpretation is clear: The PDP shows the marginal effect of a given feature X_s on the average prediction.
2. Easy to implement: The PDP does not require re-estimation of the model.
3. Causal interpretation: We intervene on a feature and measure the changes in the predictions (Zhao and Hastie (2019)).

Partial Dependence Plot: Advantages and Limits

Limits

1. Assumption of independence is the biggest issue with PDP.
   - *PDPs assume that the feature of interest X_s is **independent** of the other features X_c.*
   - *In practice, this assumption is often violated: features are correlated, and PDPs may display unrealistic combinations of values.*
   - **Solution:** Accumulated Local Effects (ALE) curves.
2. Maximum number of features: the PDP analysis is limited to 2 features.
3. Feature distribution: some PDP do not show the feature distribution and it can be misleading because you might overinterpret regions with almost no data.
4. Heterogeneous effects across instances might be hidden because PDP only show the *average* marginal effects.
   - **Solution:** Individual Conditional Expectation (ICE) curves.

Limitation of Partial Dependence Plots

Example: PDP and Correlated Features

Consider a housing price prediction model. Let $X_s = \textit{House Size}$ and $X_c = \textit{Number of Rooms}$. These two features are strongly correlated.

When computing $pd_s(x_s)$, the PDP may evaluate predictions for unrealistic combinations such as:

$$\text{House Size} = 300\,\text{m}^2 \qquad \text{with} \qquad \text{Number of Rooms} = 1.$$

Such combinations do not occur in practice, and the resulting PDP can therefore be misleading.

Implementing Partial Dependence Plots in `scikit-learn`

How to Compute and Visualize PDPs with `scikit-learn`?

There are two complementary approaches to implement PDPs:

1. Using `partial_dependence`

- Returns the underlying arrays with values of the feature (grid) and the corresponding average predictions.
- Gives flexibility to build custom visualizations (e.g., line plots for continuous variables, bar charts for categorical variables).

2. Using `PartialDependenceDisplay`

- High-level API that directly produces PDP figures.
- Simplifies the plotting of both continuous and categorical regressors.
- Allows customization of axes, labels, and layout.

In practice, one can use `partial_dependence` when numerical values are needed for further analysis, and `PartialDependenceDisplay` when the goal is quick visualization.

Partial Dependence in `scikit-learn`

partial_dependence

```
sklearn.inspection.partial_dependence(estimator, X, features, *,
sample_weight=None, categorical_features=None, feature_names=None,
response_method='auto', percentiles=(0.05, 0.95), grid_resolution=100,
custom_values=None, method='auto', kind='average') #
```

Partial dependence of `features`.

Partial dependence of a feature (or a set of features) corresponds to the average response of an estimator for each possible value of the feature.

Source: scikit-learn documentation

Partial Dependence Display in `scikit-learn`

PartialDependenceDisplay

```
class sklearn.inspection.PartialDependenceDisplay(pd_results, *, features,
feature_names, target_idx, deciles, kind='average', subsample=1000,
random_state=None, is_categorical=None)
```

Partial Dependence Plot (PDP) and Individual Conditional Expectation (ICE).

It is recommended to use `from_estimator` to create a `PartialDependenceDisplay`. All parameters are stored as attributes.

For general information regarding `scikit-learn` visualization tools, see the Visualization Guide. For guidance on interpreting these plots, refer to the Inspection Guide.

For an example on how to use this class, see the following example: Advanced Plotting With Partial Dependence.

Source: scikit-learn documentation

Partial Dependence Display

Example: Partial Dependence Display

```
model = GradientBoostingRegressor(random_state=42)
model.fit(X_data, y_data)

plt.figure()
PartialDependenceDisplay.from_estimator(
    model,
    X_data,
    features=["Variable Name"],
    grid_resolution=50
    )
plt.show()
```

Google Colab: Click here.

Exercise: Partial Dependence Plot

Exercise: Partial Dependence Plot for an Unemployment Model

Consider quarterly U.S. macroeconomic data (1959Q1–2009Q3) from the `statsmodels` *macrodata* dataset. We are interested in predicting the **unemployment rate** using the following regressors:

- GDP growth rate,
- Consumption growth rate,
- Investment growth rate,
- Inflation rate,
- 3-month T-bill rate.

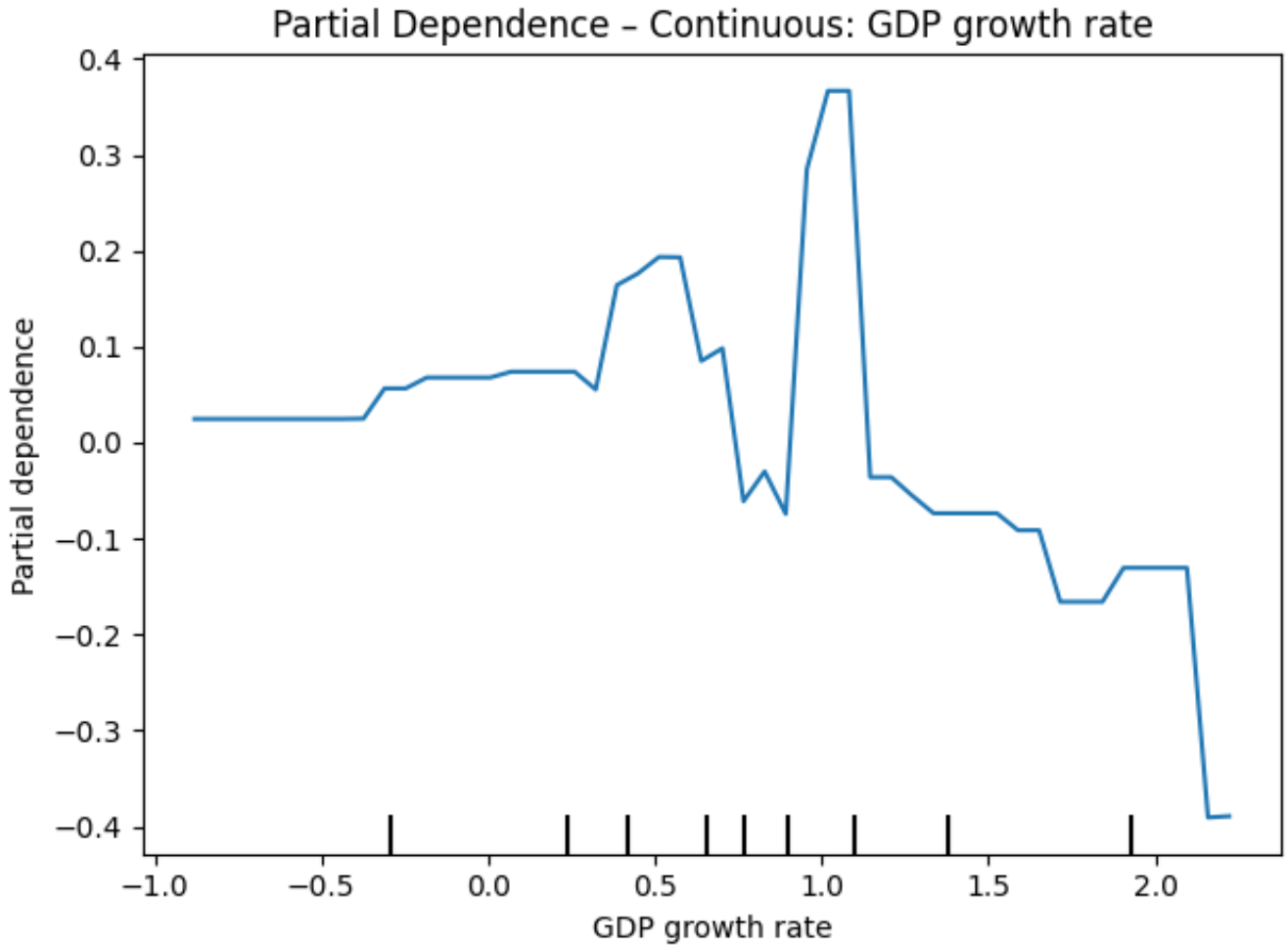
The T-bill rate is transformed into a categorical variable (low/mid/high bins), and a Gradient Boosting Regressor is fitted on this dataset.

The objective is to analyze the model using **Partial Dependence Plots (PDP)**:

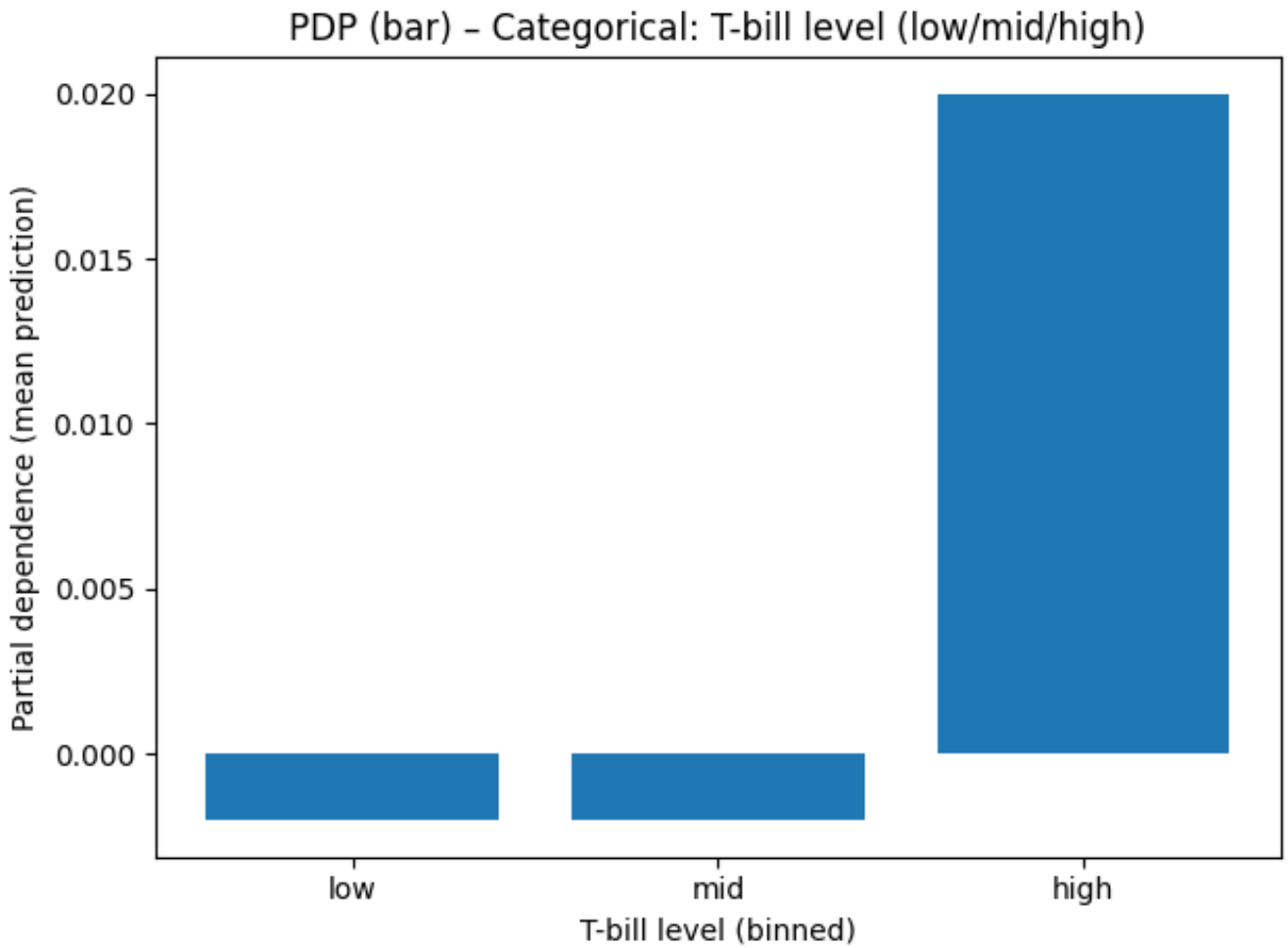
- Plot the PDP for the continuous regressor **GDP growth rate**.
- Plot the PDP for the **T-bill rate** as a categorical regressor.

Google Colab: Click here.

Exercise: Python Output



Exercise: Python Output



Accumulated Local Effects (ALE)

Accumulated Local Effects: Definition

Definition: Accumulated Local Effects

Accumulated Local Effects (ALE) plots illustrate how features influence the predictions of a machine learning model **on average**, while explicitly accounting for potential dependencies among the features.

Apley, D.W. (2016). Visualizing the effects of predictor variables in black-box supervised learning models. *arXiv preprint* arXiv:1612.08468.

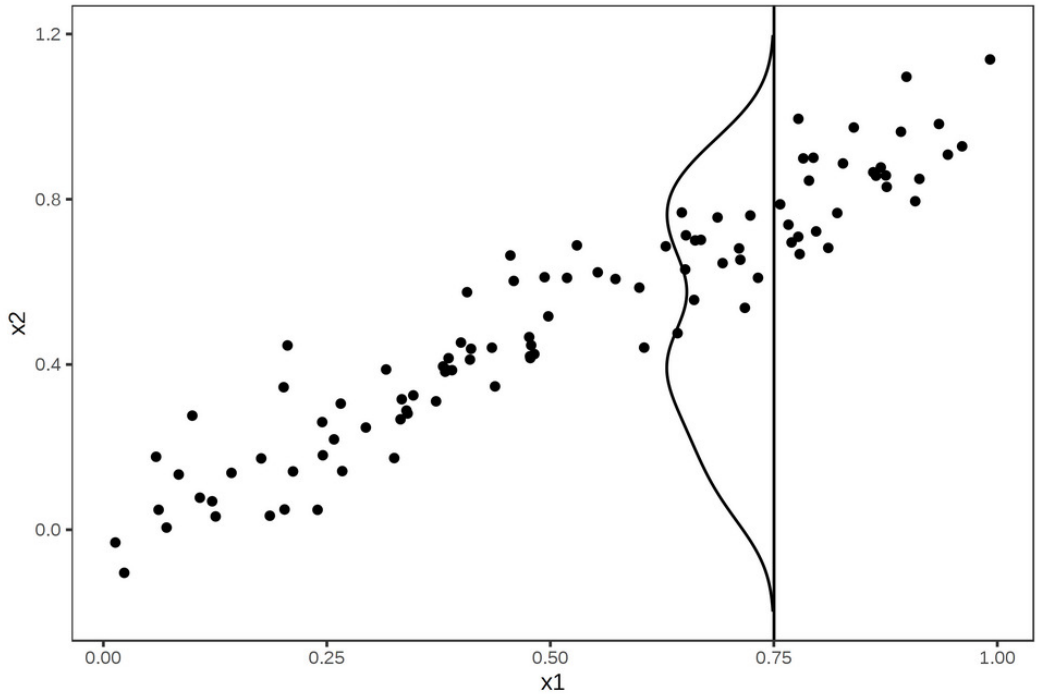
Independence assumption

Independence assumption

- To calculate the feature effect of x_1 at a given value, say 0.75, the PDP replaces x_1 of all instances with 0.75.
- It leads to falsely assume that the distribution of x_2 at $x_1 = 0.75$ is the same as the marginal distribution of x2.
- This results in unlikely combinations of x_1 and x_2 (e.g. $x_2 = 0.2$ at $x_1 = 0.75$), which the PDP uses for the calculation of the average effect.
- The figure displays two correlated features and illustrates the fact that PDP average predictions of unlikely instances.

Marginal Distribution

Figure 3: Marginal Distribution



Conditional expectation and M-plots

Conditional expectation

- We could average over the **conditional distribution** of the feature, meaning at a grid value of x_1, we average the predictions of instances with a similar x_1 value.
- The solution for calculating feature effects using the conditional distribution is called Marginal Plots or M-Plots
- M-plots are obtained like PDP but using the conditional distribution:

$$\mathbb{E}_{X_c|X_s=x}\left[\widehat{f}(x, X_c)\right]$$

PDP vs. M-Plots

Formal Definitions of PDP and M-Plots

Partial Dependence Plots (PDPs) average the predictions over the **marginal distribution** of the other features:

$$pd_s(x) = \mathbb{E}_{X_c}\left[\widehat{f}(x, X_c)\right], \qquad \forall x \in \Omega_s.$$

M-Plots average the predictions over the **conditional distribution** of the other features given $X_s = x$:

$$M_s(x) = \mathbb{E}_{X_c|X_s=x}\left[\widehat{f}(x, X_c)\right], \qquad \forall x \in \Omega_s.$$

Limits of the M-plot

Example: Limits of the M-plot

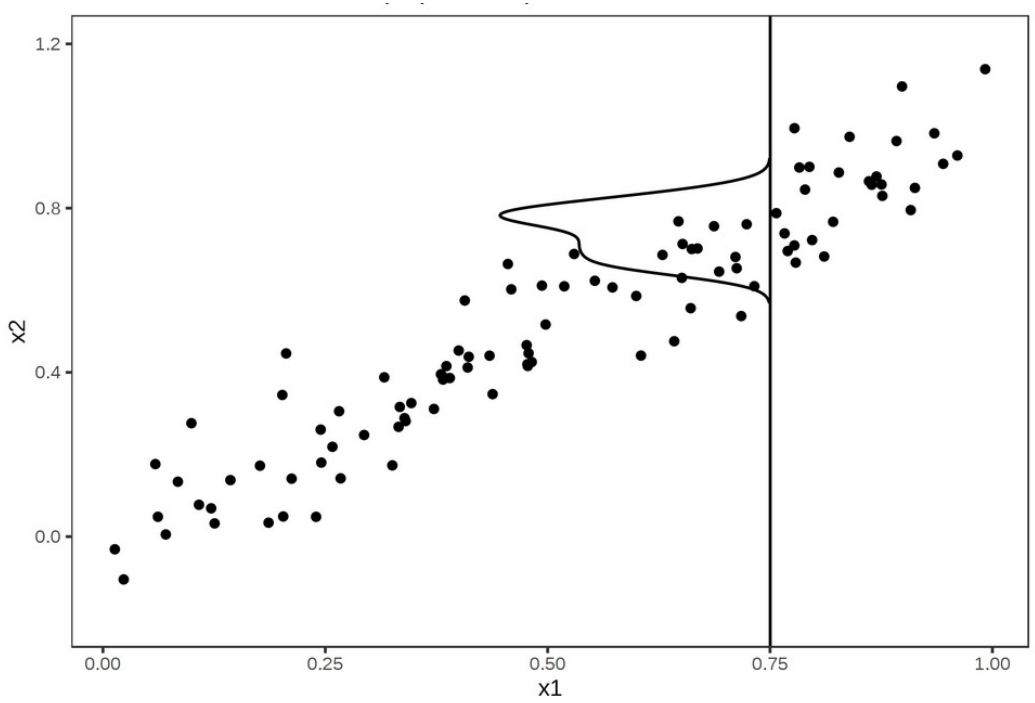
Consider a model which predicts the value of a house depending on the number of rooms and the size of the living area.

If we average the predictions of all houses of about 30 m^2, we estimate the **combined effect** of livingarea and number of rooms, because of their correlation.

- Suppose that the living area has no effect on the predicted value of a house, only the number of rooms has.
- The M-Plot would still show that the size of the living area increases the predicted value, since the number of rooms increases with the living area.

Conditional Distribution

Figure 4: Conditional Distribution



Accumulated Local Effects

Accumulated Local Effect

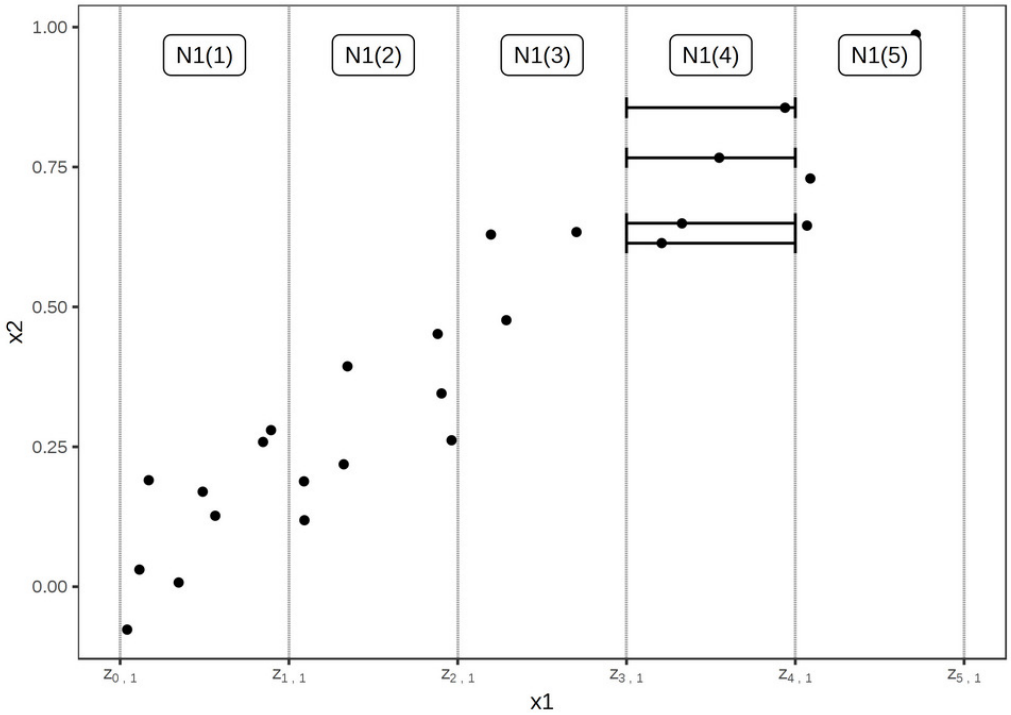
ALE solves the combined-effect problem by calculating differences in predictions instead of averages, yet using conditional distributions.

For the calculation of ALE for feature x_1, which is correlated with x_2:

1. First, we divide the feature space of x_1 into several intervals.
2. In each data point in a given interval, we calculate the difference in predictions when replacing the feature value by, respectively, the upper and lower limit of the interval.
3. In each interval, we compute the average difference in predictions.
4. Average differences are then accumulated.

Accumulated Local Effects: Intuition

Figure 5: Intuition of the ALE



Source: Molnar (2025)

Example of Accumulated Local Effects

Example: Accumulated Local Effects

For the calculation of ALE for the *living area*, which is correlated with the number of rooms:

1. First, we divide the feature space of the *living area* into several intervals.
2. For each house between 79 and 81 m^2, we calculate the difference in estimated price when replacing the *living area* value by 81 and then by 79.
3. For this interval, we compute the average difference in estimated price.
4. We do the same in all other intervals: (77-79), (75-77), ...
5. Average differences are then accumulated.

Accumulated Local Effects: Definition

Definition: Accumulated Local Effect

The **Accumulated Local Effect (ALE)** function averages the local changes in the model predictions with respect to a feature X_s, and then accumulates these changes over the range of X_s values:

$$ALE_s(x_s) = \int_{z_{0,s}}^{x} \mathbb{E}_{X_c|X_s=z_s}\left[\frac{\partial \widehat{f}(X_s, X_c)}{\partial X_s}\middle| X_s = z_s\right] \mathrm{d}z_s \quad \forall x_s \in \Omega_s$$

where $z_{0,s}$ is the minimum value of X_s at which the ALE curve is computed.

Remarks

Remarks

1. A discrete transcription of the ALE can be written as

$$ALE_s(x) = \sum_{x_S = z_{0,s}}^{x} \mathbb{E}_{X_c | X_S = x_S} \left[\Delta_s \widehat{f}(x_s, X_c) \right]$$

wher $\Delta_s \widehat{f}(x_s, X_c)$ denotes the marginal variation of the outcome model obtained for a marginal increase of X_s.

2. The ALE can also be expressed as

$$\begin{aligned} ALE_s(x) &= \int_{z_{0,s}}^{x} \mathbb{E}_{X_c | X_S = z_S} \left[\frac{\partial \widehat{f}(X_s, X_c)}{\partial X_s} \middle| X_s = z_s \right] dz_s \\ &= \int_{z_0,s}^{x} \int_{x_c} \left[\frac{\partial \widehat{f}(X_s, X_c)}{\partial X_s} \middle| X_s = z_s \right] \mathbb{P}(x_c | z_s) \, dx_c dz_s \end{aligned}$$

Accumulated Local Effects: Interpretation

Interpretation

$$ALE_s(x_s) = \int_{z_{0,s}}^{x} \mathbb{E}_{X_c|X_s=z_s}\left[\frac{\partial \widehat{f}(X_s, X_c)}{\partial X_s}\middle| X_s = z_s\right] \mathrm{d}z_s \quad \forall x_s \in \Omega_s$$

1. ALE averages the changes of predictions (defined as the gradient), not the predictions itself. The change is defined as the gradient.
2. The derivative isolates the effect of the feature of interest and blocks the effect of correlated features.

Centered ALE

Definition: Centered ALE

The **centered Accumulated Local Effect (ALE)** function is defined as

$$ALE_s^{\text{cent}}(x) = ALE_s(x) - \mathbb{E}_{X_s}[ALE_s(X_s)] \,.$$

Centering ensures that the ALE function has mean zero over the distribution of X_s, which makes it directly comparable across features.

Estimation of Accumulated Local Effects

Estimation

- To estimate local effects, we divide the feature into **many intervals** and compute the **differences in the predictions**.
- The relevant dimension (i.e. on X_s) of the feature space Ω_s is divided into K intervals beginning with the starting point $z_{0,s}$.
- The upper boundary of the k^{th} interval is denoted by $z_{k,s}$, the lower boundary by $z_{k-1,s}$.
- Denote by $N_s(k)$ the k^{th} interval, i.e. $]z_{k-1,s}, z_{k,s}]$.
- Denote by $n_s(k)$ the number of instance in this interval.

Estimation of Accumulated Local Effects

Estimator of the conditional expectation

$$ALE_s(x_s) = \int_{z_{0,s}}^{x} \mathbb{E}_{X_c|X_s=z_s}\left[\frac{\partial \widehat{f}(X_s, X_c)}{\partial X_s}\middle| X_s = z_s\right] dz_s \quad \forall x_s \in \Omega_s$$

- Each instance within an interval is shifted to the upper and lower limit of the interval and the total difference of the prediction is calculated.

- By averaging these approximations over all observations within the k^{th} interval, we get a rough estimator for the conditional expectation

$$\frac{1}{n_s(k)} \sum_{i:x_s^{(i)} \in N_S(k)} \frac{\widehat{f}\left(z_{k,s}, x_c^{(i)}\right) - \widehat{f}\left(z_{k-1,s}, x_c^{(i)}\right)}{z_{k,s} - z_{k-1,s}} \xrightarrow{p} \mathbb{E}_{X_c|X_s}\left(\frac{\partial \widehat{f}(X_s, X_c)}{\partial X_s}\middle| N_s(k)\right)$$

Accumulated Local Effects

Definition: Estimated ALE

The (uncentered) **estimated Accumulated Local Effect (ALE)** function is defined as

$$\widehat{ALE}_s(x) = \int_{z_{0,s}}^{x} \sum_{k=1}^{K} \frac{1_{\{x_S \in N_S(k)\}}}{n_s(k)} \sum_{i:x_s^{(i)} \in N_S(k)} \frac{\widehat{f}(z_{k,s}, x_c^{(i)}) - \widehat{f}(z_{k-1,s}, x_c^{(i)})}{z_{k,s} - z_{k-1,s}} \, \mathrm{d}z_s,$$

where $N_s(k)$ denotes the k-th interval (neighborhood) of the feature X_s, and $n_s(k)$ is the number of instances falling in this interval.

Accumulated Local Effects: Numerical Example

Toy Example

Consider the model

$$\widehat{f}(x_1, x_2) = 2x_1 + x_2,$$

with X_1 as the feature of interest and X_2 as the other feature. Suppose X_1 takes values in $[0, 2]$, split into two intervals:

$$N_1(1) = [0, 1], \qquad N_1(2) = [1, 2].$$

For an observation with $x_2 = 3$:

$$\widehat{f}(1, 3) - \widehat{f}(0, 3) = (2 \cdot 1 + 3) - (2 \cdot 0 + 3) = 2,$$

$$\widehat{f}(2, 3) - \widehat{f}(1, 3) = (2 \cdot 2 + 3) - (2 \cdot 1 + 3) = 2.$$

Thus, the local effect in each interval is

$$\Delta = 2.$$

Accumulating from 0 to x_1 gives

$$ALE_1(x_1) = 2 \cdot x_1.$$

Interpretation: the ALE curve recovers the true marginal effect of X_1, which is constant in this linear model.

Accumulated Local Effects

Remarks

The notation

$$\sum_{k=1}^{K} \mathbf{1}_{x_S \in N_S(k)} \frac{1}{n_s(k)} \sum_{i:x_s^{(i)} \in N_S(k)} \frac{\widehat{f}\left(z_{k,s}, x_c^{(i)}\right) - \widehat{f}\left(z_{k-1,s}, x_c^{(i)}\right)}{z_{k,s} - z_{k-1,s}}$$

correspond to a step function given by

$$g(x_s) = \begin{cases} \frac{1}{n_S(1)} \sum_{i:x_s^{(i)} \in N_S(1)} \frac{\widehat{f}\left(z_{1,s}, x_c^{(i)}\right) - \widehat{f}\left(z_{0,s}, x_c^{(i)}\right)}{z_{1,s} - z_{0,s}} & \text{if } x_s \in N_s(1) \\ \ldots\ldots & \\ \frac{1}{n_S(K)} \sum_{i:x_s^{(i)} \in N_S(K)} \frac{\widehat{f}\left(z_{K,s}, x_c^{(i)}\right) - \widehat{f}\left(z_{K-1,s}, x_c^{(i)}\right)}{z_{K,s} - z_{K-1,s}} & \text{if } x_s \in N_s(K) \end{cases}$$

Accumulated Local Effects

Accumulated Local Effects

By integrating over this step function (locally estimated derivatives) the (local) changes are accumulated and we get a continous function by parts.

$$g(x_s) = \begin{cases} a \\ a+b \\ a+b+c \end{cases} \quad => \quad \int_0^x g(z_s)\, dz_s = \begin{cases} ax + \gamma_1 \\ (a+b)\, x + \gamma_2 \\ (a+b+c)\, x + \gamma_3 \end{cases}$$

Accumulated Local Effects

Definition: Centered Estimated ALE

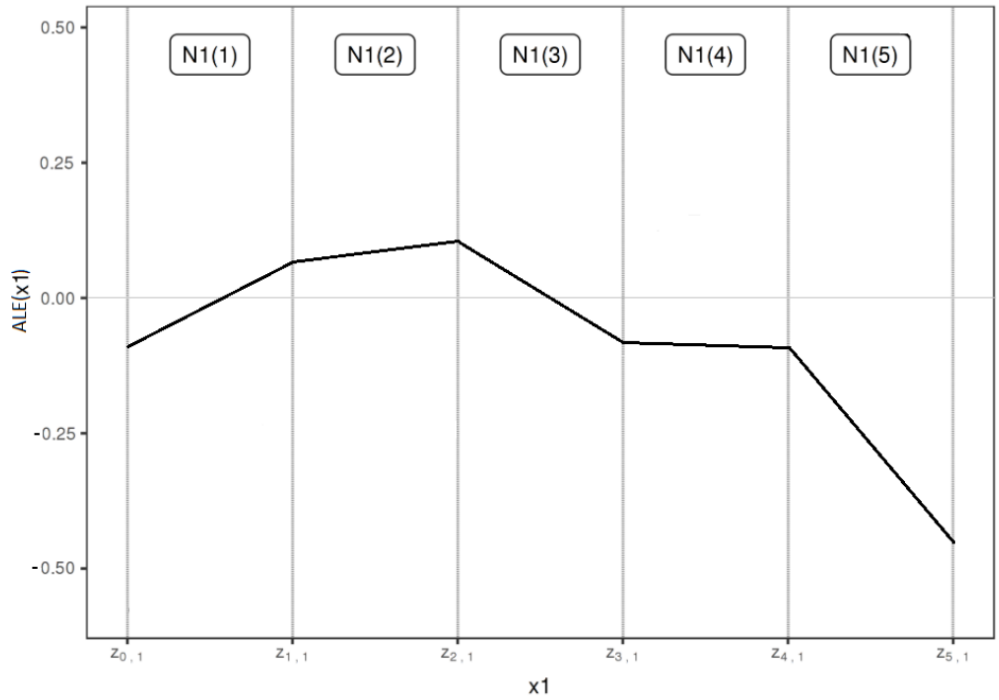
The **centered estimated Accumulated Local Effect (ALE)** function is defined as

$$\widehat{ALE}_s^{\text{cent}}(x) = \widehat{ALE}_s(x) - \frac{1}{n}\sum_{i=1}^{n} \widehat{ALE}_s\left(x_s^{(i)}\right) .$$

Centering ensures that the estimated ALE function has mean zero across the sample, which removes arbitrary vertical shifts and facilitates comparison across features.

Example of Accumulated Local Effects

Figure 6: Example with 5 intervals



Source: Molnar (2025)

Apley's formula

Apley's formula

- As both the centered and the uncentered ALE estimations are piecewise linear functions (integration over a step function), one can first calculate the ALE at the interval boundaries and interpolate in a second step.
- Denote by $k_s(x)$ the number of the interval that contains x, i.e. $x \in]z_{k_s(x)-1,s}, z_{k_s(x),s}]$.

Apley, D.W. (2016), Visualizing the effects of predictor variables in black box supervised learning models. arXiv preprint arXiv:1612.08468.

Apley's Formula

Definition: Apley's Formula

The **estimated Accumulated Local Effect (ALE)** function can be computed using Apley's formula:

$$\widehat{ALE}_s(x) = \sum_{k=1}^{k_s(x)} \frac{1}{n_s(k)} \sum_{i:x_s^{(i)} \in N_s(k)} \left(\widehat{f}(z_{k,s}, x_c^{(i)}) - \widehat{f}(z_{k-1,s}, x_c^{(i)}) \right),$$

where $N_s(k)$ denotes the k-th interval of feature X_s, $n_s(k)$ the number of observations in this interval, and $k_s(x)$ the index of the last interval up to x.

Apley's Formula: Numerical Example

Toy Example

Consider the model

$$\widehat{f}(x_1, x_2) = x_1 + 2x_2,$$

and let X_1 be the feature of interest. Suppose the domain of X_1 is partitioned into two intervals:

$$N_1(1) = [0, 1], \qquad N_1(2) = [1, 2],$$

with grid points $z_{0,1} = 0$, $z_{1,1} = 1$, $z_{2,1} = 2$.
For an observation with $x_2 = 1$:

$$\widehat{f}(1, 1) - \widehat{f}(0, 1) = (1 + 2 \cdot 1) - (0 + 2 \cdot 1) = 1,$$

$$\widehat{f}(2, 1) - \widehat{f}(1, 1) = (2 + 2 \cdot 1) - (1 + 2 \cdot 1) = 1.$$

Averaging over all instances in each interval gives the same result, $\Delta = 1$. By Apley's formula, the estimated ALE is

$$\widehat{ALE}_1(x) = \sum_{k=1}^{k_1(x)} \Delta = k_1(x).$$

Interpretation: the ALE grows stepwise with x_1, showing that each unit increase in x_1 raises the prediction by 1 on average.

Accumulated Local Effects

$$\widehat{ALE}_s(x) = \sum_{k=1}^{k_S(x)} \frac{1}{n_s(k)} \sum_{i:x_s^{(i)} \in N_S(k)} \left(\widehat{f}(z_{k,s}, x_c^{(i)}) - \widehat{f}(z_{k-1,s}, x_c^{(i)}) \right).$$

- This formula yields a **step function**: within each interval, the value corresponds to the accumulated average of the local differences.
- To obtain a continuous estimator of the ALE function, one can **linearly interpolate** between the points

  $$(z_{k,s},\ \widehat{ALE}_s(z_{k,s})), \qquad k = 1, \ldots, K,$$

  with the convention $\widehat{ALE}_s(z_{0,s}) = 0$.
- Since the integral is replaced by a summation, this formulation is simpler to implement in practice.

Accumulated Local Effects

Interpretation of ALE

The **(uncentered) ALE** measures the accumulated effect of feature X_s relative to a fixed baseline $z_{0,s}$ (the lower bound of the feature domain).

Example: Interpretation of ALE

For example, if

$$\widehat{ALE}_1(2) = 1,$$

this means that when $X_1 = 2$, the model prediction is expected to be **1 unit higher** on average than the prediction at the reference point $X_1 = z_{0,1}$, after averaging over the conditional distribution of the other features.

The interpretation of uncentered ALE therefore **depends on the choice of** $z_{0,1}$, which may limit comparability across features or datasets.

Accumulated Local Effects

Interpretation of Centered ALE

The **centered ALE** measures how the local effect of X_s at a given value differs from its average effect.

Example: Interpretation of Centered ALE

Centering subtracts the average effect of the feature X_s across the sample.

$$\widehat{ALE}_1^{\text{cent}}(2) = 2,$$

means that when $X_1 = 2$, the **first-order effect** of X_1 on the model prediction is **2 units higher** than the **average first-order effect** of X_1 across the data distribution.

ALE for Continuous Features and Regression Model

Figure 7: Example of **centered ALE** for continuous features and a regression model

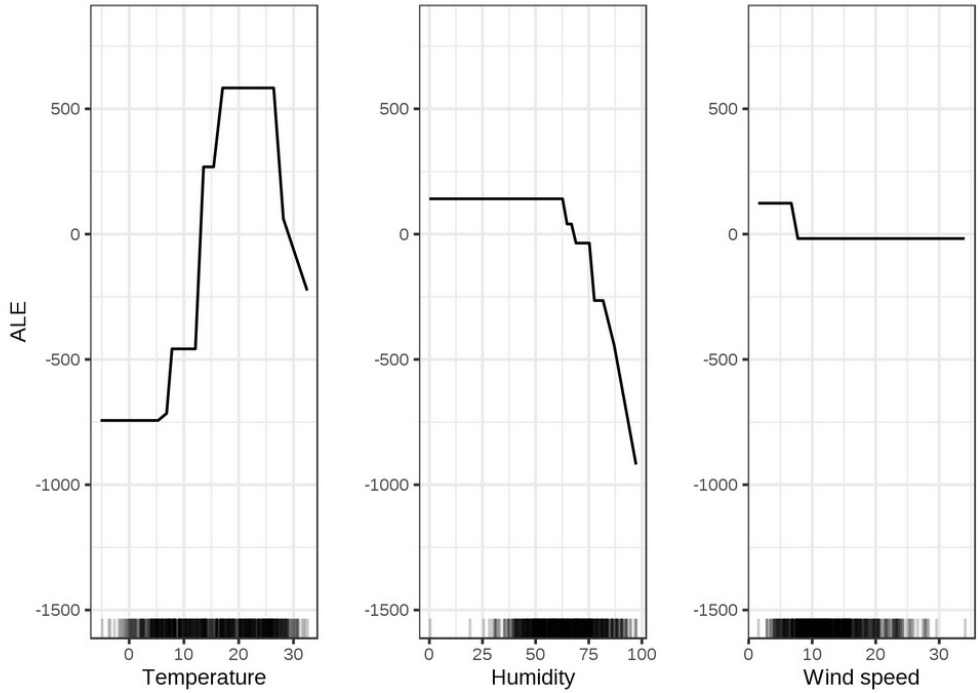


FIGURE 5.16: ALE plots for the bike prediction model by temperature, humidity and wind speed. The temperature has a strong effect on the prediction. The average prediction rises with increasing temperature, but falls again above 25 degrees Celsius. Humidity has a negative effect: When above 60%, the higher the relative humidity, the lower the prediction. The wind speed does not affect the predictions much.

Source: Molnar (2025)

Comparison with PDP

Figure 8: Comparison with PDP

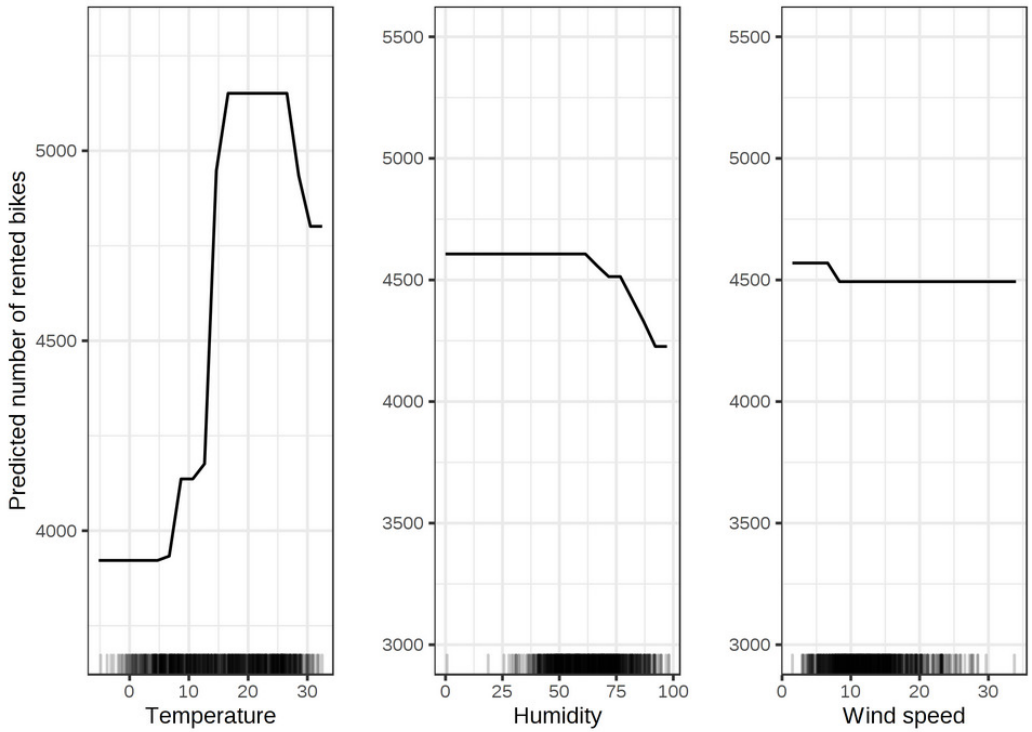


FIGURE 5.18: PDPs for temperature, humidity and wind speed. Compared to the ALE plots, the PDPs show a smaller decrease in predicted number of bikes for high temperature or high humidity. The PDP uses all data instances to calculate the effect of high temperatures, even if they are, for example, instances with the season "winter". The ALE plots are more reliable.

Source: Molnar (2025)

ALE for Continuous Features and Regression Model

Figure 9: Example of **centered ALE** for continuous features and a regression model

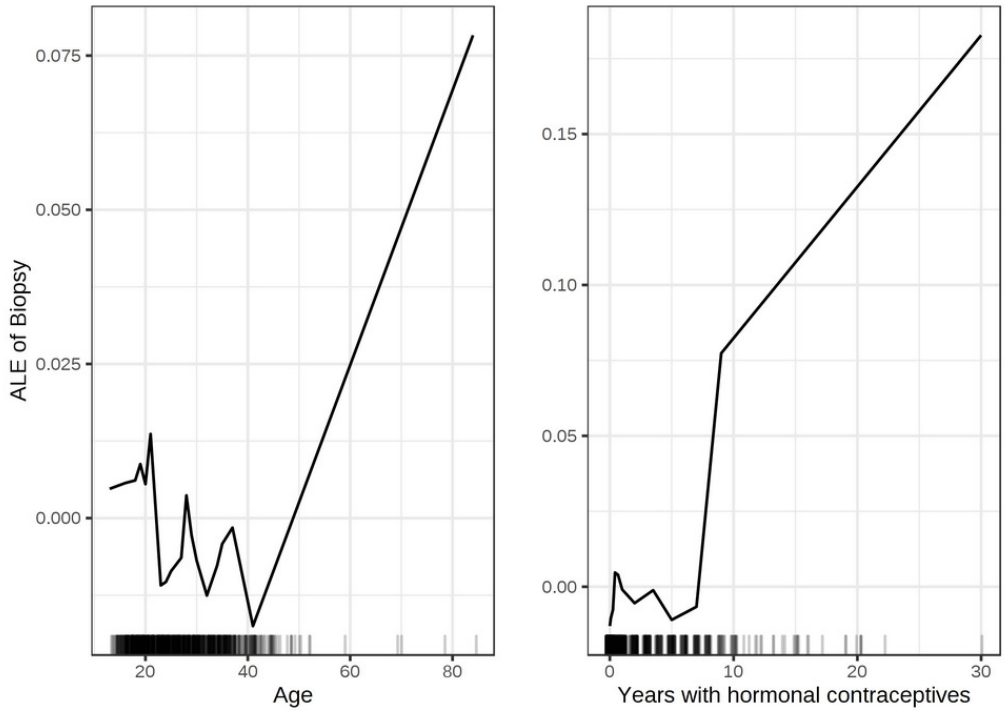


FIGURE 5.22: ALE plots for the effect of age and years with hormonal contraceptives on the predicted probability of cervical cancer. For the age feature, the ALE plot shows that the predicted cancer probability is low on average up to age 40 and increases after that. The number of years with hormonal contraceptives is associated with a higher predicted cancer risk after 8 years.

Source: Molnar (2025)

Advantages and Limits

Advantages

1. Unbiased: ALE plots still work when features are correlated.
2. Faster to compute than PDP. For PDP, we need to compute $n \times k$ predictions and compute k averages. For ALE, we need to compute $2 \times n$ predictions and compute at most k averages.
3. Interpretation is clear. For each feature value, the centered ALE curve indicates the difference between a particular average prediction at this feature value (ALE) and a particular average prediction (average ALE).

Advantages and Limits

Limits

1. ALE plots can become a bit shaky (many small ups and downs) with a high number of intervals.
2. No perfect solution for setting the number of intervals: If the number is too small, the ALE plots might not be very accurate. If the number is too high, the curve can become shaky.

How to Compute ALE with Python

Available Python Libraries

Accumulated Local Effects (ALE) are not implemented in `scikit-learn`.

To compute ALE in Python, one can use:

- Alibi Explain: provides an ALE explainer with visualization utilities.
- PyALE: a standalone package for ALE computation and plotting.

ALE Implementation with Alibi

Accumulated Local Effects

Overview

Accumulated Local Effects (ALE) is a method for computing feature effects based on the paper Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models by Apley and Zhu. The algorithm provides model-agnostic (*black box*) global explanations for classification and regression models on tabular data.

ALE addresses some key shortcomings of Partial Dependence Plots (PDP), a popular method for estimating first order feature effects. We discuss these limitations and motivate ALE after presenting the method usage.

Usage

Initialize the explainer by passing a black-box prediction function and optionally a list of feature names and target (class) names for interpretation:

```
from alibi.explainers import ALE
ale = ALE(predict_fn, feature_names=feature_names, target_names=target_names)
```

Source: Alibi Explain

ALE Implementation with Alibi

Example: ALE Implementation with Alibi

```
from alibi.explainers import ALE

ale = ALE(predict_fn, feature_names=feature_names, target_names=
     target_names)

grid_points = {0: np.array([0.1, 0.5, 1.0]),
               1: np.array([0.1, 0.5, 1.0])}

exp = ale.explain(X, features=[0, 1], grid_points=grid_points)
```

Google Colab: Click here.

Exercise: ALE vs PDP

Exercise: ALE and Partial Dependence for an Unemployment Model

Consider quarterly U.S. macroeconomic data (1959Q1–2009Q3) from the `statsmodels` *macrodata* dataset. We aim to predict the **unemployment rate** using:

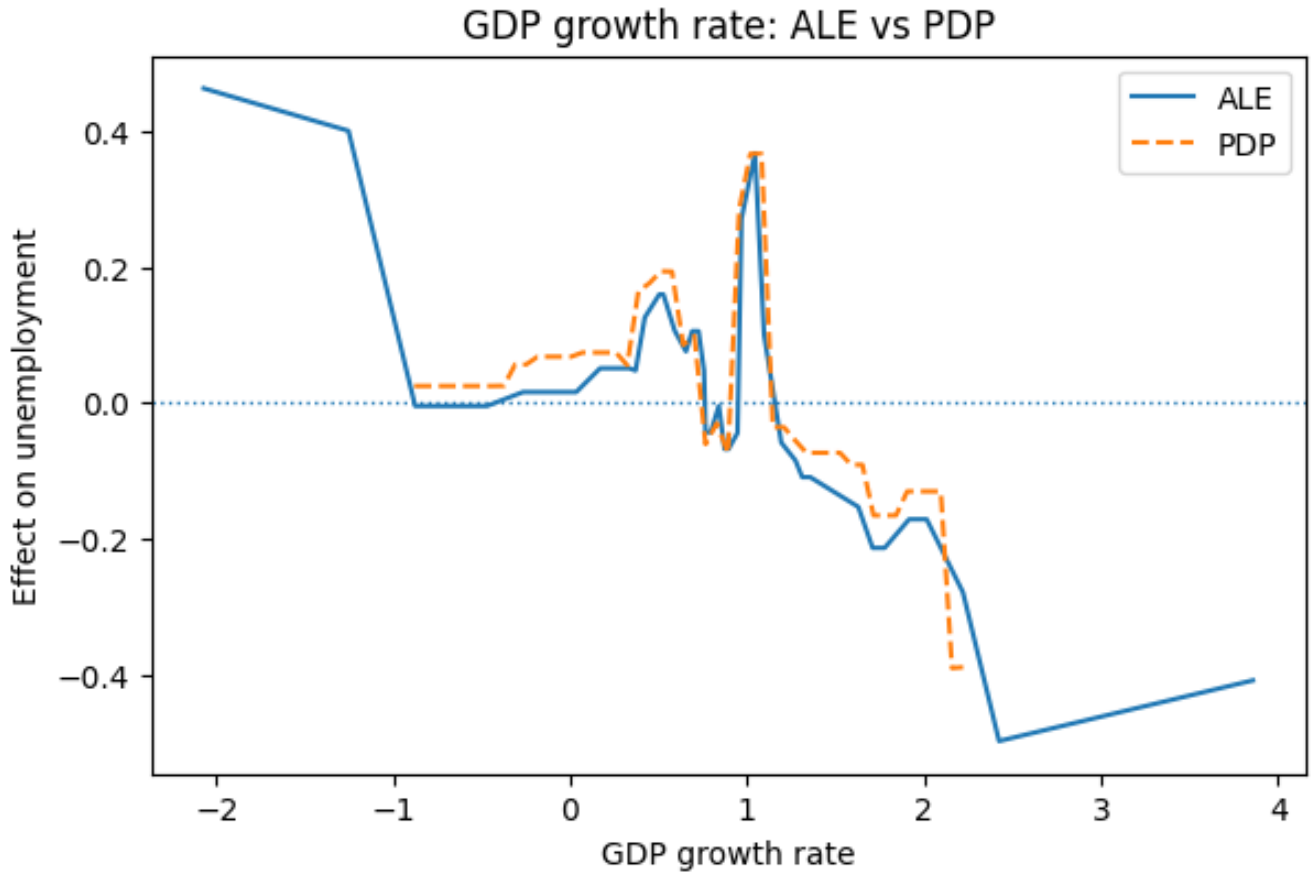
- GDP growth rate,
- Consumption growth rate,
- Investment growth rate,
- Inflation rate,
- 3-month T-bill rate discretized into three categories (*low/mid/high*).

A Gradient Boosting is trained on the feature set. The objective is to analyze the model with **Accumulated Local Effects (ALE)** and to compare them to PDPs.

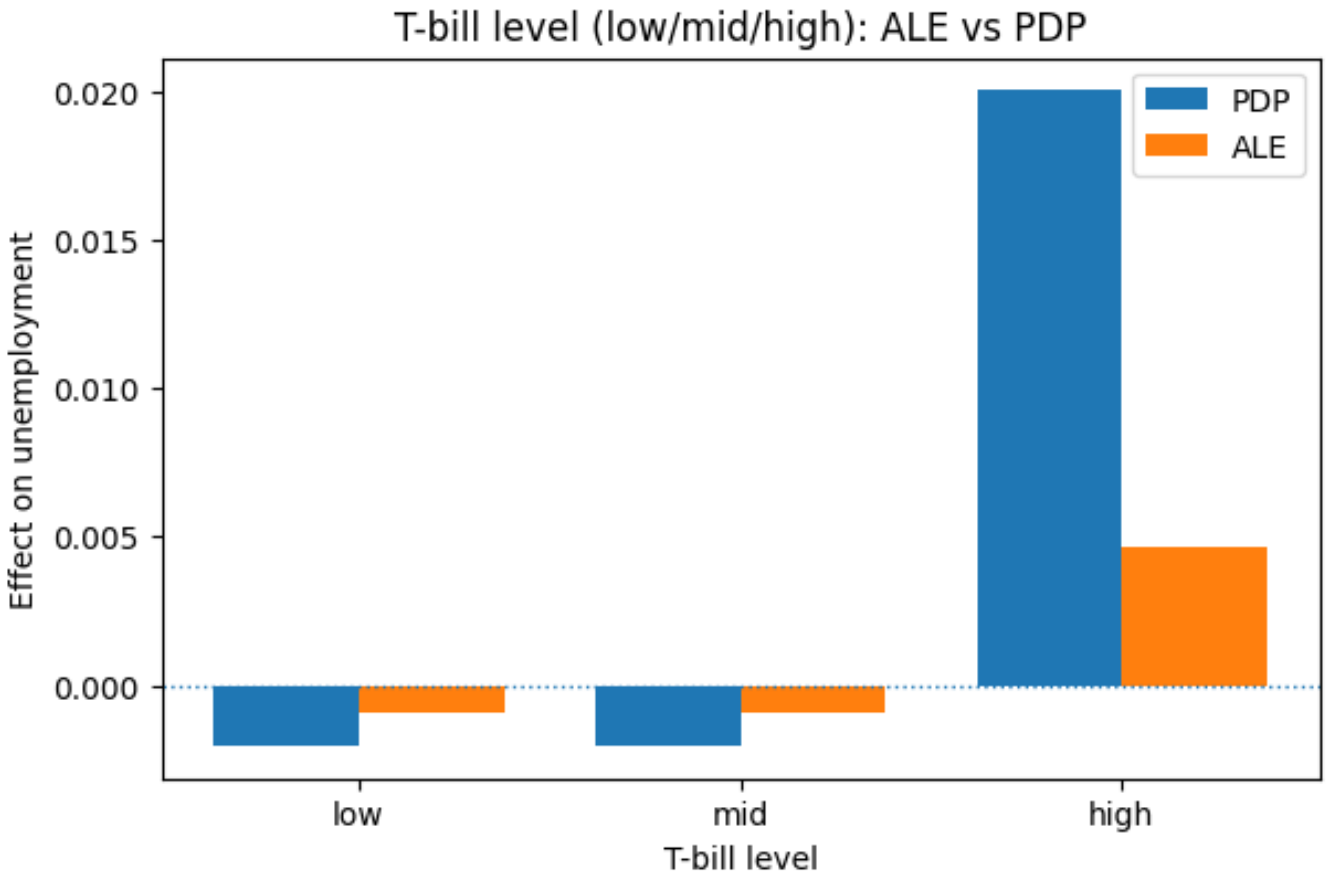
Compare and interpret the two methods: discuss when ALE and PDP yield similar or different patterns, particularly in the presence of feature correlations and heterogeneous effects across the data distribution.

Google Colab: Click here.

Exercise: Python Output



Exercise: Python Output



Global Post hoc Interpretability

Key concepts

1. Global surrogate model
2. Partial Dependence function
3. Partial Dependence Plot (PDP)
4. Assumption of independence
5. Accumulated Local Effects (ALE)

Outline

Local Post hoc Interpretability

In this section, we introduce three local model-agnostic interpretation methods:

1. **Local interpretable model-agnostic explanations (LIME)**
2. **Individual Conditional Expectation (ICE)**
3. **Shapley values**

Individual Conditional Expectation (ICE)

Individual Conditional Expectation

Individual Conditional Expectation (ICE) plots display one curve of partial dependence per instance, showing how the predicted outcome for that instance changes when a specific feature varies.

Definition: Individual Conditional Expectation (ICE)

For each instance $\{x_s^{(i)}, x_c^{(i)}\}$, the **Individual Conditional Expectation (ICE)** associated with the feature X_s is defined as

$$ICE_{s,i}(x_s) = \widehat{f}(x_s, x_c^{(i)}) \quad \forall x_s \in \Omega_s,$$

where x_s denotes the value of the feature of interest and $x_c^{(i)}$ represents the fixed values of all other features for instance i.

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1), 44–65.

Example of Individual Conditional Expectation

Figure 10: Example of ICE for a classification model and a numerical feature

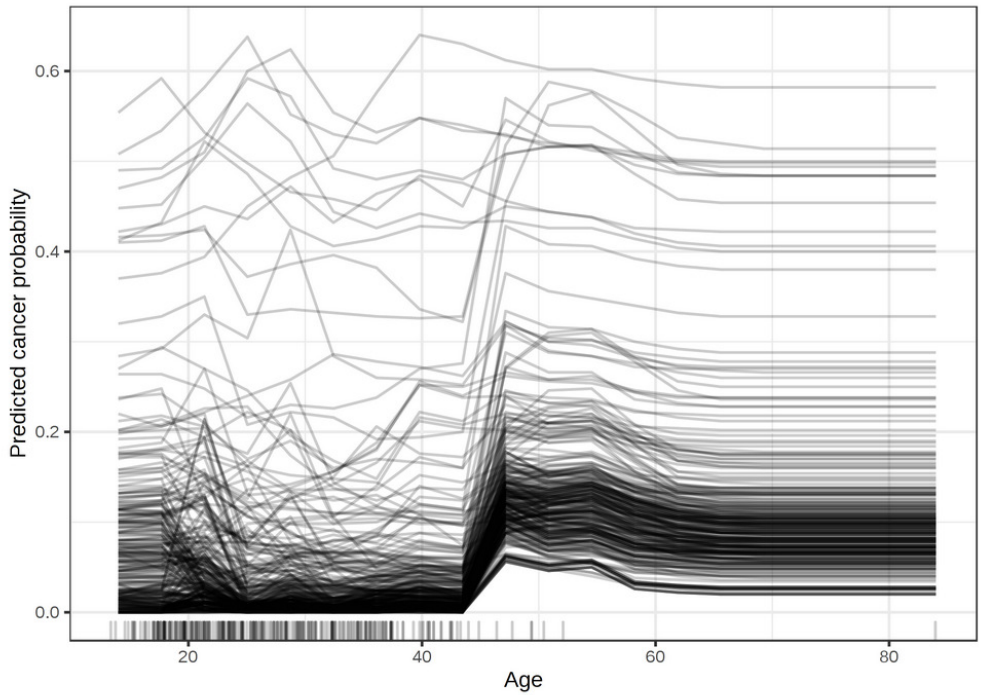


FIGURE 5.6: ICE plot of cervical cancer probability by age. Each line represents one woman. For most women there is an increase in predicted cancer probability with increasing age. For some women with a predicted cancer probability above 0.4, the prediction does not change much at higher age.

Source: Molnar (2025)

Example of Individual Conditional Expectation

Figure 11: Example of ICE for a regression model and 3 numerical features

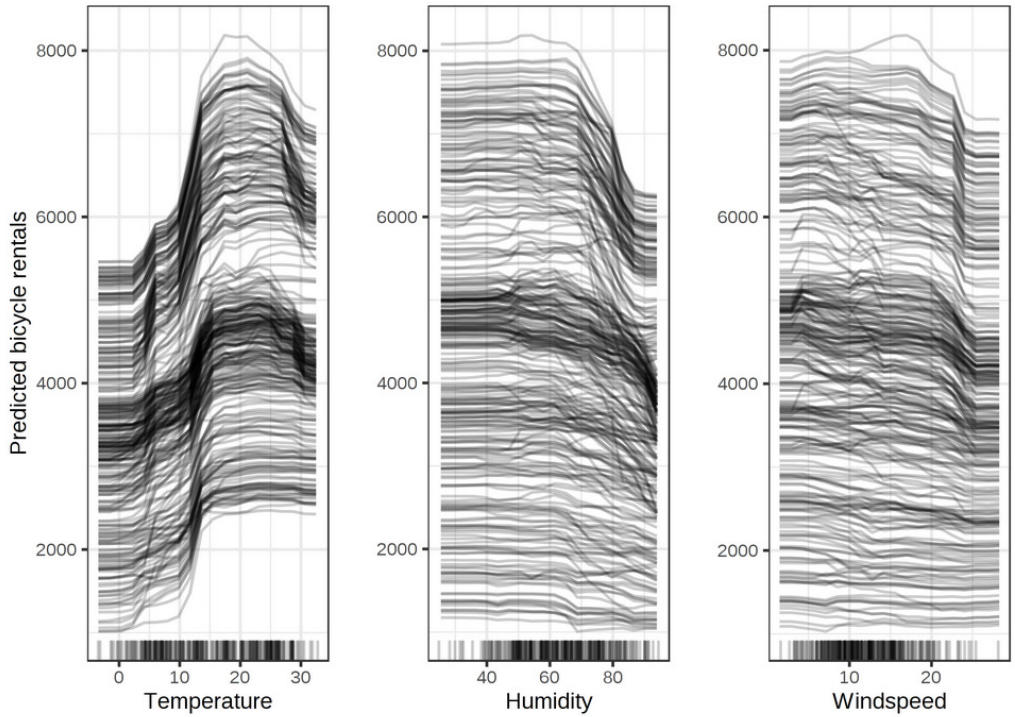


FIGURE 5.7: ICE plots of predicted bicycle rentals by weather conditions. The same effects can be observed as in the partial dependence plots.

Source: Molnar (2025)

Centered ICE: Definition

Centered ICE

- It can be hard to tell whether ICE curves differ across individuals because they start at different levels.
- A simple solution is to center the curves and only display the difference with respect to the reference point.

Definition: Centered ICE

The **centered ICE** is defined as

$$ICE_{s,i}^{cent}(x_s) = \widehat{f}\left(x_s, x_c^{(i)}\right) - \widehat{f}\left(x_a, x_c^{(i)}\right) \quad \forall x_s \in \Omega_s$$

where x_a is the anchor point.

Centered ICE: Definition

Centered ICE

- In general, the anchor point is defined as:

$$x_a = \min_{x_s \in \Omega_s} x_s$$

- Thus, we have:

$$ICE_{s,i}^{cent}(x_s) = \begin{cases} 0 & \text{if } x_s = x_a \\ \widehat{f}\left(x_s, x_c^{(i)}\right) - \widehat{f}\left(x_a, x_c^{(i)}\right) & \text{if } x_s > x_a \end{cases}$$

- All the instances have the same (null) ICE for the value x_a, i.e.

$$ICE_{s,i}^{cent}(x_a) = 0, \quad \forall i = 1, \ldots, n$$

Example of Centered ICE

Figure 12: Example of Centered ICE

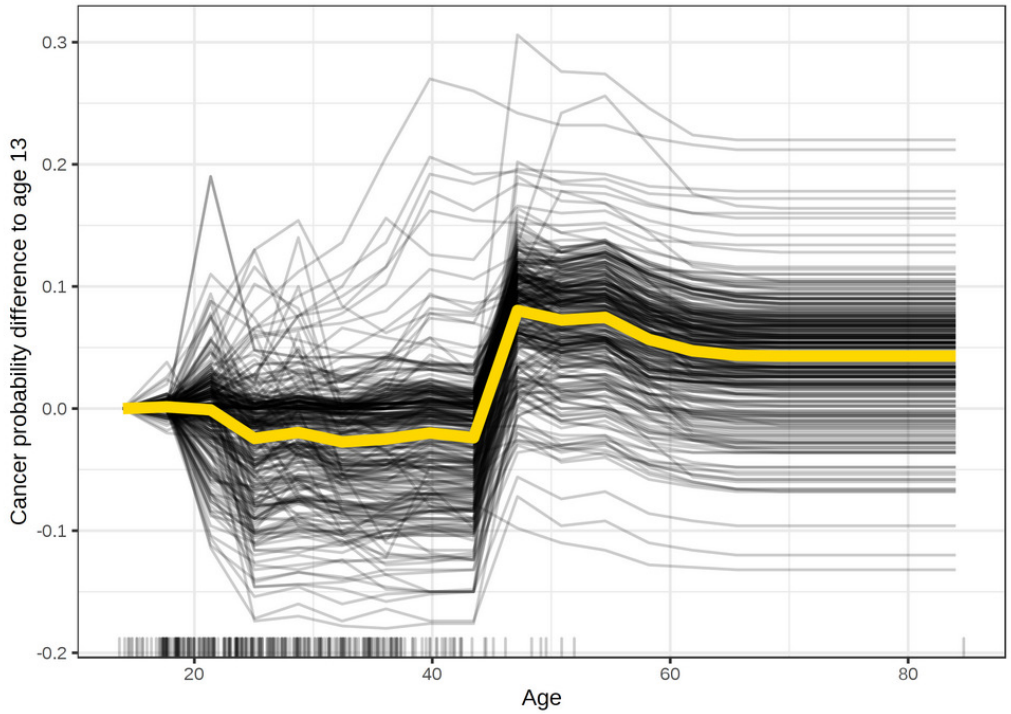


FIGURE 5.8: Centered ICE plot for predicted cancer probability by age. Lines are fixed to 0 at age 14. Compared to age 14, the predictions for most women remain unchanged until the age of 45 where the predicted probability increases.

Source: Molnar (2025)

Example of Centered ICE

Figure 13: Example of centered ICE

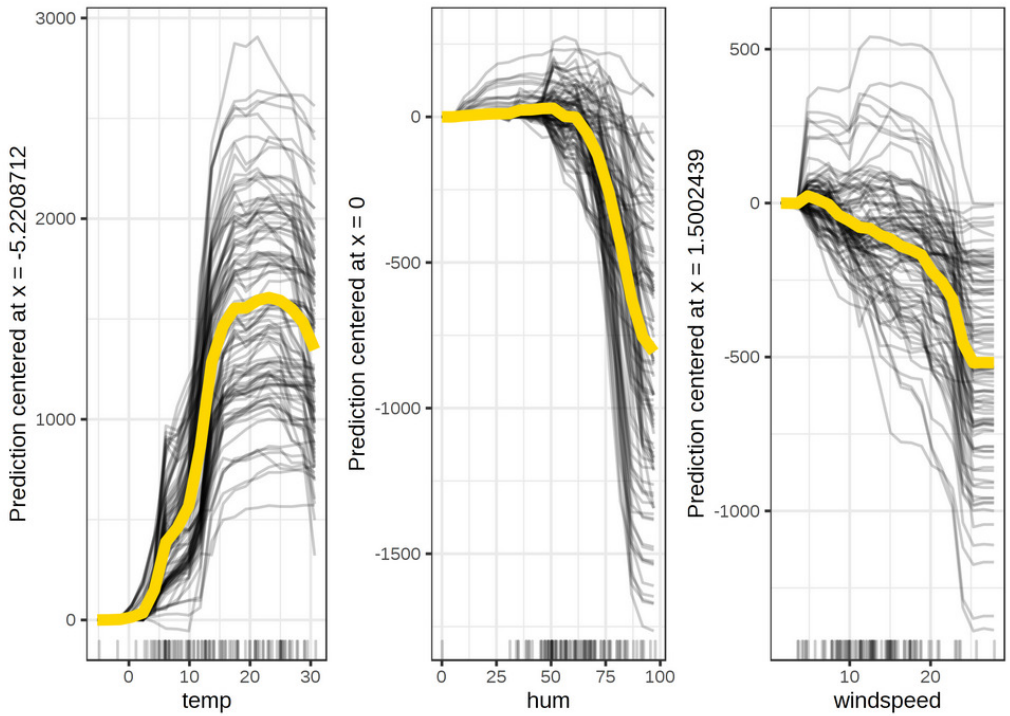


FIGURE 5.9: Centered ICE plots of predicted number of bikes by weather condition. The lines show the difference in prediction compared to the prediction with the respective feature value at its observed minimum.

Source: Molnar (2025)

Advantages and Limits

Advantages

1. Intuitive: ICE curves are even more intuitive to understand than PDP.
2. Heterogeneous effects: unlike PDP, ICE curves can uncover heterogeneous relationships.

Limits

1. Maximum number of features: ICE curves can only display one feature at the time.
2. Assumption of independence: ICE curves suffer from the same problem as PDP: If the feature of interest is correlated with the other features, then some points in the curve might be unlikely data points according to the joint feature distribution.
3. Readibility: If many ICE curves are drawn, the plot can become overcrowded.

Partial Dependence Display in `scikit-learn`

PartialDependenceDisplay

class **sklearn.inspection.PartialDependenceDisplay(*pd_results*, ***, *features*, *feature_names*, *target_idx*, *deciles*, *kind='average'*, *subsample=1000*, *random_state=None*, *is_categorical=None*)**

Partial Dependence Plot (PDP) and Individual Conditional Expectation (ICE).

It is recommended to use `from_estimator` to create a `PartialDependenceDisplay`. All parameters are stored as attributes.

For general information regarding `scikit-learn` visualization tools, see the Visualization Guide. For guidance on interpreting these plots, refer to the Inspection Guide.

For an example on how to use this class, see the following example: Advanced Plotting With Partial Dependence.

kind : *{'average', 'individual', 'both'} or list of such str, default='average'*

Whether to plot the partial dependence averaged across all the samples in the dataset or one line per sample or both.

- `kind='average'` results in the traditional PD plot;
- `kind='individual'` results in the ICE plot;
- `kind='both'` results in plotting both the ICE and PD on the same plot.

A list of such strings can be provided to specify `kind` on a per-plot basis. The length of the list should be the same as the number of interaction requested in `features`.

Source: scikit-learn documentation

ICE with Partial Dependence Display

Example: ICE with Partial Dependence Display

```
model = GradientBoostingRegressor(random_state=42)
model.fit(X_data, y_data)

plt.figure()
PartialDependenceDisplay.from_estimator(
    model,
    X_data,
    features=["Variable Name"],
    kind="individual",     # <- this makes it an ICE plot
    grid_resolution=50,
    centered=True,  # Centered ICE or uncentered ICE
    subsample=100,  # optional: speed up by sampling ICE curves
)
plt.show()
```

Google Colab: Click here.

Exercise: Individual Conditional Expectation

Exercise: ICE for an Unemployment Model

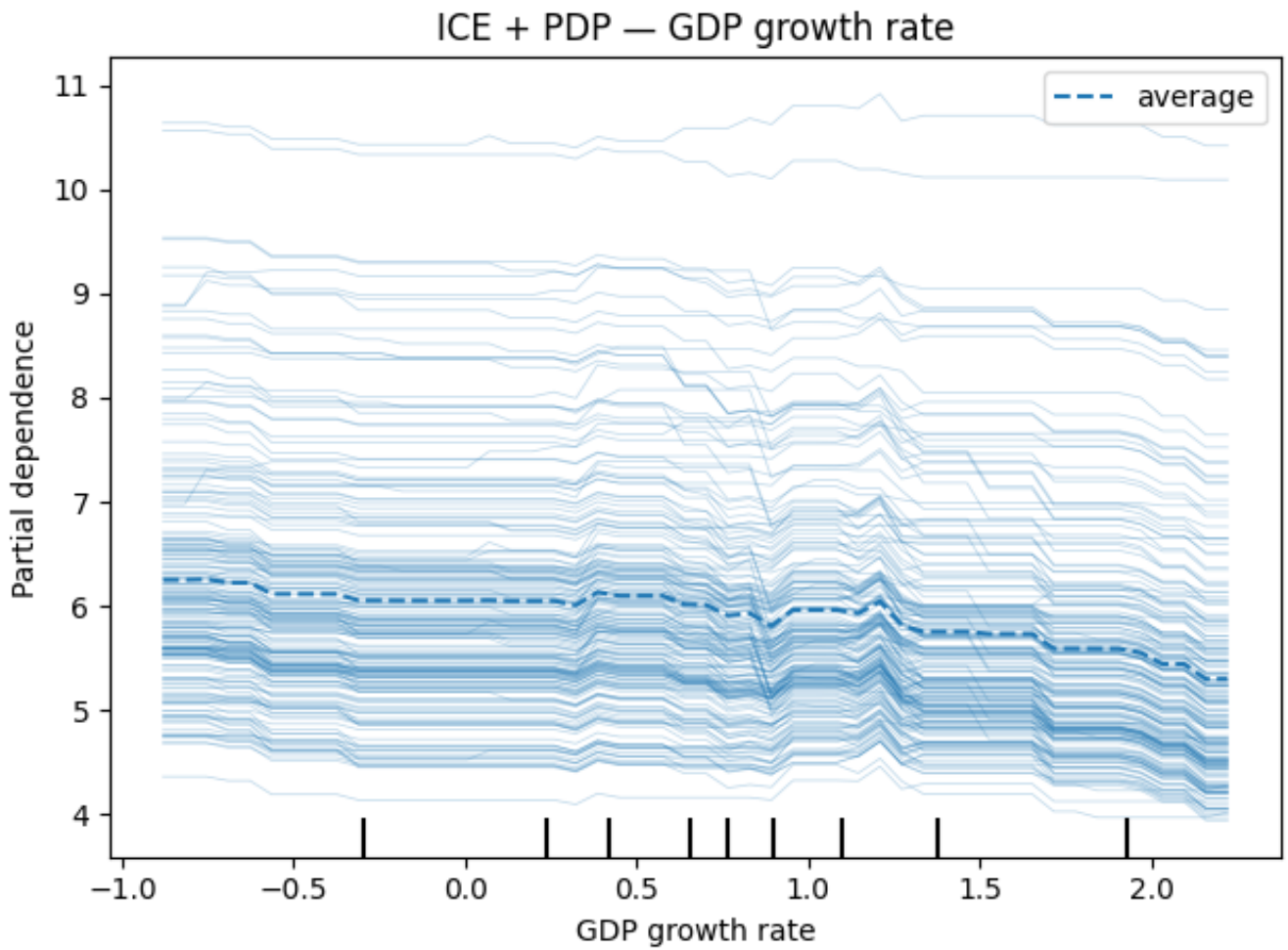
Consider quarterly U.S. macroeconomic data (1959Q1–2009Q3) from the `statsmodels` *macrodata* dataset. We aim to predict the **unemployment rate** using:

- GDP growth rate,
- Consumption growth rate,
- Investment growth rate,
- Inflation rate,
- 3-month T-bill rate, discretized into three categories (*low/mid/high*).

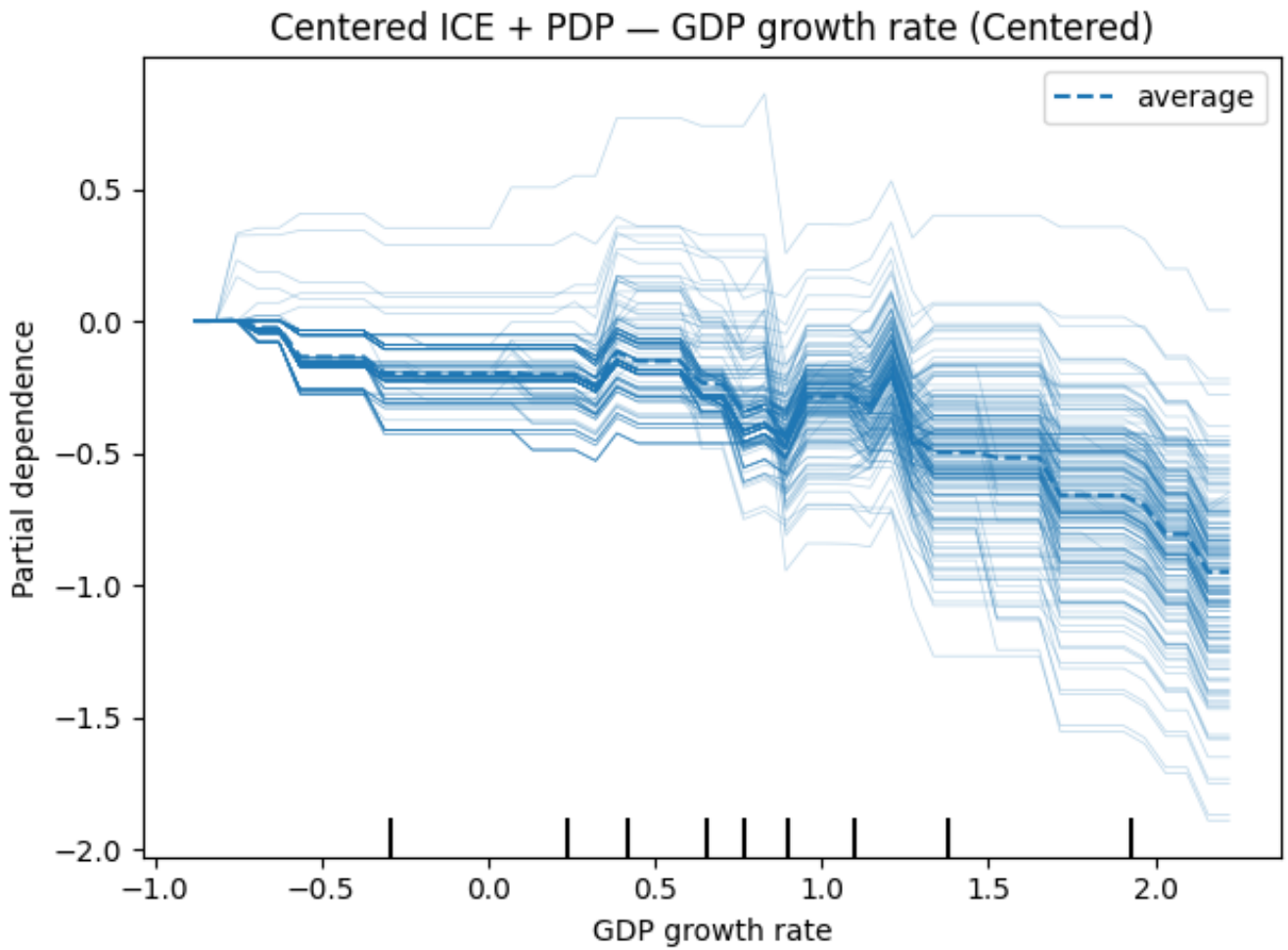
A Gradient Boosting Regressor is trained on the feature set. The objective is to analyze the influence of the **GDP growth rate** and the **T-bill rate** using **Individual Conditional Expectation (ICE)** curves. Produce ICE visualizations for each of the two variables and discuss what they reveal about heterogeneity in the model's local responses.

Google Colab: Click here.

Exercise: Python Output



Exercise: Python Output



Local interpretable model-agnostic explanations (LIME)

LIME: Definition

Definition: Local Interpretable Model-agnostic Explanations

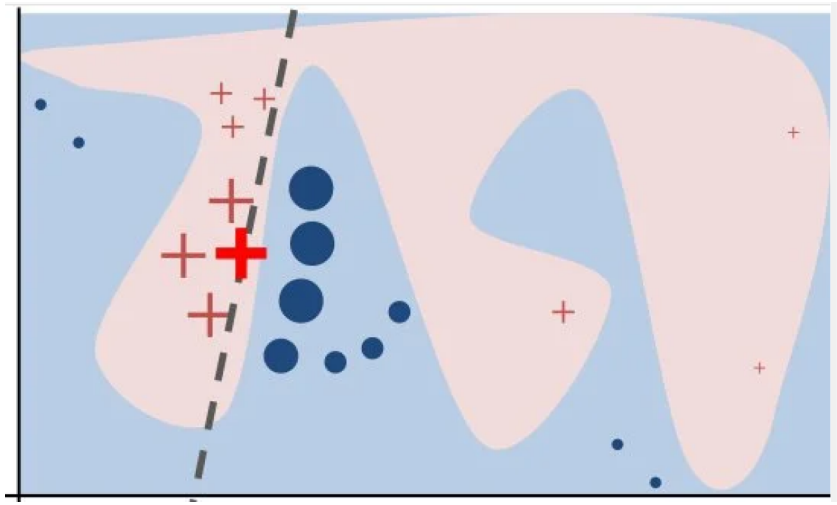
Local Interpretable Model-agnostic Explanations (LIME) is a technique that explains the prediction of *any* complex model by fitting a simple, interpretable model in the neighborhood of the instance being predicted.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Intuition of LIME

Intuition

- Complex black-box model with decision function *f* (blue/pink).
- Target instance to explain (solid red cross).
- LIME generates perturbed samples around the instance (dots and crosses).
- Predictions from *f* are obtained and weighted by proximity to the target instance (size of points).
- A simple interpretable model is fitted locally (dashed line).



Source: c3.ai

Interpretable Data Representations

Interpretable Data Representation

Raw features are not always directly meaningful to humans. An interpretable representation transforms raw features into a form that is easier for humans to understand while still being useful for explanations.

Example of Interpretable Data Representation

In images, instead of individual pixels, interpretable representations are often based on superpixels, i.e., contiguous regions of similar pixels.

Interpretable Data Representations

Interpretable Data Representations

- In image classification, pixel values can be grouped into **superpixels**, i.e., contiguous regions of similar pixels represented by binary indicators (present/absent).
- In text classification, instead of relying on complex embeddings, an interpretable representation can be the presence or absence of specific words.
- In tabular data, continuous variables can be discretized into categories (e.g., age groups), and categorical variables can be combined into interpretable groups.

LIME: Notations and Setup

Interpretable Data Representations and Models

- Let $x \in \mathbb{R}^d$ denote the original representation (features) of the instance being explained.
- Let $x' \in \{0,1\}^{d'}$ denote the interpretable representation, typically a binary vector.
- The black-box model is $f : \mathbb{R}^d \to \mathbb{R}$, where in classification $f(x)$ is the predicted probability of belonging to a class.
- An explanation is given by a surrogate model $g : \{0,1\}^{d'} \to \mathbb{R}$, with $g \in G$, where G is a family of interpretable models (e.g., linear models, shallow decision trees).

LIME: Fidelity–Interpretability Trade-off

Definition: LIME explanation

The explanation produced by **LIME** solves:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g).$$

where

- $\Omega(g)$ measures the **complexity** of the explanation $g \in G$ (e.g., depth of a tree, number of nonzero coefficients in a linear model).
- $\pi_x(z)$ defines a **locality** around x via a proximity measure between z and x (e.g., exponential kernel).
- $\mathcal{L}(f, g, \pi_x)$ measures the **unfaithfulness** of g in approximating f in the neighborhood of x.

Remark: Different choices of G, $\mathcal{L}$, and Ω yield different forms of explanations.

LIME: Sampling for Local Exploration

Model-agnostic local approximation

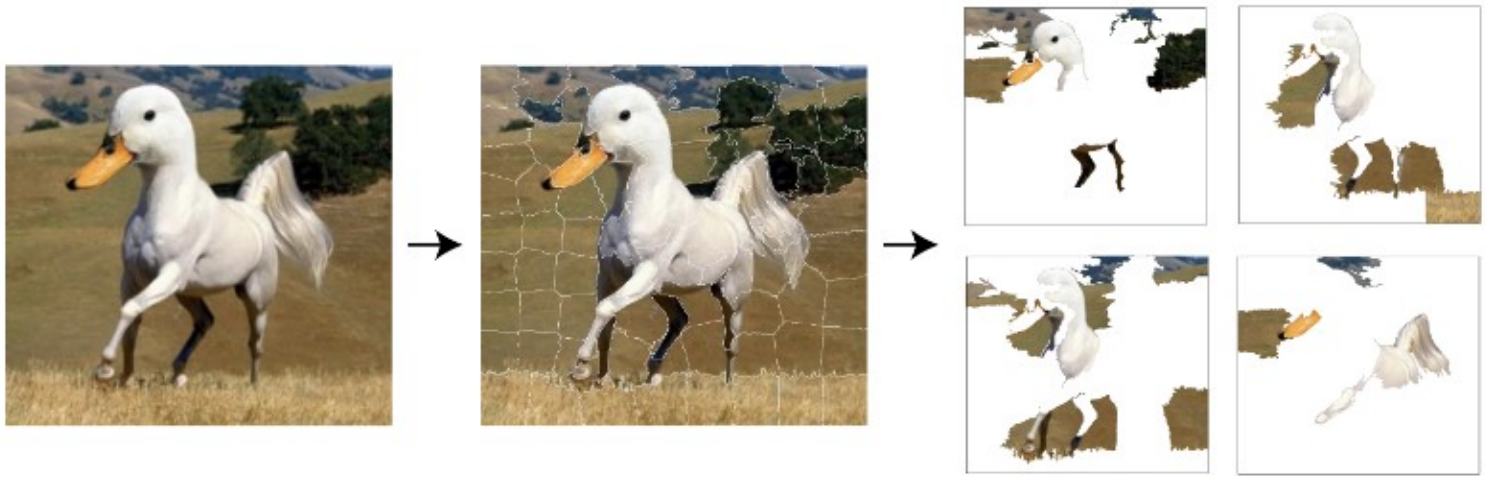
- The goal is to minimize $\mathcal{L}(f, g, \pi_x)$ without making assumptions about f (model-agnostic).
- To approximate the local behavior of f, we generate perturbed samples around the instance x', and weight them according to their proximity π_x.

Procedure: Sampling in LIME

To approximate the local behavior of the black-box model, LIME generates perturbed samples around the instance and uses them to train a simple surrogate model.

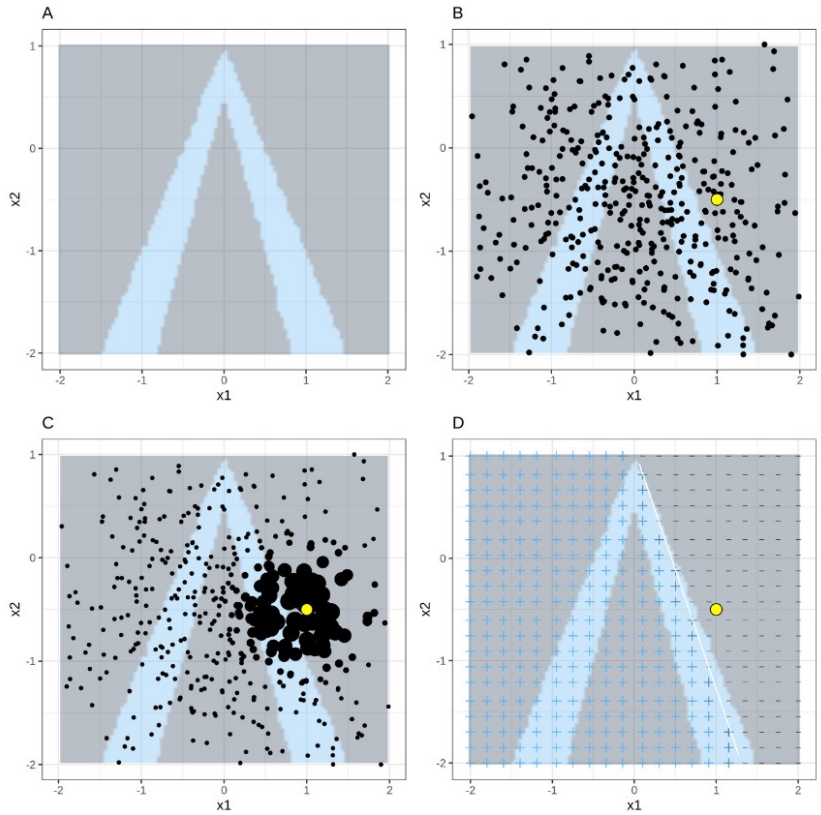
- Create perturbed interpretable instances $z' \in \{0,1\}^{d'}$ around x' (e.g., by randomly switching features on or off).
- Map each z' back to the original feature space, $z \in \mathbb{R}^d$, and compute $f(z)$.
- Build the dataset $\mathcal{Z} = \{(z', f(z))\}$ of perturbed samples and their black-box predictions.
- Fit the interpretable surrogate model g on $\mathcal{Z}$ to obtain the explanation $\xi(x)$.

LIME: Sampling for Local Exploration



Note: This image illustrates how LIME perturbs an instance: the original image is first segmented into superpixels, and then different subsets of superpixels are turned on or off to create new samples that are used to approximate the model's local behavior.

LIME: Sampling for Local Exploration



Note: This figure illustrates how LIME builds a local explanation around a given instance (yellow point). The black-box model's decision boundary is shown in light blue, and perturbed samples are generated (black dots) and weighted according to their proximity to the instance (larger dots indicate higher weights). Using these weighted samples, LIME fits a simple surrogate model (Panel D) that approximates the complex decision function locally.

LIME: Sparse Linear Explanations

Sparse Linear Explanations

The general principle of LIME is to approximate the complex model f locally with a simple and interpretable surrogate, here a sparse linear model.

- Restrict G to the class of linear models:

$$g(z') = w_g \cdot z' = \sum_{j=1}^{d'} w_{g,j} z'_j$$

- Define the locally weighted square loss:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

where

$$\pi_x(z) = \exp(-D(x, z)/\sigma^2)$$

is an exponential kernel based on a distance function D.

- Impose sparsity by limiting the number of nonzero coefficients:

$$\Omega(g) = \begin{cases} \infty & \text{if } ||w_g||_0 > K, \\ 0 & \text{otherwise,} \end{cases}$$

with

$$||w_g||_0 = \lim_{p \to 0} \sum_j |w_{g,j}|^p.$$

LIME: Sparse Linear Explanations

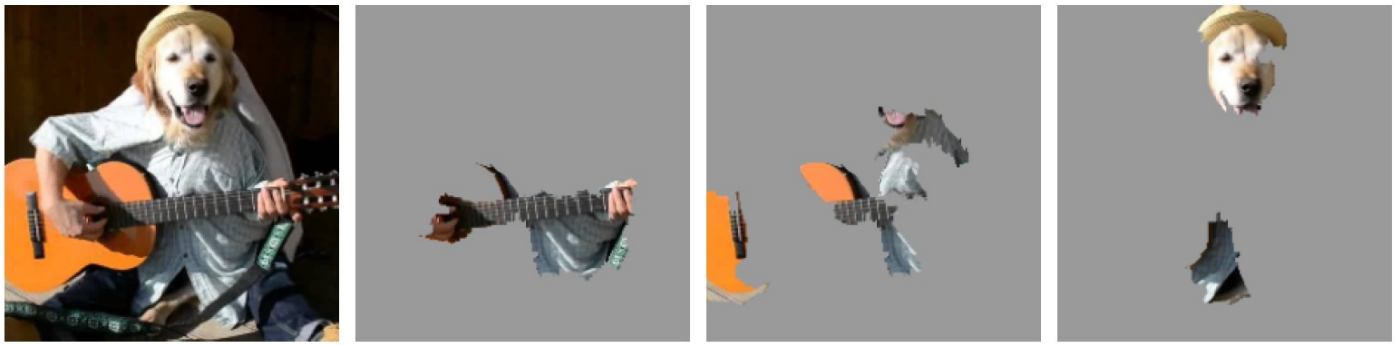
Interpretability in Practice

The explanation is interpretable because it relies on human-understandable representations and restricts their number to K.

- In text classification, the interpretable representation is a bag of words, and K limits the number of words in the explanation.
- In image classification, the interpretable representation consists of superpixels, and K limits the number of superpixels shown.
- In tabular data, continuous variables can be discretized and categorical variables grouped, with K limiting the number of categories used.

Sparse Linear Explanations using LIME

Figure 14: Example of Deep networks for images



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Ribeiro, M.T., and al. (2016), "Why Should I Trust You?" Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Sparse Linear Explanations using LIME

Figure 15: Example of error explanation

(a) Husky classified as wolf (b) Explanation

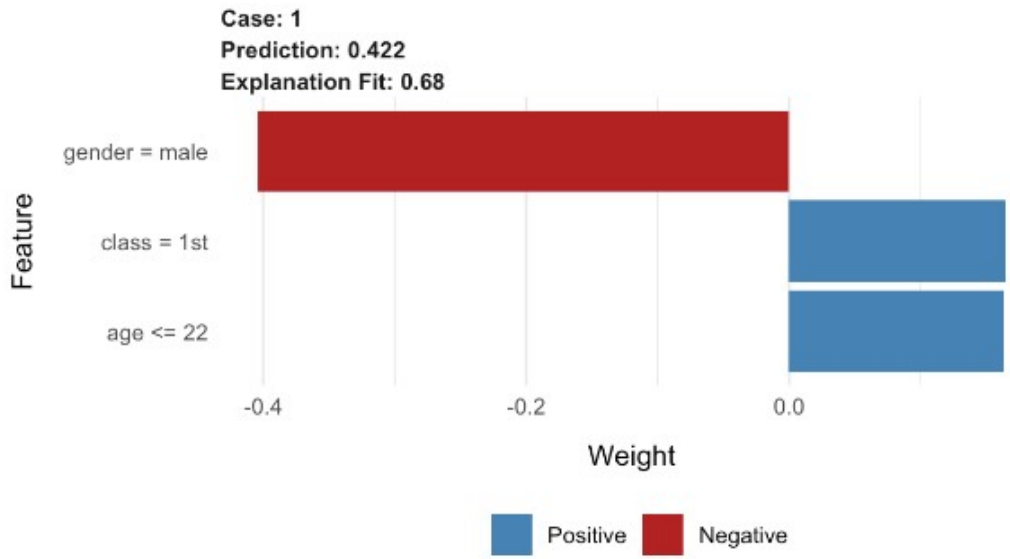
Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

Ribeiro, M.T., and al. (2016), "Why Should I Trust You?" Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Sparse Linear Explanations using LIME

Figure 16: LIME for tabular data

$$\hat{p}_{lime} = 0.55411 - 0.40381 \cdot 1_{male} + 0.16366 \cdot 1_{age<=22} + 0.16452 \cdot 1_{class=1st} = 0.47848,$$



Note: Titanic data, probability of survival of an 8-year-old passenger that travelled in the first class.

Advantages and Limits

Advantages

1. Human-friendly explanation: the resulting explanations are short (= selective) and possibly contrastive.
2. Features type: works for tabular data, text and images.
3. Fidelity measure: good idea of how reliable the interpretable model is in explaining the black box predictions in the neighborhood of the data instance of interest.

Limits

1. Neighborhood definition: sensitive results to the kernel values.
2. Sampling: creation of unrealistic instances.
3. Complexity definition: it has to be defined in advance and the choice of K is let to the user.

LIME Libraries in Python

LIME is not available in `scikit-learn`;

It is provided by a separate package on PyPI called `lime`.

- For tabular data, the relevant module is `lime.lime_tabular`.
- For textual data, use `lime.lime_text`,
- For images, use `lime.lime_image`.

LIME Implementation

Docs » Local Interpretable Model-Agnostic Explanations (lime) Edit on GitHub

Local Interpretable Model-Agnostic Explanations (lime)

In this page, you can find the Python API reference for the lime package (local interpretable model-agnostic explanations). For tutorials and more information, visit the github page.

- lime package
  - Subpackages
  - Submodules
  - lime.discretize module
  - lime.exceptions module
  - lime.explanation module
  - lime.lime_base module
  - lime.lime_image module
  - lime.lime_tabular module
  - lime.lime_text module
  - lime.submodular_pick module
  - Module contents

Source: Lime Github

Example of LIME Implementation

Example: LIME Implementation

```
import lime.lime_tabular

model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

# Create the LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,    # Training data
    feature_names=X.columns,         # Feature names
    mode='classification'            # Classif.  or regression
)

explanation = explainer.explain_instance(
    data_row=instance,     # Instance to explain
    predict_fn=model.predict_proba,  # Prediction function
    num_features=6,        # Number of features to display
    num_samples=1000       # Number of samples for LIME
)
```

Google Colab: Click here.

Exercise: LIME for Credit Scoring Model

Exercise: LIME Explanation for Credit Scoring Model

The goal is to build a **credit scoring model** with a Random Forest and to generate **local explanations** using LIME.

- The dataset contains borrower and loan characteristics in `Scoring_Database.xlsx`. The target variable is `Default` (1 if the borrower defaults, 0 otherwise).
- Split the data into training and test sets (e.g., 70%/30%) with stratification on the target.
- Train a Random Forest Classifier on the training set.
- Evaluate on the test set: report accuracy, ROC–AUC, the confusion matrix, etc.
- Apply **LIME** to explain two test instances: (i) the observation with the *highest* predicted default probability; (ii) a *random* test observation (distinct from the first).
- For each instance, provide a brief description of key raw feature values and interpret the main LIME contributions toward the default class.

Google Colab: Click here.

Exercise: Credit Scoring Dataset

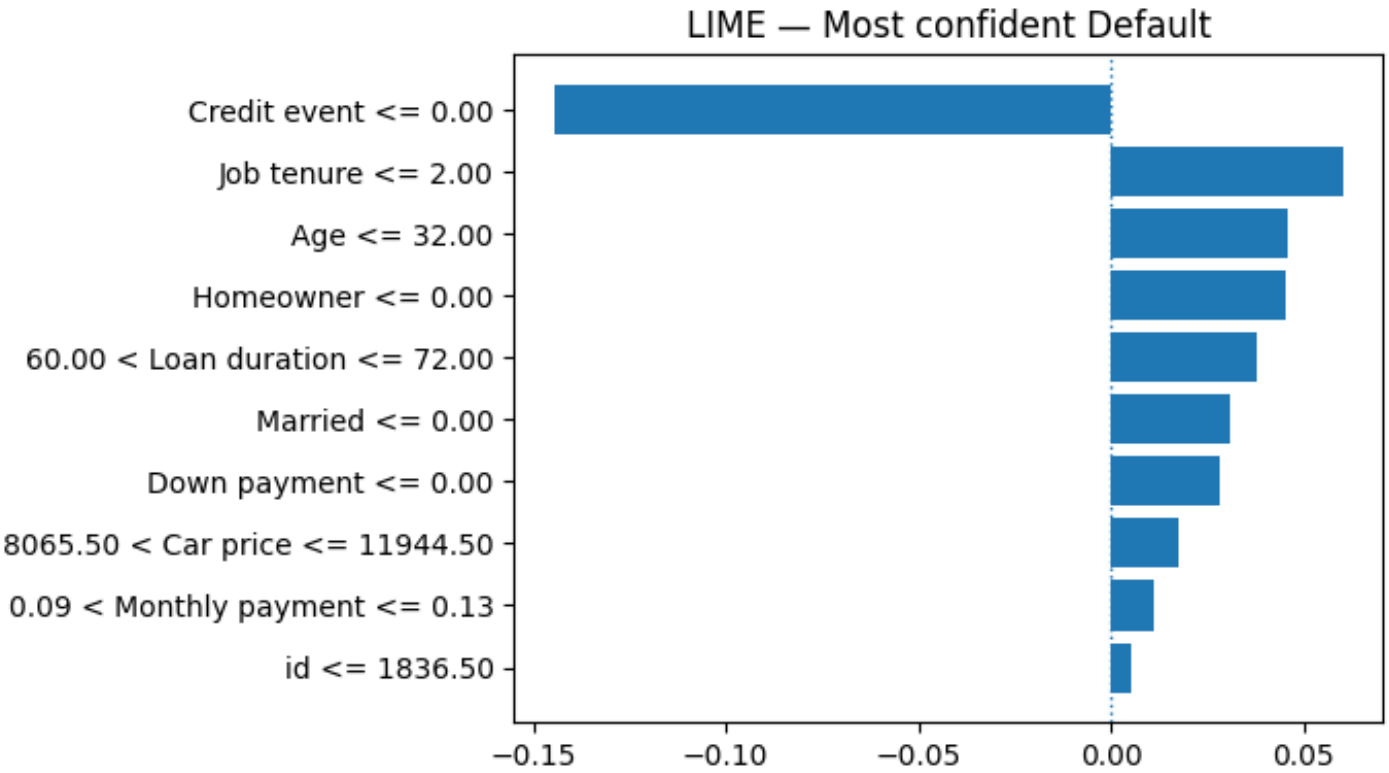
Variable	**Role**	**Type**	**Unit**	**Values / Categories**
Default indicator	Target	Binary		1 = Default within 12 months 0 = No default within 12 months
Tenure in current job	Feature	Continuous	Years	
Borrower age	Feature	Continuous	Years	
Car purchase price	Feature	Continuous	Euros	
Loan amount	Feature	Continuous	Euros	
Monthly payment as a share of monthly income	Feature	Continuous	%	
Down payment > 50% of vehicle value	Feature	Binary		1 = Yes 0 = No
Expected loan duration	Feature	Continuous	Months	
Credit event in last 6 months	Feature	Binary		1 = Yes 0 = No
Marital status (married or other)	Feature	Binary		1 = Married 0 = Other
Homeownership status	Feature	Binary		1 = Homeowner 0 = Other

Exercise: Python Output

```
=== Most confident Default (test idx=1254) ===
True label: 0 | Predicted: 1 | P(Default)= 0.918
Key raw feature values:
  - Age: 20.0
  - id: 1255.0
  - Job tenure: 0.0
  - Monthly payment: 0.1282758620689655
  - Loan duration: 72.0
  - Married: 0.0
  - Funding amount: 12416.0
  - Car price: 11000.0
```

```
LIME top contributions (towards 'Default'):
                          feature    weight
           Credit event <= 0.00 -0.144729
             Job tenure <= 2.00  0.060124
                   Age <= 32.00  0.045649
              Homeowner <= 0.00  0.045476
 60.00 < Loan duration <= 72.00  0.038032
                Married <= 0.00  0.031008
           Down payment <= 0.00  0.028201
8065.50 < Car price <= 11944.50  0.017193
 0.09 < Monthly payment <= 0.13  0.011202
                  id <= 1836.50  0.005181
```

Exercise: Python Output

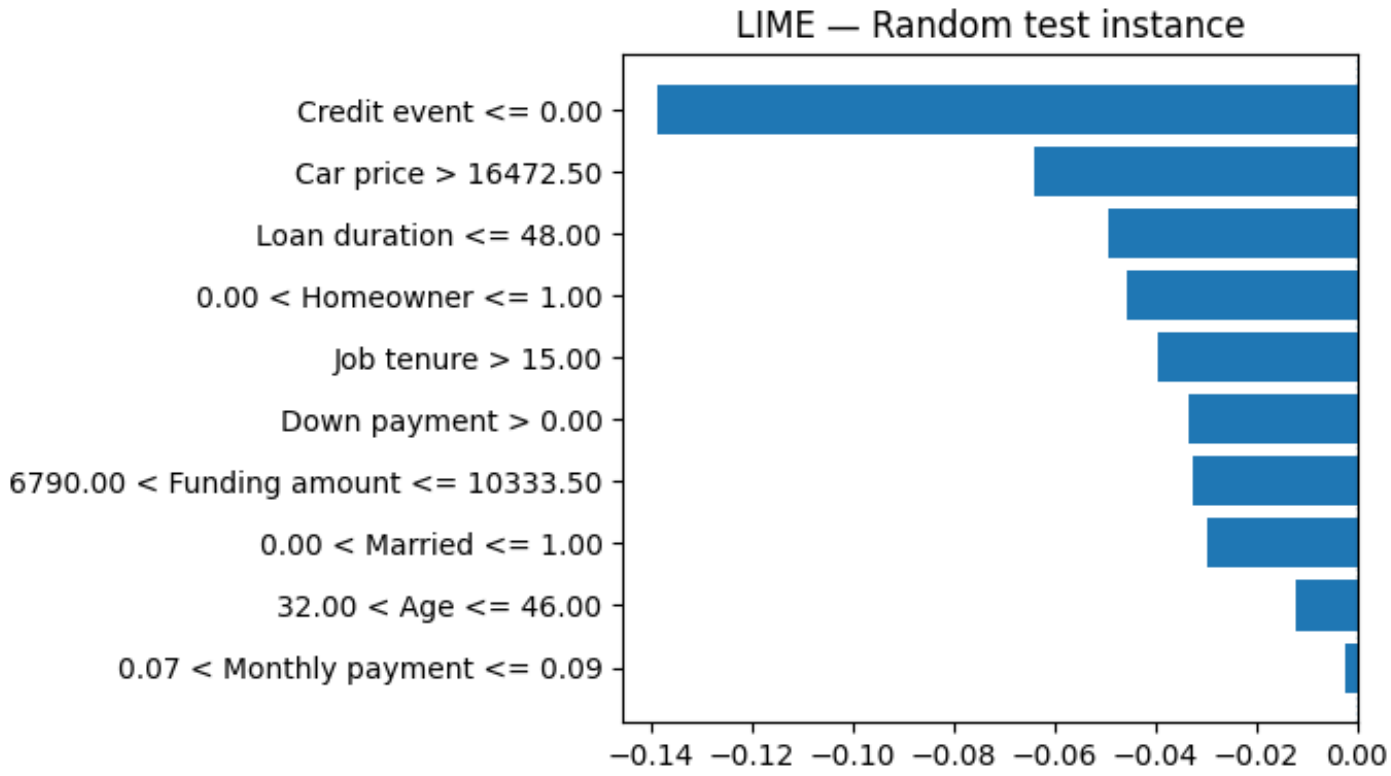


Exercise: Python Output

```
=== Random test instance (test idx=5735) ===
True label: 0 | Predicted: 0 | P(Default)= 0.014
Key raw feature values:
  - Job tenure: 18.0
  - Car price: 21000.0
  - id: 5736.0
  - Loan duration: 48.0
  - Down payment: 1.0
  - Homeowner: 1.0
  - Age: 37.0
  - Funding amount: 7549.0
```

```
LIME top contributions (towards 'Default'):
                              feature    weight
                Credit event <= 0.00 -0.138580
                Car price > 16472.50 -0.064008
              Loan duration <= 48.00 -0.049280
            0.00 < Homeowner <= 1.00 -0.045537
                  Job tenure > 15.00 -0.039533
                 Down payment > 0.00 -0.033171
6790.00 < Funding amount <= 10333.50 -0.032579
              0.00 < Married <= 1.00 -0.029461
                32.00 < Age <= 46.00 -0.012253
     0.07 < Monthly payment <= 0.09 -0.002433
```

Exercise: Python Output



Shapley Values

Shapley Values: Definition

Definition: Shapley Values

Shapley values provide a principled way to fairly allocate the total payoff (or contribution) of a coalition among individual players, based on their marginal contributions across all possible coalitions.

- Originates from cooperative game theory. Each player's value reflects their average contribution when joining all possible subsets of players.
- Introduced by Lloyd Shapley in 1953, later awarded the Nobel Prize in Economics in 2012.
- In machine learning, Shapley values are used for post hoc interpretability: they quantify how much each feature contributes to a model's prediction, ensuring fairness and consistency in feature attribution.

Shapley, L. S. (1953). *A value for n-person games*. Contributions to the Theory of Games, 2(28), 307–317.

Intuition of Shapley Values

Intuition of Shapley Values

Three friends, Pierre, Eve, and Aminata, go out for dinner. They order and share fries, wine, and pies. Since they did not eat equal portions, it is not clear how much each should pay.

The question is: how can we fairly split the the total bill of 73?

We have the following information about the possible coalitions:

- If Pierre eats alone, the bill is 25.
- If Eve eats alone, the bill is 16.
- If Aminata eats alone, the bill is 19.
- If Pierre and Eve eat together, the bill is 50.
- If Pierre and Aminata eat together, the bill is 56.
- If Eve and Aminata eat together, the bill is 42.
- If all three eat together, the bill is 73.

Intuition of Shapley Values

How Shapley Values Work

1. Consider all possible orders in which the three friends could join the table.
2. For each order, compute the marginal contribution of the newly arriving guest to the bill.
3. The Shapley value for each person is the average of their marginal contributions across all possible orders.

Example: One Ordering

Consider the sequence (Pierre, Eve, Aminata). First, Pierre pays 25. Then, when Eve joins, the bill rises to 50, so her marginal contribution is 25. Finally, when Aminata joins, the bill rises to 73, so her marginal contribution is 23.

Intuition of Shapley Values

Intuition of Shapley Values

We repeat the same exercise for each possible order of arrival of the three friends and obtain the corresponding marginal contributions:

- (Pierre, Eve, Aminata) $\rightarrow$ (25, 25, 23)
- (Eve, Pierre, Aminata) $\rightarrow$ (34, 16, 23)
- (Eve, Aminata, Pierre) $\rightarrow$ (31, 16, 26)
- (Aminata, Pierre, Eve) $\rightarrow$ (37, 17, 19)
- (Aminata, Eve, Pierre) $\rightarrow$ (31, 23, 19)
- (Pierre, Aminata, Eve) $\rightarrow$ (25, 17, 31)

Intuition of Shapley Values

Intuition of Shapley Values

The Shapley value for each friend is simply the average of their marginal contributions across all possible orders:

- Pierre: $(25 + 34 + 31 + 37 + 31 + 25)/6 = 30.5$
- Eve: $(25 + 16 + 16 + 17 + 23 + 17)/6 = 19$
- Aminata: $(23 + 23 + 26 + 19 + 19 + 31)/6 = 23.5$
- Total: $30.5 + 19 + 23.5 = 73$ (the full bill).

This final allocation provides a fair split of the total cost according to each person's average contribution.

Shapley Values for ML Interpretability

Shapley Values for ML Interpretability

- The game is the prediction task, considered for a single instance.
- The players are the features that "cooperate" to generate the prediction, just as the friends in the dinner example contributed to the bill.
- The objective is to decompose the prediction into feature contributions that explain the difference between the actual prediction and the average prediction of the model.

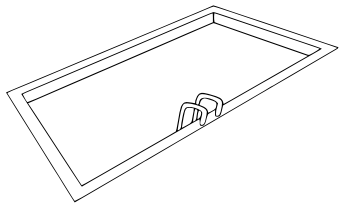
Example of Shapley Values for ML Interpretability

Example: Shapley Values

Consider an ML model used to predict house prices. For a particular house with a garage, a private pool, and an area of 50 square yards, the predicted price is €510,000, while the average prediction across all houses is €500,000.

Question: How much does each feature (garage, pool, area) contribute to the €10,000 difference between this house's prediction and the average prediction?

has_garage



has_pool

50 Yards

area

Remarks on Shapley Values

Remark 1: Shapley values represent the individual contribution of each feature to the prediction, relative to the average prediction of the model. They show how much each feature pushes the prediction upward or downward.

Remark 2: The efficiency property: the sum of all Shapley values for a given instance is exactly equal to the difference between the individual prediction $\hat{y}_i$ and the average model prediction $\mathbb{E}[\hat{y}]$.

Remark 3: Shapley values guarantee a fair allocation of contributions among features, respecting symmetry (identical features get equal credit) and nullity (irrelevant features have zero contribution).

Shapley Values for a Classification Model

Figure 17: Example of Shapley values for a classification model

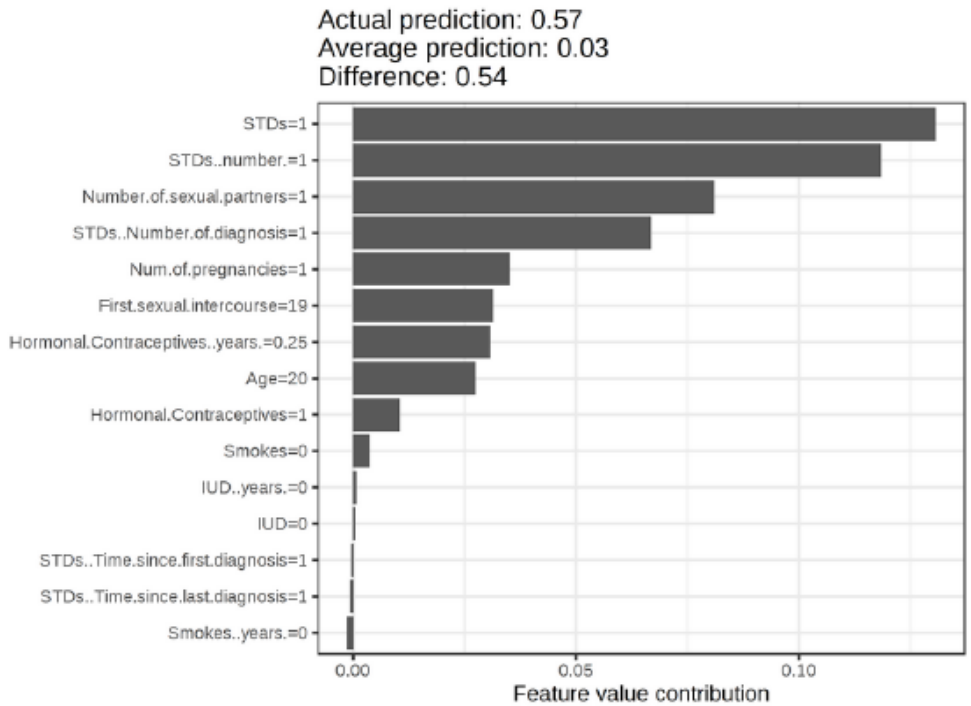


FIGURE 5.46: Shapley values for a woman in the cervical cancer dataset. With a prediction of 0.57, this woman's cancer probability is 0.54 above the average prediction of 0.03. The number of diagnosed STDs increased the probability the most. The sum of contributions yields the difference between actual and average prediction (0.54).

Shapley Values for a Regression Model

Figure 18: Example of Shapley values for a regression model

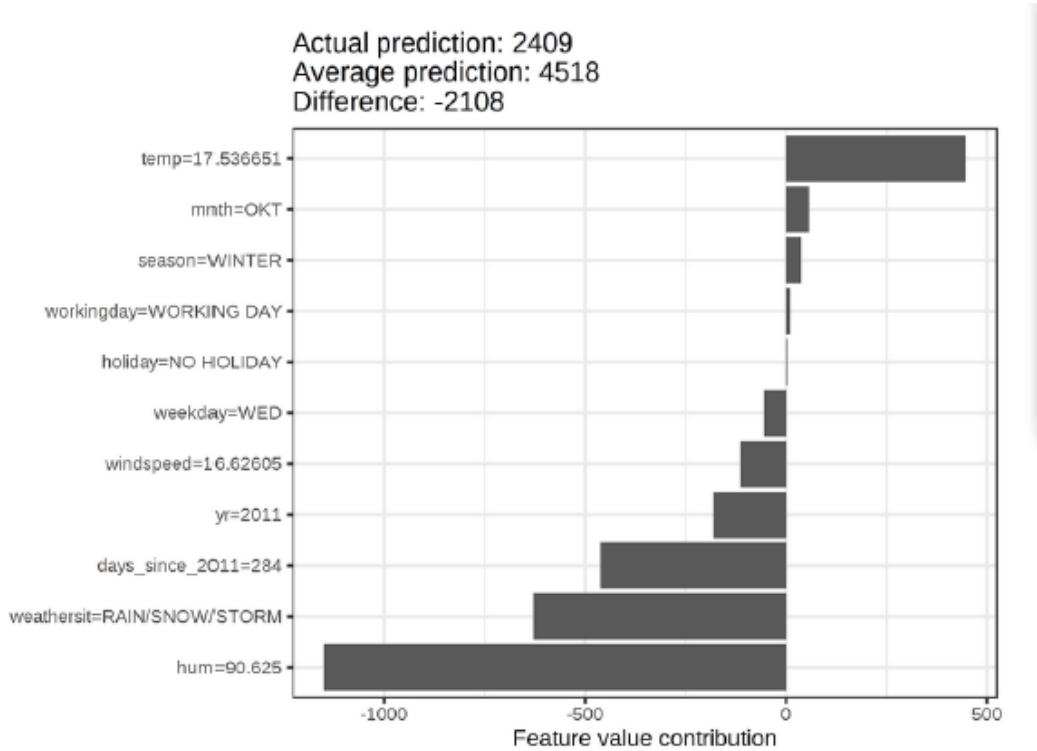


FIGURE 5.47: Shapley values for day 285. With a predicted 2409 rental bikes, this day is -2108 below the average prediction of 4518. The weather situation and humidity had the largest negative contributions. The temperature on this day had a positive contribution. The sum of Shapley values yields the difference of actual and average prediction (-2108).

Notations and Framework

Notations

- S is a subset of the features used in the model.
- x is the vector of feature values of the instance to be explained.
- p the number of features.
- We are interested in quantifying how each feature contributes to the prediction for a given instance.

Shapley Values: Definition

Definition: Shapley Value

The **Shapley value** of feature x_j is the weighted average of its **marginal contributions** across all possible subsets $S \subseteq \{x_1, \ldots, x_p\} \setminus \{x_j\}$:

$$\phi_j(\widehat{f}) = \sum_{S \subseteq \{x_1, \ldots, x_p\} \setminus \{x_j\}} \frac{|S|!(p - |S| - 1)!}{p!} \left(\widehat{f}(S \cup \{j\}) - \widehat{f}(S) \right).$$

Remarks on the Shapley Definition

Remarks

- $\widehat{f}(S)$ denotes the model prediction using only the features in subset S.
- The weight $\frac{|S|!(p-|S|-1)!}{p!}$ ensures that all possible orderings of features are treated equally (symmetry).
- Subsets can have sizes $|S| = 1, \ldots, p-1$.
- If we have $p-1$ features, $C_{|S|}^{p-1}$ denotes the number of groups of size $|S|$ we can form with them.

$$p \times C_{|S|}^{p-1} = p \times \begin{pmatrix} p-1 \\ |S| \end{pmatrix} = \frac{p!}{|S|!\,(p-1-|S|)!}$$

Example: Shapley Values for a Logit Model

Example: Shapley Values for a Logit Model

Consider a logit model with three features $X = (X_1, X_2, X_3)$:

$$\widehat{f}(X) = \Pr(Y = 1|X) = \Lambda(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3)\,,$$

where $\Lambda(z) = 1/(1 + \exp(-z))$ is the logistic cdf.

To compute the Shapley value $\phi_1(\widehat{f})$ associated with feature X_1, we evaluate its marginal contribution over all possible subsets $S \subseteq \{X_2, X_3\}$, with weights

$$\omega_S = \frac{|S|!(3 - |S| - 1)!}{3!}.$$

S	$S \cup \{X_1\}$	ω_S	Marginal contribution
$\varnothing$	$\{X_1\}$	$1/3$	$\Lambda(\beta_0 + \beta_1 X_1) - \Lambda(\beta_0)$
$\{X_2\}$	$\{X_1, X_2\}$	$1/6$	$\Lambda(\beta_0 + \beta_1 X_1 + \beta_2 X_2) - \Lambda(\beta_0 + \beta_2 X_2)$
$\{X_3\}$	$\{X_1, X_3\}$	$1/6$	$\Lambda(\beta_0 + \beta_1 X_1 + \beta_3 X_3) - \Lambda(\beta_0 + \beta_3 X_3)$
$\{X_2, X_3\}$	$\{X_1, X_2, X_3\}$	$1/3$	$\Lambda(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3) - \Lambda(\beta_0 + \beta_2 X_2 + \beta_3 X_3)$

Example: Shapley Values for a Logit Model

Example: Shapley Values for a Logit Model (continued)

Collecting all terms, the Shapley value of X_1 is:

$$\begin{aligned}\phi_1(\widehat{f}) = & \frac{1}{3}\Big(\Lambda(\beta_0+\beta_1 X_1) - \Lambda(\beta_0)\Big) \\ & + \frac{1}{6}\Big(\Lambda(\beta_0+\beta_1 X_1+\beta_2 X_2) - \Lambda(\beta_0+\beta_2 X_2)\Big) \\ & + \frac{1}{6}\Big(\Lambda(\beta_0+\beta_1 X_1+\beta_3 X_3) - \Lambda(\beta_0+\beta_3 X_3)\Big) \\ & + \frac{1}{3}\Big(\Lambda(\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_3) - \Lambda(\beta_0+\beta_2 X_2+\beta_3 X_3)\Big).\end{aligned}$$

Intuition: $\phi_1(\widehat{f})$ is the **average marginal contribution of** X_1 to the prediction, taken across all possible subsets of the other features.

Example: Shapley Values in a Linear Model

Example: Shapley Values and Linear Models

Consider the **linear prediction model**:

$$\widehat{f}(x) \;=\; \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p.$$

The **Shapley contribution** ϕ_j of the j^{th} feature to the prediction $\widehat{f}(x)$ can be expressed as the deviation of the feature's effect from its average effect:

$$\phi_j(x) \;=\; \hat{\beta}_j x_j \;-\; \mathbb{E}\Big[\hat{\beta}_j x_j\Big] \;=\; \hat{\beta}_j \big(x_j - \mathbb{E}[x_j]\big).$$

Thus, the prediction can be decomposed as:

$$\widehat{f}(x) \;=\; \mathbb{E}[\widehat{f}(x)] \;+\; \sum_{j=1}^{p} \phi_j(x).$$

Properties of the Shapley Values

Properties of the Shapley Values

The Shapley values satisfy the following **axioms**:

Efficiency: The feature contributions must add up to the difference of prediction for x and the average.

$$\sum_{i=1}^{p} \phi_j\left(\widehat{f}\right) = \widehat{f}(x) - \mathbb{E}\left(\widehat{f}(x)\right)$$

Symmetry: The contributions of two feature values j and k should be the same if they contribute equally to all possible coalitions.

$$\widehat{f}\left(S \cup \{x_j\}\right) = \widehat{f}\left(S \cup \{x_k\}\right) \Longleftrightarrow \phi_j\left(\widehat{f}\right) = \phi_k\left(\widehat{f}\right)$$

Dummy: A feature j that does not change the predicted value – regardless of which coalition of feature values it is added to – should have a Shapley value of 0.

$$\widehat{f}\left(S \cup \{x_j\}\right) = \widehat{f}(S) \Longleftrightarrow \phi_j\left(\widehat{f}\right) = 0$$

Properties of the Shapley Values

Properties of the Shapley Values

The Shapley values satisfy the following **axioms**: Additivity: For a game with combined payouts $\widehat{f}_1 + \widehat{f}_2$ the respective Shapley values are as follows:

$$\phi_j\left(\widehat{f}_1\right) + \phi_j\left(\widehat{f}_2\right)$$

Note : Suppose you trained a random forest, which means that the prediction is an average of many decision trees. The Additivity property guarantees that for a feature value, you can calculate the Shapley value for each tree individually, average them, and get the Shapley value for the feature value for the random forest.

Feasibility of Shapley Values

Remarks

- This definition of the Shapley Value becomes infeasible when p is large as the total number of coalitions required to compute $\phi_1\left(\widehat{f}\right), \ldots, \phi_p\left(\widehat{f}\right)$ is equal to $p \times 2^{p-1}$.
- Example : $5,120$ coalitions for $p = 10$, $485,760$ coalitions for $p = 20$, etc.
- For large p, Lundberg and Lee (2016), and Lundberg et al. (2019) have developed fast algorithms for the estimation of $\phi_j\left(\widehat{f}\right)$.


Lundberg, S.M. and Lee, S.L. (2016) A Unified Approach to Interpreting Model Predictions, 31st Conference on Neural Information Processing Systems.


Lundberg, S.M., Erion, G.G., and Lee, S.L. (2019), Consistent Individualized Feature Attribution for Tree Ensembles, arXiv 1802.03888.

SHAP: SHapley Additive exPlanations

Definition: SHAP

SHAP (SHapley Additive exPlanations) proposed by Lundberg and Lee (2016) is a framework that estimates Shapley values in order to attribute the prediction of a model to each of its input features in an additive and consistent way.

For a model $\widehat{f}(X)$ with p input features, the prediction for an observation $x = (x_1, \ldots, x_p)$ can be decomposed as

$$\widehat{f}(x) = \phi_0 + \sum_{j=1}^{p} \phi_j,$$

where $\phi_0 = \mathbb{E}\big[\widehat{f}(X)\big]$ is the expected model prediction, and ϕ_j is the Shapley value associated with feature j.

Lundberg, S.M. and Lee, S.-I. (2016). *A Unified Approach to Interpreting Model Predictions*. Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS).

Kernel SHAP: Intuition

Problem: Computing exact Shapley values requires evaluating all 2^p coalitions of features, which is infeasible for large p.

Solution: Kernel SHAP

- Approximates Shapley values using a surrogate regression model.
- Each coalition of features is represented by a binary vector $z' \in \{0,1\}^p$, where $z'_j = 1$ if feature j is present, 0 otherwise.
- For each coalition z', compute the prediction $f(z)$ of the original model.
- Fit a weighted linear regression that learns additive contributions of features.

Key idea: SHAP values ϕ_j are obtained as the coefficients of this regression, ensuring local accuracy and consistency.

Kernel SHAP: Intuition

Surrogate regression:

$$f(z) \approx \phi_0 + \sum_{j=1}^{p} \phi_j z'_j,$$

where ϕ_j are the SHAP values.

Weighted least squares:

$$\min_{\phi} \sum_{z'} \pi_x(z') \Big(f(z) - \phi_0 - \sum_{j=1}^{p} \phi_j z'_j \Big)^2,$$

with weights

$$\pi_x(z') = \frac{(p-1)}{\binom{p}{|z'|} \, |z'| \, (p - |z'|)},$$

where $|z'|$ is the number of features included in the coalition.

Practical computation:

- Not all coalitions are used: a subset is sampled from $\{0,1\}^p$.
- The kernel $\pi_x(z')$ guarantees that the regression solution converges to exact Shapley values if all coalitions are included.

Kernel SHAP: Numerical Example (1/2)

Toy model with two features

Consider a model

$$f(x_1, x_2) = 3 + 2x_1 + x_2.$$

We want to explain the prediction at $x = (1, 1)$, i.e. $f(1, 1) = 6$.

Coalitions: Represent presence/absence of features with $z' = (z'_1, z'_2) \in \{0, 1\}^2$.

Coalition z'	Features included	$f(z)$	z_1	z_2
(0,0)	none	3	–	–
(1,0)	$\{x_1\}$	5	1	–
(0,1)	$\{x_2\}$	4	–	1
(1,1)	$\{x_1, x_2\}$	6	1	1

Kernel SHAP: Numerical Example (2/2)

Toy model with two features (continued)

Surrogate regression:

$$f(z) \approx \phi_0 + \phi_1 z_1' + \phi_2 z_2'.$$

Solving the weighted regression yields:

$$\phi_0 = 3, \qquad \phi_1 = 2, \qquad \phi_2 = 1.$$

Interpretation: The SHAP decomposition of the prediction is

$$f(1,1) = \underbrace{3}_{\phi_0} + \underbrace{2}_{\phi_1} + \underbrace{1}_{\phi_2}.$$

Each feature's contribution matches its true effect in the model.

Ways to Report SHAP Values

Local vs Global Plots

Local plots (e.g. force plot, waterfall plot) explain single predictions, while global plots (e.g. summary plot, bar plot) show overall model behavior.

Common visualizations:

- Force plot: Local explanation that shows how each feature pushes the prediction above or below the baseline.
- Waterfall plot: Stepwise visualization of how each feature contribution moves the prediction from the baseline to the final value.
- Summary plot: Global explanation combining feature importance (mean absolute SHAP values) and distribution of effects for all observations.
- Feature Importance (Bar) plot: Average absolute SHAP values across the dataset to rank features by importance.
- Dependence plot: Scatter plot of SHAP values of a feature against its actual values, showing interaction patterns.

SHAP Force Plot

Definition: SHAP Force Plot

A **force plot** provides a local explanation of a single prediction by visualizing how each feature's SHAP value pushes the output away from the baseline (mean model prediction) toward the final prediction.

Principle:

- Start from the baseline value $\phi_0 = \mathbb{E}[f(X)]$, i.e. the mean prediction of the model.
- Visualize features as "forces" that either push the prediction upward or downward.
- The final position after all pushes corresponds to the model prediction.

Interpretation:

- Positive SHAP values push the prediction upward from the mean prediction (shown in red).
- Negative SHAP values push the prediction downward from the mean prediction (shown in blue).
- The balance of these forces explains the final model output.

SHAP Force Plot

Figure 19: SHAP force plot

FIGURE 5.50: SHAP values to explain the predicted cancer probabilities of two individuals. The baseline -- the average predicted probability -- is 0.066. The first woman has a low predicted risk of 0.06. Risk increasing effects such as STDs are offset by decreasing effects such as age. The second woman has a high predicted risk of 0.71. Age of 51 and 34 years of smoking increase her predicted cancer risk.

Source: Molnar (2025)

SHAP Waterfall

Definition: SHAP Waterfall

A **waterfall plot** provides a local explanation of a single prediction by showing how each feature's SHAP value moves the output from the baseline (mean model prediction) to the final prediction.

Principle:

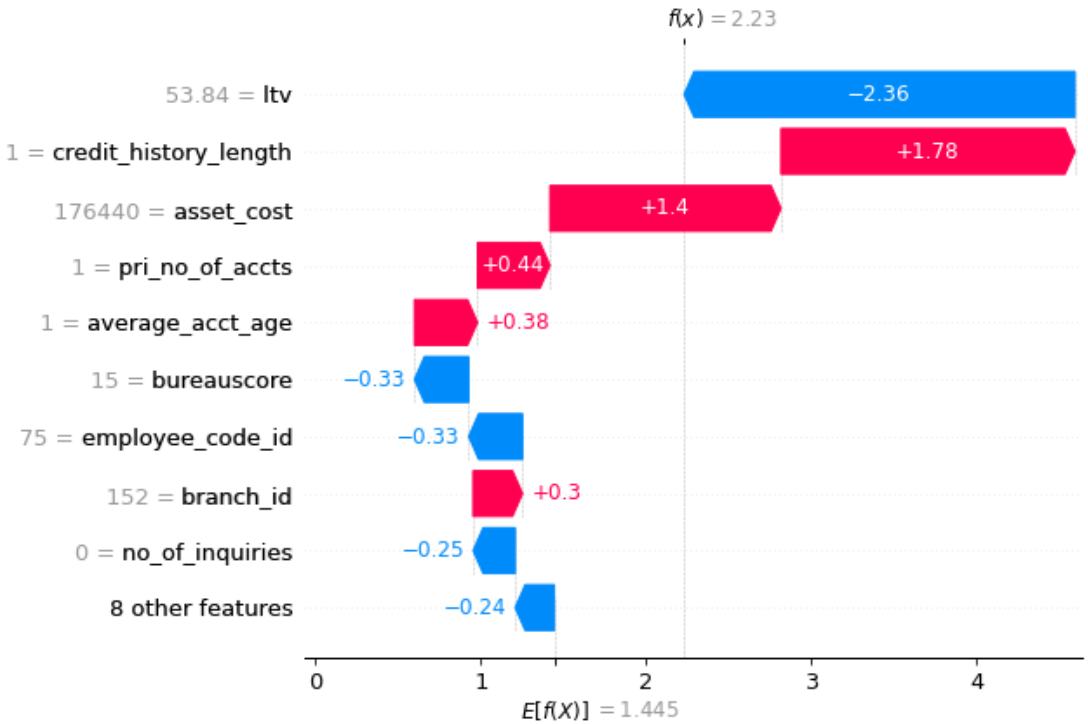
- Start from the baseline value $\phi_0 = \mathbb{E}[f(X)]$, i.e. the mean prediction of the model.
- Add contributions ϕ_j one by one, ordered by magnitude or importance.
- Visualize the cumulative effect as a stepwise path (like a waterfall).

Interpretation:

- Positive SHAP values push the prediction upward from the mean prediction.
- Negative SHAP values push the prediction downward from the mean prediction.
- The final step equals the model prediction for the instance.

SHAP Waterfall

Figure 20: SHAP feature importance



Source: Molnar (2025)

SHAP Bar Plot of Feature Importance

Definition: SHAP Feature Importance

The **SHAP feature importance** for feature j is defined as

$$I_j = \sum_{i=1}^{n} \left| \phi_j^{(i)} \right|,$$

where $\phi_j^{(i)}$ is the SHAP value of feature j for observation i.

Interpretation: Features with larger absolute SHAP values on average have a stronger impact on model predictions, and are therefore considered more important.

SHAP Bar Plot of Feature Importance

Figure 21: SHAP feature importance

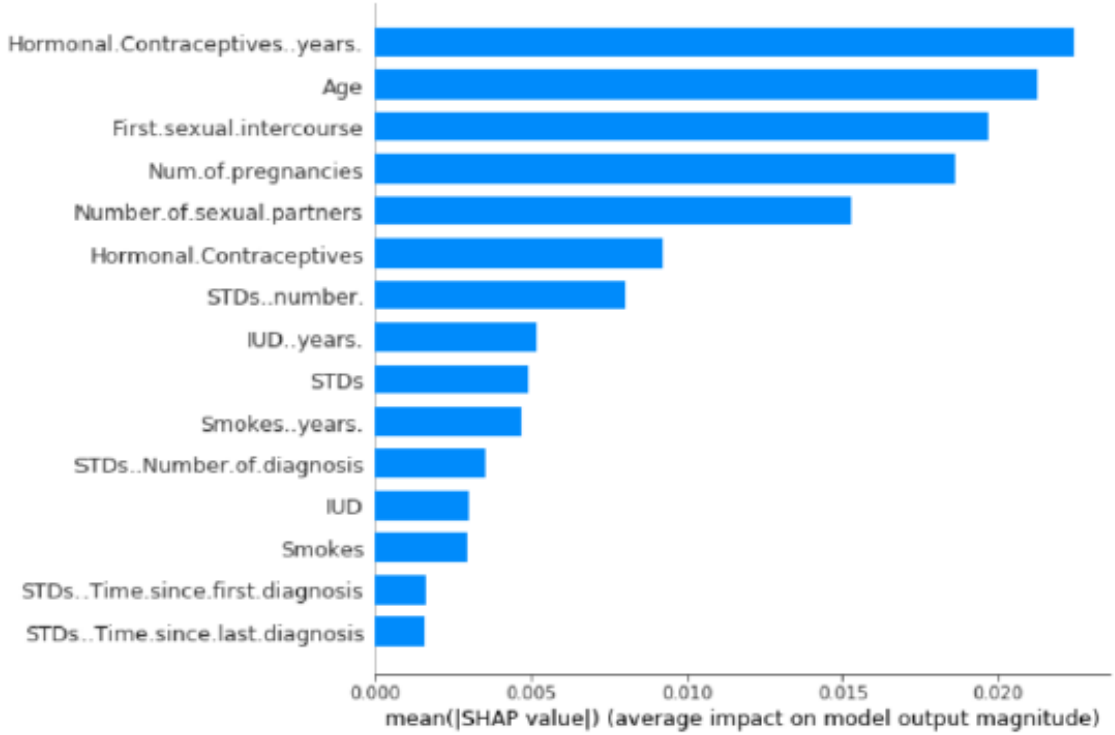


FIGURE 5.51: SHAP feature importance measured as the mean absolute Shapley values. The number of years with hormonal contraceptives was the most important feature, changing the predicted absolute cancer probability on average by 2.4 percentage points (0.024 on x-axis).

Source: Molnar (2025)

SHAP Summary Plot

Definition: SHAP Summary Plot

A **summary plot** provides a global explanation of a model by combining feature importance with the distribution of SHAP values across all observations.

Principle:

- Each point corresponds to a Shapley value $\phi_j^{(i)}$ for feature j and instance i.
- The y-axis lists features ranked by importance.
- The x-axis shows the Shapley value (effect on the prediction).
- The color encodes the original feature value (low to high).

Interpretation:

- The spread of points along the x-axis shows the strength and variability of feature effects.
- The vertical ordering indicates overall importance.
- The color gradient reveals how the feature value relates to its contribution (e.g. high values increase the prediction).

SHAP Summary Plot

Figure 22: SHAP summary plot

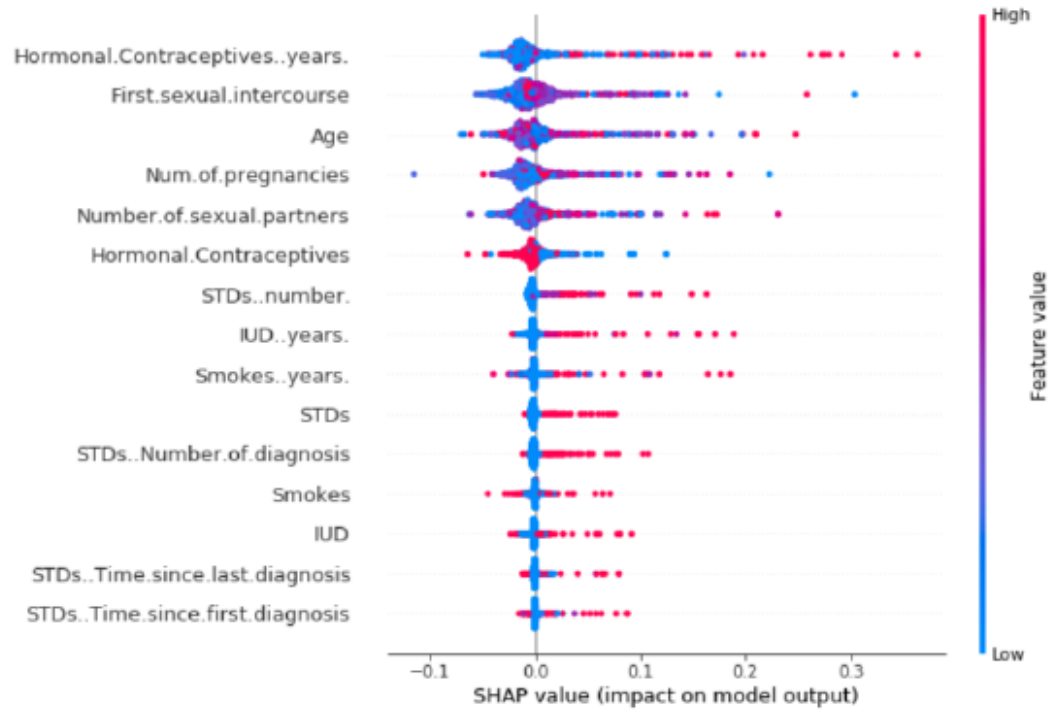


FIGURE 5.52: SHAP summary plot. Low number of years on hormonal contraceptives reduce the predicted cancer risk, a large number of years increases the risk. Your regular reminder: All effects describe the behavior of the model and are not necessarily causal in the real world.

Source: Molnar (2025)

SHAP Dependence Plot

Definition: SHAP dependence plot

The **SHAP dependence plot** simply displays the scatter plot of the feature value and the associated Shapley values for all the instances.

SHAP Dependence Plot

Figure 23: SHAP dependence plot

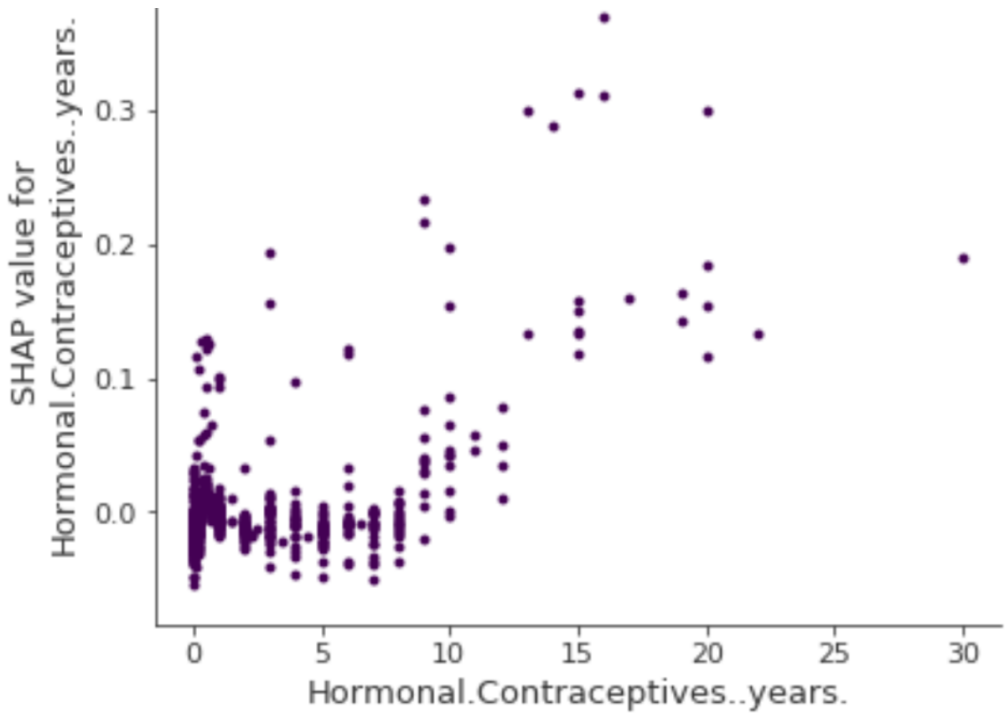


FIGURE 5.53: SHAP dependence plot for years on hormonal contraceptives. Compared to 0 years, a few years lower the predicted probability and a high number of years increases the predicted cancer probability.

Source: Molnar (2025)

SHAP: Advantages and Limits

Advantages

1. Strong theoretical foundation. SHAP is based on Shapley values from cooperative game theory, the only attribution method that uniquely satisfies axioms of efficiency, symmetry, dummy, and additivity.
2. Consistency and local accuracy. Feature contributions sum exactly to the model prediction, starting from the mean prediction.
3. Model-agnostic. SHAP can be applied to any predictive model, including black-box models such as ensembles or neural networks.

Limits

1. Computational cost. Exact computation requires 2^p coalitions, which is infeasible for most real-world problems. In practice, SHAP relies on approximations (e.g. Kernel SHAP, Tree SHAP).
2. Correlation between features. SHAP assumes feature independence when building coalitions, which can bias attributions in the presence of strong correlations.
3. Interpretation challenges. Large numbers of features or instances may make SHAP visualizations harder to interpret without aggregation.

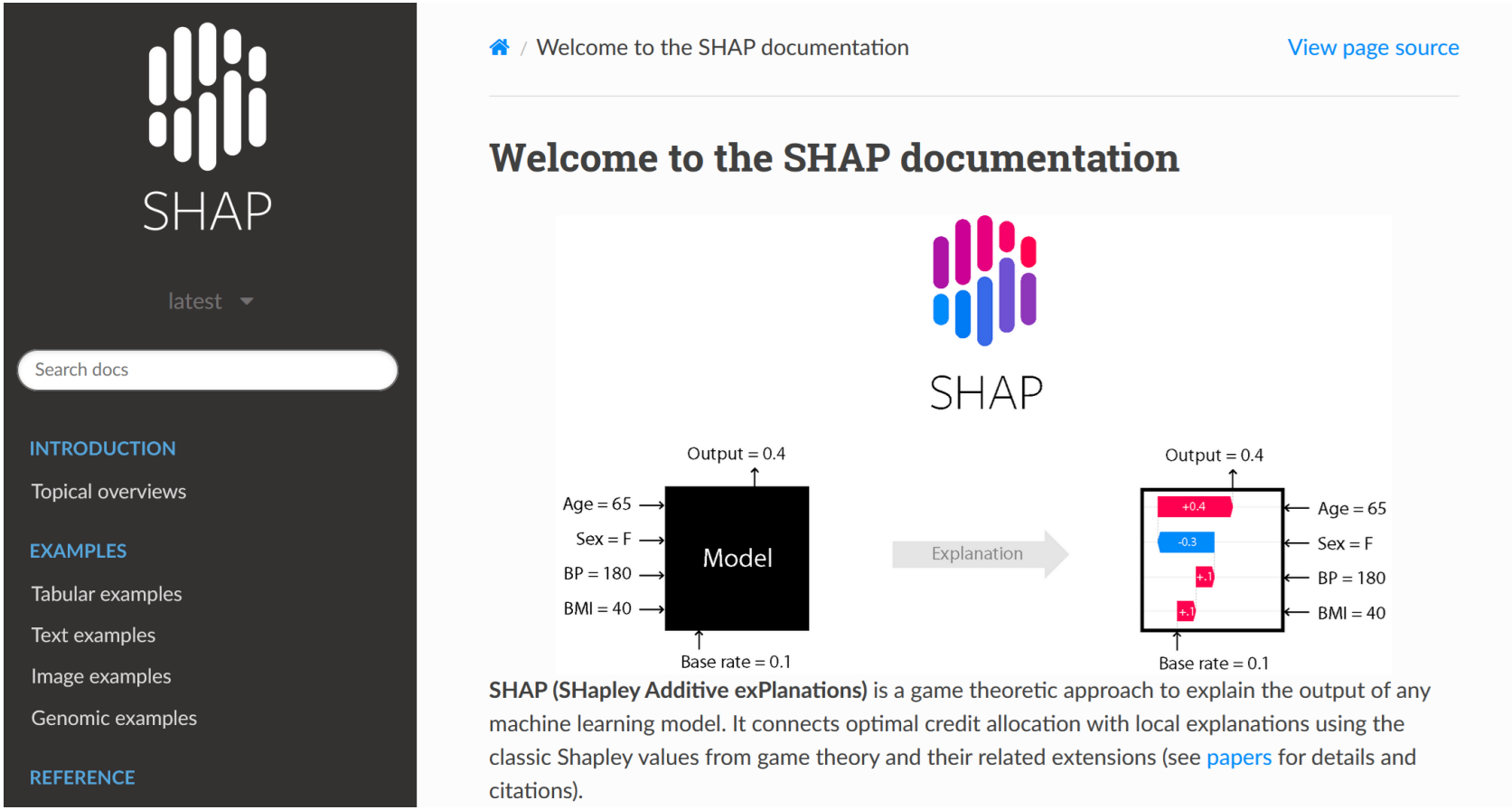
HERE

SHAP Libraries in Python

SHAP values in Python are estimated with the `shap` library.

- The library offers a unified interface via `shap.Explainer`, which selects or wraps the appropriate explainer for a given model.
- For tree ensembles, `TreeExplainer` implements the exact and efficient TreeSHAP algorithm.
- For arbitrary black-box models, `KernelExplainer` provides a model-agnostic approximation of SHAP values.
- For linear models, `LinearExplainer` delivers fast analytic SHAP values.
- The package includes rich visualizations—`summary_plot`, `force`, `waterfall`, and `decision_plot`—for both global and local explanations.
- `scikit-learn` does not compute SHAP values; the `shap` library works with scikit-learn models by using their `predict` or `predict_proba` outputs.

SHAP Implementation



Source: SHAP documentation

Example of SHAP Implementation

Example: SHAP Implementation

```
import shap

# Model Fit
model = RandomForestRegressor(n_estimators=100).fit(X, y)

# SHAP values
explainer = shap.Explainer(model, X)
shap_values = explainer(X_test)

# Visualization: Beeswarm summary
shap.plots.beeswarm(shap_values, max_display=10)
```

Google Colab: Click here.

Exercise: SHAP for Credit Scoring Model

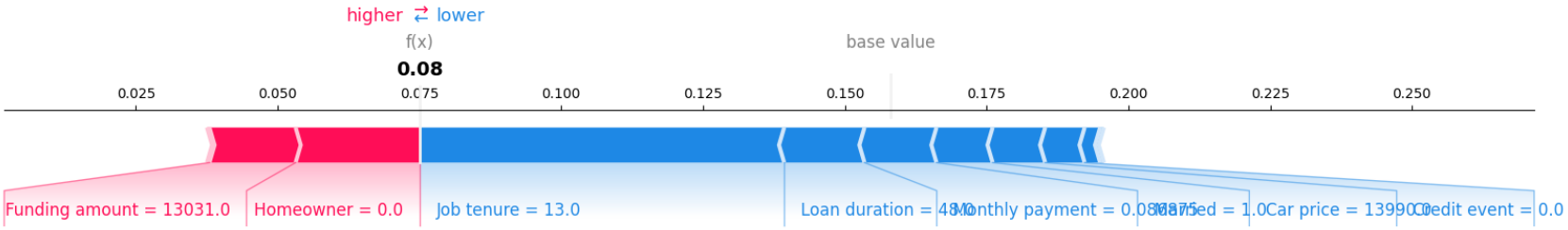
Exercise: SHAP for Local & Global Explanations

Build a **credit scoring model** with a Random Forest and explain it using **SHAP values**.

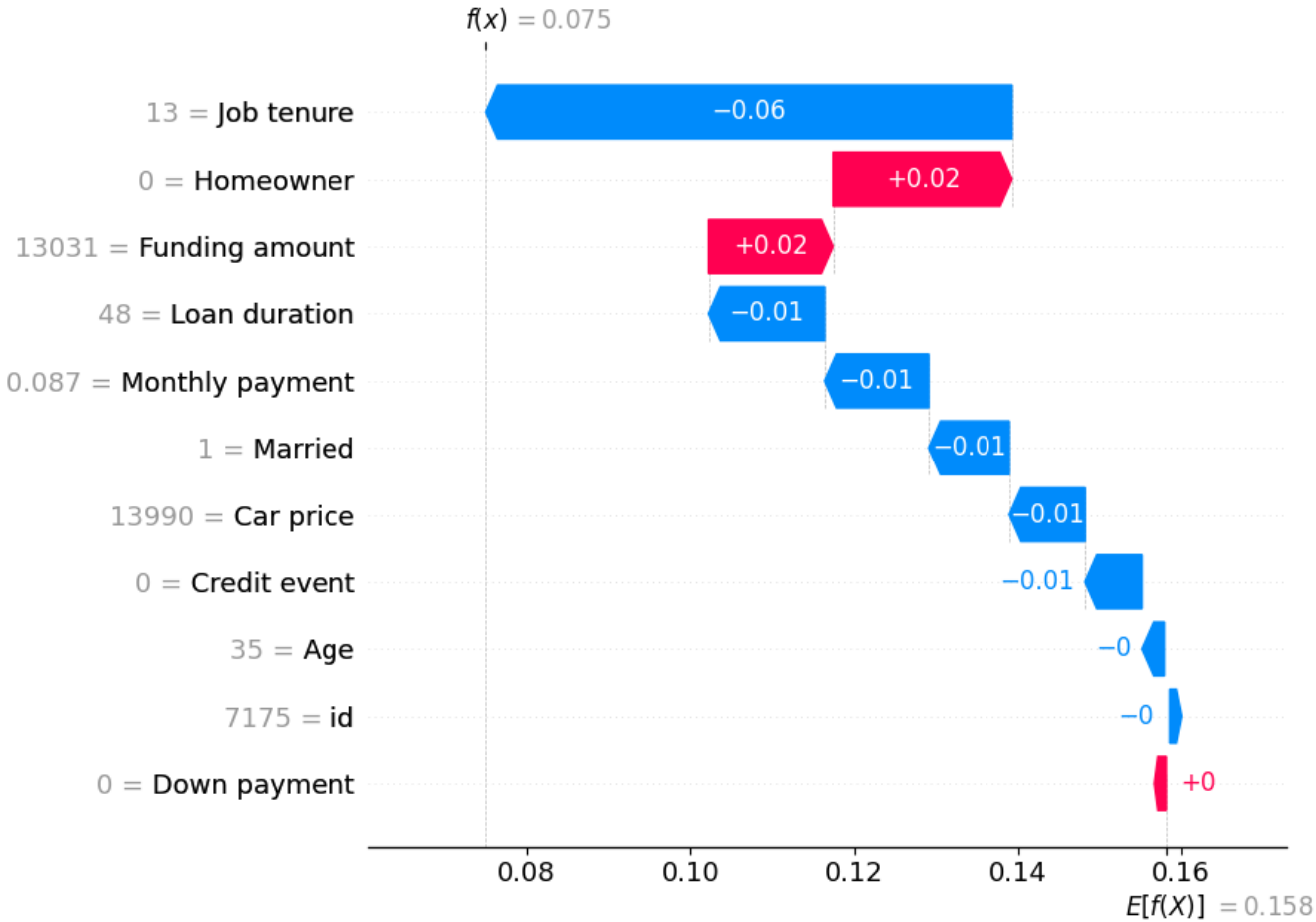
- Use the dataset `Scoring_Database.xlsx` (sheet `Data`); the target is `Default` (1 = default, 0 = no default).
- Split the data into training and test sets (e.g., 70%/30%) with stratification; preprocess numerics/categoricals (imputation, one-hot encoding).
- Train a Random Forest Classifier and evaluate on the test set its predictive performance (accuracy, ROC–AUC, etc.).
- Compute the **SHAP** values on the transformed test features.
- Produce the following SHAP visualizations:
  - A *force plot* for the test instance n°15.
  - A *waterfall plot* for the same instance.
  - A *summary (beeswarm) plot* over the test set.
  - A *feature-importance bar plot* based on mean absolute SHAP values.

Google Colab: Click here.

Exercise: Python Output

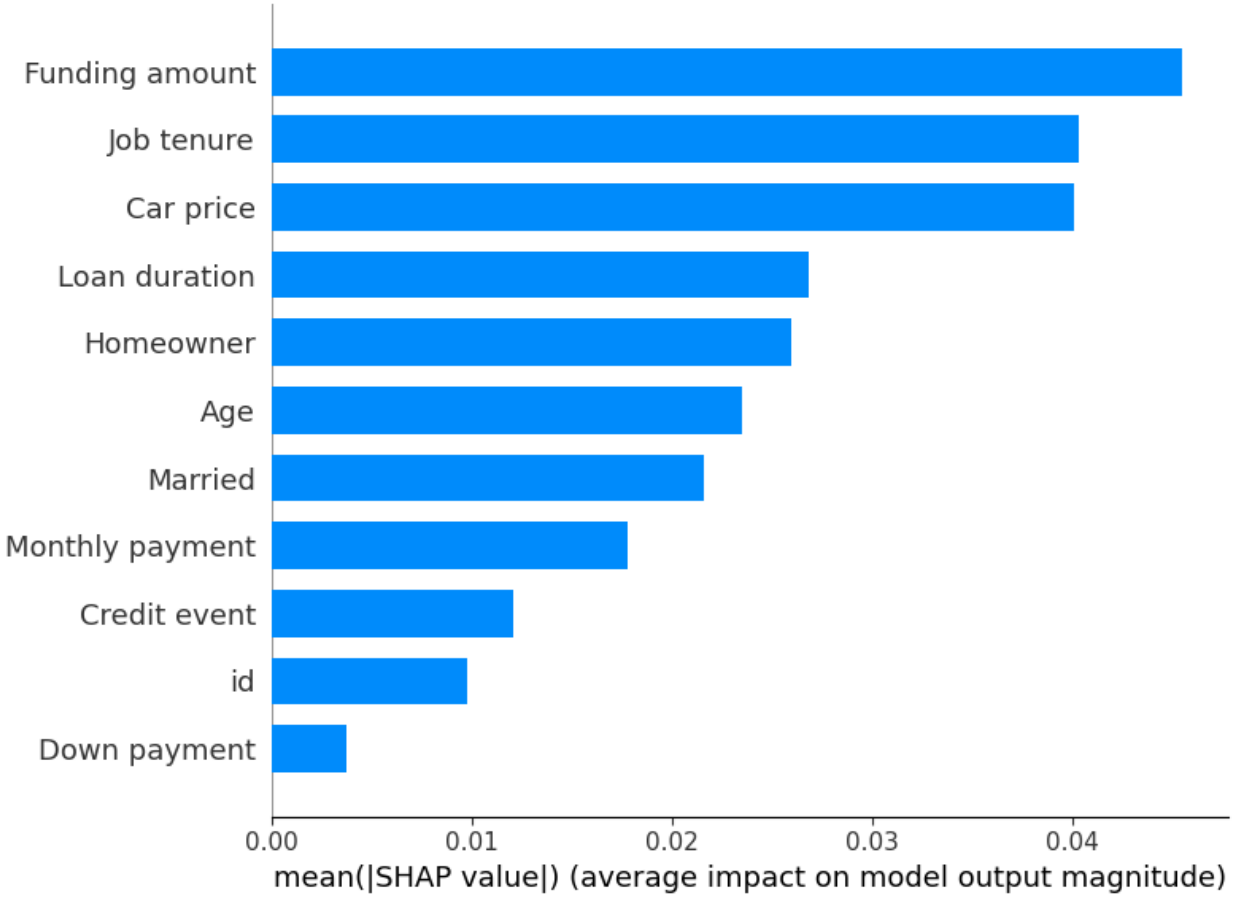


Exercise: Python Output



Exercise: Python Output

Exercise: Python Output



Local Post hoc Interpretability

Key concepts

1. LIME (Local Interpretable Model-agnostic Explanations)
2. Individual Conditional Expectation (ICE)
3. Shapley value
4. SHAP (SHapley Additive exPlanations)

Outline

Which Tool to Choose? Global vs Local Explanations

Post-hoc interpretability methods differ in scope and reliability. They provide only a partial understanding of the model, and their usefulness may vary considerably from case to case.

- *Graphical tools (PDP, ALE, ICE):* Show the effect of an explanatory variable on predictions, usually at the global level.
- *Feature importance measures:* Rank explanatory variables by their average contribution to predictions.
- *Surrogate models:* Approximate the global functioning of a complex model with a simpler interpretable model.
- *Shapley values (SHAP):* Allocate contributions of each feature to individual predictions based on cooperative game theory.
- *LIME:* Provide simple local approximations of the model in the vicinity of an observation.

Each method sheds light on a different facet of the model but none offers a complete explanation on its own.

The Limits of Post-hoc Interpretability Methods

Key criticisms:

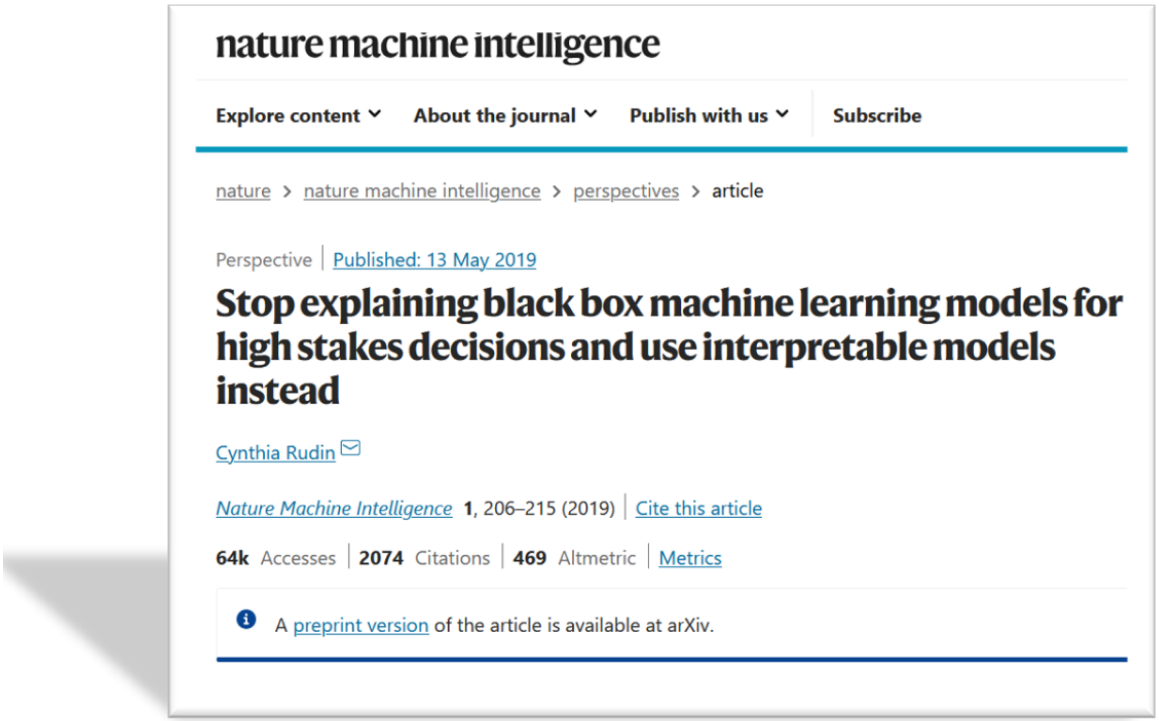
1. By definition, post-hoc methods may provide explanations that are not fully faithful to what the original model calculates (Rudin, 2019).
2. Explanations can be unstable, varying across runs or depending on small changes in the data (Chen et al., 2023).
3. Different methods may yield contradictory explanations for the same prediction (Krishna et al., 2022).
4. Many methods assume feature independence, which is rarely valid in real-world applications.


Krishna, S. et al. (2022). The disagreement problem in explainable machine learning: A practitioner's perspective. arXiv, 2022.


Chen Y, Calabrese R., and B. Martin-Barragan (2023). Effects of imbalanced datasets on interpretable machine learning. 2023; EJOR.

Post-hoc interpretability

Takeaway: For high-stakes applications, interpretable models may be preferable.



nature machine intelligence

Explore content | About the journal | Publish with us | Subscribe

nature > nature machine intelligence > perspectives > article

Perspective | Published: 13 May 2019

Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead

Cynthia Rudin

Nature Machine Intelligence **1**, 206–215 (2019) | Cite this article

64k Accesses | **2074** Citations | **469** Altmetric | Metrics

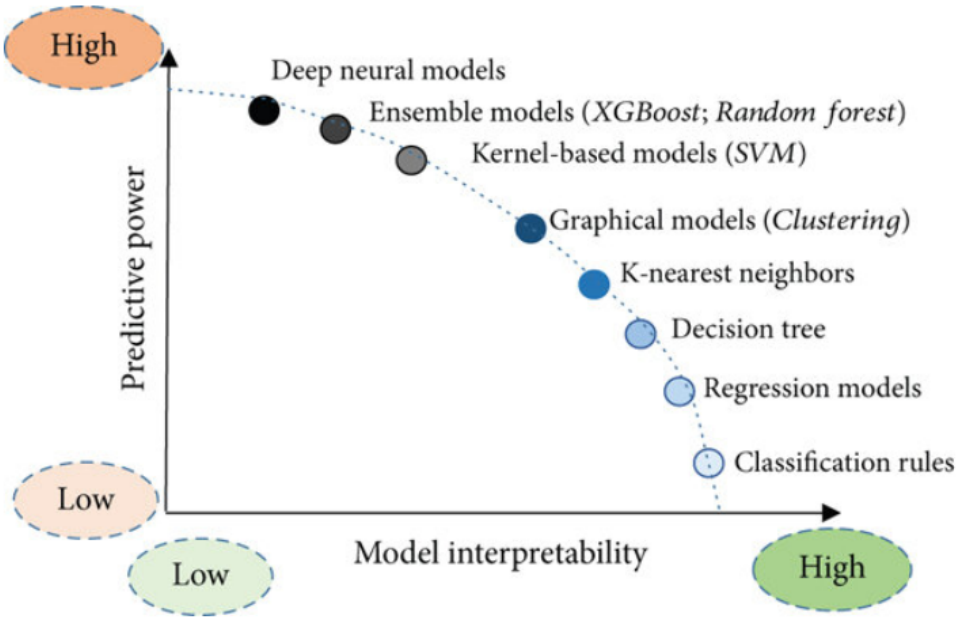
A preprint version of the article is available at arXiv.

Rudin (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature*, Machine Learning Intelligence, 1, 206–215.

Trade-off between Performance and Interpretability

Why are we still using post-hoc interpretability methods?

- A common belief among data scientists is that there exists a trade-off between model performance and interpretability.
- If such a trade-off holds, it justifies the use of post-hoc interpretability methods despite their limitations, given the potential gains in predictive accuracy.



Example of the belief in a trade-off between predictive performance and interpretability: Kumar et al. (2021)

The Rashomon Set: Rethinking the Trade-off

The theory of Rashomon Set: Semenova et al., (2022):

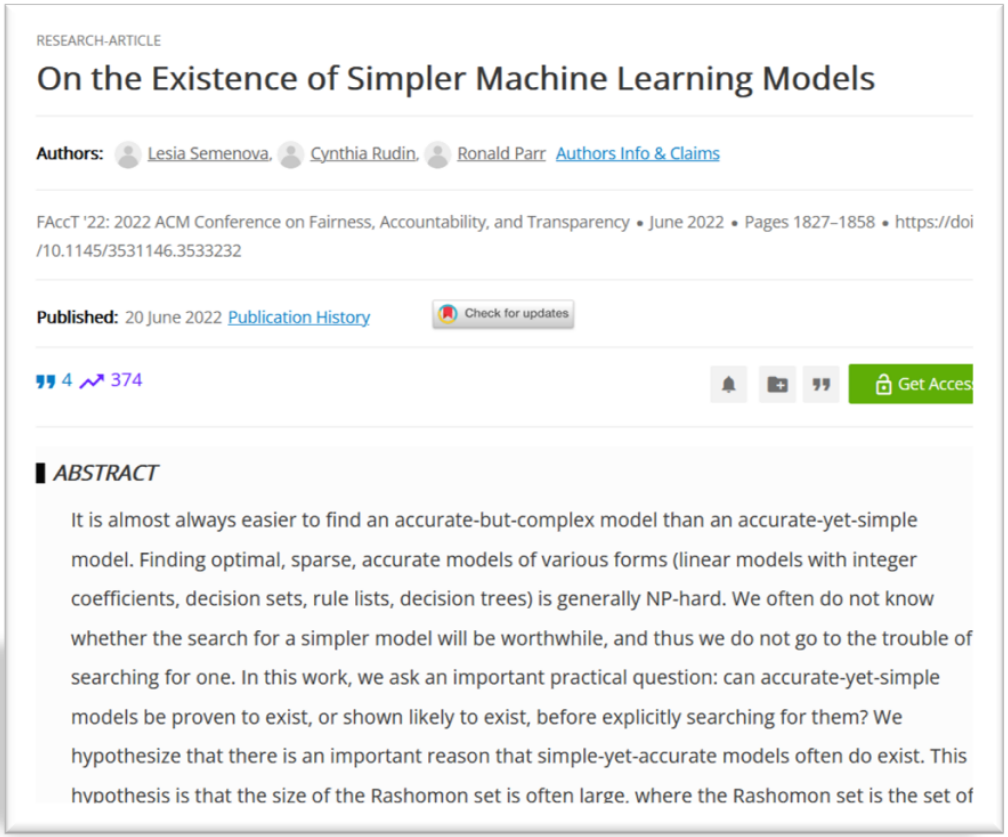
- Semenova et al., (2022) introduce the theory of Rashomon set.
- The Rashomon set is the collection of predictive models whose accuracy is very close to that of the optimal black-box model.
- If this set is sufficiently large, it is highly probable that it contains interpretable models.
- In such cases, one can achieve both high accuracy and interpretability, eliminating the trade-off.

Implication:

- If the Rashomon set exists for a given task, interpretable models can replace black-box models without significant loss of accuracy.
- No need to rely on post-hoc explanations (e.g. Shapley values, LIME, counterfactuals).

Semenova, L., Rudin, C. and Parr. R., (2022). On the existence of simpler machine learning models, *ACM Conference on Fairness, Accountability, and Transparency*, 1827-1858.

Simpler Machine Learning Models



RESEARCH-ARTICLE

On the Existence of Simpler Machine Learning Models

Authors: Lesia Semenova, Cynthia Rudin, Ronald Parr Authors Info & Claims

FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency • June 2022 • Pages 1827–1858 • https://doi/10.1145/3531146.3533232

Published: 20 June 2022 Publication History

Check for updates

4 374

Get Acces

ABSTRACT

It is almost always easier to find an accurate-but-complex model than an accurate-yet-simple model. Finding optimal, sparse, accurate models of various forms (linear models with integer coefficients, decision sets, rule lists, decision trees) is generally NP-hard. We often do not know whether the search for a simpler model will be worthwhile, and thus we do not go to the trouble of searching for one. In this work, we ask an important practical question: can accurate-yet-simple models be proven to exist, or shown likely to exist, before explicitly searching for them? We hypothesize that there is an important reason that simple-yet-accurate models often do exist. This hypothesis is that the size of the Rashomon set is often large, where the Rashomon set is the set of

Semenova, L., Rudin, C. and Parr. R., (2022). On the existence of simpler machine learning models, *ACM Conference on Fairness, Accountability, and Transparency*, 1827-1858.

Inherently Interpretable Machine Learning Models

Definition: Inherently Interpretable Models

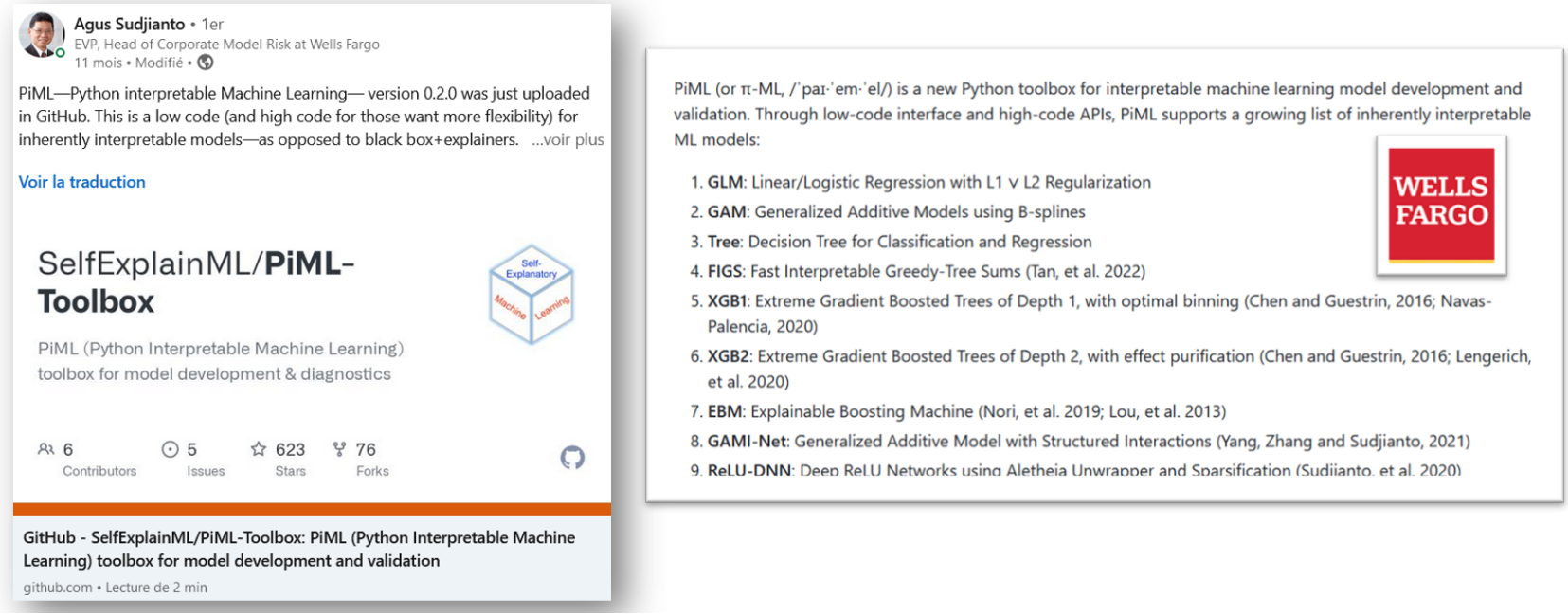
An **inherently interpretable machine learning model** is a model whose structure and parameters can be directly understood by humans, without requiring post-hoc explanation methods.

Examples include sparse linear models, rule-based models, and generalized additive models (GAMs).

Key properties:

- *Transparency*: The prediction process can be directly inspected and explained.
- *Simplicity*: The number of parameters, rules, or terms is limited and interpretable.
- *Faithfulness*: Explanations reflect exactly how the model makes predictions (unlike post-hoc methods, which approximate).

PiML: Python interpretable Machine Learning



An example of toolbox of interpretable ML models: Python interpretable Machine Learning (PiML)

Inherently Interpretable Machine Learning Models

In this section, we introduce two interpretable machine learning models (among many others):

1. **Penalised Logistic Tree Regression (PLTR)**
2. **Generalized Additive Models (GAM)**

PLTR: Penalised Logistic Tree Regression

PLTR: Penalised Logistic Tree Regression

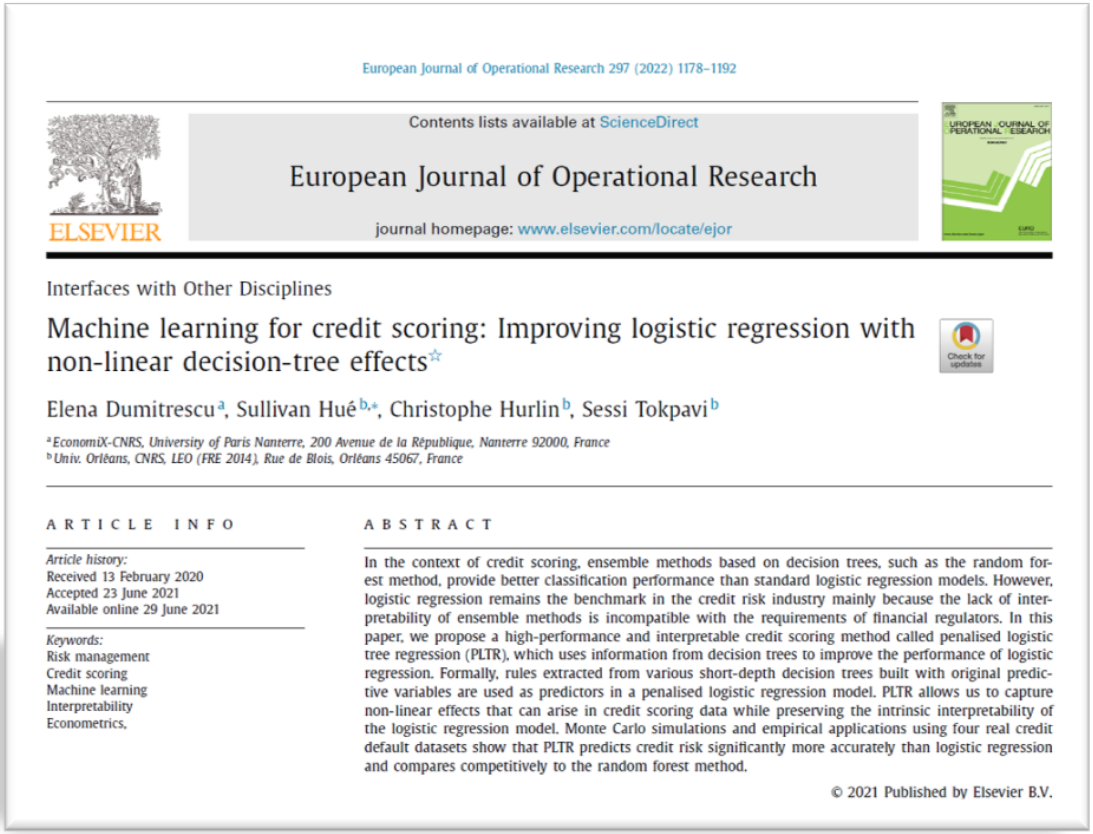
Definition: Penalised Logistic Tree Regression (PLTR)

The **Penalised Logistic Tree Regression (PLTR)** is high performance and interpretable credit scoring method which uses information from decision trees to improve the performance of logistic regression.

Remark: This method is not restricted to credit scoring applications and can be used for other classification and regression (after some small adjustments) problems.

Dumitrescu, E., Hué, S., Hurlin, C. and Tokpavi, S. (2022), Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects, *European Journal of Operational Research*, 297(3), 1178-1192.

PLTR: Reference



European Journal of Operational Research 297 (2022) 1178–1192

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Interfaces with Other Disciplines

Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects☆

Elena Dumitrescu[a], Sullivan Hué[b,*], Christophe Hurlin[b], Sessi Tokpavi[b]

[a] EconomiX-CNRS, University of Paris Nanterre, 200 Avenue de la République, Nanterre 92000, France
[b] Univ. Orléans, CNRS, LEO (FRE 2014), Rue de Blois, Orléans 45067, France

ARTICLE INFO

Article history:
Received 13 February 2020
Accepted 23 June 2021
Available online 29 June 2021

Keywords:
Risk management
Credit scoring
Machine learning
Interpretability
Econometrics,

ABSTRACT

In the context of credit scoring, ensemble methods based on decision trees, such as the random forest method, provide better classification performance than standard logistic regression models. However, logistic regression remains the benchmark in the credit risk industry mainly because the lack of interpretability of ensemble methods is incompatible with the requirements of financial regulators. In this paper, we propose a high-performance and interpretable credit scoring method called penalised logistic tree regression (PLTR), which uses information from decision trees to improve the performance of logistic regression. Formally, rules extracted from various short-depth decision trees built with original predictive variables are used as predictors in a penalised logistic regression model. PLTR allows us to capture non-linear effects that can arise in credit scoring data while preserving the intrinsic interpretability of the logistic regression model. Monte Carlo simulations and empirical applications using four real credit default datasets show that PLTR predicts credit risk significantly more accurately than logistic regression and compares competitively to the random forest method.

Dumitrescu, E., Hué, S., Hurlin, C. and Tokpavi, S. (2022), Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects, *European Journal of Operational Research*, 297(3), 1178-1192.

PLTR: Summary of the Approach

Penalised Logistic Tree Regression (PLTR) combines the flexibility of decision trees with the interpretability of logistic regression.

It proceeds in three main steps:

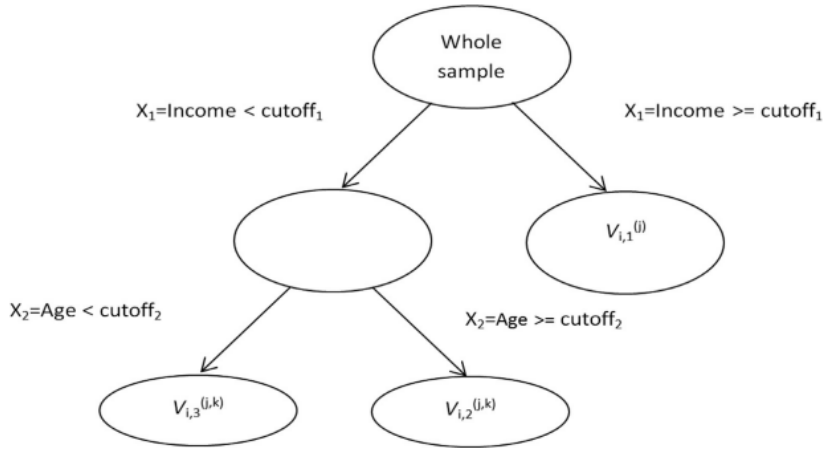
1. Tree-based splits: Build univariate and bivariate trees (with at most two variables). For each tree, construct binary variables associated with the leaves (all leaves except one are coded as binary dummies).
2. Logistic regression with thresholds: Introduce these binary variables into a logistic regression model, capturing nonlinear and interaction effects in a transparent way.
3. Penalisation: Estimate the logistic model using adaptive Lasso to ensure sparsity and enhance interpretability.

PLTR: First Step

First Step: Splitting process:

Consider a pair of features (j, k), e.g. (Income, Age). Construct binary variables $\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}, \mathcal{V}_{i,3}^{(j,k)}$ associated with the leaves for an instance i, where:

- The first binary variable $\mathcal{V}_{i,1}^{(j)}$ captures univariate threshold effects. It takes the value one if the income of individual i is higher than an estimated income threshold, and zero otherwise.
- The second and third binary variables, $\mathcal{V}_{i,2}^{(j,k)}$ and $\mathcal{V}_{i,3}^{(j,k)}$, capture bivariate threshold effects. They equal one when the person's income is below its threshold while his/her age is above (or below) an estimated age threshold, and zero otherwise.



Source: Dumitrescu et al. (2022)

PLTR: Second Step

Second Step: Logistic Regression

All the binary variables representing the univariate and bivariate threshold effects obtained in the previous step are included in the logistic regression:

$$\mathbb{P}(y_i = 1 | \mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) = \frac{1}{1 + exp\left[-\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta)\right]}$$

with $\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) = \beta_0 + \sum_{j=1}^{p} \alpha_j x_j + \sum_{j=1}^{p} \beta_j \mathcal{V}_{i,1}^{(j)} + \sum_{j=1}^{p-1} \sum_{k=j+1}^{p} \gamma_{j,k} \mathcal{V}_{i,2}^{(j,k)}$.

The corresponding likelihood is

$$\mathcal{L}(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i log[F(\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta))] \right] + \frac{1}{N} \sum_{i=1}^{N} \left[(1 - y_i) log[1 - F(\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta))] \right].$$

PLTR: Third Step

Third Step: Penalization

The logistic regression model is estimated by an adaptive Lasso penalized regression:

$$\hat{\Theta}_{alasso}(\lambda) = \underset{\Theta}{argmin} - \mathcal{L}(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) + \lambda \sum_{v=1}^{V} w_v |\theta_v|.$$

PLTR: Applications in Credit Scoring

Empirical Applications: Credit Scoring

In Dumitrescu et al. (2022), the Penalised Logistic Tree Regression (PLTR) model has been applied to multiple credit scoring datasets (Kaggle, Australian, Taiwan, Housing).

- Compared with benchmark models: linear and non-linear logistic regression, random forest, support vector machines, and neural networks.
- Performance assessed using standard criteria: AUC, Brier Score, Kolmogorov-Smirnov statistic, Percentage of Correctly Classified cases (PCC), and Partial Gini Index (PGI).
- PLTR consistently improves upon logistic regression and performs competitively with random forest, while preserving interpretability.
- Across datasets, PLTR also leads to significant reductions in misclassification costs, providing both statistical and economic benefits.

PLTR Performances

Methods	AUC	PGI	PCC	KS	BS
Taiwan dataset					
Linear Logistic Regression	0.6310	0.2099	0.7586	0.2506	0.2344
Non-Linear Logistic Regression	0.5963	0.0984	0.7035	0.1927	0.2965
Non-Linear Logistic Regression + ALasso	0.7596	0.5029	0.7871	0.3926	0.1447
Random Forest	0.7722	0.4924	0.8102	0.4177	0.1362
PLTR	0.7780	0.5156	0.7959	0.4257	0.1352
Support Vector Machine	0.7102	0.3207	0.8195	0.3382	0.1461
Neural Network	0.7304	0.4226	0.7879	0.3885	0.1401
Housing dataset					
Linear Logistic Regression	0.7904	0.5508	0.8103	0.4450	0.1228
Non-Linear Logistic Regression	0.7965	0.5425	0.8239	0.4650	0.1199
Non-Linear Logistic Regression + ALasso	0.8113	0.5754	0.8217	0.4815	0.1125
Random Forest	0.9387	0.8157	0.9036	0.7455	0.0736
PLTR	0.9011	0.7341	0.8818	0.6694	0.0844
Support Vector Machine	0.7890	0.5514	0.8093	0.4444	0.1254
Neural Network	0.7910	0.5478	0.8132	0.4470	0.1208

Source: Dumitrescu et al. (2022)

PLTR Interactions

Table 3
Decision rules and average marginal effects: full sample Kaggle dataset.

#	Decision rules	Average marginal effects
1	"NumberOfTime60-89DaysPastDueNotWorse <0.5"	-0.0392
2	"NumberOfTimes90DaysLate<0.5" & "RevolvingUtilizationOfUnsecuredLines<0.59907"	-0.0389
3	"NumberOfTimes90DaysLate<0.5" & "NumberOfTime60-89DaysPastDueNotWorse<0.5"	-0.0342
4	"NumberOfTime60-89DaysPastDueNotWorse<0.5" & "NumberOfTime30-59DaysPastDueNotWorse<0.5"	-0.0326
5	"NumberOfTimes90DaysLate<0.5"	-0.0326
6	"NumberOfTime60-89DaysPastDueNotWorse>=0.5" & "NumberOfTime60-89DaysPastDueNotWorse<1.5"	-0.0300
7	"RevolvingUtilizationOfUnsecuredLines>=0.69814" & "RevolvingUtilizationOfUnsecuredLines<1.001"	-0.0285
8	"RevolvingUtilizationOfUnsecuredLines<0.69814"	-0.0281
9	"NumberOfTimes90DaysLate<0.5" & "NumberOfTime30-59DaysPastDueNotWorse<0.5"	-0.0277
10	"NumberOfTimes90DaysLate<0.5" & "NumberOfTime30-59DaysPastDueNotWorse<0.5"	-0.0231

Note: The table provides the list of the decision rules associated with the 10 largest absolute values of the marginal effects (with respect to the probability of defaulting) derived from the PLTR model estimated using the full sample. See Table A.1 in the online appendix for a precise description of the variables.

Source: Dumitrescu et al. (2022)

GAM: Generalized Additive Models

GAM: Generalized Additive Models

Generalized Additive Models (GAM)

Generalized Additive Models (GAMs) provide an extension to standard linear models by allowing the incorporation of non-linear functions for each variable while preserving the principle of additivity.

Hastie, T. and Tibshirani, R. (1986), Generalized additive models. Statistical Science, Vol.1, No. 3, 297-318.

GAM: Generalized Additive Models

Definition: Generalized Additive Models (GAM)

In a **Generalized Additive Model (GAM)**, the conditional mean $\mu(X) = \mathbb{E}(Y|X)$ of a target variable Y is related to an additive function of the features via a link function $g(.)$:

$$g(\mu(X)) = \alpha + f_1(X_1) + \cdots + f_p(X_p),$$

where $f_1(X_1), \ldots, f_p(X_p)$ correspond to (smooth) non-linear functions of the features.

Example of a GAM for Classification

Example of a GAM for Classification

Consider a binary target variable $Y \in \{0, 1\}$. A bivariate Generalized Additive Model assumes

$$g(\mu(x)) = \alpha + f_1(x) + f_2(x),$$

where $\mu(x) = \mathbb{E}[Y|X = x]$ denotes the conditional mean of Y given $X = x$.

Example with a logit link:

$$\mu(x) = \Pr(Y = 1|X = x) = \frac{1}{1 + \exp\big[- (\alpha + f_1(x_1) + f_2(x_2))\big]},$$

$$g(z) = \log\left(\frac{z}{1 - z}\right).$$

Example with a probit link:

$$\mu(x) = \Pr(Y = 1|X = x) = \Phi\big(\alpha + f_1(x_1) + f_2(x_2)\big),$$

$$g(z) = \Phi^{-1}(z),$$

where Φ is the standard normal cdf.

Example of a GAM for Regression

Example of a GAM for Regression

A Generalized Additive Model assumes

$$g(\mu(x)) = \alpha + f_1(x_1) + f_2(x_2),$$

where $\mu(x) = \mathbb{E}[Y|X = x]$ denotes the conditional mean of Y given $X = x$.

Example with an identity link (Gaussian response):

$$\mu(x) = \mathbb{E}[Y|X = x] = \alpha + f_1(x_1) + f_2(x_2),$$

$$g(z) = z,$$

In this case, GAM extends the linear regression model by allowing non-linear smooth functions $f_j(\cdot)$ for the predictors.

GAM: Generalized Additive Models

Figure 24: Example of GAM for a regression model

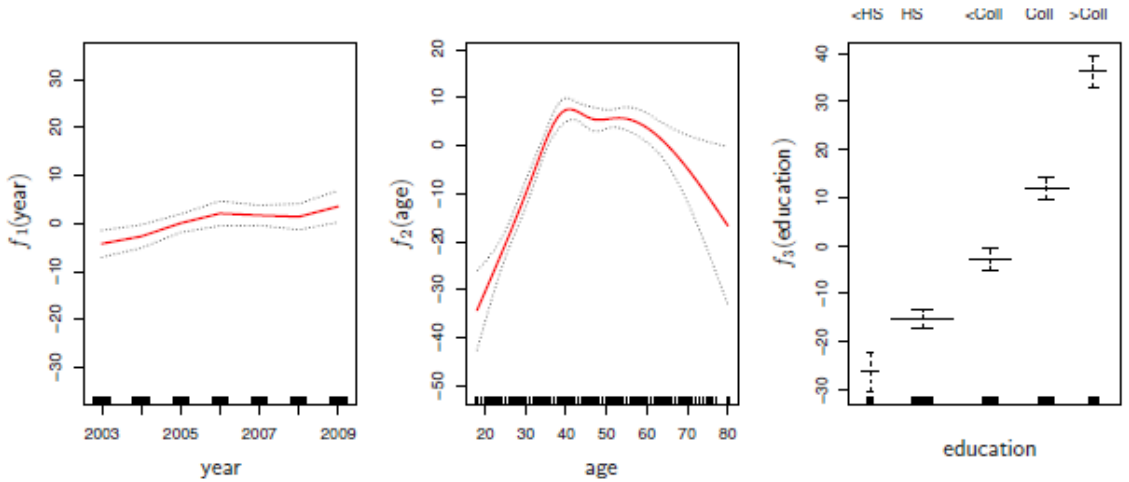


FIGURE 7.11. *For the* Wage *data, plots of the relationship between each feature and the response,* wage*, in the fitted model (7.16). Each plot displays the fitted function and pointwise standard errors. The first two functions are natural splines in* year *and* age*, with four and five degrees of freedom, respectively. The third function is a step function, fit to the qualitative variable* education*.*

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon \qquad (7.16)$$

Source: An introduction to Statistical Learning: With Application in R (2021)

GAM: Generalized Additive Models

Figure 25: Example of GAM for a classification model

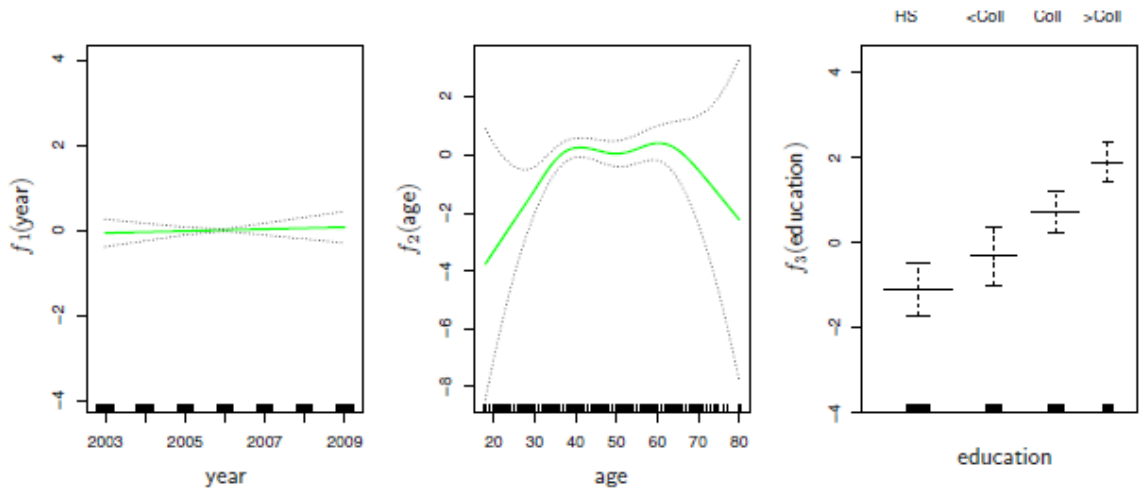


FIGURE 7.14. *The same model is fit as in Figure 7.13, this time excluding the observations for which* `education` *is* `<HS`*. Now we see that increased education tends to be associated with higher salaries.*

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 \times \texttt{year} + f_2(\texttt{age}) + f_3(\texttt{education}), \quad (7.19)$$

where

$$p(X) = \Pr(\texttt{wage} > 250 | \texttt{year}, \texttt{age}, \texttt{education}).$$

Source: An introduction to Statistical Learning: With Application in R (2021)

GAM: Generalized Additive Models

Advantages

1. Automatically model non-linear relationships that standard linear regression will miss.
2. The non-linear fits can potentially make more accurate predictions of the target.
3. Easily examine the marginal effect of each feature on the target.

Limits

1. Interactions can be missed as the model is restricted to be additive. However, as with linear regression, we can manually add interaction terms to the GAM. By doing so, the GAM are subject to overfitting issues.
2. Solution: See the GAM(L)A approach of Flachaire and al. (2024) for an alternative.

Flachaire, E., Hué, S., Laurent, S., Hacheme, G., (2024), Interpretable Machine Learning using partial linear models, *Oxford Bulletin of Economics and Statistics*, 86(3), 519-540.

Other Inherently Interpretable ML Models

Beyond PLTR and GAM, there exist many other inherently interpretable machine learning models that combine predictive performance with transparency.

Takeaway

These approaches are beyond the scope of this lecture, but they illustrate that one should not always believe in a systematic trade-off between performance and interpretability.

Examples:

- **GLM**: Linear and Logistic Regression with regularization
- **FIGS**: Fast Interpretable Greedy-Tree Sums (Tan et al., 2022)
- **EBM**: Explainable Boosting Machine (Nori et al., 2019)
- **XGB1**/**XGB2**: Shallow Gradient Boosted Trees with effect purification
- **GAMI-Net**: GAM with Structured Interactions (Yang et al., 2021)
- **ReLU-DNN**: Sparse Deep ReLU Networks (Sudjianto et al., 2020)
- etc.

Beyond Post-hoc: Inherently Interpretable ML

Key concepts

1. Limits of Post-hoc Interpretability Methods
2. Trade-off between accuracy and interpretability
3. Rashomon set
4. Inherently interpretable models
5. Penalised Logistic Tree Regression (PLTR)
6. Generalized Additive Models (GAM)

End of Session

Christophe Hurlin (University of Orléans and IUF)