Lecture 10: Model Visualization

AC295

Advanced Practical Data Science

Guest Lecturer: Yongwhan Lim

Yongwhan Lim

Guest Lecturer

- **Research Software Engineer at Google Brain Research**
- http://yongwhan.github.io/
- **Stanford**
  - CS (BS '11 & MS '12)
  - Math (BS '11)
- **MIT**
  - Operations Research (PhD, on extended leave)

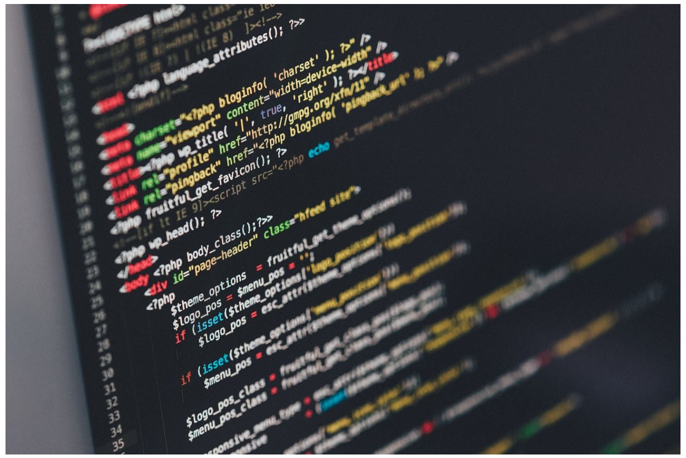
Outline

Overview

- ❑ Motivation
- ❑ Model Explanation Methods
- ❑ Model-Agnostic Methods
- ❑ “The Big Picture”
- ❑ Available Tools
- ❑ Lecture Outline

Motivation

- As a data scientist, you need to know **ins and outs** of the machine learning models.
- Often you need to provide a **trustworthy**, **transparent**, and **accountable** <u>explanation</u> to stakeholders beyond the final evaluation metrics (e.g., Accuracy, ROC-AUC score, etc.).
- To provide a good justification, we need an easily **generalizable** framework to explain the machine learning model.



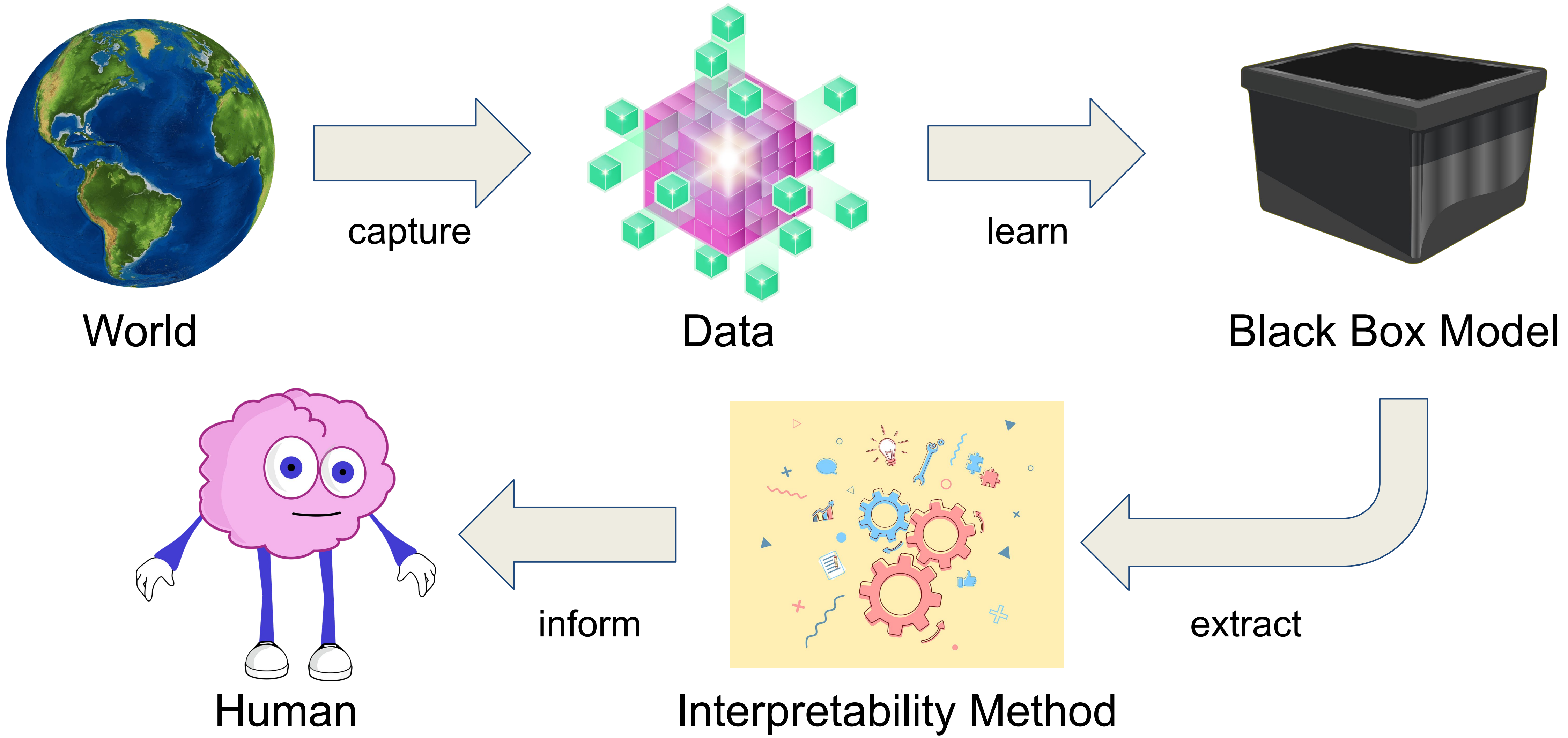
Model Explanation Methods

- There are inherent trade-off between model interpretability and accuracy.
- There are many ways to explain the machine learning models:
    - Use only interpretable models
        - Linear regression, logistic regression, decision trees, etc.
        - Most times too simple!
    - Use model specific interpretation methods
        - Lacks generalization because it is model dependent!
    - Use **model-agnostic methods**

Model-Agnostic Methods

- Some aspects for model-agnostic interpretation methods are:
  - Model flexibility
    - The method can work with any machine learning model including neural networks.
  - Explanation flexibility
    - Not binded to a particular form of explanation
    - linear formula, feature importance graphics, etc.
  - Representation flexibility
    - The explanation system should be able to use a different feature representation.

“The Big Picture”



Available Tools

- For interpreting machine learning models, there are many Python libraries already available. Some of the popular options are:
  - ELI5
  - LIME
  - SHAP
  - Yellowbrick
  - Alibi
  - Lucid
  - Skater
  - MLxtend
  - …

Available Tools

- In this lecture, we will go through the following three most popular libraries in this order:
  - **ELI5**
  - **LIME**
  - **SHAP**

Lecture Outline

- We will cover some **theoretical background** for each of the libraries first. We will then list few references that cover the topic in more depth.

- Then, we will take a look at few **simple colabs** to demonstrate how they can be used in practice.

Outline

ELI5

- Overview
- Highlight
- Demos

ELI5: Overview

- ELI5 (“Explain Like I am 5 year old”) is a Python package which helps to debug machine learning classifiers and explain their predictions.

- Provides support for many machine learning frameworks and packages including:
    - **scikit-learn**
    - **keras**
    - XGBoost
    - lightGBM
    - etc.

ELI5: Highlight

- ELI5 visualization <u>shows weights for each feature</u> depicting how influential it might have been in contributing to the final prediction decision.

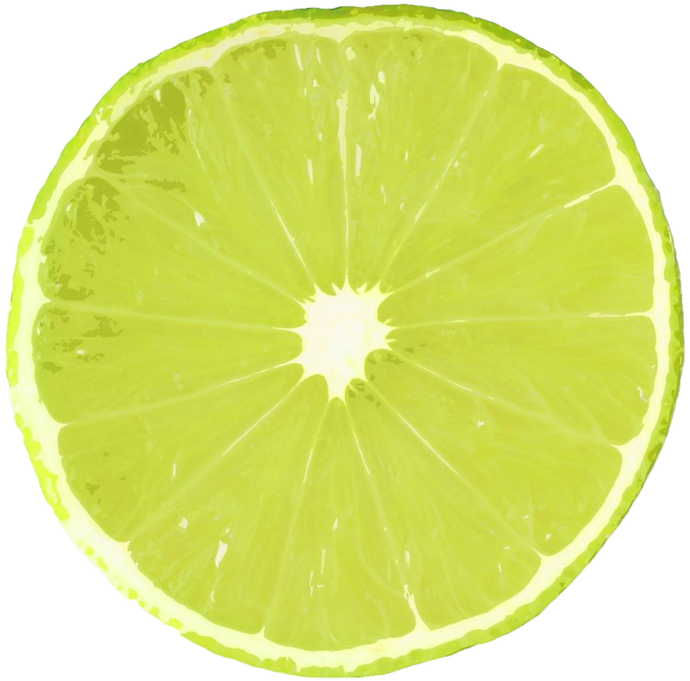
- This is a good step in direction of model-agnostic interpretation but not entirely model-agnostic like, we will see it later, for LIME.

ELI5: Demos

- Let's dive right into the demo. We will show how we apply ELI5 on:
  - Debugging **scikit-learn** text classification pipeline
  - Explaining **Keras** image classifier predictions with Grad-CAM

Outline

LIME



- Overview
- Objective
- Training Recipe
- “Intuition”
- Sparse Linear Example
- Pros and Cons
- Demos

LIME: Overview

- The creators of LIME (Local Interpretable Model-Agnostic Explanations) outline four basic criteria that must be satisfied:

LIME: Overview

- The creators of LIME (Local Interpretable Model-Agnostic Explanations) outline four basic criteria that must be satisfied:
    - **Interpretable**: The explanation must be easy to understand depending on the target demographic.

LIME: Overview

- The creators of LIME (Local Interpretable Model-Agnostic Explanations) outline four basic criteria that must be satisfied:
  - **Interpretable**: The explanation must be easy to understand depending on the target demographic.
  - **Local fidelity**: The explanation should be able to explain how the model behaves for individual predictions.

LIME: Overview

- The creators of LIME (Local Interpretable Model-Agnostic Explanations) outline four basic criteria that must be satisfied:
  - **Interpretable**: The explanation must be easy to understand depending on the target demographic.
  - **Local fidelity**: The explanation should be able to explain how the model behaves for individual predictions.
  - **Model-agnostic**: The method should be able to explain any model.

LIME: Overview

- The creators of LIME (Local Interpretable Model-Agnostic Explanations) outline four basic criteria that must be satisfied:
    - **Interpretable**: The explanation must be easy to understand depending on the target demographic.
    - **Local fidelity**: The explanation should be able to explain how the model behaves for individual predictions.
    - **Model-agnostic**: The method should be able to explain any model.
    - **Global perspective**: The model, as a whole, should be considered while explaining it.

LIME: Objective

- Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models.
- The objective can be expressed as:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

LIME: Objective

- Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models.
- The objective can be expressed as:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

- The explanation model for instance x is the model (e.g., linear regression model) that minimizes loss L (e.g., mean squared error), which measures how close the explanation is to the prediction of the original model f (e.g., deep learning model), while the model complexity Ω is kept low (e.g., prefer fewer features).
- G is the family of possible explanations.
- $\boldsymbol{\pi}$ is a proximity measure defining a neighborhood around instance x we consider for an explanation.

LIME: Objective

- Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models.
- The objective can be expressed as:

$$\text{explanation}(x) = \overset{\textbf{LIME}}{\boxed{\arg\min_{g \in G} L(f, g, \pi_x)}} + \overset{\textbf{USER}}{\boxed{\Omega(g)}}$$

- The explanation model for instance x is the model (e.g., linear regression model) that minimizes loss L (e.g., mean squared error), which measures how close the explanation is to the prediction of the original model f (e.g., deep learning model), while the model complexity Ω is kept low (e.g., prefer fewer features).
- G is the family of possible explanations.
- π is a proximity measure defining a neighborhood around instance x we consider for an explanation.

LIME: Training Recipe

- The 5-step training recipe:

LIME: Training Recipe

- The 5-step training recipe:
  - **Select your instance of interest** for which you want to have an explanation of its black box prediction.

LIME: Training Recipe

- The 5-step training recipe:
    - **Select your instance of interest** for which you want to have an explanation of its black box prediction.
    - **Perturb your dataset** and get the black box **predictions** for these new points.

LIME: Training Recipe

- The 5-step training recipe:
  - **Select your instance of interest** for which you want to have an explanation of its black box prediction.
  - **Perturb your dataset** and get the black box **predictions** for these new points.
  - **Weight the new samples** according to their proximity to the instance of interest.

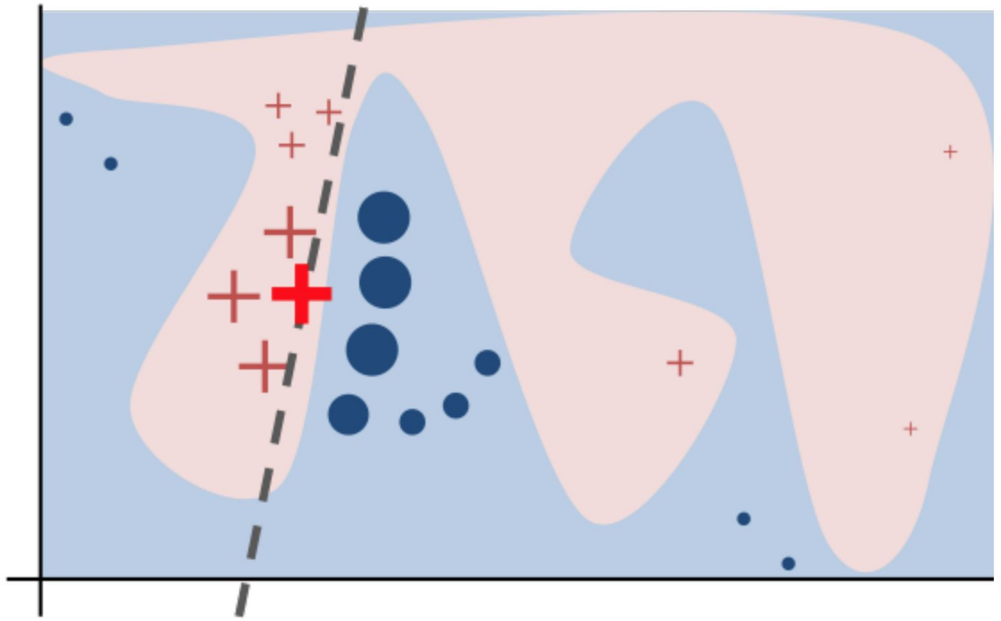
LIME: Training Recipe

- The 5-step training recipe:
  - **Select your instance of interest** for which you want to have an explanation of its black box prediction.
  - **Perturb your dataset** and get the black box **predictions** for these new points.
  - **Weight the new samples** according to their proximity to the instance of interest.
  - **Train a weighted, interpretable model** on the dataset with the variations.

LIME: Training Recipe

- The 5-step training recipe:
  - **Select your instance of interest** for which you want to have an explanation of its black box prediction.
  - **Perturb your dataset** and get the black box **predictions** for these new points.
  - **Weight the new samples** according to their proximity to the instance of interest.
  - **Train a weighted, interpretable model** on the dataset with the variations.
  - Explain the prediction by **interpreting the local model**.

LIME: “Intuition”



- “Toy example” to present intuition for LIME.
- model’s complex decision function is blue/pink background.
- **bold red cross** is the instance being explained.
- LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by **size**).
- The dashed line is the learned explanation that is locally (but not globally) faithful.

LIME: Sparse Linear Example

- Let G be the class of linear models. Then,

$$g(z') = w_g \cdot z'$$

LIME: Sparse Linear Example

- Let G be the class of linear models. Then,

$$g(z') = w_g \cdot z'$$

- We use locally weighted square loss $\mathcal{L}$ and exponential kernel $\pi_x(z)$:

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

$$\pi_x(z) = \exp\left(\frac{-D(x, z)^2}{\sigma^2}\right)$$

- Here, $D(x, z)$ is a distance function (e.g., cosine for text, L^2 for images).

LIME: Sparse Linear Example

- Let G be the class of linear models. Then,

$$g(z') = w_g \cdot z'$$

- We use locally weighted square loss $\mathcal{L}$ and exponential kernel $\pi_x(z)$:

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

$$\pi_x(z) = \exp\left(\frac{-D(x, z)^2}{\sigma^2}\right)$$

- Here, $D(x, z)$ is a distance function (e.g., cosine for text, L^2 for images).
- Finally, we take $\Omega(g)$ as:

$$\Omega(g) = \infty \mathbb{1}[\|w_g\|_0 > K]$$

LIME: Pros and Cons

- We can apply LIME to tabular data, text, and images. We will look at this in more detail in demo.

LIME: Pros and Cons

- We can apply LIME to tabular data, text, and images. We will look at this in more detail in demo.
- **Advantages**
  - Even if you replace the underlying machine learning model, you can still use the same local, interpretable model for explanation.
  - They make **human-friendly** explanations.
  - Very **easy to use**.
  - The **fidelity measure** gives us a good idea of how reliable the interpretable model is in explaining the black box predictions.

LIME: Pros and Cons

- We can apply LIME to tabular data, text, and images. We will look at this in more detail in demo.
- **Advantages**
  - Even if you replace the underlying machine learning model, you can still use the same local, interpretable model for explanation.
  - They make **human-friendly** explanations.
  - Very **easy to use**.
  - The **fidelity measure** gives us a good idea of how reliable the interpretable model is in explaining the black box predictions.
- **Disadvantages**
  - The correct definition of the neighborhood is a very big, unsolved problem when using LIME with tabular data leading to **instability**.

LIME: Demos

- Let’s dive right into the demo. We will show how we apply LIME on:
  - Black-box interpretation of models using LIME - Part I
  - Black-box interpretation of models using LIME - Part II

Outline

SHAPLEY

- Shapley Values
- SHAP

Shapley Values

- ❑ General Idea
- ❑ Concrete Example
- ❑ Details
- ❑ Advantages
- ❑ Disadvantages

Shapley Values: General Idea

- The Shapley value, coined by Shapley (1953), is 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.
  - The **game** is the prediction task for a single instance of the dataset.
  - The **gain/loss** is the actual prediction for this instance minus the average prediction for all instances.
  - The **players** are the feature values of the instance that collaborate to receive the gain.

Shapley Values: Concrete Example

- You have trained a machine learning model to predict apartment prices. For a certain apartment it predicts \$3M and you need to explain this prediction (game). The apartment has a size of **50 m^2**, is located on the **2nd floor**, has a **park nearby** and **cats are banned**.

Shapley Values: Concrete Example

- You have trained a machine learning model to predict apartment prices. For a certain apartment it predicts \$3M and you need to explain this prediction (game). The apartment has a size of **50 m^2**, is located on the **2nd floor**, has a **park nearby** and **cats are banned**.
- The average prediction for all apartments is \$3.1M. How much has each feature value contributed to the prediction compared to the average prediction?

Shapley Values: Concrete Example

- You have trained a machine learning model to predict apartment prices. For a certain apartment it predicts \$3M and you need to explain this prediction (game). The apartment has a size of **50 m^2**, is located on the **2nd floor**, has a **park nearby** and **cats are banned**.
- The average prediction for all apartments is \$3.1M. How much has each feature value contributed to the prediction compared to the average prediction?
- the feature values (park-nearby, cat-banned, area-50, and floor-2nd) are **players** that worked together to achieve the prediction of \$3M, a **loss** of \$100K.

Shapley Values: Concrete Example

- You have trained a machine learning model to predict apartment prices. For a certain apartment it predicts \$3M and you need to explain this prediction (game). The apartment has a size of **50 m^2**, is located on the **2nd floor**, has a **park nearby** and **cats are banned**.
- The average prediction for all apartments is \$3.1M. How much has each feature value contributed to the prediction compared to the average prediction?
- the feature values (park-nearby, cat-banned, area-50, and floor-2nd) are **players** that worked together to achieve the prediction of \$3M, a **loss** of \$100K.
- The answer could be: park-nearby (+\$300K), size-50 (+\$100K), floor-2nd (+\$0), and cat-banned (-\$500K). These add up to -\$100K, which is (final prediction) - (average predicted apartment price).

Shapley Values: Concrete Example

- The Shapley value is the average marginal contribution of a feature value across all possible coalitions.
- The following figure shows all coalitions of feature values that are needed to determine the Shapley value for cat-banned:
    - no feature values: { }
    - {park-nearby}
    - {size-50}
    - {floor-2nd}
    - {park-nearby, size-50}
    - {park-nearby, floor-2nd}
    - {size-50, floor-2nd}
    - {park-nearby, size-50, floor-2nd}

Shapley Values: Concrete Example

- For each of these coalitions we compute the predicted apartment price with and without the feature value cat-banned and take the difference to get the marginal contribution.
- The Shapley value is the (weighted) average of marginal contributions.
- We replace the feature values of features that are not in a coalition with random feature values from the apartment dataset to get a prediction from the machine learning model.
- If we estimate the Shapley values for all feature values (park-nearby, cat-banned, area-50, and floor-2nd), we get the complete distribution of the prediction minus the average among the feature values.

Shapley Values: Concrete Example

- For each of these coalitions we compute the predicted apartment price with and without the feature value cat-banned and take the difference to get the marginal contribution.
- The Shapley value is the (weighted) average of marginal contributions.
- We replace the feature values of features that are not in a coalition with random feature values from the apartment dataset to get a prediction from the machine learning model.
- If we estimate the Shapley values for all feature values (park-nearby, cat-banned, area-50, and floor-2nd), we get the complete distribution of the prediction minus the average among the feature values.
- In general, we do **Monte Carlo sampling** when there are many feature values.

Shapley Values: Details

- **Four** Axioms:

Shapley Values: Details

- **<u>Four</u>** Axioms:
  - **Efficiency**: the feature contributions must add up to the difference of prediction for x and the average.

Shapley Values: Details

- **Four** Axioms:
  - **Efficiency**: the feature contributions must add up to the difference of prediction for x and the average.
  - **Symmetry**: the contributions of two feature values j and k should be the same if they contribute equally to all possible coalitions.

Shapley Values: Details

- **<u>Four</u>** Axioms:
  - **Efficiency**: the feature contributions must add up to the difference of prediction for x and 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.

Shapley Values: Details

- **Four** Axioms:
  - **Efficiency**: the feature contributions must add up to the difference of prediction for x and 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**: for combined payouts, their Shapley values are additive.

Shapley Values: Details

- **<u>Four</u>** Axioms:
    - **Efficiency**: the feature contributions must add up to the difference of prediction for x and 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**: for combined payouts, their Shapley values are additive.
- The Shapley value is the **<u>only attribution method</u>** that satisfies the properties Efficiency, Symmetry, Dummy and Additivity, which together can be considered a definition of a **fair** payout.

Shapley Values: Advantages (over LIME)

- Due to **efficiency property** of Shapley values, Shapley values might be a **legally compliant** method.
- **ontrastive explanation**: instead of comparing a prediction to the average prediction of the entire dataset, you could compare it to a subset or even to a single data point.
- The Shapley value is the explanation method with a **solid theory**: the four axioms (efficiency, symmetry, dummy, additivity) give it a reasonable foundation.

Shapley Values: Disadvantages

- Computationally **prohibitively expensive**.
- The Shapley value can be **misinterpreted**!
- Shapley value method **always use all the features**.
- Shapley value **does not have a prediction ability** like LIME.
- **Access to the data** is required if you want to calculate the Shapley value for a new data instance.
- the Shapley value method suffers from inclusion of **unrealistic data instances** when features are correlated.

SHAP

- Overview
- Properties
- Core Explainers
- Kernel SHAP Highlight
- Kernel SHAP Explanation
- Tree SHAP
- Demos

SHAP: Overview

- **SHAP** (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations, uniting several previous methods.

SHAP: Overview

- **SHAP** (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations, uniting several previous methods.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

- where g is the explanation model
- $z' \in \{0, 1\}^M$ is the coalition vector
- M is the maximum coalition size
- ϕ_j is the feature attribution (Shapley values) for a feature j.

SHAP: Properties

- **Local accuracy** = Shapley efficiency
- **Missingness**: a missing feature gets an attribution of zero
- **Consistency**: if a model changes so that the marginal contribution of a feature value increases or stays the same (regardless of other features), the Shapley value also increases or stays the same.
  - **<u>FACT</u>**: From Consistency the Shapley properties Additivity, Dummy, and Symmetry follow (Appendix of Lundberg and Lee).

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:
    - **Tree Explainer**: Uses Tree SHAP algorithms to explain the output of ensemble tree models.

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:
    - **Tree Explainer**: Uses Tree SHAP algorithms to explain the output of ensemble tree models.
    - **Gradient Explainer**: Support TensorFlow and Keras models.

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:
  - **Tree Explainer**: Uses Tree SHAP algorithms to explain the output of ensemble tree models.
  - **Gradient Explainer**: Support TensorFlow and Keras models.
  - **Deep Explainer** (DEEP SHAP): Meant to approximate SHAP values for deep learning models.

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:
  - **Tree Explainer**: Uses Tree SHAP algorithms to explain the output of ensemble tree models.
  - **Gradient Explainer**: Support TensorFlow and Keras models.
  - **Deep Explainer** (DEEP SHAP): Meant to approximate SHAP values for deep learning models.
  - **Kernel Explainer** (Kernel SHAP): Uses the Kernel SHAP method to explain the output of any function.

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:
  - **Tree Explainer**: Uses Tree SHAP algorithms to explain the output of ensemble tree models.
  - **Gradient Explainer**: Support TensorFlow and Keras models.
  - **Deep Explainer** (DEEP SHAP): Meant to approximate SHAP values for deep learning models.
  - **Kernel Explainer** (Kernel SHAP): Uses the Kernel SHAP method to explain the output of any function.
  - **Sampling Explainer**: This is an extension of the Shapley sampling values explanation method.

SHAP: Core Explainers

- SHAP provides multiple explainers for different kind of models:
  - **Tree Explainer**: Uses Tree SHAP algorithms to explain the output of ensemble tree models.
  - **Gradient Explainer**: Support TensorFlow and Keras models.
  - **Deep Explainer** (DEEP SHAP): Meant to approximate SHAP values for deep learning models.
  - **Kernel Explainer** (Kernel SHAP): Uses the Kernel SHAP method to explain the output of any function.
  - **Sampling Explainer**: This is an extension of the Shapley sampling values explanation method.
  - **Partition Explainer; Linear Explainer; Permutation Explainer**; **Additive Explainer;**

SHAP: Kernel SHAP Highlight

- Kernel SHAP consists of five steps:
  - Sample coalitions $z_k' \in \{0, 1\}^M$, $k \in \{1, \ldots, K\}$ (1 = feature present in coalition, 0 = feature absent).
  - Get prediction for each z_k' by first converting z_k' to the original feature space and then applying model f: $f(h_x(z_k'))$.
  - Compute the weight for each z_k' with the SHAP kernel.
  - Fit weighted linear model.
  - Return Shapley values ϕ_k, the coefficients from the linear model.

SHAP: Kernel SHAP Explanation

- We can create a random coalition by repeated coin flips until we have a chain of 0's and 1's. For example, the vector of (1,0,1,0) means that we have a coalition of the first and third features. The K sampled coalitions become the dataset for the regression model.
- For tabular data, $\mathbf{h_x}$ maps 0's to the values of another instance that we sample from the data. This means that we equate "feature value is absent" with "feature value is replaced by random feature value from data".

SHAP: Kernel SHAP Explanation

- We can create a random coalition by repeated coin flips until we have a chain of 0's and 1's. For example, the vector of (1,0,1,0) means that we have a coalition of the first and third features. The K sampled coalitions become the dataset for the regression model.
- For tabular data, $\mathbf{h_x}$ maps 0's to the values of another instance that we sample from the data. This means that we equate "feature value is absent" with "feature value is replaced by random feature value from data".
- **SHAP Kernel**:

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

- where M is the maximum coalition size and |z′| is the number of present features in instance z′.

SHAP: Kernel SHAP Explanation

- Build our weighted linear regression model:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

- We train the linear model g by optimizing the following loss function L:

$$L(f, g, \pi_x) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_x(z')$$

- where Z is the training data.
- This is the good old sum of squared errors that we usually optimize for linear models.
- The estimated coefficients of the model, the ϕ_j's are the Shapley values.

SHAP: Tree SHAP

- Lundberg et. al (2018) proposed TreeSHAP, a variant of SHAP for tree-based machine learning models such as decision trees, random forests and gradient boosted trees.
- TreeSHAP was introduced as a **fast, model-specific alternative** to KernelSHAP but it turned out that it can produce unintuitive feature attributions.
- TreeSHAP defines the value function using the **conditional expectation** instead of the marginal expectation.
- Compared to exact KernelSHAP, it reduces the computational complexity from TL^2M to TLD^2 where M is maximum coalition size, T is the number of trees, L is the maximum number of leaves in any tree, and D the maximal depth of any tree.

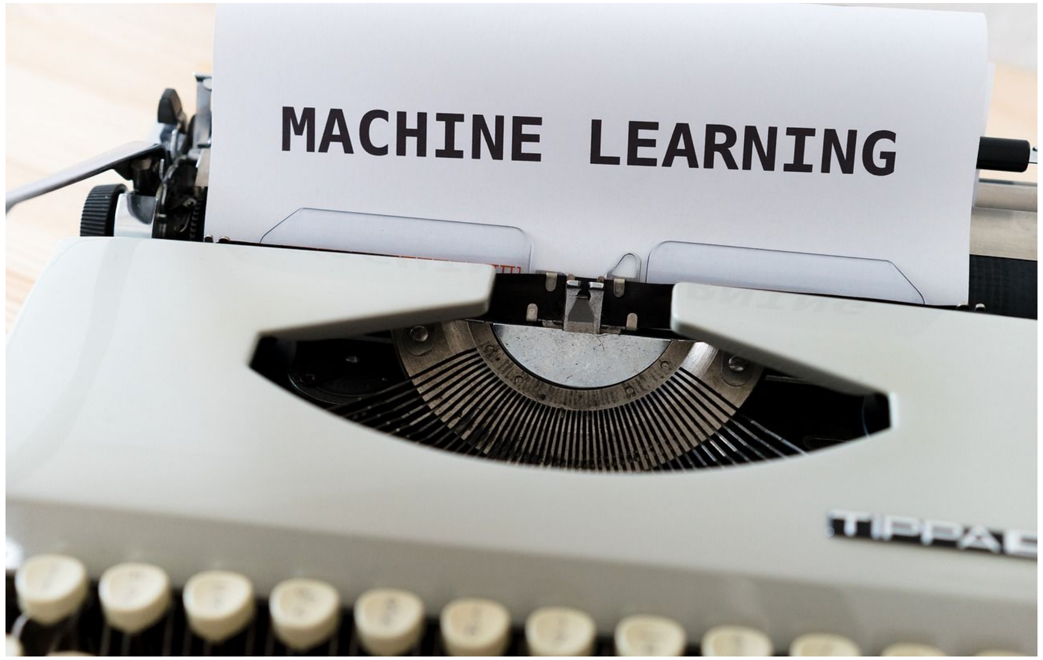
SHAP: Demos

- Model agnostic example with KernelExplainer
- Tree ensemble example with SHAP TreeExplainer

Outline

Conclusion

- We looked at ELI5, LIME, and SHAP as a way to explain the machine learning model in model agnostic way.
- There is a tension between explainability and accuracy of the machine learning models.



References

- Interpretable ML: https://christophm.github.io/interpretable-ml-book
- ELI5: https://eli5.readthedocs.io/en/latest/index.html
- LIME: https://github.com/marcotcr/lime
- SHAP: https://github.com/slundberg/shap

THANK YOU!