Interpretability

CSE547

May 23 2024

Margaret Li

What is Interpretability?

Definition

Interpretability is:

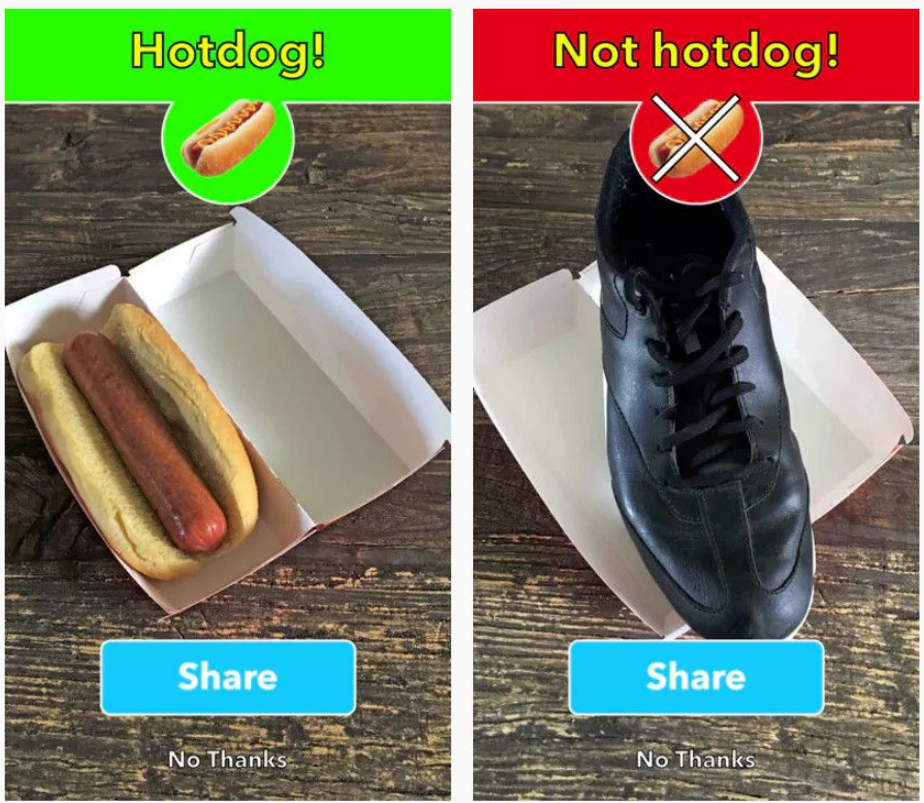
> the degree to which a human can *understand* the cause of a decision
>
> (Miller, Tim. “Explanation in artificial intelligence: Insights from the social sciences.” arXiv Preprint arXiv:1706.07269. (2017).)
>
> the degree to which a human can consistently *predict* the model’s result
>
> (Kim, Been, Rajiv Khanna, and Oluwasanmi O. Koyejo. “Examples are not enough, learn to criticize! Criticism for interpretability.” Advances in Neural Information Processing Systems (2016).)

The higher the interpretability of a machine learning model, the easier it is for someone to comprehend why certain decisions or predictions have been made.

Why (do we care about) Interpretability?

Sometimes we don't (as much)

Importance



Importance

ML models are increasingly given power to influence life-changing decisions:

- Rent prices
- Medical options
- Job offers
- Legal document processing
- ...
- Netflix recommendations

Importance

Opinion:

It's *not enough* to hand-wave and say that interpretability is important

We are responsible for weighing the *impact* of our models

From there, we can decide whether we should pay the *cost* of various interpretability methods and whether we can tolerate their *limitations*

What kinds of interpretability methods exist?

Grouped by **model assumption**

- **Model specific**

  Works for **certain types** of models only

  Makes assumptions about the inner workings of the model

- **Model agnostic**

  Works for ***any* model**

  Usually operates only on the inputs and outputs, treats the model as a black box

Grouped by **globality**

- **Global methods**

  Describe the model behavior ***averaged*** across examples

- **Local methods**

  Describe the model behavior on ***individual* examples**

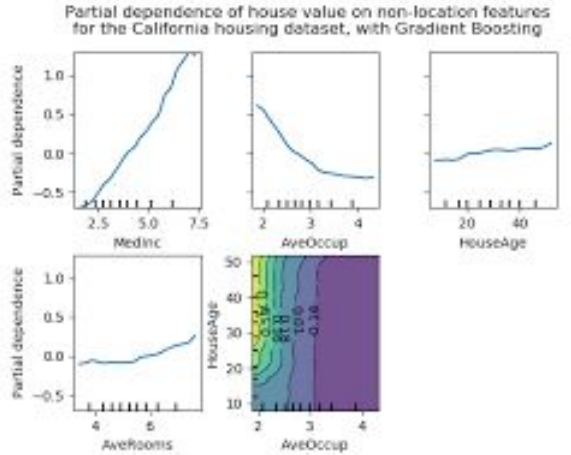
Grouped by **return**

- **Feature summary**
  - **Statistic**

    A single number per feature, e.g., feature importance:
    "color is 80% responsible for this prediction"
    Or a more complex result, e.g, pairwise feature interaction

  - **Visualization**

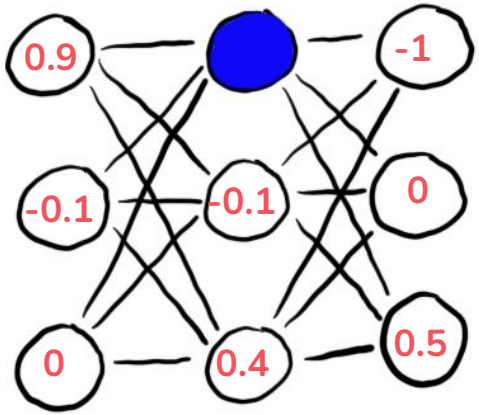
    Some feature summaries are only meaningful if visualized.
    E.g., partial dependence plots show a feature and the average predicted outcome. The best way to present partial dependencies is to draw the curve instead of printing the coordinates.



Grouped by **return**

- **Model internals (e.g. learned weights)**

  E.g., weights in linear models or the learned tree structure (the features and thresholds used for the splits) of decision trees, or visualization of feature detectors learned in convolutional neural networks.
  By definition model-specific.



Grouped by **return**

- **Data point**

  E.g., counterfactual explanations. Changing some of the features in an input, looking for changes in prediction, like a flip in the predicted class.

  To be useful, this requires that the data points themselves can be interpreted. This works well for images and texts, but is less useful for tabular data with hundreds of features.

118 %

Results | Messages

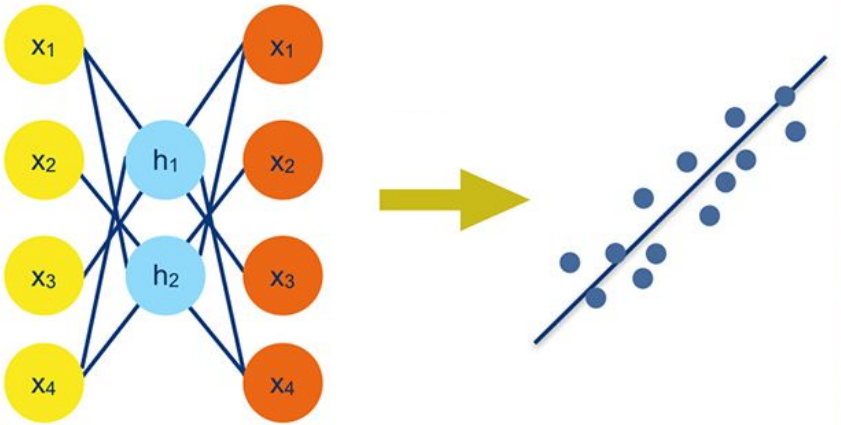
	allocation_unit_id	type	type_desc	container_id	data_space_id	total_pages	used_pages	data_pages
82	844424932884480	1	IN_ROW_DATA	844424932884480	1	0	0	0
83	844424933408768	1	IN_ROW_DATA	844424933408768	1	2	2	1
84	844424935768064	1	IN_ROW_DATA	844424935768064	1	0	0	0
85	1125899909070...	1	IN_ROW_DATA	1125899909070...	1	17	10	8
86	7177611906514...	2	LOB_DATA	281474980642816	1	12	6	0
87	7205759403799...	1	IN_ROW_DATA	281474980511744	1	2	2	1
88	7205759403805...	1	IN_ROW_DATA	562949957222400	1	2	2	1
89	7205759403819...	1	IN_ROW_DATA	281474983067648	1	2	2	1
90	7205759403825...	1	IN_ROW_DATA	562949959778304	1	2	2	1

vs

Grouped by **return**

- **Intrinsically interpretable model**

  Approximate black box models (either globally or locally) with an interpretable model,
  which is interpreted by looking at internal model parameters or feature summary statistics.



Today

- **Model-specific methods**
  Choose a model designed with some inherently interpretable properties
- **Model-agnostic methods**
  Interpret models which have been trained, regardless of model design
  - Global methods
  - Local methods
- **Neural net-specific study**

Today

- Model-specific methods
  Choose a model designed with some inherently interpretable properties
- Model-agnostic methods
  Interpret models which have been trained, regardless of model design
  - Global methods
  - Local methods
- Neural net-specific study

Brief Discussion

Focus today

Today

- **Model-specific methods**
  Choose a model designed with some inherently interpretable properties
- Model-agnostic methods
  Interpret models which have been trained, regardless of model design
  - Global methods
  - Local methods
- Neural net-specific study

Model-Specific methods

If your goal is interpretability, it makes sense to design the model for this.

However:

This limits the models and methods you can use for your problem.

Some problems may not be feasible to learn with an interpretable model.

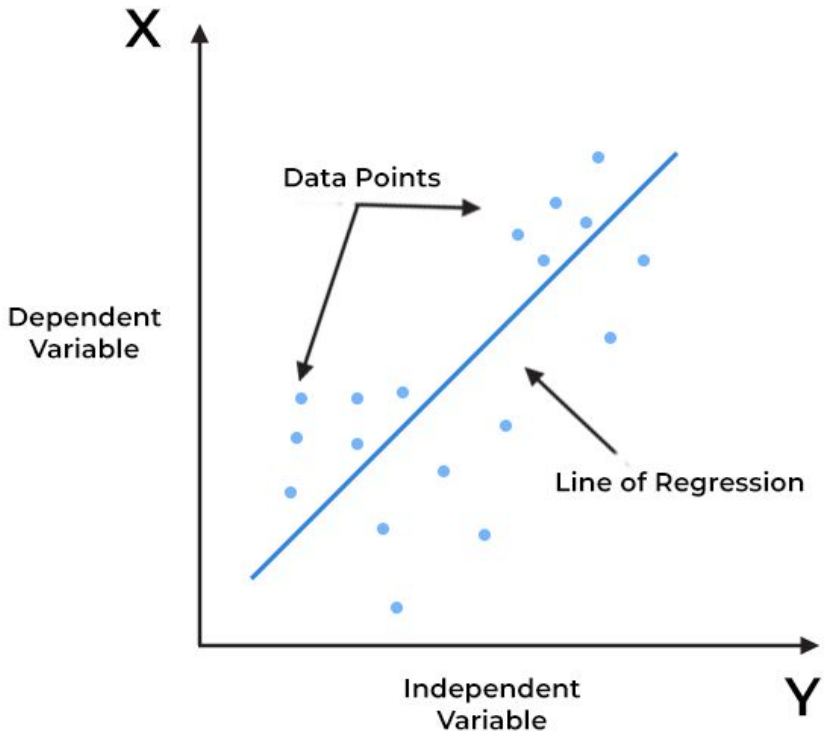
Interpretable Models

- **Regressions (Linear, Logistic)**
  Fit a simple function to the features
- **Decision Tree**
  Fit a set of single-feature rules (i.e. nested if-then statements)
- **Decision Rule**
  Fit a single rule with more complex conditions (i.e. a switch/case statement)
- **RuleFit**
  Fit a simple function on complex interactions between features
- **& More**

Linear Regression

Given p features, learn p+1 coefficients (including an intercept) to fit a line:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$$



Interpretation

- Of a **numerical** feature
  x_k := x_k **+1** -> y + β_k
- Of a **binary** feature
  x_k := (**1 -** x_k) -> y + β_k
- Of a **categorical** feature
  x_k := **a_{k, i}** -> y + β_k

Interpretation

- Of the **intercept β_0**
  For an instance with all numerical feature values at zero and the categorical feature values at the reference categories, the model prediction is the intercept weight.
  Usually not relevant because instances with all features values at zero often make no sense.
  However, if the features have been standardised (mean of 0, standard deviation of 1), then the intercept reflects the predicted outcome of an instance where all features are at their mean value.

Interpretation

Feature Importance

The importance of a feature in a linear regression model can be measured by the absolute value of its t-statistic. The t-statistic is the estimated weight scaled with its standard error.

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)}$$

The importance of a feature increases with increasing weight.
The more variance the estimated weight has (= the less certain we are about the correct value), the less important the feature is.

Interpretation

Model "goodness" (R-squared measurement)

R-squared tells you how much of the total variance of your target outcome is explained by the model:

$$R^2 = 1 - SSE/SST$$

SSE is the squared sum of the error term and tells you how much variance remains after fitting the linear model.
SST is the total variance of the target outcome.

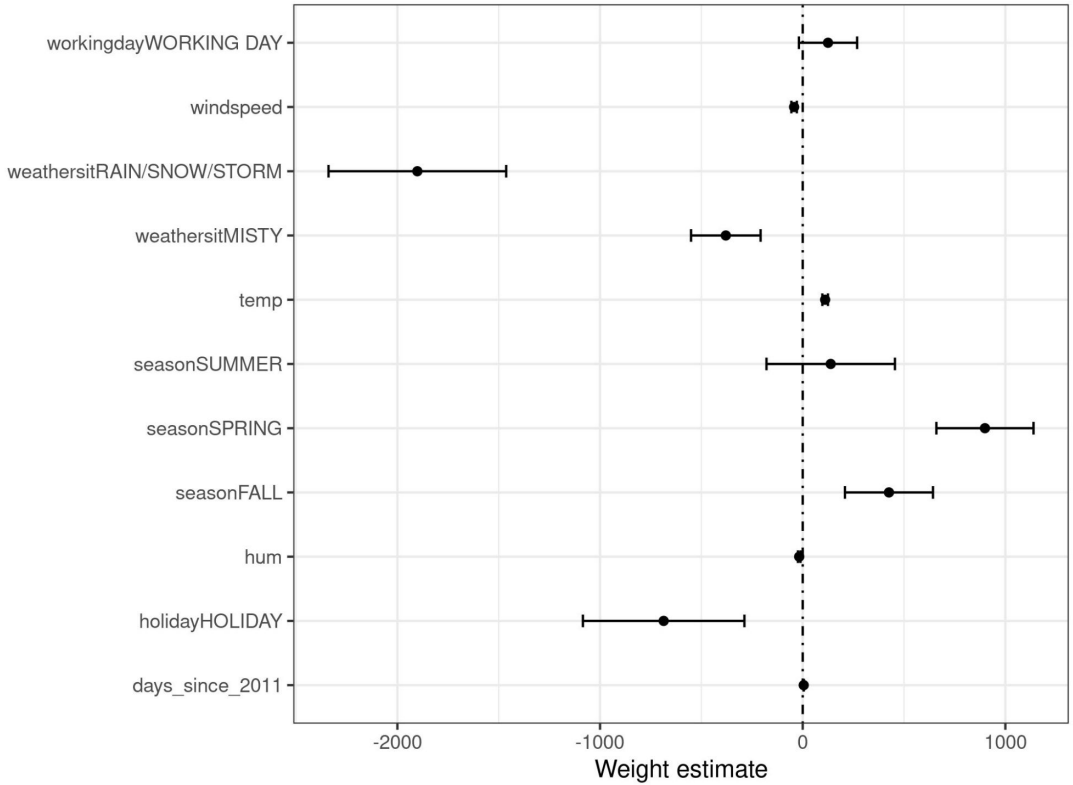
R-squared tells you how much of your variance can be explained by the linear model. R-squared usually ranges between 0 for models where the model does not explain the data at all and 1 for models that explain all of the variance in your data.

Interpretation

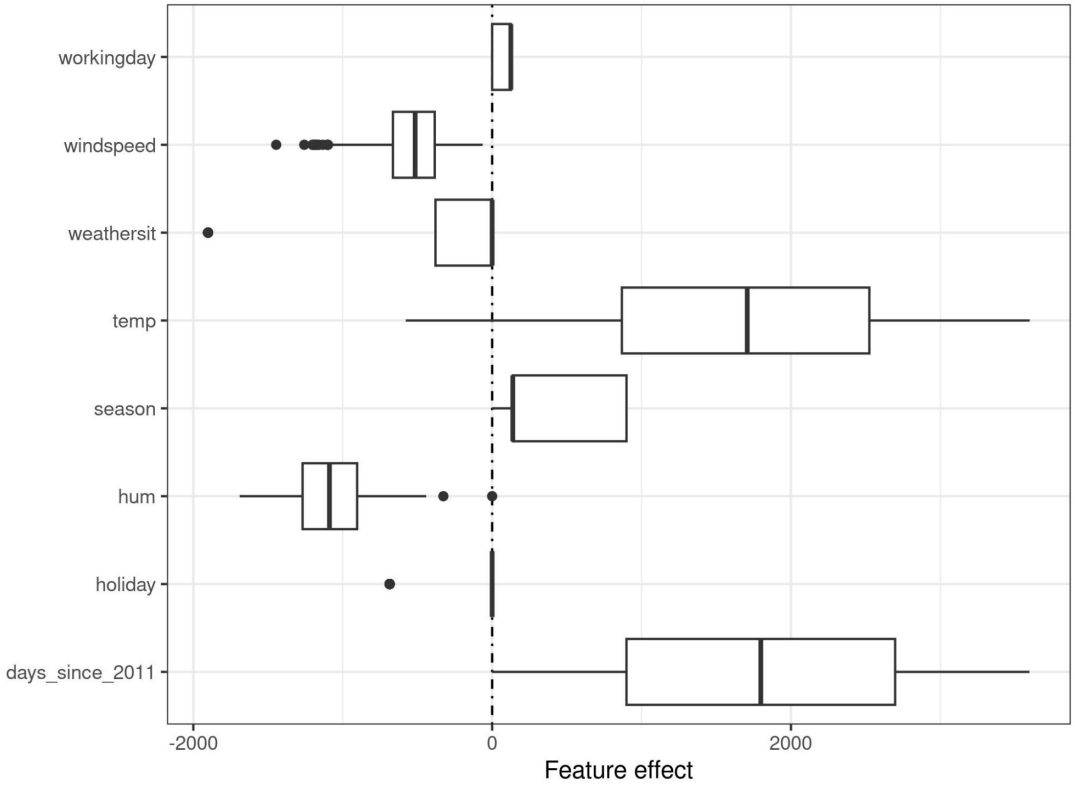
Would you want to interpret a model with very low (adjusted) R-squared?

No, because such a model basically does not explain much of the variance. Any interpretation of the weights would not be meaningful.

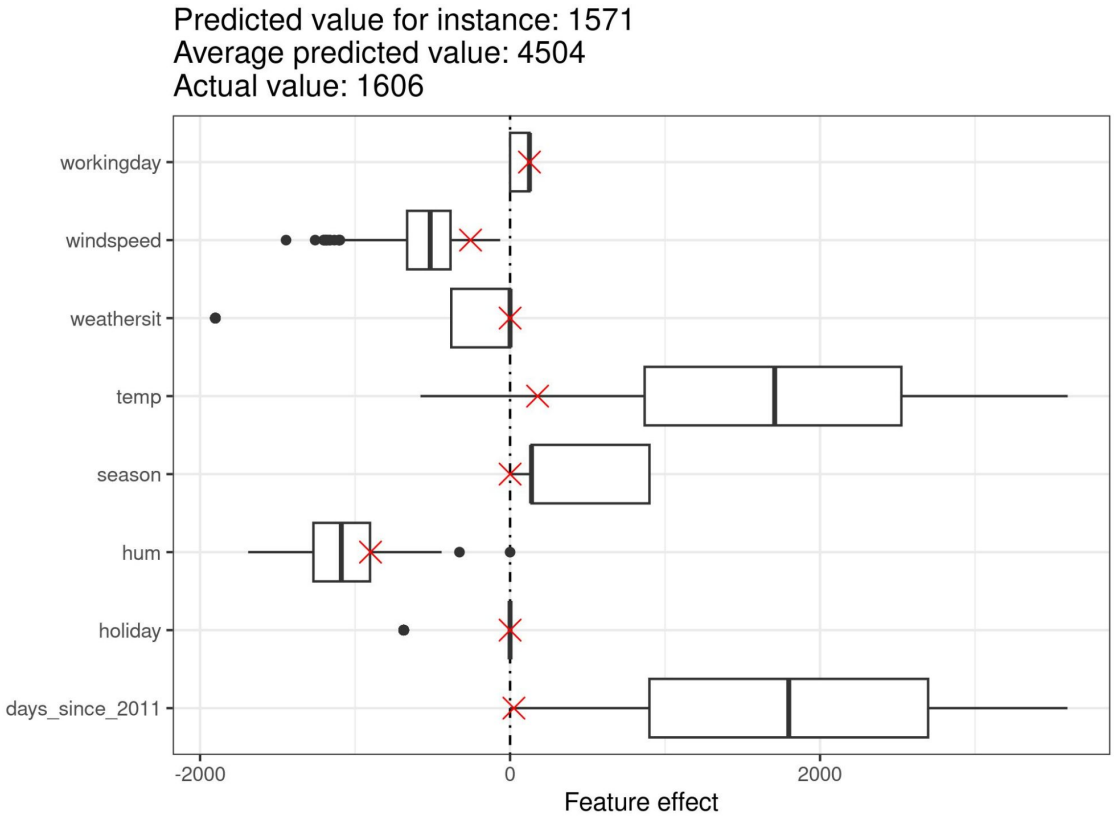
Interpretation - Weight Plot



Interpretation - Effect Plot



Interpretation - Effect Plot for 1 instance



Advantages

Transparency

The weighted sum makes it transparent how predictions are produced.

Widely used & Extended

In many places, linear regressions are accepted for predictive modeling and doing inference. There is a high level of collective experience and expertise, including teaching materials on linear regression models and software implementations. Linear regression can be found in R, Python, Java, Julia, Scala, Javascript, ...

There are also many extensions of the linear regression model (GLM, GLAM, etc)

Theory

Together with the weights you get confidence intervals, tests, and solid statistical theory.

Disadvantages

Linearity

Linear regression models can only represent linear relationships, i.e. a weighted sum of the input features. Each nonlinearity or interaction has to be hand-crafted and explicitly given to the model as an input feature.

Low predictive power

Possibly low predictive performance, because the relationships that can be learned are so restricted and usually oversimplify.

Correlated features decrease interpretability

The interpretation of a weight can be unintuitive because it depends on all other features.
A feature with high positive correlation with the outcome y and another feature might get a negative weight in the linear model, because, given the other correlated feature, it is negatively correlated with y in the high-dimensional space.

Today

- Model-specific methods
  Choose a model designed with some inherently interpretable properties
- Model-agnostic methods
  Interpret models which have been trained, regardless of model design
  - Global methods
  - Local methods
- Neural net-specific study

Today

- Model-specific methods
  Choose a model designed with some inherently interpretable properties
- **Model-agnostic methods**
  Interpret models which have been trained, regardless of model design
  - **Global methods**
  - **Local methods**
- Neural net-specific study

Today

What does the model do *on average?*

- **Global methods**

What does the model do in *specific instances?*

- **Local methods**

Global methods

What happens to input predictions on average as you vary a feature?

- **Partial Dependence Plot (PDP)**
- **Accumulated Local Effects (ALE)**

How do the features interact with each other to decide predictions?

- **Feature Interaction**
- **Functional Decomposition**

What is the importance of each feature to inputs' predictions?

- **Permutation Feature Importance**

Can you represent the actual model with a simpler model?

- **Global Surrogate**
- **Prototypes & Criticisms**

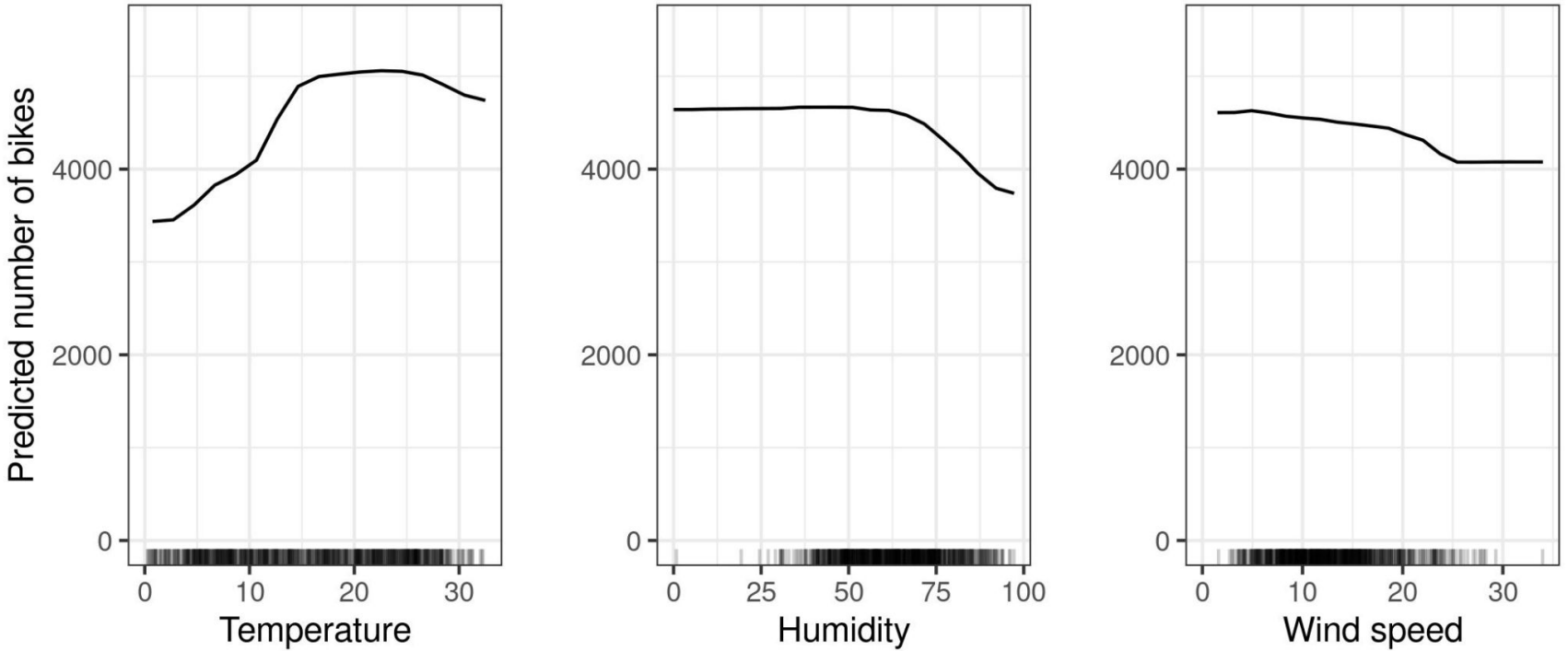
Partial Dependence Plots (PDPs)

PDPs show the marginal effect one or two features have on the predicted outcome of a machine learning model

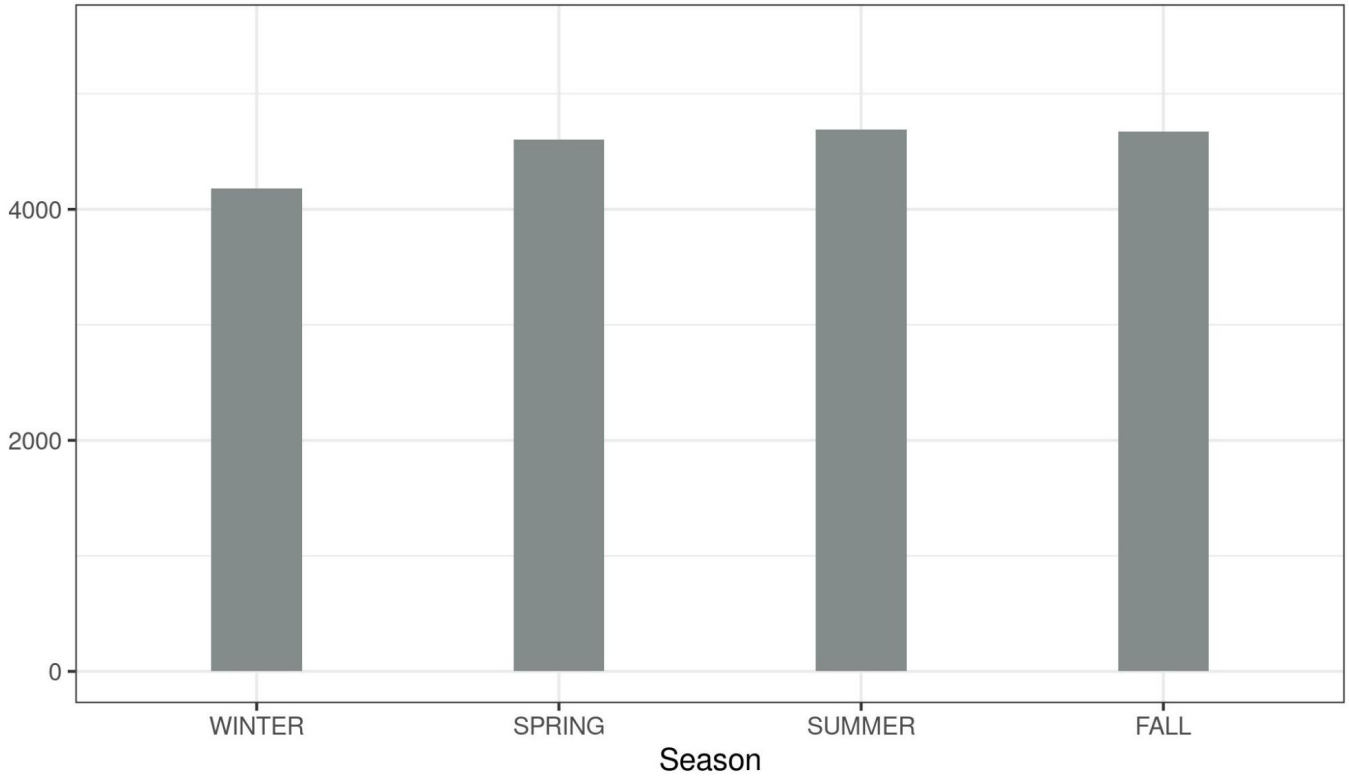
PDPs can show whether the relationship between the target and a feature is linear, monotonic or more complex

Partial dependence function = for each feature value, what would be the average prediction if we force all data points to assume that value?

Partial Dependence Plots



Partial Dependence Plots



Partial Dependence Plots (PDPs)

- **Intuitive**
- **Easy to implement.**
- **Has a causal interpretation.**

  We intervene on a feature and measure the changes in the predictions. In doing so, we analyze the causal relationship between the feature and the prediction.The relationship is causal for the model – because we explicitly model the outcome as a function of the features – but not necessarily for the real world!

Partial Dependence Plots (PDPs)

- **2 factors only**
  The maximum number of features in a partial dependence function is two. This is not the fault of PDPs, but of the 2-dimensional representation (paper or screen) and also of our inability to imagine more than 3 dimensions.
- **Do not always show the feature distribution**
  Omitting the distribution can be misleading, because you might overinterpret regions with almost no data. This problem is easily solved by showing a rug (indicators for data points on the x-axis) or a histogram.
- **Assumption of independence**
  It is assumed that the feature(s) for which the partial dependence is computed are not correlated with other features. When the features are correlated, we create new data points in areas of the feature distribution where the actual probability is very low (for example it is unlikely that someone is 2 meters tall but weighs less than 50 kg).
- **Heterogeneous effects might be hidden**
  Suppose that for a feature half your data points have a positive association with the prediction – the larger the feature value the larger the prediction – and the other half has a negative association.
  The PD curve could be a horizontal line, since the effects of both halves of the dataset could cancel each other out.

Local methods

What happens to a specific input's prediction if you change one feature?

- **Individual Conditional Expectations**

What features do you need to change in order to change a specific input's prediction?

- **Counterfactual Explanations**

Which features of a specific input decide (anchor) its prediction?

- **Scoped Rules (Anchors)**

What is the contribution of each feature to a specific input's prediction?

- **Shapley Values**
- **SHAP (SHapley Additive exPlanations)**

Can you represent the actual model with a simpler model?

- **Local Surrogate (LIME)**

Local methods

What happens to a specific input's prediction if you change one feature?

- Individual Conditional Expectations

What features do you need to change in order to change a specific input's prediction?

- Counterfactual Explanations

What is the contribution of each feature to a specific input's prediction?

- **Shapley Values**
- **SHAP (SHapley Additive exPlanations)**

Which features of a specific input decide (anchor) its prediction?

- Scoped Rules (Anchors)

Can you represent the actual model with a simpler model?

- Local Surrogate (LIME)

Shapley Values

Shapley values come from cooperative games.

Imagine you are on a capture the flag team, and you get a team prize payout for your overall performance. How do you split the prize?

Shapley values tell us how much each player contributed to the final performance, and thus how to fairly distribute the payout among them.

The very same player may have a *different contribution and Shapley value* on different teams.

Shapley Values

A method for **assigning payouts** to players depending on their **contribution** to the total payout. Players cooperate in a **coalition** and receive a certain **profit** from this cooperation.

Applied to interpretability:

- The “game” = the prediction task for a single instance of the dataset
- The “gain” = the actual prediction for this instance minus the average prediction for all instances
- The “players” = the feature values of the instance that collaborate to receive the gain (= predict a certain value).

Shapley Values

Key Idea:

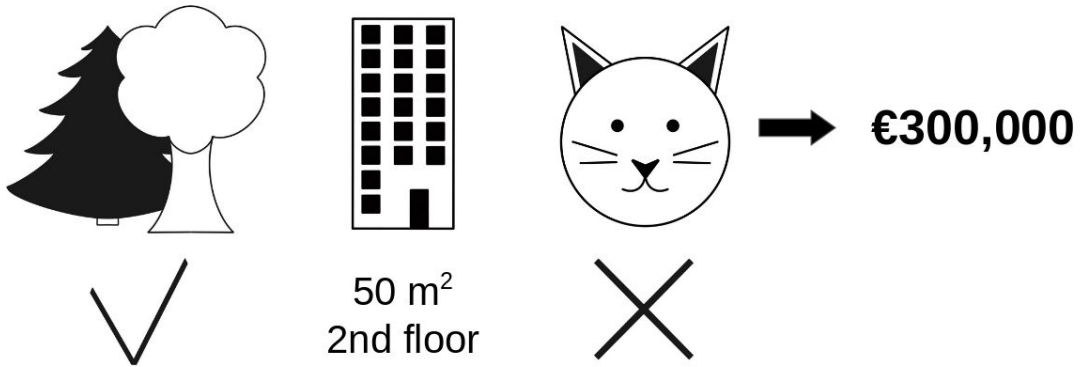
Assume that each input feature is a “player” in a game where the prediction is the payout.

Shapley values tell us how much each "player" contributed, and thus how to fairly distribute the “payout” among them.

In other words: **how responsible was each feature for influencing the final prediction?**

Example

You have trained a machine learning model to predict apartment prices.
Apartment 1 has an **area of 50 m^2**, is located on the **2nd floor**, has a **park nearby** and **cats are banned**. Your model predicts **€300,000** and you need to explain this prediction:

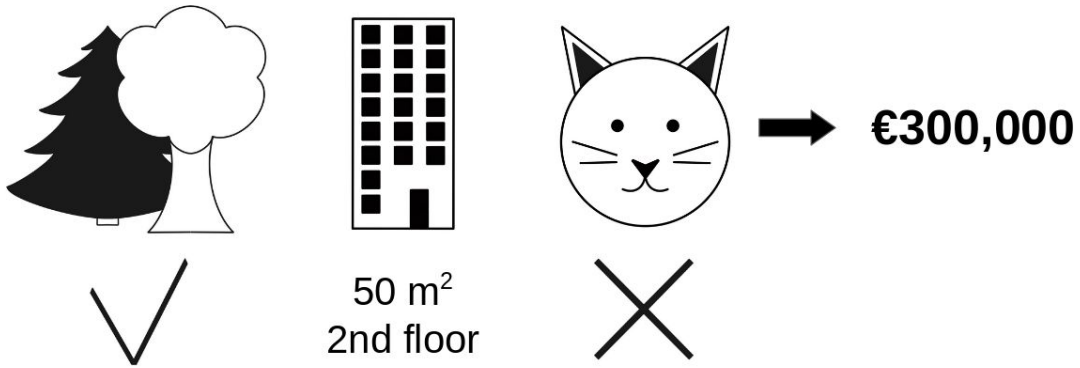


The average prediction for all apartments is **€310,000**.

Example

Why does your model predict **€300,000** instead of the average of **€310,000**?

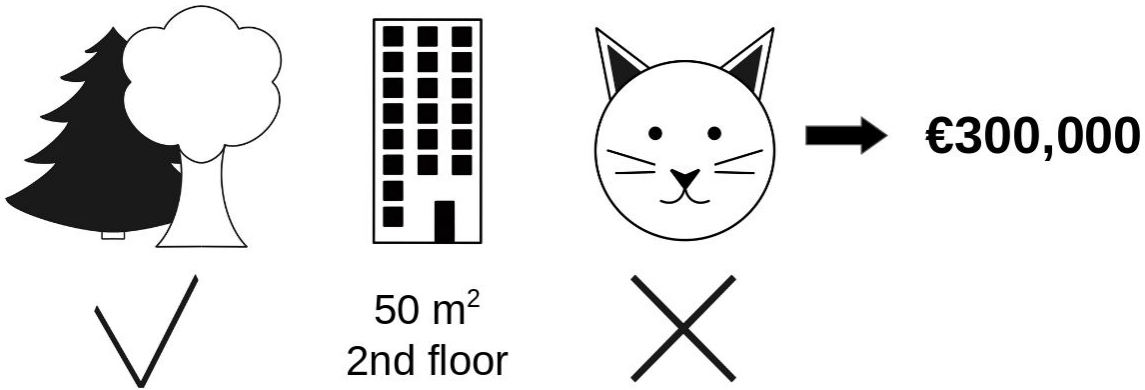
How much has **each feature value contributed** to this difference?



Example

In our apartment example:
the players = feature values (**park-nearby, cat-banned, area-50, floor-2nd**) work together to achieve the prediction of **€300,000**.

Our goal is to explain the difference between the actual prediction (**€300,000**) and the average prediction (**€310,000**): a difference of **-€10,000.**



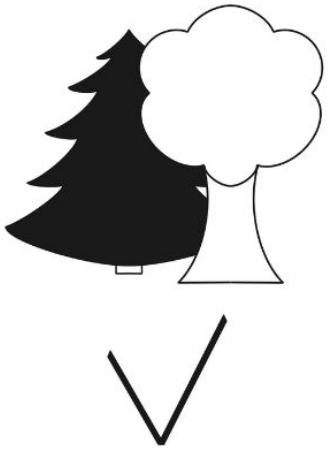
Example

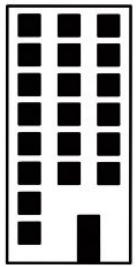
Specifically, how much did each feature value "contribute?"

As an example, perhaps:

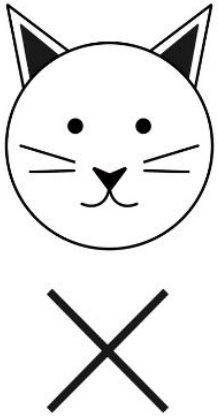
- ***park-nearby*** contributed **€30,000**;
- ***area-50*** contributed **€10,000**;
- ***floor-2nd*** contributed **€0**;
- ***cat-banned*** contributed **-€50,000**.

The contributions add up to **-€10,000**, the final prediction minus the average predicted apartment price.





50 m^2
2nd floor



€30,000 **€0** **€50,000**

Example

How much has each feature value contributed to the prediction compared to the average prediction?

If your model is a:

- Linear Regression:

  The effect of each feature is the weight of the feature times the feature value.

$$\beta_j x_j - \beta_j E(X_j)$$

- More complex:

  Consider Shapley Values

Definition

The Shapley value is the

average marginal contribution of a

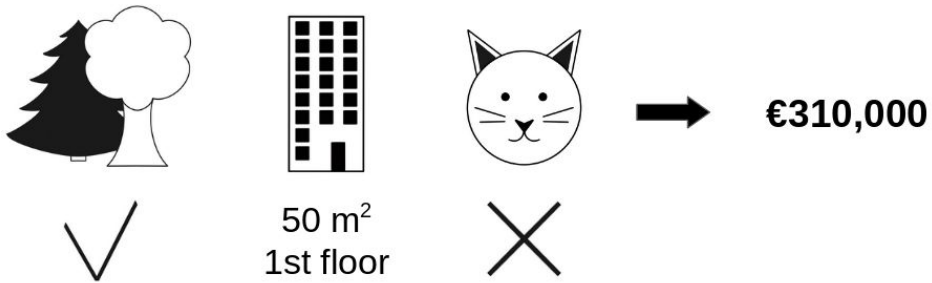
feature value across

all possible coalitions (combinations).

Example

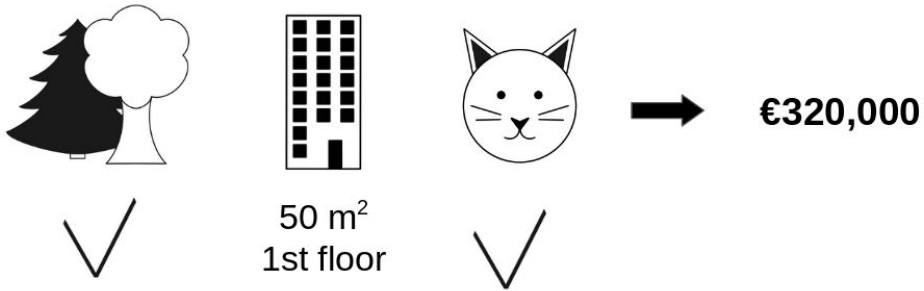
Let's evaluate the contribution of the ***cat-banned*** feature value when it is added to a coalition of ***park-nearby*** and ***area-50***.

In step 1, we simulate that only ***park-nearby***, ***cat-banned*** and ***area-50*** are in a coalition by randomly drawing another apartment from the data and using its value for the floor feature. The value ***floor-2nd*** was replaced by the randomly drawn ***floor-1st***.



Example

In step 2, we remove ***cat-banned*** from the coalition by replacing it with a random value of the ***cat allowed/banned*** feature from the randomly drawn apartment.
In this example, we drew ***cat-allowed***, but it could have been *cat-banned* again. We predict the apartment price for the coalition of ***park-nearby*** and ***area-50*** and get **€320,000**.



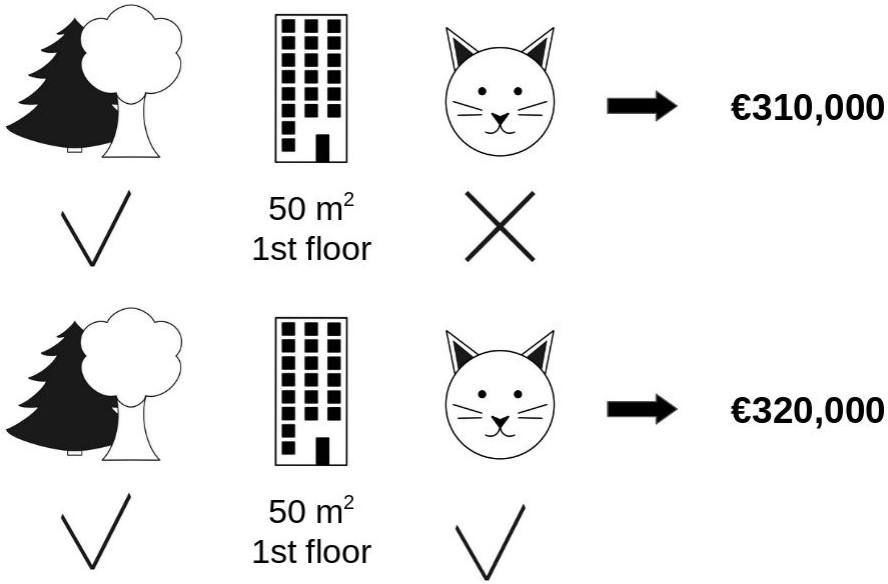
Example

The contribution of *cat-banned* was **€310,000 - €320,000 = -€10,000**.

This estimate depends on the values of the randomly drawn apartment that served as a “donor” for the cat and floor feature values.

We will get better estimates if we repeat this sampling step and average the contributions.

We repeat this computation for all possible coalitions.



Misconception

Common misconception:

The Shapley value is the average contribution of a feature value to the prediction in different coalitions.

The Shapley value is **NOT** the difference in prediction when we would remove the feature from the model.

This means that you can't measure the Shapley value by simply removing the feature and getting a new prediction.

Misconception

Why not?

Shapley Values

Returning to the capture the flag example, imagine:

The players are drafted in random order.
The Shapley value of a player is the average change in the final performance that the coalition already in the room receives when the player joins.

Player A is an exceptional runner.

If A joins the team when it has 5 fast runners but 0 good guards, then A's joining results in relatively small change.

If A joined as the first player, however, A would result in a large change. Similarly if the team has no good runners.

If we average over all orders of the players entering, we get the contribution of A.

Shapley Values

The Shapley value is the **average marginal contribution** of a **feature value** across **all possible coalitions** (combinations).

Imagine:

The feature values enter a room in random order.
All feature values in the room participate in the game (= contribute to the prediction).
The Shapley value of a feature value is the average change in the prediction that the coalition already in the room receives when the feature value joins them.

Shapley Values

The Shapley value is the **average marginal contribution** of a **feature value** across **all possible coalitions** (combinations).

All possible coalitions (sets) of feature values have to be evaluated with and without the j-th feature to calculate the exact Shapley value.

Theoretical Properties

- **Efficiency**

The feature contributions must add up to the difference of prediction for x and the average.
In other words, they always *exactly* explain the prediction difference from the average.

- **Symmetry**

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

- **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.

- **Additivity**

If the prediction can be decomposed as p_a + p_b, the Shapley value can also be decomposed thus. For each feature, you can calculate the Shapley value it has for each prediction sub-part and add them.

Advantages

- **Contrastive Explanations**

Instead of comparing a prediction to the average prediction of the entire dataset, you could compare it to a subset or a single data point. This contrastiveness is something certain local models *cannot* have.

- **Solid Theory**

The Shapley value is the only explanation method with a solid theory.
The axioms – efficiency, symmetry, dummy, additivity – give the explanation a reasonable foundation.
Some methods assume linear behavior of the ML model locally, but there is no theory as to why.

- **"Full" explanations**

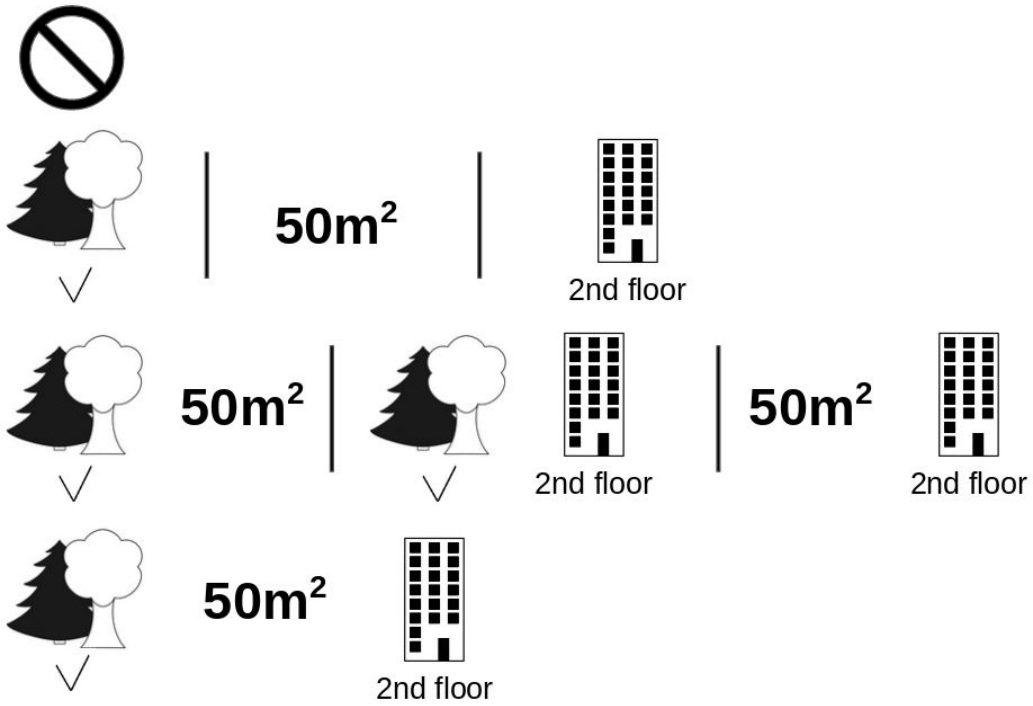
In situations where the law requires explainability – like EU's "right to explanations" – the Shapley value might be the only legally compliant method, because it is based on a solid theory and distributes the effects fairly. I am not a lawyer, so this reflects only my intuition about the requirements.

Disadvantages

- **Computationally expensive**

The computation time increases exponentially with the number of features.

One solution to keep the computation time manageable is to compute contributions for only a few samples of the possible coalitions.



Disadvantages

- **Need data access**

To calculate the Shapley value for a new data instance, you need to replace parts of the instance with values from randomly drawn data. Or, generate data instances that look "real" but are not.

- **Assumes independent features**

The Shapley value method includes unrealistic data instances when features are correlated.
To simulate that a feature value is missing from a coalition, we sample values from the feature's marginal distribution, which may yield values that do not make sense for this instance.
One solution might be to group correlated features together and get one mutual Shapley value for them.

- **Too many features**

Explanations created with Shapley values always use all features. Humans prefer selective explanations.

- **No predictions**

Cannot make statements about changes in prediction for changes in the input, e.g., "If cats were allowed, the rent would increase by 20,000."

SHAP

SHAP (SHapley Additive exPlanations) builds on the game theoretically optimal Shapley values.

Specifically, SHAP combines Shapley values and *local surrogate models*, which allow you to approximate complex models using simpler ones.

SHAP inherits the advantages of Shapley values as well as local surrogate methods (LIME).

Today

- Model-specific methods
  Choose a model designed with some inherently interpretable properties
- Model-agnostic methods
  Interpret models which have been trained, regardless of model design
  - Global methods
  - Local methods
- **Neural net-specific study**

Neural Net specific methods

Why are we talking about these methods separately?

Deep neural nets frequently take in high-dimensional inputs

E.g., feature = a pixel, expressed in RGBA

Deep neural nets also learn *large numbers of learned features (parameters)*

Modern LLMs / multi-modal models have hundreds of billions of parameters

Neural Net specific methods

What does each part of the neural net learn?

- **Learned Features**

What does each raw feature of the input mean / contribute?

- **Pixel Attribution**

What *user-defined* concepts (e.g., "cats") does the model learn?

- **Detecting Concepts**

What happens to the model's predictions if you *perturb* the input?

- **Adversarial Examples**

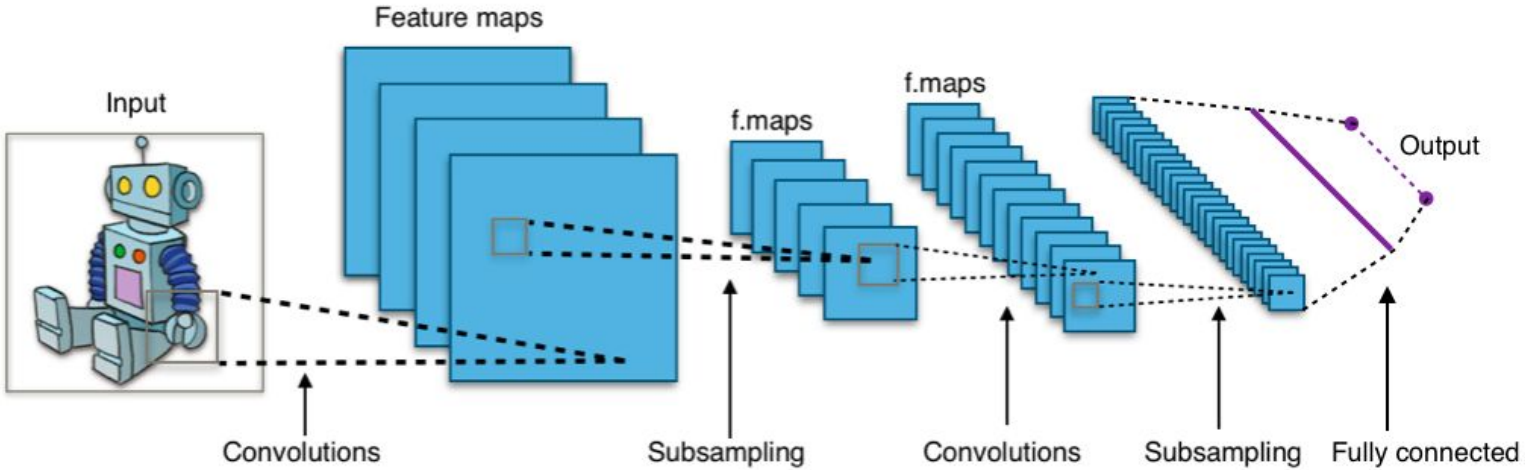
What training instances changed the model the most?

- **Influential Instances**

Neural Net methods

We'll focus on Image Classification Models for this lecture.

Specifically, Convolutional Neural Networks (CNNs)



Neural Net specific methods

What does each part of the neural net learn?

- **Learned Features**

What does each raw feature of the input mean / contribute?

- Pixel Attribution

What *user-defined* concepts (e.g., "cats") does the model learn?

- Detecting Concepts

What happens to the model's predictions if you *perturb* the input?

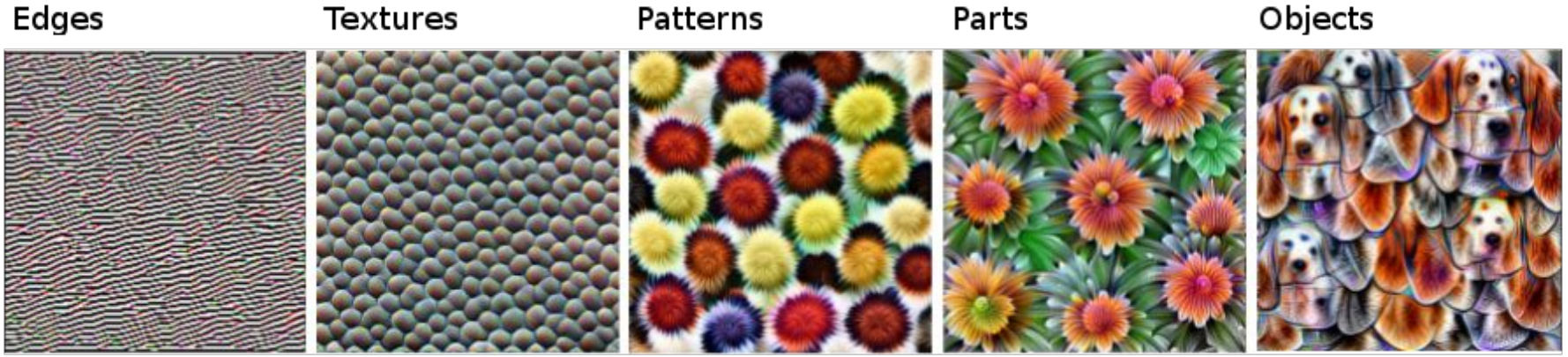
- Adversarial Examples

What training instances changed the model the most?

- Influential Instances

Learned Features

***Learned* features** are representations picked up by the model during the training process from data. It is difficult to explain *exactly* what they are, as they also differ by task, data, model, etc.



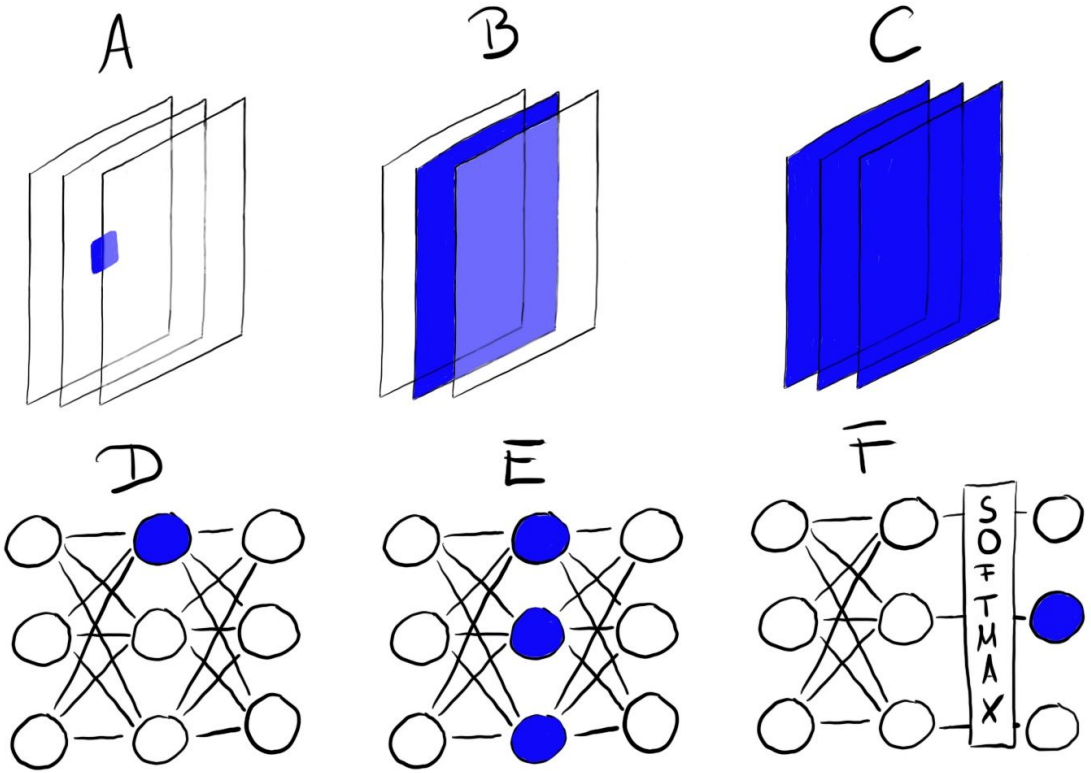
Features learned by a convolutional neural network (Inception V1) trained on the ImageNet data. The features range from simple features in the lower convolutional layers (left) to more abstract features in the higher convolutional layers (right). Figure from Olah, et al. (2017, CC-BY 4.0)

Learned Features

We'll use “unit” to refer any part of the neural net we can "draw a box" around:

- individual neurons
- channels (also called feature maps)
- entire layers
- or the final class probability in classification
  (or the corresponding pre-softmax neuron, which is recommended)

Units



A) Convolution neuron
B) Convolution channel
C) Convolution layer
D) Neuron
E) Hidden layer
F) Class probability neuron (or corresponding pre-softmax neuron)

Learned Features

Key idea: find the input that maximizes the activation of a particular unit

How do you find this input?

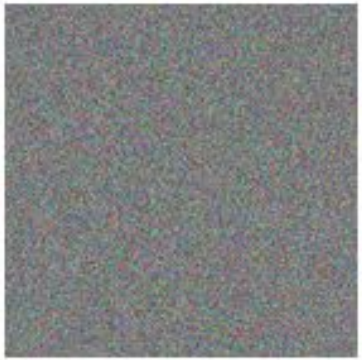
It could be:

- "Real" or "natural" input – images taken from an existing dataset (likely your train/validation data)
- Synthetic images

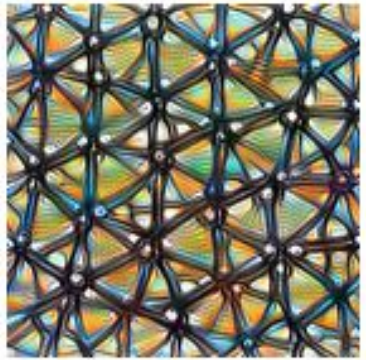
But how do you generate the synthetic images?

Learned Features

But how do you generate the synthetic images?



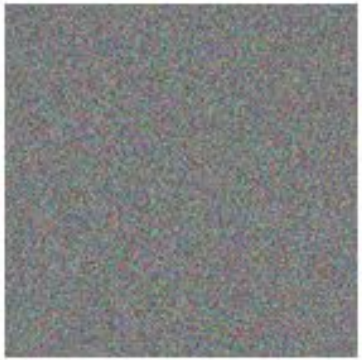
Step 0



Step 2048

Learned Features

But how do you generate the synthetic images?

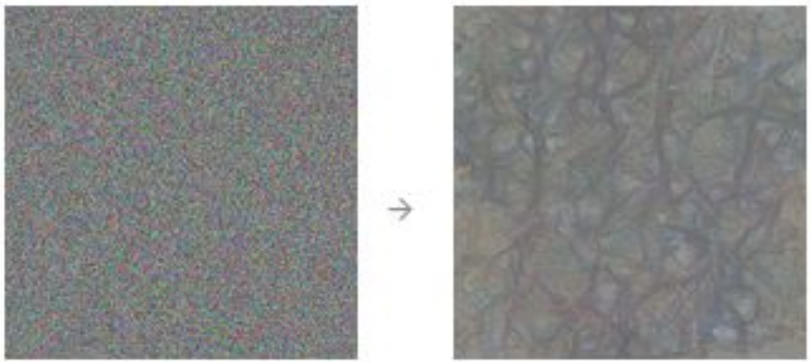


Step 0

1. Initialize with random noise

Learned Features

But how do you generate the synthetic images?

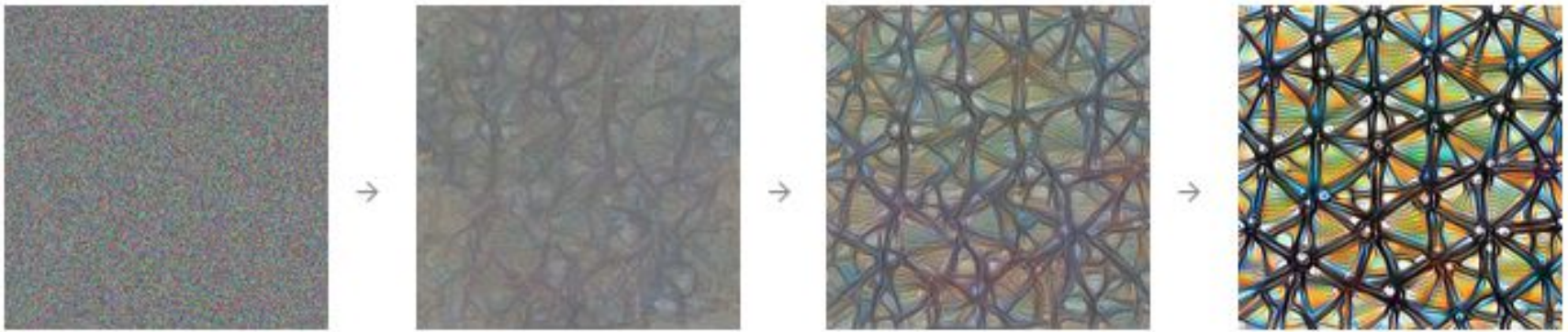


2. Take steps in the direction of increasing activations

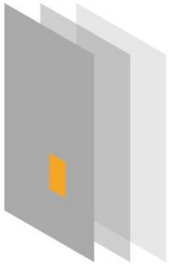
This is pretty clear cut for individual neurons (high value) but more complex for, e.g., a whole layer

Learned Features

But how do you generate the synthetic images?

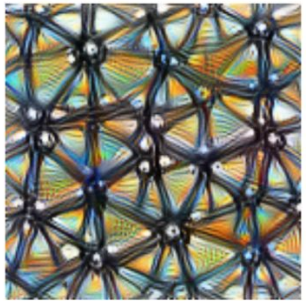


Learned Features



Neuron

`layer`$_n$`[x,y,z]`

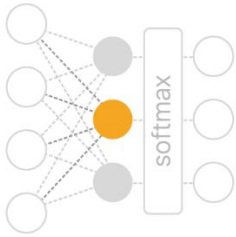


Channel

`layer`$_n$`[:,:,z]`

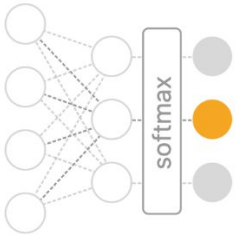
Layer/DeepDream

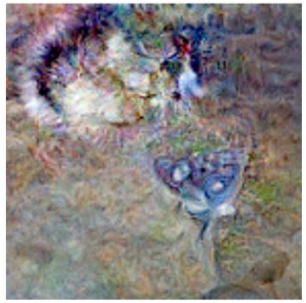
`layer`$_n$`[:,:,:]`^{2}



Class Logits

`pre_softmax[k]`





Class Probability

`softmax[k]`

Learned Features

What if, instead of generating or finding images that embody learned features from unit activations, you look for specific learned features that you want to have by measuring activations in the presence of those features?

To look for where the model is learning the concept of an eye: 1) show the model many pictures of eyes, 2) look for units with high activation

Learned Features

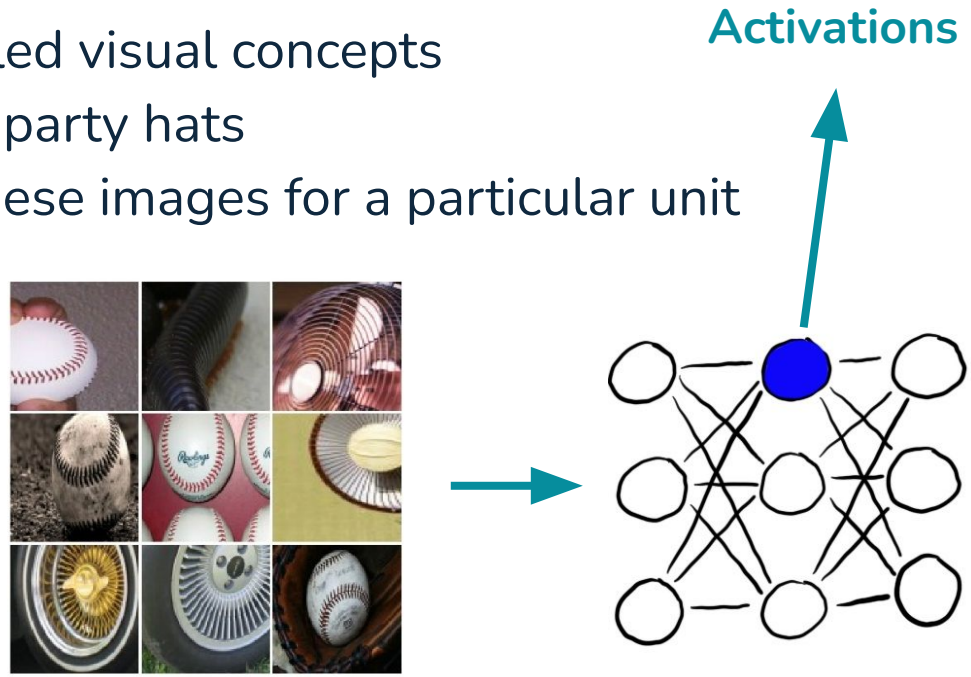
More formally:

1. Get images with human-labeled visual concepts
   E.g., stripes, fur, red, bicycles, party hats

Learned Features

More formally:

1. Get images with human-labeled visual concepts
   E.g., stripes, fur, red, bicycles, party hats
2. Measure the activations for these images for a particular unit
   E.g., a channel, or a neuron



Learned Features

More formally:

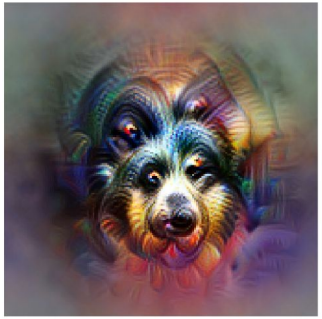
1. Get images with human-labeled visual concepts
   E.g., stripes, fur, red, bicycles, party hats
2. Measure the activations for these images for a particular unit
   E.g., a channel, or a neuron
3. Quantify the alignment of activations and labeled concepts
   High alignment = the unit "knows" the human definition of the concept

Learned Features

But *careful*:
How do you know that your stated concepts are accurately being defined or found?

What other concepts correlate with your concept?

Baseball—or stripes?



Animal faces—or snouts?

Clouds—or fluffiness?

Buildings—or sky?

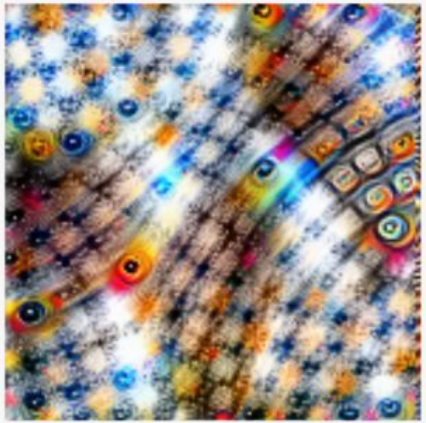
Learned Features

Slightly positive activation examples

Maximum activation examples

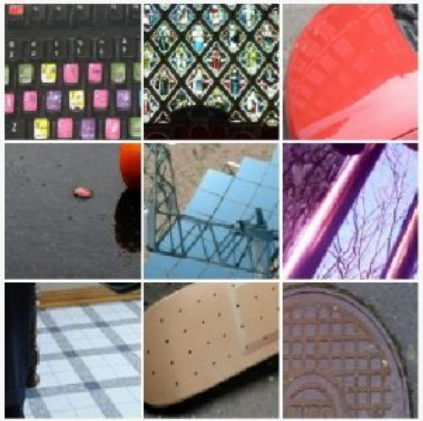
Positive optimized

Learned Features



Negative optimized

Minimum activation examples



Slightly negative activation examples

Learned Features

Negative optimized

Minimum activation examples

Slightly negative activation examples

Slightly positive activation examples

Maximum activation examples

Positive optimized

Negative optimized

Minimum activation examples

Slightly negative activation examples

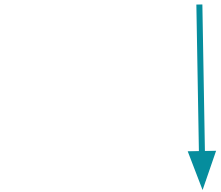
Slightly positive activation examples

Maximum activation examples

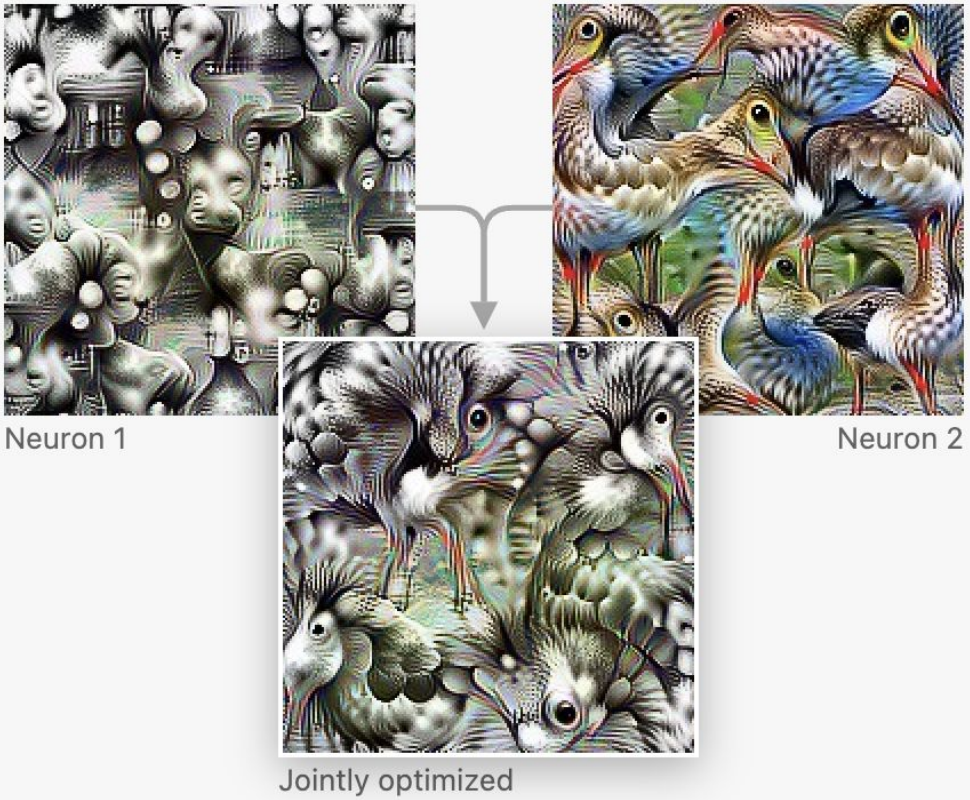
Positive optimized

Learned Features

Black & White + Bird Heads



Black & White Bird Heads



Learned Features

Layer 4a, Unit 476

Layer 4a, Unit 455

Mosaic?

Mosaics about People?

Corners? Boxes?

Learned Features

Layer 4b, Unit 475

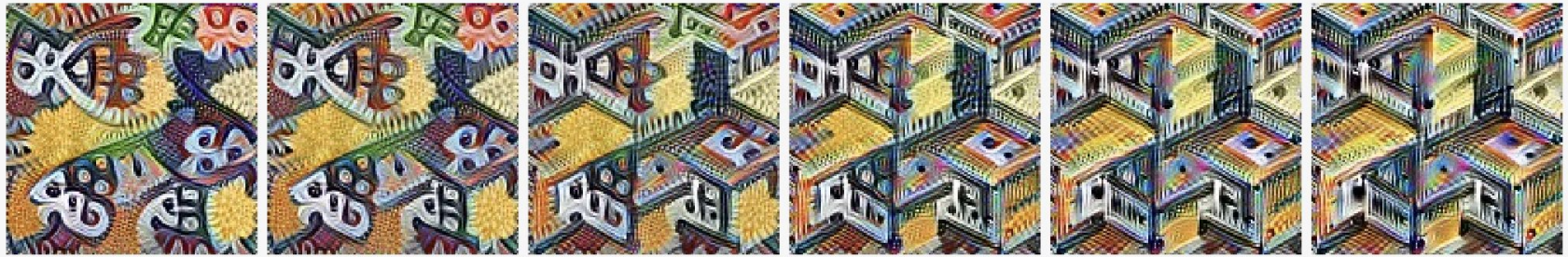
Layer 4a, Unit 476

People?

Mosaics about People?

Mosaics?

Learned Features



Layer 4a, Unit 476

Layer 4a, Unit 455

Layer 4b, Unit 475

Layer 4a, Unit 476

Neural Net specific methods

What does each part of the neural net learn?

- Learned Features

What does each raw feature of the input mean / contribute?

- **Saliency Maps / Pixel Attribution**

What *user-defined* concepts (e.g., "cats") does the model learn?

- Detecting Concepts

What happens to the model's predictions if you *perturb* the input?

- Adversarial Examples

What training instances changed the model the most?

- Influential Instances

Feature Attribution

Recall (from SHAP):

You can measure the "influence" of a feature by perturbing it, either to see marginal changes, or to compare to an average or baseline prediction

OR, other methods may measure the gradient of the prediction w.r.t. the feature.

Pixel attribution is a special case of feature attribution in general

E.g., the larger the absolute value of the gradient, the stronger the effect of a change of this pixel.

Saliency Maps

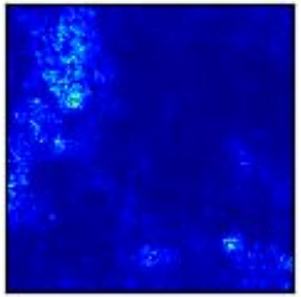
Greyhound (vanilla)

Soup Bowl (vanilla)

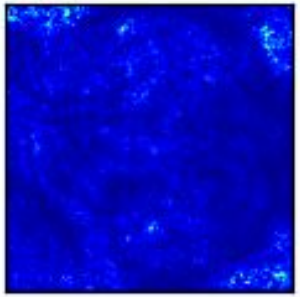
Eel (vanilla)

Saliency Maps

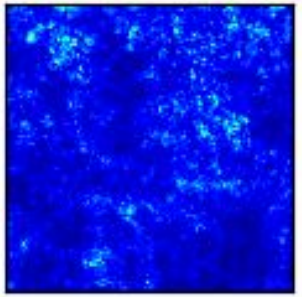
Greyhound (vanilla)



Soup Bowl (vanilla)

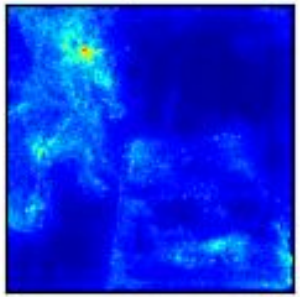


Eel (vanilla)

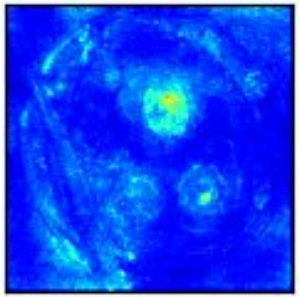


Saliency Maps

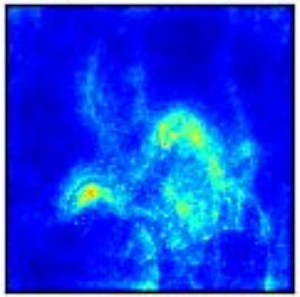
Greyhound (Smoothgrad)



Soup Bowl (Smoothgrad)

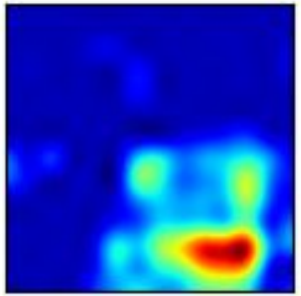


Eel (Smoothgrad)

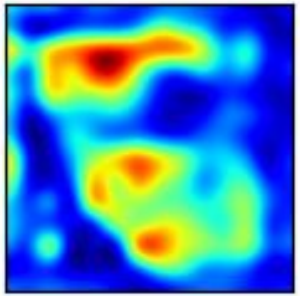


Saliency Maps

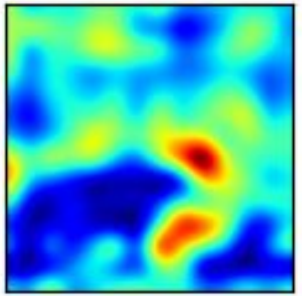
Greyhound (Grad-Cam)



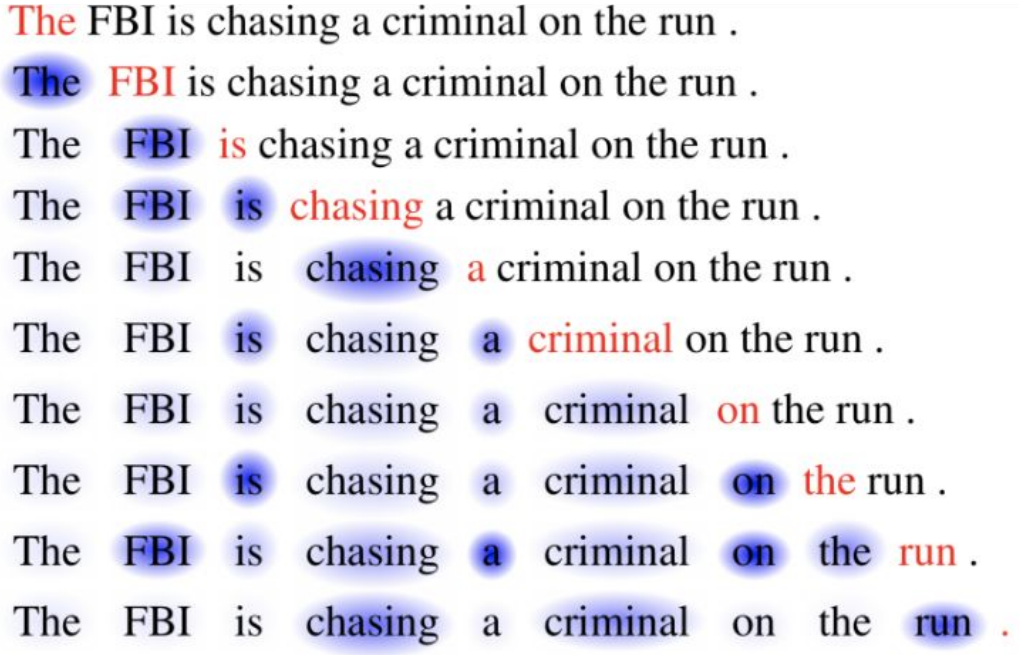
Soup Bowl (Grad-Cam)



Eel (Grad-Cam)



Attention Patterns



Attention Patterns

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

Saliency Maps

If you can identify the specific part of an image that has the most influence on a prediction, you know what the model has learned about that prediction class.

E.g., if it has the largest gradients on the cat's ears, it's learned that ears are what define a cat.

Neural Net specific methods

What does each part of the neural net learn?

- Learned Features

What does each raw feature of the input mean / contribute?

- Pixel Attribution

What *user-defined* concepts (e.g., "cats") does the model learn?

- Detecting Concepts

What happens to the model's predictions if you *perturb* the input?

- **Adversarial Examples**

What training instances changed the model the most?

- Influential Instances

Adversarial Examples

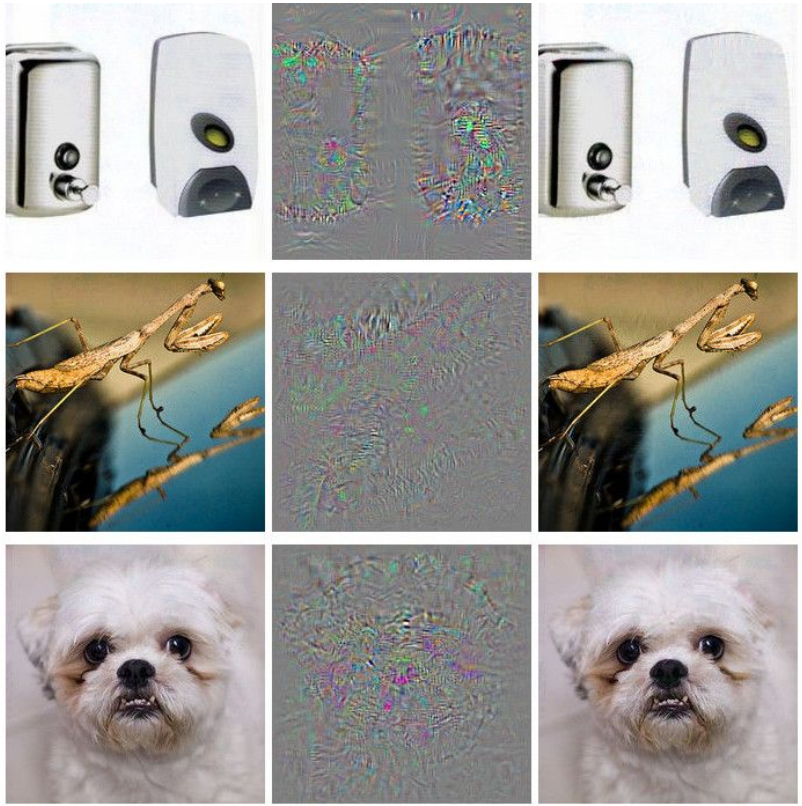
What is "adversarial" about an example?

The example is an adversary to the model's "expected" behavior.

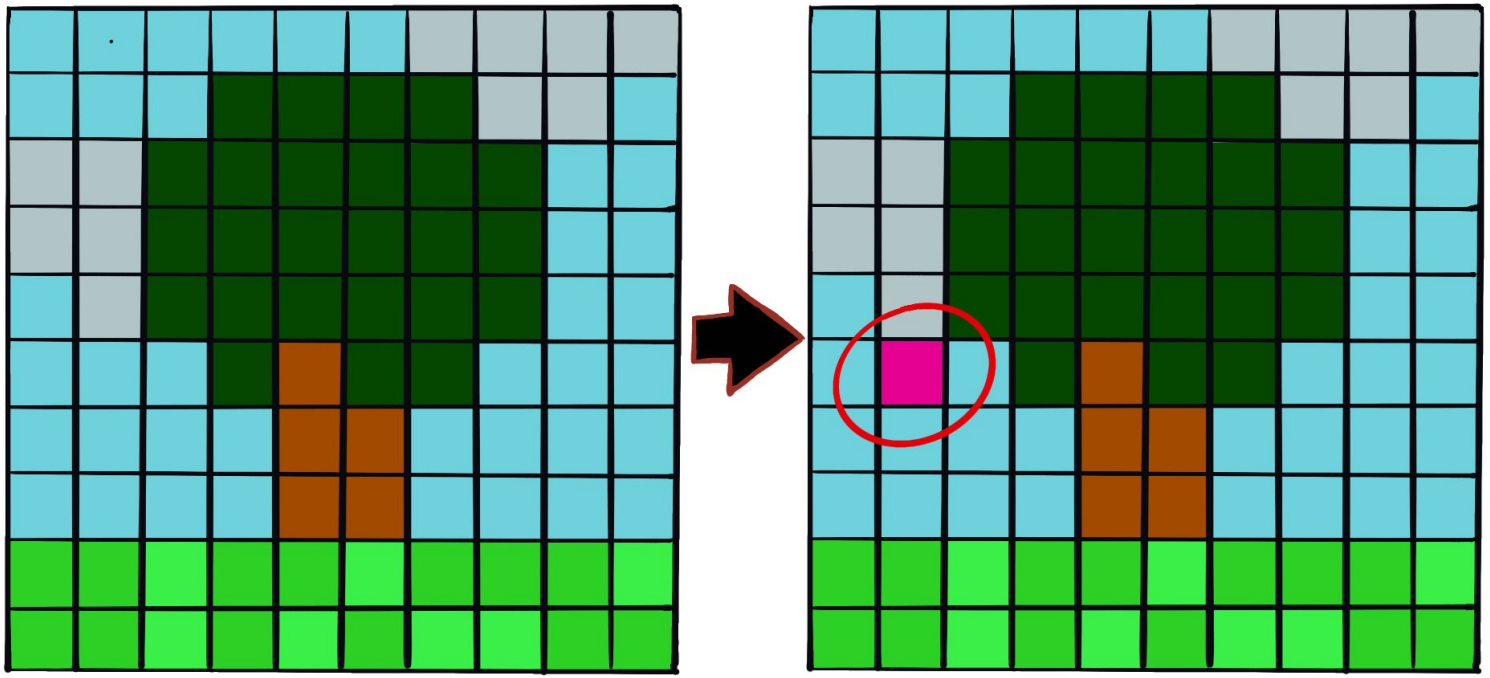
Adversarial Examples

What's the difference?

Adversarial Examples



Adversarial Examples



Adversarial Examples

Why study adversarial examples?

If we can break a model unexpectedly, that can tell us about what the model learns overall, where it has weaknesses, how we might want to add to the training data, or how to evaluate the model.

Neural Net specific methods

What does each part of the neural net learn?

- Learned Features

What does each raw feature of the input mean / contribute?

- Pixel Attribution

What *user-defined* concepts (e.g., "cats") does the model learn?

- Detecting Concepts

What happens to the model's predictions if you *perturb* the input?

- Adversarial Examples

What training instances changed the model the most?

- **Influential Instances**

Influential Instances

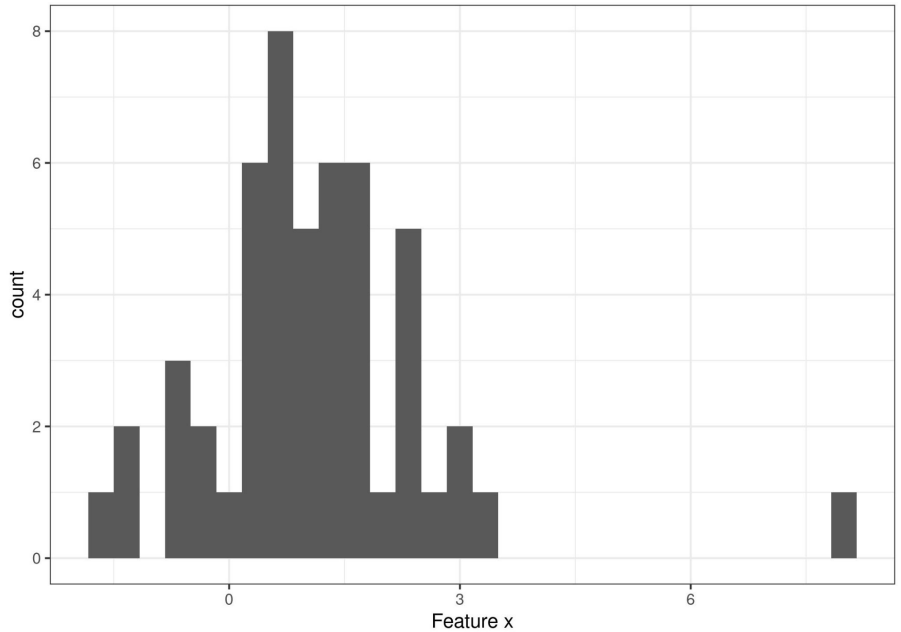
How do we define "influential?"

We call a training instance “influential” when its deletion from the training data considerably changes the parameters or predictions of the model.

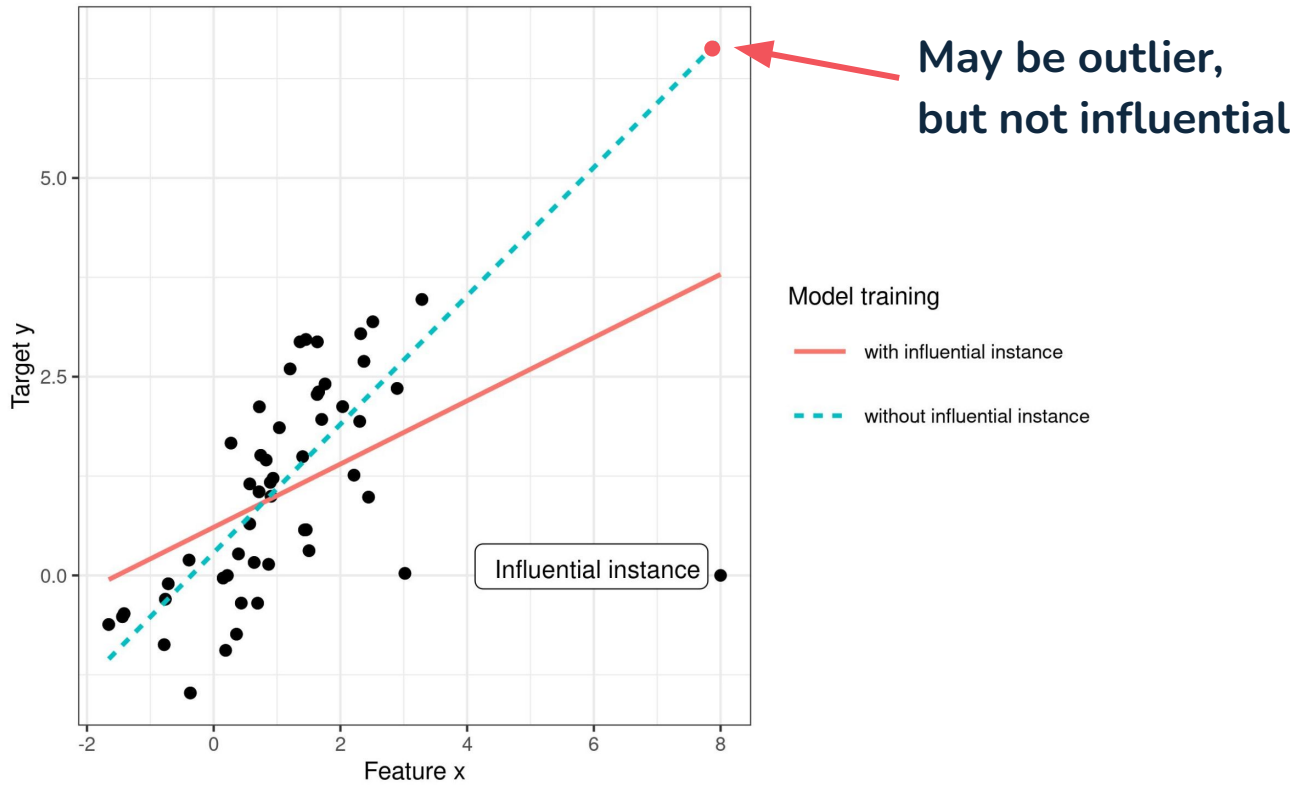
Influential Instances

Are they the same as outliers?

Not always

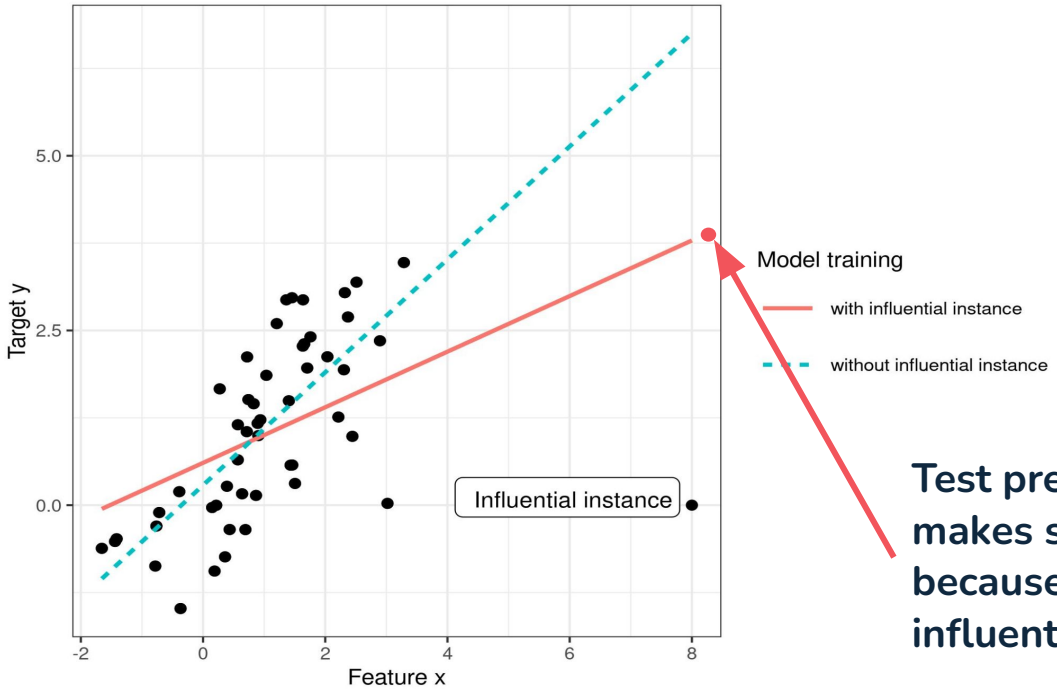


Influential Instances



Influential Instances

Why study influential instances?



Influential Instances

Why study influential instances?

Training data is composed of many individual points

If we can identify how individual decide the overall model, we may better understand what the model learns, how it learns, and what the data is saying

Discussion time:
Where is interpretability going?

Looking forward (Chris Molnar's thoughts)

The data scientists will automate themselves.

For this to happen, the tasks must be well-defined and there must to be some processes and routines around them. Today, these routines and processes are missing, but data scientists are working on them. As machine learning becomes an integral part of many industries and institutions, many of the tasks will be automated.

Machine learning will be automated and, with it, interpretability.

An already visible trend is the automation of model training. That includes automated engineering and selection of features, hyperparameter optimization, comparison of different models, and ensembling or stacking of the models.
Model-agnostic interpretation methods can be automatically applied to any model that emerges from the automated machine learning process: Automatically compute the feature importance, plot the partial dependence, train a surrogate model, and so on. Everyone will be able to train machine learning models without knowing how to program, but there will still be a need for machine learning experts.

Looking forward (Chris Molnar's thoughts)

Robots and programs will explain themselves.

We need more intuitive interfaces to machines and programs that make heavy use of machine learning. Some examples: A self-driving car that reports why it stopped abruptly (“70% probability that a kid will cross the road”); A credit default program that explains to a bank employee why a credit application was rejected (“Applicant has too many credit cards and is employed in an unstable job.”); A robot arm that explains why it moved the item from the conveyor belt into the trash bin (“The item has a craze at the bottom.”).

Interpretability could boost machine intelligence research.

I can imagine that by doing more research on how programs and machines can explain themselves, we can improve our understanding of intelligence and become better at creating intelligent machines.

Looking forward (Chris Molnar's thoughts)

The focus will be on model-agnostic interpretability tools.

It is much easier to automate interpretability when it is decoupled from the underlying machine learning model. The advantage of model-agnostic interpretability lies in its modularity. We can easily replace the underlying machine learning model and interpretation method separately.

We do not analyze data, we analyze models.

The raw data itself is always useless. Interpretable machine learning is a great way to distill knowledge from data. You can probe the model extensively, the model automatically recognizes if and how features are relevant for the prediction (many models have built-in feature selection), the model can automatically detect how relationships are represented, and – if trained correctly – the final model is a very good approximation of reality.

Where is interpretability going?
Who knows?

Feedback

(It'll only take 30 sec)

https://bit.ly/ml_05232024

Credits

& Further Reading

https://christophm.github.io/interpretable-ml-book/the-future-of-interpretability.html

https://distill.pub/2017/feature-visualization/