



DMIF, University of Udine

---

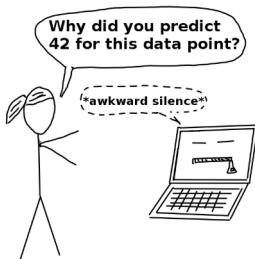
# Interpretability in Machine Learning

Andrea Brunello

[andrea.brunello@uniud.it](mailto:andrea.brunello@uniud.it)

June 10, 2022

- Interpretability is one of the most **interesting** topics in the machine learning realm, but it is also very **complex**
- This comes from the fact that interpretability is a **not a well defined** nor understood concept
- “We know when we see it”



*Interpretability is the degree to which a human can understand the cause of a decision*

*(Miller 2019)*

*Interpretability is the degree to which a human can consistently predict the model's result*

*(Kim, Khanna, and Koyejo 2016)*

Two kinds of interpretability:

- **Local interpretability:** it refers to a single instance. We look for an *explanation*, that relates the feature values of the instance to its model prediction in a way that can be understood by a human
- **Global interpretability:** get an overall insight about how the model behaves; which patterns, regularities are present in the data and how they are exploited by the model



# Importance of interpretability

- Why do we not just trust the model and ignore why it made a certain decision?
- After all, if it works *very well*, that should be enough. . .
- There are **several reasons** supporting interpretability:
  - Human curiosity and learning
  - Guaranteeing a good behaviour
  - Bug discovery and correction
  - Detecting biases
  - Ensuring social acceptance

# Human curiosity and learning

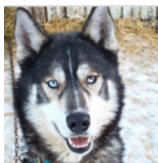
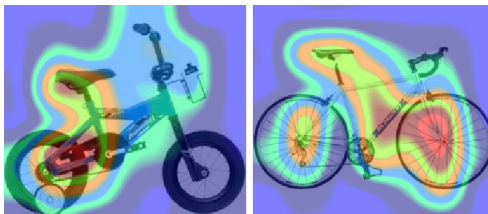
```

conversation time <= 7
|   conversation time <= 0
|   |   dialling time <= 30
|   |   |   dialling time <= 11: busy_or_nonexistent
|   |   |   dialling time > 11
|   |   |   |   dialling time <= 14: busy_or_nonexistent
|   |   |   |   dialling time > 14: no_answer
|   |   dialling time > 30: no_answer
|   conversation time > 0
|   |   postcall time <= 1
|   |   |   dialling time <= 29: fax_or_answermachine
|   |   |   dialling time > 29
|   |   |   |   conversation time <= 1: no_answer
|   |   |   |   conversation time > 1: fax_or_answermachine
|   |   postcall time > 1
|   |   |   conversation time <= 4: fax_or_answermachine
|   |   |   conversation time > 4: spoken_no_survey
conversation time > 7
|   conversation time <= 76
|   |   conversation time <= 11
|   |   |   postcall time <= 1
|   |   |   |   conversation time <= 9
|   |   |   |   |   dialling time <= 22
|   |   |   |   |   |   conversation time <= 8: fax_or_answermachine
|   |   |   |   |   |   conversation time > 8: spoken_no_survey
|   |   |   |   |   dialling time > 22: fax_or_answermachine
|   |   |   |   |   conversation time > 9: spoken_no_survey
|   |   |   |   postcall time > 1: spoken_no_survey
|   |   conversation time > 11: spoken_no_survey
|   conversation time > 76
|   |   conversation time <= 87
|   |   |   postcall time <= 0: spoken_no_survey
|   |   |   postcall time > 0: survey_made
|   |   conversation time > 87: survey_made

```

# Guaranteeing a good behaviour

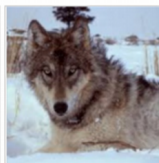
E.g., resistance to adversarial examples, or ensuring that only causal relationships are picked up



Predicted: **wolf**  
True: **husky**



Predicted: **husky**  
True: **husky**



Predicted: **wolf**  
True: **wolf**



$$\begin{aligned} houseSellingPrice = & (-26,6882 * houseSize) + \\ & (7,0551 * lotSize) + \\ & (43166,0767 * bedrooms) + \\ & (42292,0901 * bathroom) \\ & -21661,1208 \end{aligned}$$

The weight assigned by the linear regression model to the feature *houseSize* is rather strange, and may indicate the presence of a multicollinearity phenomenon.



# Detecting biases

- Training datasets are **biased by our vision of the world**
- For instance, let us say that we want to build a model that tries to predict if a person is going to commit a felony
- We assemble a training dataset where each instance is a person described by age, gender, neighbourhood, study title, ..., and felon\_or\_not is the label
- We train a logistic regression model; inspecting it we notice that some social groups are systematically discriminated
- This might be due to an unbalanced training set
- E.g., people living in specific neighbourhoods might be assigned a higher probability of being a felon, simply because those neighbourhoods are highly patrolled by the police, thus more crimes are discovered in them



- John Doe has requested a loan from the bank
- The bank refuses it, based on the prediction of a machine learning model
- John Doe asks for the reason(s) why
- As a result, the bank provides the *explanation* related to the specific prediction: his income level is too low

Providing an explanation may be required for legal reasons:

- European Union's GDPR: "right to explanation"
- *When an individual is subject to 'a decision based solely on automated processing' that 'produces legal effects ... or similarly significantly affects him or her', the GDPR creates rights to 'meaningful information about the logic involved'*
- [academic.oup.com/idpl/article/7/4/233/4762325](https://academic.oup.com/idpl/article/7/4/233/4762325)



# Do we always need interpretability?

Sometimes, you do not need interpretability:

- the model has **no significant impact**
  - e.g., Mike is working on a machine learning side project to predict where his friends will go for their holidays
- the **problem is well studied**
  - there is enough practical experience with the model, so all problems related with it have been solved over time
  - optical character recognition is a good example: it is clear how OCR models work, and we are not really interested in gaining additional insights about the task at hand
- interpretability might enable people or programs to **manipulate** the system
  - e.g., loan application



# How can we achieve interpretability?

## Intrinsic interpretability

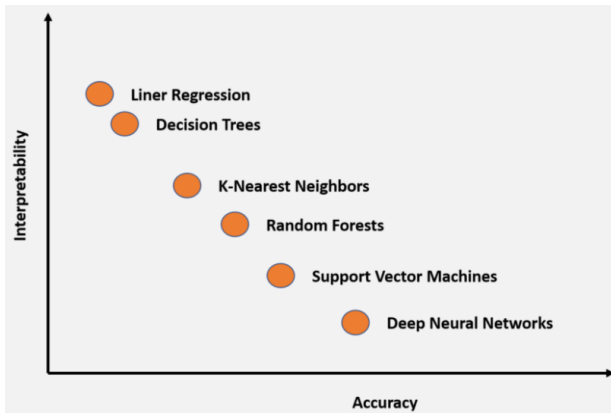
**Strategy 1:** rely on models that are inherently interpretable (intrinsic interpretability)

- linear models: if the pre-conditions for their application are satisfied; also, always remember *ceteris paribus*
- decision trees: height is an important parameter here
- naive bayes: if you do not have a high number of features, otherwise reason component-wise
- rules: lists, sets
- k-NN: it really depends on your training instances

# How can we achieve interpretability?

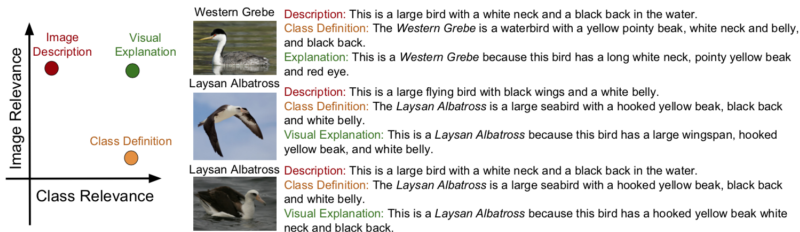
## Interpretability vs Accuracy

As for the Strategy 1, it is typically a matter of **trading-off** accuracy for interpretability



**Strategy 2:** modify the models and their training algorithms to include interpretability characteristics

- (Hendricks et al. 2016)
- deep learning approach
- Learn visual explanations that are both *class discriminative* and that *accurately describe* a specific image instance





# How can we achieve interpretability?

## Post hoc approaches


**Strategy 3:** rely on a non-interpretable model, but apply methods that analyze the model after training (post hoc)

- such methods can treat the model as a black box, analyzing only the input-output relationships (model agnostic methods)
- or, they can consider the inner structure of the model (model specific methods, e.g., saliency maps)
- we will focus here on the first class, model agnostic methods, given their high generality and portability

# Permutation Feature Importance

- Simple idea: to determine the importance of a feature, we consider the increase in the (test set) prediction error of the model that we obtain if we permute the feature values
- A feature is “important” if shuffling its values increases the model error, because that means the model relied on the feature for the prediction
- Originally proposed by (Breiman 2001), it can be considered as a global interpretability approach

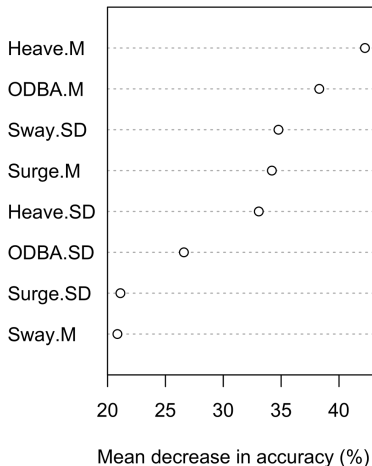
Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24





# Permutation Feature Importance

Available on scikit-learn: [https://scikit-learn.org/stable/modules/permutation\\_importance.html](https://scikit-learn.org/stable/modules/permutation_importance.html)



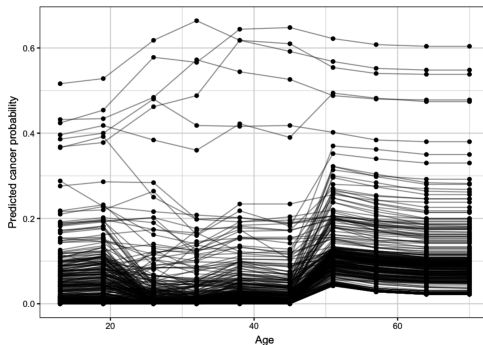




# ICE plots (Goldstein et al. 2015)

- For a chosen feature, Individual Conditional Expectation (ICE) plots display one line per instance, representing how the instance prediction changes when the feature changes
- The values for a line can be computed by leaving all other features the same (*ceteris paribus*), creating variants of the instance by replacing the considered feature's value with values from a grid and letting the black box model make the predictions with these newly derived instances

# ICE plots (Goldstein et al. 2015)



Most women with a low cancer probability in younger years see an increase in predicted cancer probability (*ceteris paribus*), except for a few of them that already have a high estimated cancer probability



# ICE plots (Goldstein et al. 2015)

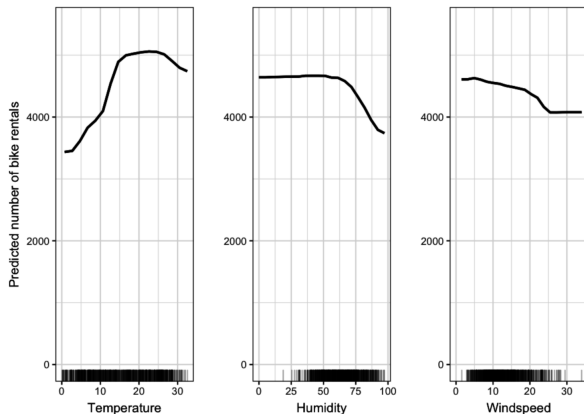
- ICE curves can only display one feature meaningfully
- If the feature of interest is correlated with other features, then some points in the lines might be invalid according to the joint feature distribution
- ICE curves can uncover heterogeneous relationships between the considered feature and the model prediction
- Observe that ICE plots can easily become overcrowded
- In Python, ICE plots are built into scikit-learn starting with version 0.24.0



# Partial Dependence Plots (Friedman 2001)

- Partial Dependence Plots (PDPs) show the marginal effect of a feature on the predicted outcome of a model
- For instance, we may be able to determine whether the relationship between the target and the feature is linear, monotonic or more complex
- Values are computed similarly to ICE, but then the curves are averaged to obtain a single curve for each feature
- Heterogeneous effects might be hidden, since the PDP plot only shows the average over the observations
- PDP plots can be considered as the “global interpretability version” of the ICE plots
- In Python, also partial dependence plots are built into scikit-learn starting with version 0.24.0

# Partial Dependence Plots (Friedman 2001)



The bars at the bottom show the distribution of the data, which is useful to give the correct meaning to the graphs



# Shapley values (Shapley 1953)

- Assume that we are in a setting where some players play a game, cooperating in a coalition, and they obtain a certain gain from that cooperation. The Shapley value is a method for fairly distributing payouts to the players depending on their contribution towards the total payout
- The *game* is the prediction task for a **single instance** of the dataset. The *gain* is the actual prediction for the given instance minus the average prediction of all instances. The *players* are the feature values of the instance, which collaborate to obtain such a gain
- Shapley values are thus focused on local interpretability



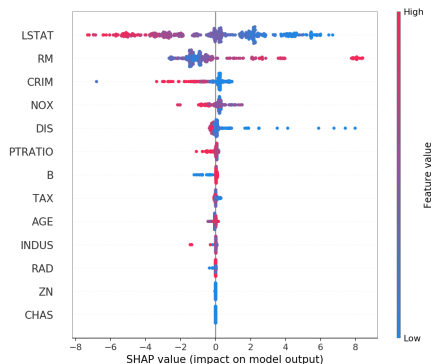
# Shapley values (Shapley 1953)

Consider the following scenario:

- You have trained a machine learning model to predict apartment prices
- For a certain apartment the model predicts € 300.000 and you need to explain this prediction
- The apartment has an area of  $50 \text{ m}^2$ , is located on the 2nd floor, has a park nearby and cats are banned
- The average prediction for all apartments is € 310.000
- How much has each feature value contributed to the prediction compared to the average prediction?
- Our goal is to explain the difference between the actual prediction and the average prediction, i.e., € -10.000
- The answer could be: the park-nearby contributed € 30.000; area of  $50 \text{ m}^2$  contributed € 10.000; floor-2nd contributed € 0; cats-banned contributed € -50.000

SHAP, Python package that calculates Shap values:

<https://github.com/slundberg/shap>



Note how, putting together several *local* explanations, we can derive a *global* interpretability





- Global interpretability approach
- An interpretable model is trained to approximate the predictions of a black box model
- General idea:
  - 1 given a dataset  $X$ , get the predictions  $\hat{y}$  of the black box model
  - 2 train the interpretable model on the dataset  $X$  and predictions  $\hat{y}$
  - 3 measure how well the interpretable model replicates the predictions of the black box model



- The surrogate model method is flexible: any interpretable model can be used as the surrogate
- Be aware that you are drawing conclusions about the behaviour of the black box model and not about the data, since the surrogate model never sees the real labels
- How much accurate should the surrogate model be?



# Global surrogates – Caveats (Rudin 2019)

*"Explanations must be wrong. They cannot have perfect fidelity with respect to the original model. If the explanation was completely faithful to what the original model computes, the explanation would equal the original model, and one would not need the original model in the first place, only the explanation."*

*"An explainable model that has a 90% agreement with the original model indeed explains the original model most of the time. However, an explanation model that is correct 90% of the time is wrong 10% of the time."*

*"Even an explanation model that performs almost identically to a black box model might use completely different features, and is thus not faithful to the computation of the black box."*

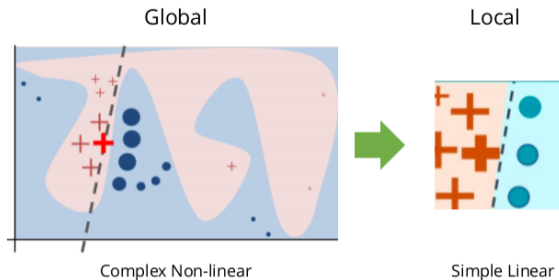


- Local Interpretable Model-agnostic Explanations (LIME)
- Method for fitting local, interpretable models that can explain single predictions made by any black box machine learning model (Python's library *lime*)
- It makes use of **local surrogate models**, i.e., interpretable models that are learned starting from single predictions of the original black box model
- The focus is on explaining why single predictions were made by the original black box model



General idea:

- 1 Choose an instance of interest for which you want to get an explanation of its black box prediction
- 2 Perturb the instance generating a dataset, and get the black box predictions for these new points
- 3 Assign weights to the new samples according to their proximity to the instance of interest
- 4 Fit a weighted, interpretable model on the perturbed dataset
- 5 Explain the prediction by interpreting the local model



The complex, black box model's decision function is represented in blue/pink colors.

The bold red cross is the instance being explained, and the other crosses and circles are the local perturbations (weighted).

The dashed line is the learned explanation, locally (but not globally) faithful.



- The explanations created with local surrogate models can use other (interpretable) features than the original model was trained on
- Of course, these interpretable features must be derived from the data instances
- For example, a deep learning model can rely on abstract word embeddings as features, but the explanation can be based on the presence or absence of words in a sentence (BoW approach)



# LIME (Ribeiro, Singh, and Guestrin 2016)

- The correct definition of the neighborhood of the explained instance is a very big, unsolved problem when using LIME
- Which kinds of perturbations should be performed?
  - Of course, tabular/image/text data require different transformations
- What about their extent?
- Another big problem is the instability of the explanations, which may vary a lot even for very close points
- Finally, observe that LIME's results do not indicate how faithful they are, as LIME solely learns a decision boundary that best approximates the model (locally, but how much locally?) given the perturbation space





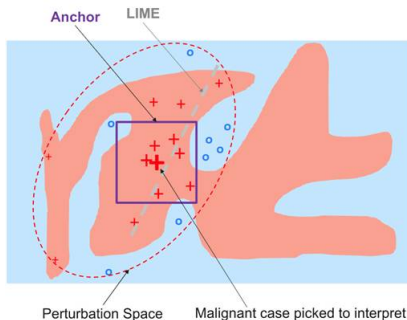
# Anchors (Ribeiro, Singh, and Guestrin 2018)

- Anchors answer the question: “Which features are sufficient to anchor a specific prediction, i.e., changing the other features does not change the prediction of the black box model?”
- In other words, anchors represent local, “sufficient” conditions for a prediction
- Anchors are expressed as **scoped** IF-THEN rules, i.e., such rules precisely specify to which other, possibly unseen, instances they apply, and to what extent (in terms of *precision* and *coverage*)
- To build and estimate the precision and coverage of the anchors, also here the perturbation space of a given instance is considered (Python’s package *anchor*)



Let us consider the following scenario:

- we want to interpret a classifier that, based on a person's age, loan amount and credit history (good/bad), determines if the individual will be a defaulter or not
- a first anchor may refer to just a single attribute, such as *IF age > 55 THEN defaulter*, with 80% coverage and 60% precision
- another anchor may consider two features, for example *IF age > 55 AND loan amount > 50.000 THEN defaulter*, with 50% coverage and 85% precision
- finally, a third anchor may be something like *IF age > 55 AND loan amount > 50.000 AND credit history = bad THEN defaulter*, with 20% coverage and 100% precision



LIME works bad for the considered instance; instead, the anchor represented by the box (e.g., an IF with 4 conditions, 2 on the  $x$  axis and 2 on the  $y$  axis) has a very high precision, and a recall around 50% considering the perturbation space

# Counterfactual explanations

- A counterfactual explanation of a prediction describes the **smallest change** to the feature values that changes the prediction to a desired output
- For instance, considering our John Doe that was refused a loan application by his bank, he could reformulate his “Why?” question as follows: *What is the smallest change to the features (income, number of credit cards, age, ...) that would change the prediction from rejected to approved?*
- Counterfactuals are very attractive, because they represent highly human-friendly explanations
- Also, they are selective, meaning that they usually focus on a small number of feature changes

- On the downside, counterfactuals suffer from the **Rashomon effect**
- Rashomon is a Japanese movie in which the murder of a Samurai is told by different people
- Each of the stories explains the outcome equally well, but the stories contradict each other
- The same can also happen with counterfactuals, since there are usually multiple different counterfactual explanations, that might contradict each other as for the changes to perform to the attributes (number and extent) in order to get the desired output from the model

What defines a good counterfactual?

- it produces the desired output **as closely as possible** (e.g., in terms of probability)
- it should be as similar as possible to the original instance in terms of feature values (distance in the feature space)
- it should change as few features as possible
- it should have feature values that are likely (e.g., not describe an apartment of  $50m^2$  with 10 rooms)



## How to generate counterfactuals?

- It is often desirable to generate multiple diverse counterfactual explanations
- A possible approach is that of employing multi-objective evolutionary algorithms
- Python-based framework *DiCE*:  
<https://github.com/interpretml/DiCE>

We want to find a counterfactual explanation for this instance, representing a customer that has been predicted to have a good credit risk with a probability of 24.2% (thus, she has been denied the credit by the bank)

age	sex	job	housing	savings	amount	duration	purpose
58	f	unskilled	free	little	6143	48	car



The counterfactuals should answer how the input features need to be changed to get a predicted probability larger than 50%

age	sex	job	amount	duration	$o_2$	$o_3$	$o_4$	$\hat{f}(x')$
		skilled		-20	0.108	2	0.036	0.501
		skilled		-24	0.114	2	0.029	0.525
		skilled		-22	0.111	2	0.033	0.513
-6		skilled		-24	0.126	3	0.018	0.505
-3		skilled		-24	0.120	3	0.024	0.515
-1		skilled		-24	0.116	3	0.027	0.522
-3	m			-24	0.195	3	0.012	0.501
-6	m			-25	0.202	3	0.011	0.501
-30	m	skilled		-24	0.285	4	0.005	0.590
-4	m		-1254	-24	0.204	4	0.002	0.506



# Last words: Multiple models, same accuracy

- Assume that we have at our disposal the perfect method to derive the interpretation of a model
- Still, given a problem, there might be multiple models that achieve a very similar error rate
- Such models may use the input features in different ways
- For instance, in the prediction of a house value, a model may consider the surface of the house, while another one the number of rooms
- Thus, we may still in principle obtain very different interpretations for the same predictions



# References I

- Leo Breiman (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.
- Jerome H Friedman (2001). "Greedy function approximation: a gradient boosting machine". In: *Annals of Statistics*, pp. 1189–1232.
- Alex Goldstein et al. (2015). "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation". In: *Journal of Computational and Graphical Statistics* 24.1, pp. 44–65.
- Lisa Anne Hendricks et al. (2016). "Generating Visual Explanations". In: *Proceedings of the 14th European Conference on Computer Vision*, pp. 3–19.
- Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo (2016). "Examples are not enough, learn to criticize! criticism for interpretability". In: *Advances in neural information processing systems* 29.



## References II

- Tim Miller (2019). “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial intelligence* 267, pp. 1–38.
- Christoph Molnar (2022). *Interpretable machine learning*.  
<https://christophm.github.io/interpretable-ml-book/>.
- Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin (2016). ““Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144.
- (2018). “Anchors: High-Precision Model-Agnostic Explanations”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 1527–1535.



## References III

- Cynthia Rudin (2019). “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5, p. 206.
- Lloyd S Shapley (1953). “A value for n-person games”. In: *Contributions to the Theory of Games* 2.28, pp. 307–317.