Applied Machine Learning for Business Analytics

Lecture 7: Interpretability Methods in Machine Learning

Lecturer: Zhao Rui

Agenda

1. Interpretability in Machine Learning
2. Interpretability vs Accuracy
3. Interpretability Methods
4. Feature Evaluation

1. Interpretability in Machine Learning

Why do we need model explainability

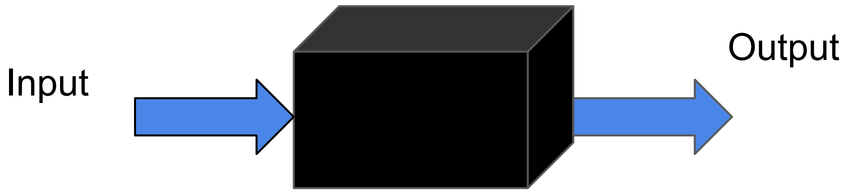
- Use Machine Learning to review resumes
  - Based on your capability or gender?
  - https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G
- Use Machine Learning to detect fraud transactions?
  - Why does the model think this transaction is suspicious?

High-stakes decision

- The above examples all belong to high-stakes decisions. The decisions have a huge impact on human well-being.
- What are those non high-stakes decisions?
    - Recommendations in E-commerces websites
    - Auto-fill in emails
    - But interpretability in machine learning sys is still valuable

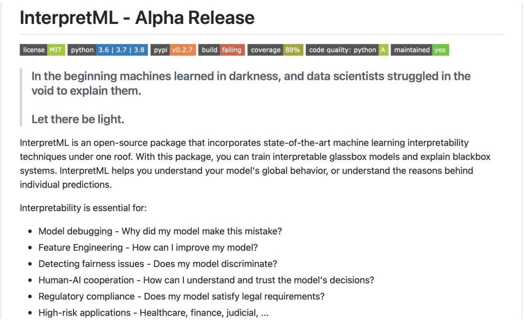
Black-box Model

- If the ML system is deployed in high-stakes decisions environment:
  - Is accuracy important?
  - Can we trust the machine learning model?
- In banking, insurance and other heavily regulated industries, model interpretability is a serious legal mandate
- In lots of critical areas such as healthcare, government, bioinformatics, etcs, rationale for models' decision is necessary for trust



Goals of Interpretability

- Model debugging
  - Why did my model make mistake?
- Feature Engineering
  - How can I improve my model?
- Detecting fairness issues
  - Does my model have biases?
- Human-AI cooperation
  - How can I understand and trust model’s decision?
- Regulatory Compliance
  - Does my model satisfy legal requirements?
- High-stake Decisions
  - Healthcare, Finance..



InterpretML - Alpha Release

license MIT | python 3.6 | 3.7 | 3.8 | pypi v0.2.7 | build failing | coverage 88% | code quality: python A | maintained yes

> In the beginning machines learned in darkness, and data scientists struggled in the void to explain them.
>
> Let there be light.

InterpretML is an open-source package that incorporates state-of-the-art machine learning interpretability techniques under one roof. With this package, you can train interpretable glassbox models and explain blackbox systems. InterpretML helps you understand your model's global behavior, or understand the reasons behind individual predictions.

Interpretability is essential for:

- Model debugging - Why did my model make this mistake?
- Feature Engineering - How can I improve my model?
- Detecting fairness issues - Does my model discriminate?
- Human-AI cooperation - How can I understand and trust the model's decisions?
- Regulatory compliance - Does my model satisfy legal requirements?
- High-risk applications - Healthcare, finance, judicial, ...

2. Interpretability vs Accuracy

Linear models first

- Prediction is the linear combinations of the features values, weighted by the model coefficients.

Students A's chance = 0.2 + 0.1* GPA - 0.005 * Hours on Tiktok

One student's feature
GPA: 5
Hours on Tiktok: 60 hours

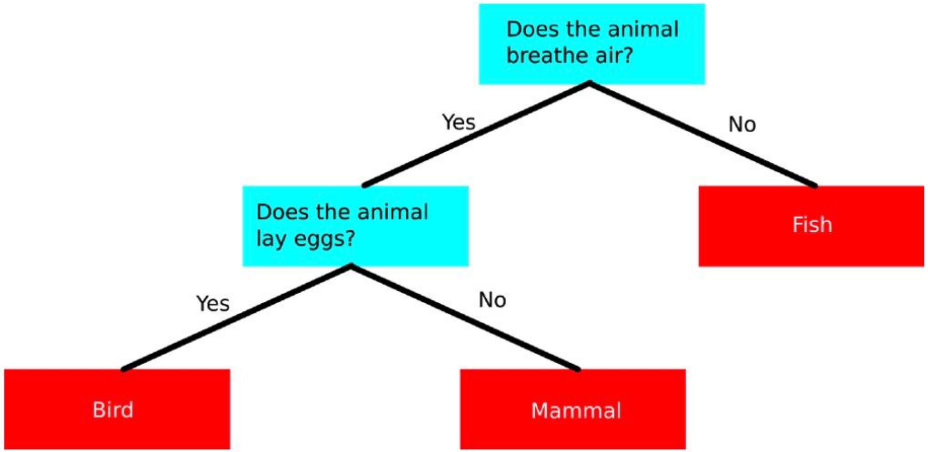
→

Feature Contributions
GPA: 0.5
Hours on Tiktok: -0.3

- Capability of linear models is limited.

Decision tree

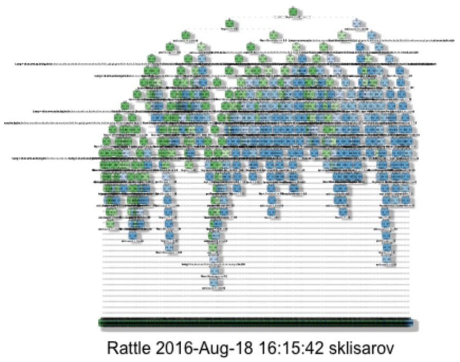
- It is “interpretable”
- More powerful compared to linear models.



Source:
https://towardsdatascience.com/a-beginners-guide-to-decision-tree-classification-6d3209353ea

Decision tree can be complex

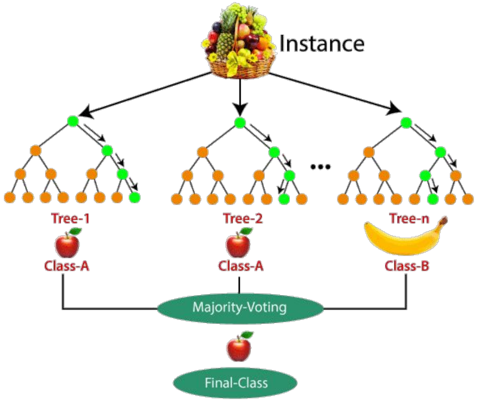
It can be a huge and complex tree.



My goal is to extract some useful rules from the entire process to implement in a score card.

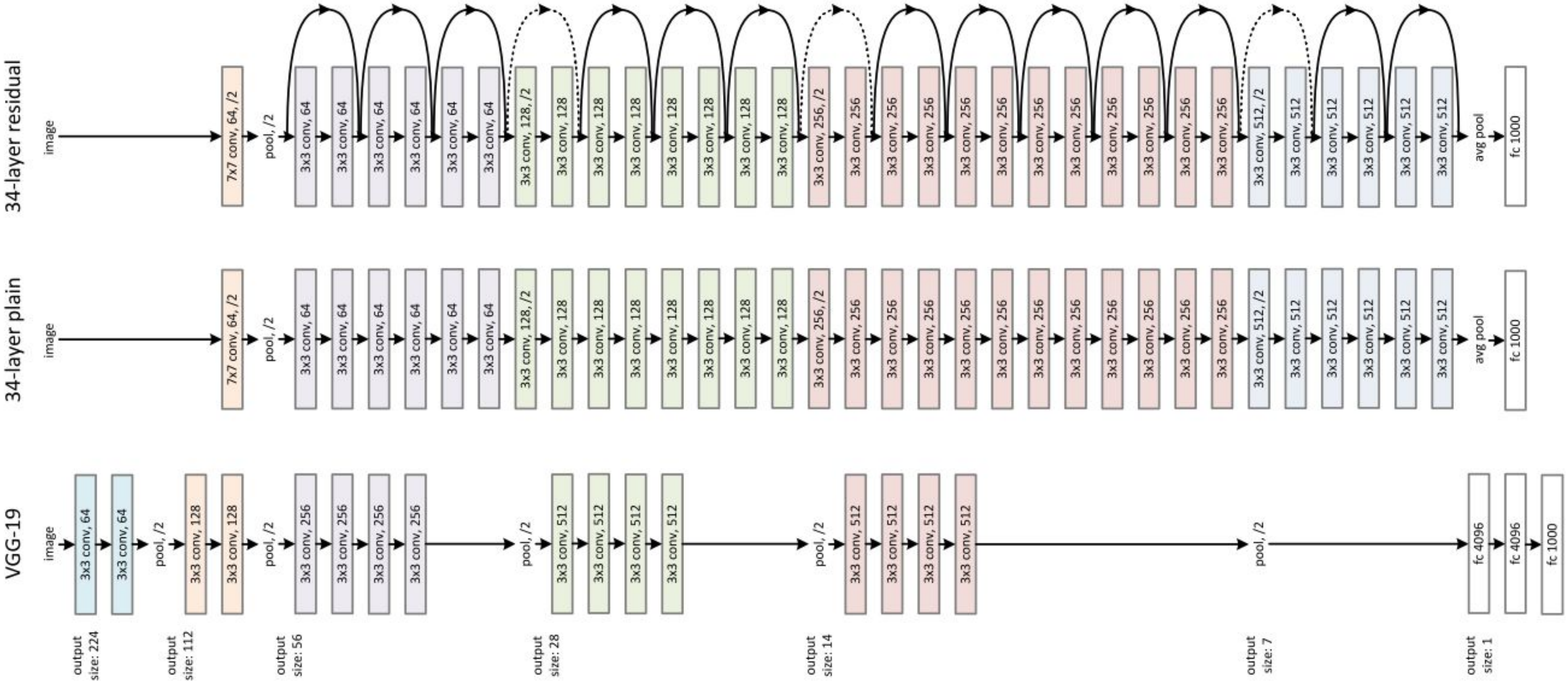
https://stats.stackexchange.com/questions/230581/decision-tree-too-large-to-interpret

It can be a forest



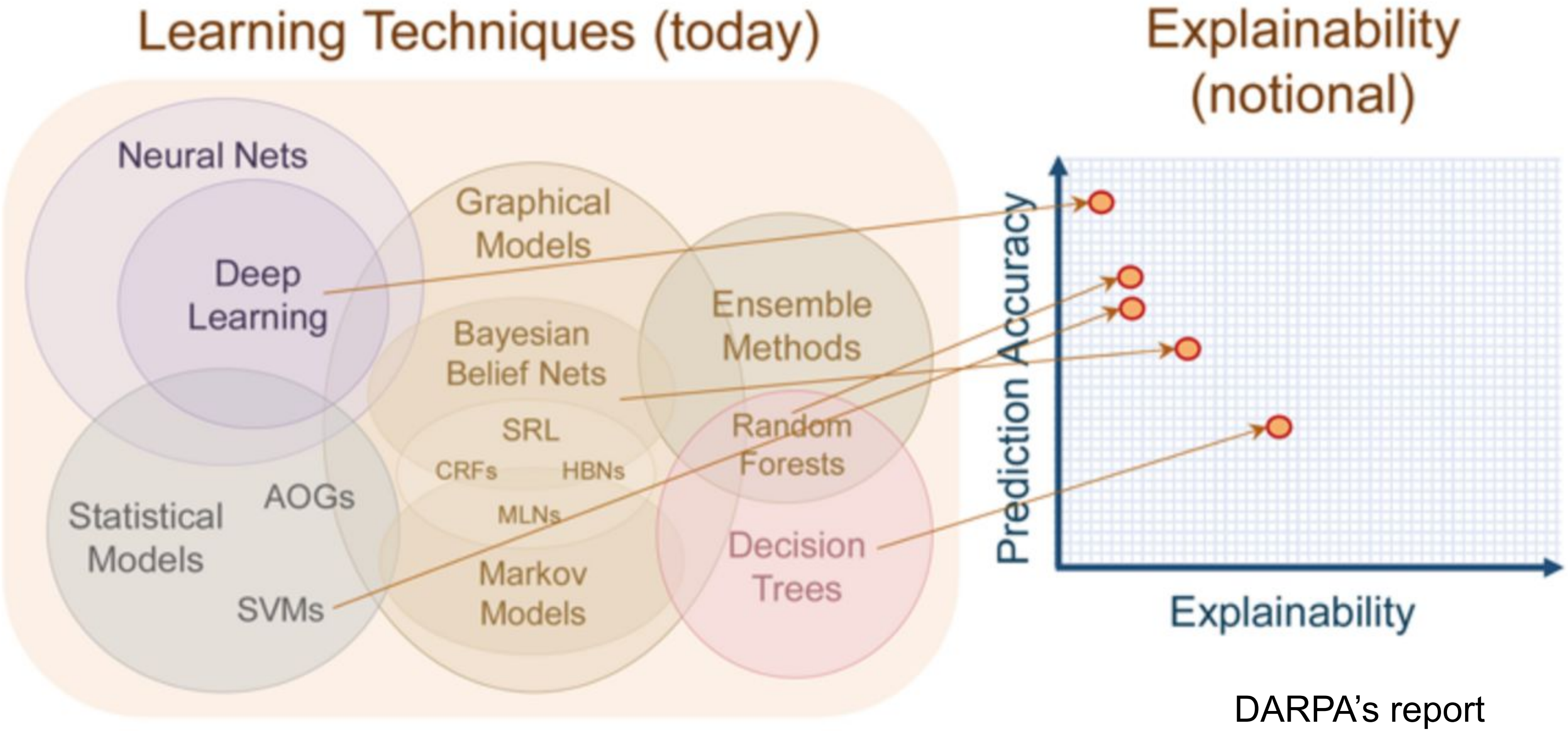
https://www.javatpoint.com/machine-learning-random-forest-algorithm

Complex models



For imagenet, they use 152 layers, which firstly achieved lower error rate compared to Humans in image recognition tasks.

Trade-off



DARPA's report

3. Interpretability Methods

Categorization of interpretability

- Self-Explaining
  - Directly interpretable
  - Generates the explanations at the same time as the prediction
  - Rule-based System, Decision Trees, Logistic Regression, Hidden Markov Model, etc.

- Post hoc:
  - Additional operation is performed after the predictions are made
  - Open-source packages: tf-keras-vis (gradient-based methods for deep learning), LIME, SHAP, etc

Categorization of interpretability

- Global
  - Explanation or justification by revealing how the model's predictive process works
  - What do you think pokemon looks like?
- Local:
  - Provide information or justification for the model's prediction on a specific input
  - Why do you think this image is pokemon?

Post-Hoc

Perform additional operations to explain the entire model's predictive reasoning

Explain a single prediction by performing additional operations (after the model has made the prediction)

Global

Local

Use the predictive model itself to explain the entire model's predictive reasoning (directly interpretable model)

Explain a single prediction using the model itself (calculated from information made available from the model as part of making the prediction)

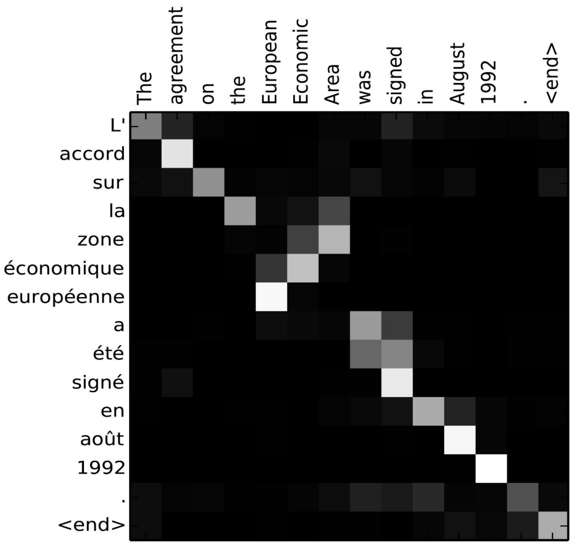
Self-Explaining

Example-driven

- Reasoning with examples
  - Explain the prediction of an input instance by identifying and presenting other instances
    - Eg. patient A has a tumor because he is similar to these k other data points with tumors
- Similar to nearest neighbor-based approaches

Feature importance

- Derive explanation by investigating the importance scores of different features used to output the final prediction
- It can be computed from
  - Attention Layer Approach
  - Gradient-based Saliency Approach



https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Gradient-based method

- Explain the decision made by the model
  - Eg, Why do you think this image is pokemon not digimon?

- Motivation: we want to know the contribution of each **<u>component/feature</u>** in the input data for prediction

Pixel, Segment in Images

Word in text

This is BT5153

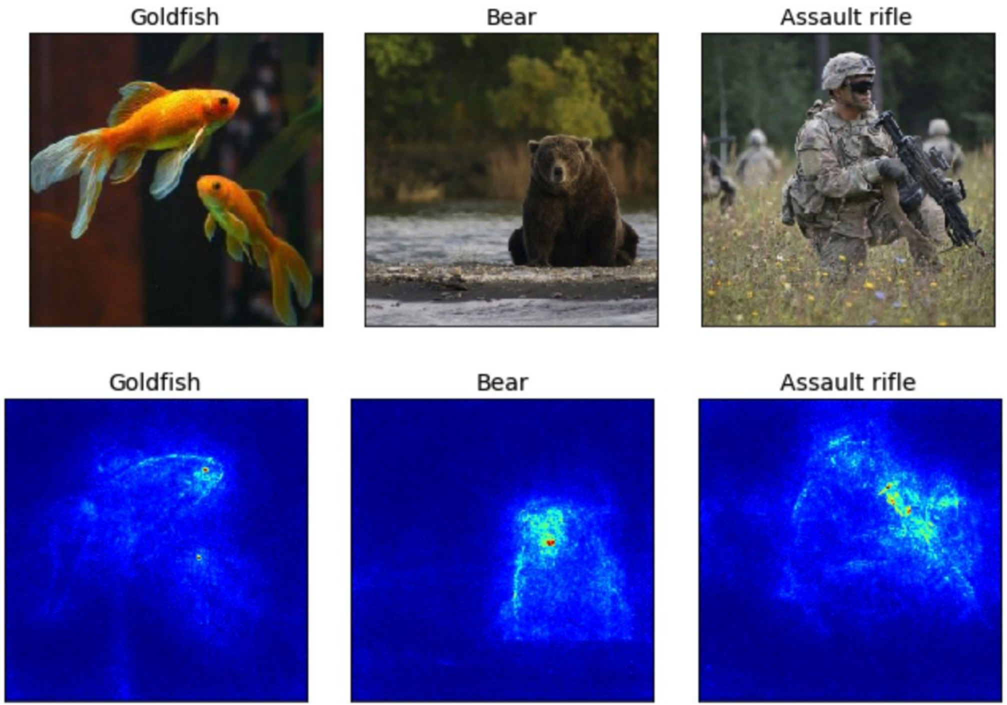
- Solution: Removing or modifying the partial parts of the components, observing the change of decision.

Saliency map

$$\{x_1, \ldots, x_i, \ldots, x_n\} \qquad \{x_1, \ldots, x_i + \Delta x, \ldots, x_n\}$$

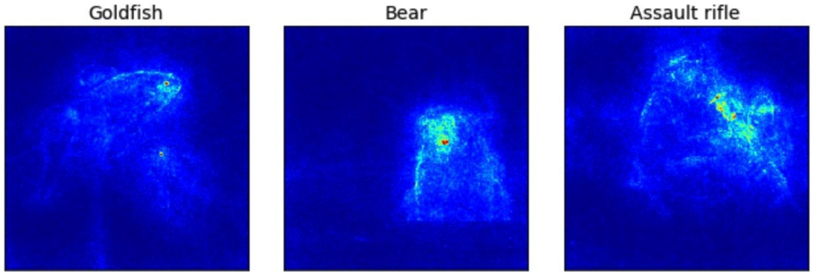
$$y_k \qquad y_k + \Delta y$$

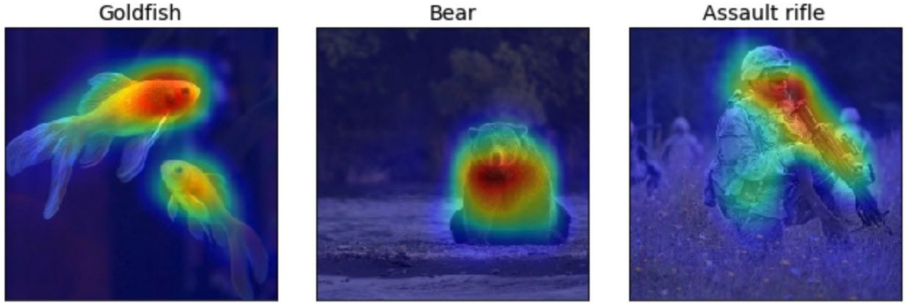
$$|\frac{\Delta y}{\Delta x}| \qquad |\frac{\partial y}{\partial x}|$$



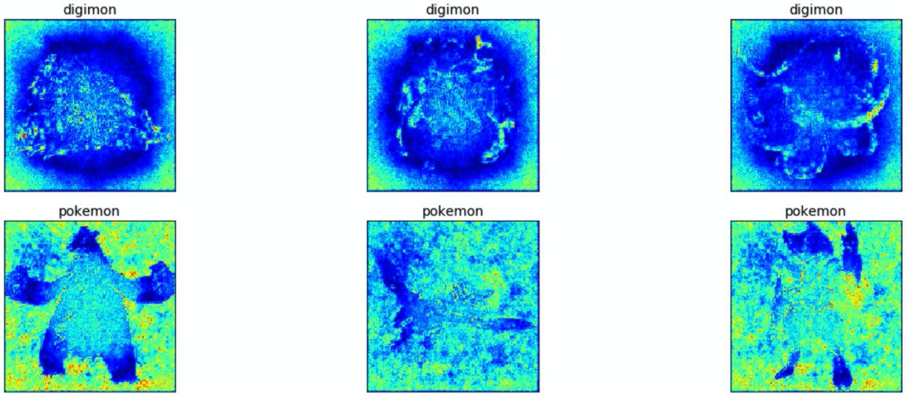
Andrea Vedaldi, Andrew Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR, 2014

Saliency map

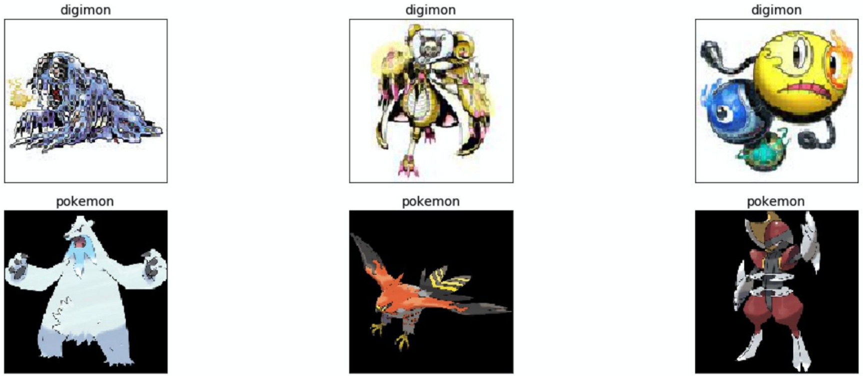




Pokemon vs Digimon



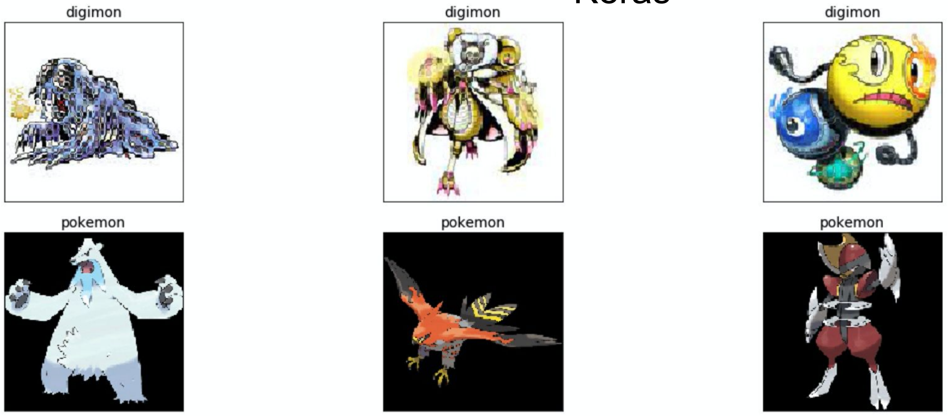
Saliency map



Input image

Pokemon vs Digimon

Loaded by Keras



CNN only learns to classify pokemon and digimon based on background colors.

PNG all appear in full black background

https://stackoverflow.com/questions/55720576/keras-loaded-png-appears-full-black

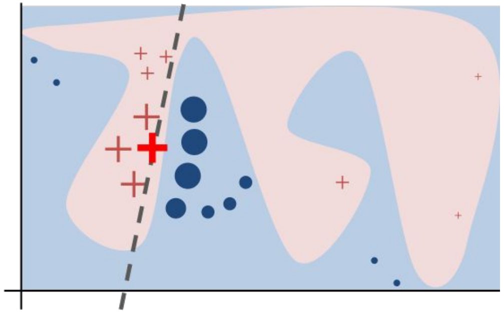
Do not celebrate when your model accuracy/AUC is over 99%

Surrogate model

- Model predictions are explained by learning a second, usually more explainable model, as a proxy
- Model-agnostic (applicable for any machine learning models)l prediction
- The learned surrogate models and the original models may have completely different mechanisms to make predictions

Surrogate model: local explanations

- Hard to explain a complex model in its entirety
    - How about explaining smaller regions?
    - Explain decisions of any model in a local region around a particular point
    - Learns sparse linear model

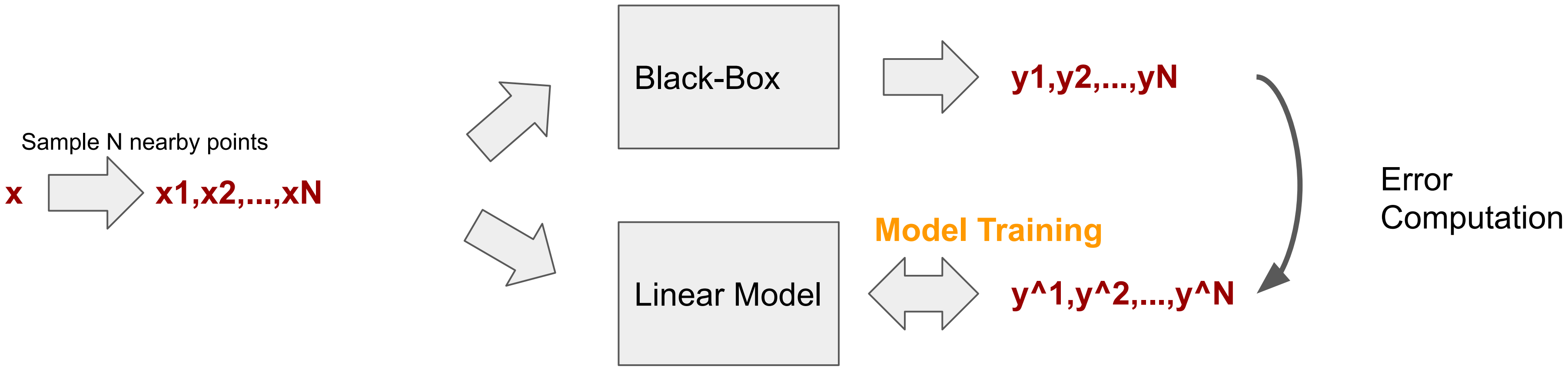


LIME (Ribeiro et. al)

Linear model can not mimic neural networks..but it may mimic a local region

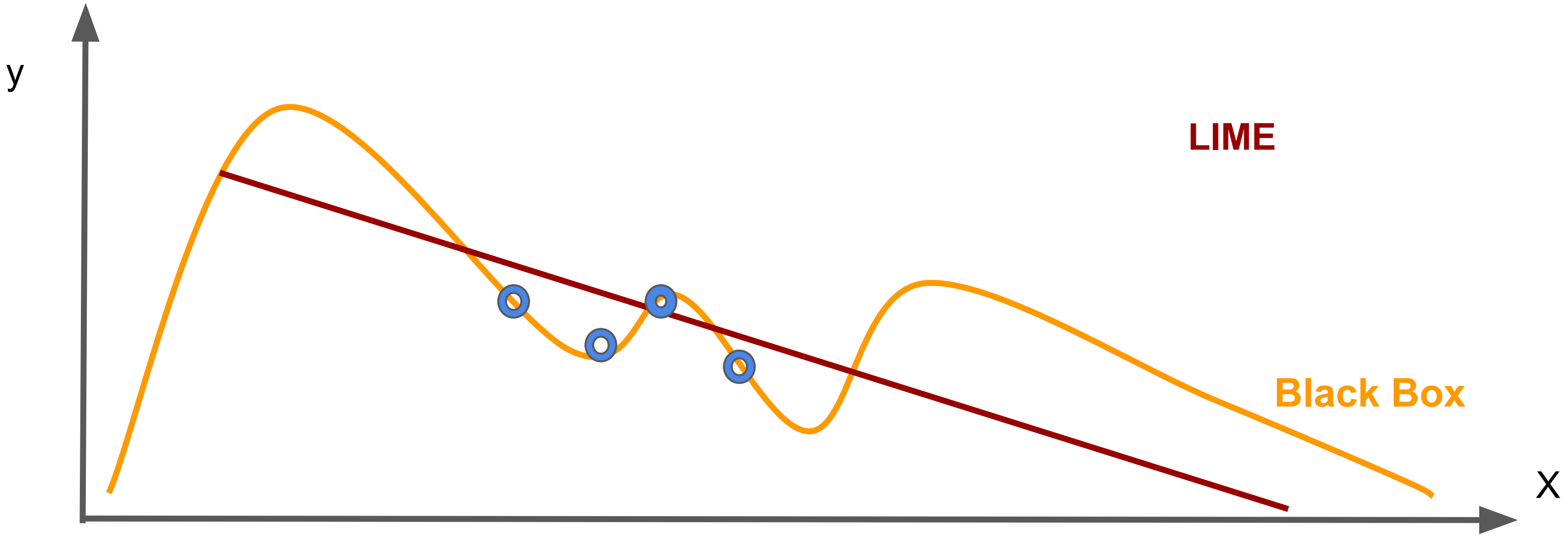
Surrogate model: local explanations

- Interpretable model can be used to mimic the actions of an complex model



Local interpretable model-agnostic explanations

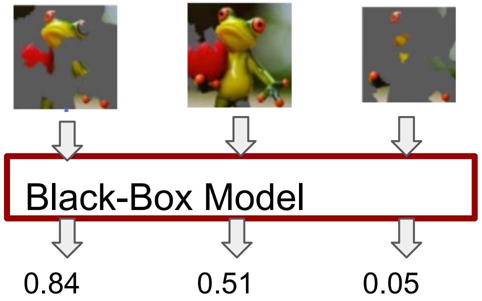
- Given a data point you want to explain
- Sample at the nearby
- Fit with linear model (or other interpretable models)
- Interpret the linear model



LIME on image

- Given a data point you want to explain

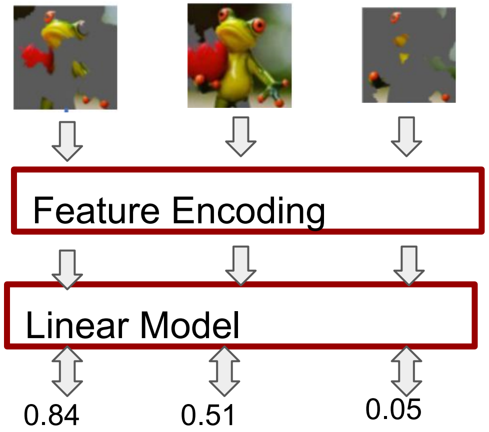
- Sample at the nearby
  - Each image is represented as a set of superpixels (segments)
  - Randomly delete some segments



Compute the probability of “frog” by black box

LIME on image

- Fit with linear model



Indicating whether the segment is removed or not

$$x_1^i \ldots x_m^i \ldots x_M^i$$

Indice of nearby data samples

The number of segments

$$x_m^i = \begin{cases} 0 & \text{if segment m in sample i is deleted} \\ 1 & \text{if segment m in sample i exists} \end{cases}$$

LIME on image

- Interpret the linear model

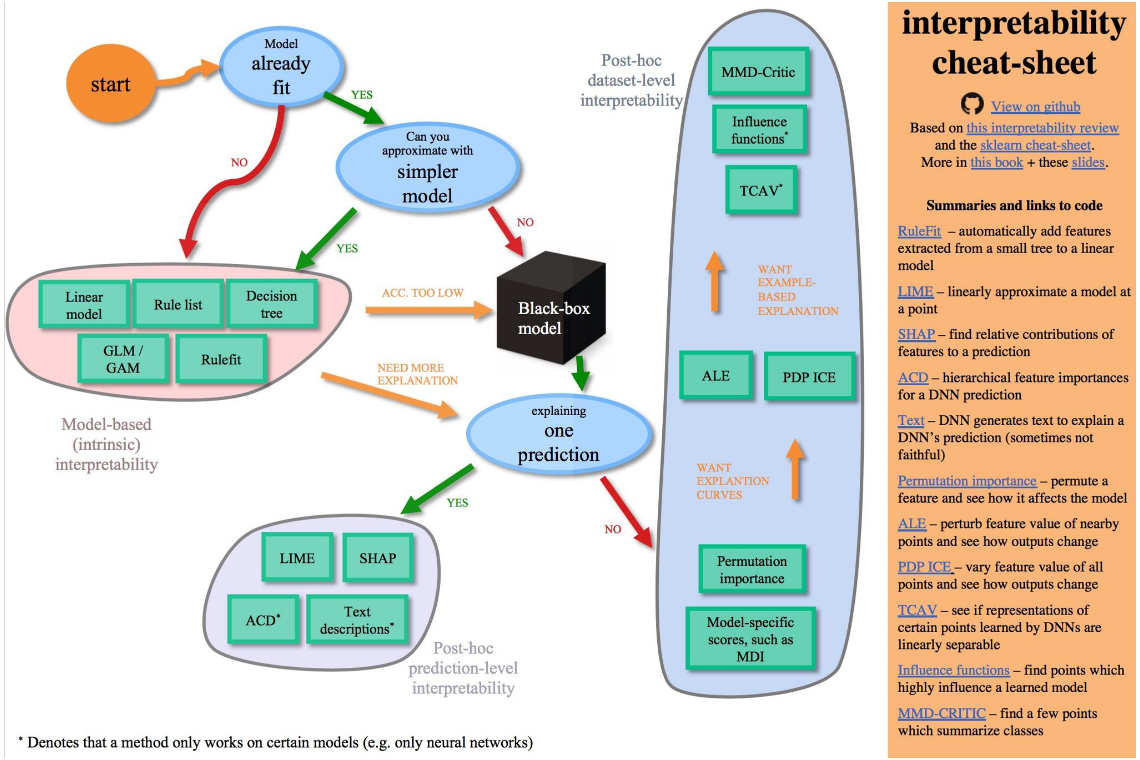
$$y = w_1 x_1 + \cdots + w_m x_m + \cdots + w_M x_M$$

$$x_m^i = \begin{cases} 0 & \text{if segment m in sample i is deleted} \\ 1 & \text{if segment m in sample i exists} \end{cases}$$

$$w_m \approx 0$$

$$w_m > 0$$

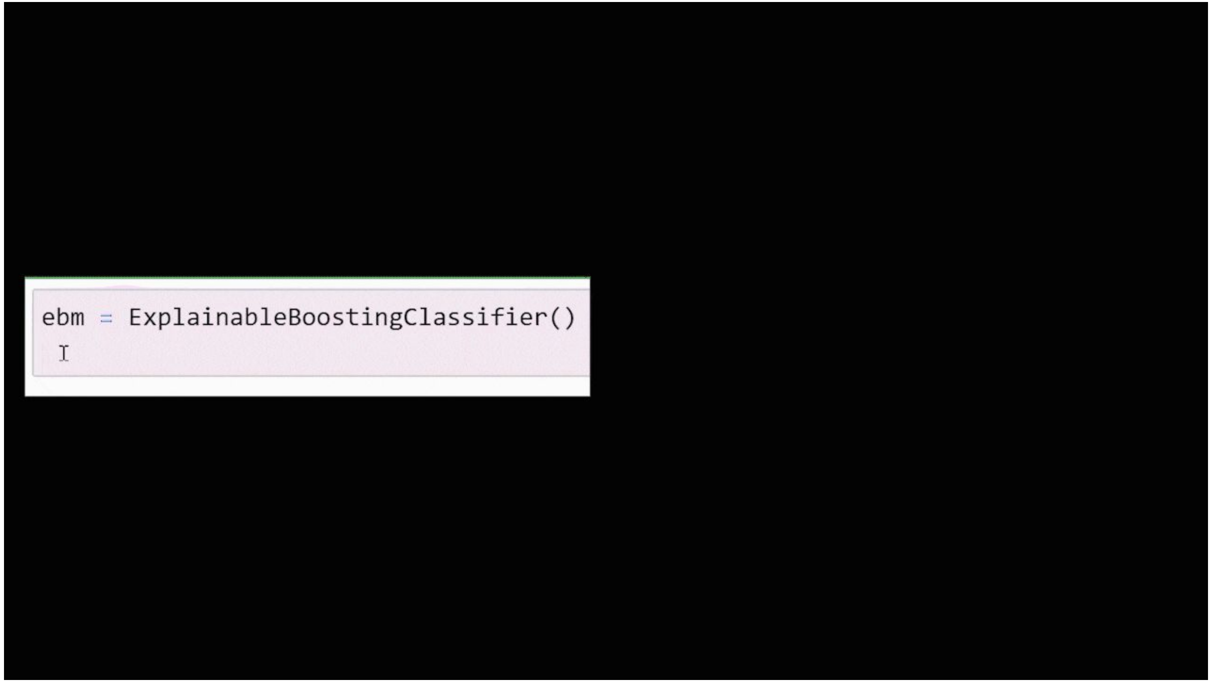
$$w_m < 0$$



Source: https://github.com/csinva/csinva.github.io/blob/master/_notes/cheat_sheets/interp.pdf

Interpretable Machine Learning:
https://christophm.github.io/interpretable-ml-book/

Interpretable Machine Learning Toolkit: **https://github.com/interpretml/interpret**



4. Feature Evaluation

Don't dump too many uncorrelated and unnecessary features.
Reduce the feature set to essentials.

Evaluation features’ importance

How much the model performance deteriorates
if a feature or a set of features containing that feature
is removed from the model?

Don't toss too many uncorrelated and unnecessary features. Reduce, reduce, reduce down to the essentials!

XGBoost’s feature importance

- XGBoost use boosting to combine weak learners to make accurate predictions.
- Feature importance for XGBoost could be checked in the following methods:
  - Build-in function
  - Permutation method
  - SHAP method

Boston dataset

Miscellaneous Details

- **Origin**
  The origin of the boston housing data is **Natural**.
- **Usage**
  This dataset may be used for **Assessment**.
- **Number of Cases**
  The dataset contains a total of **506** cases.
- **Order**
  The order of the cases is **mysterious**.
- **Variables**
  There are **14** attributes in each case of the dataset. They are:

  1. CRIM - per capita crime rate by town
  2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
  3. INDUS - proportion of non-retail business acres per town.
  4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
  5. NOX - nitric oxides concentration (parts per 10 million)
  6. RM - average number of rooms per dwelling
  7. AGE - proportion of owner-occupied units built prior to 1940
  8. DIS - weighted distances to five Boston employment centres
  9. RAD - index of accessibility to radial highways
  10. TAX - full-value property-tax rate per $10,000
  11. PTRATIO - pupil-teacher ratio by town
  12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
  13. LSTAT - % lower status of the population
  14. MEDV - Median value of owner-occupied homes in $1000's

Housing price prediction

Source: https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

XGBoost built-in function

- XGBoost

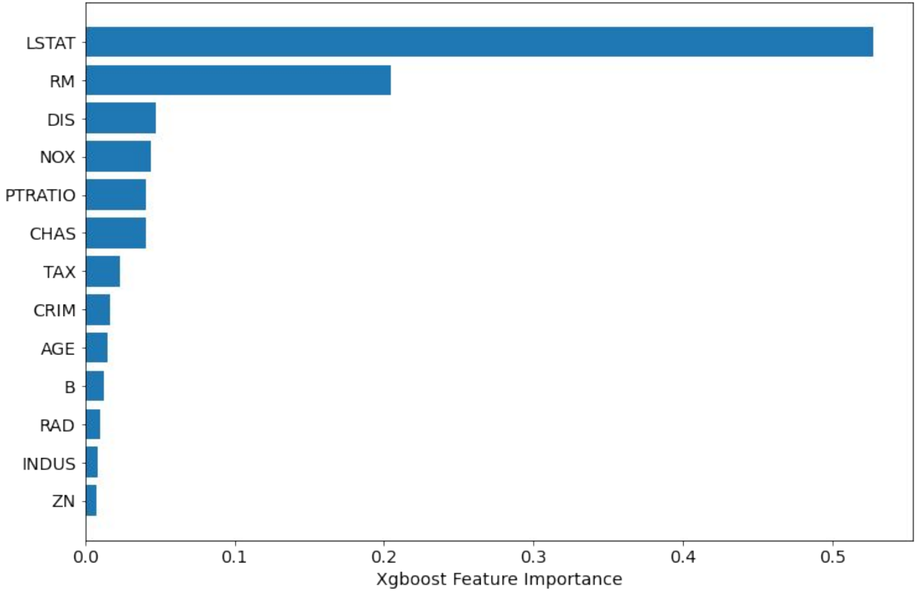
`get_score`(*fmap='', importance_type='weight'*)

Get feature importance of each feature. For tree model Importance type can be defined as:

- 'weight': the number of times a feature is used to split the data across all trees.
- 'gain': the average gain across all splits the feature is used in.
- 'cover': the average coverage across all splits the feature is used in.
- 'total_gain': the total gain across all splits the feature is used in.
- 'total_cover': the total coverage across all splits the feature is used in.

Xgboost feature importance

- gain:



Permutation based feature importance

- Randomly shuffle each feature and compare the model's performance change. The most important feature impact the performance the most
- Well-supported in sklearn
- A bit slow!

sklearn.inspection.permutation_importance

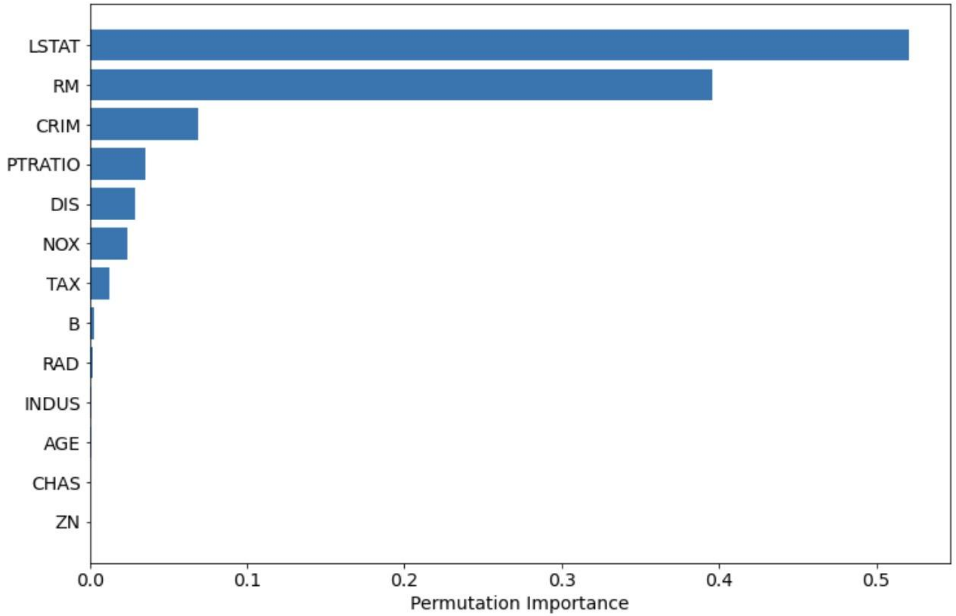
sklearn.inspection.permutation_importance(*estimator*, *X*, *y*, *, *scoring=None*, *n_repeats=5*, *n_jobs=None*, *random_state=None*, *sample_weight=None*, *max_samples=1.0*) [source]

Permutation importance for feature evaluation [BRE].

The estimator is required to be a fitted estimator. `X` can be the data set used to train the estimator or a hold-out set. The permutation importance of a feature is calculated as follows. First, a baseline metric, defined by scoring, is evaluated on a (potentially different) dataset defined by the `X`. Next, a feature column from the validation set is permuted and the metric is evaluated again. The permutation importance is defined to be the difference between the baseline metric and metric from permutating the feature column.

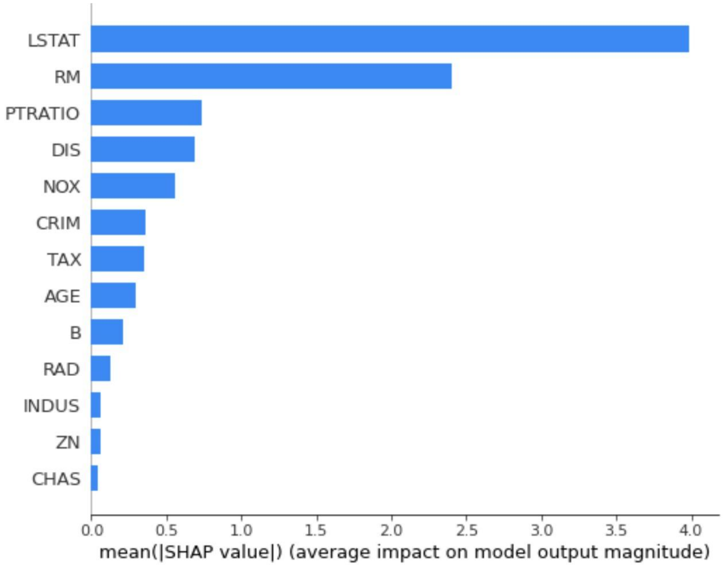
Read more in the User Guide.

Permutation importance



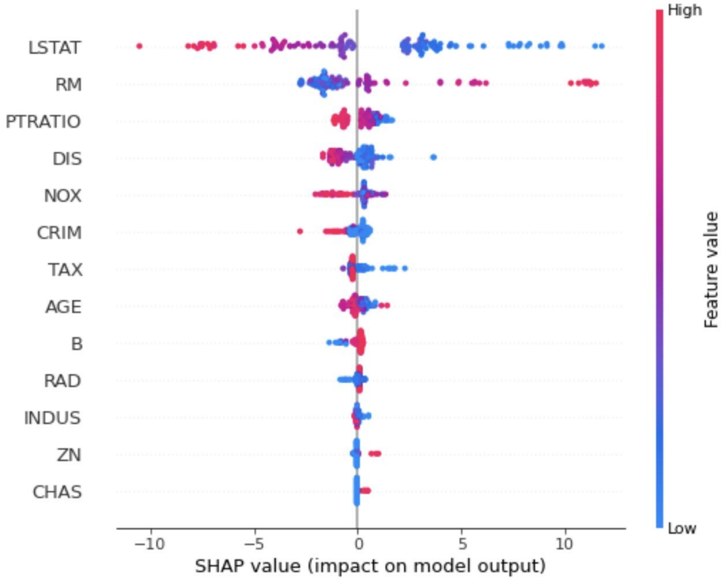
SHAP: SHapley Additive exPlanations

- Similar to LIME, it is also model-agnostic and compute the shapley value based on game theory to measure the contribution from each feature
- Also slow



SHAP over entire model

- Measuring the feature importance to the entire model



SHAP over a single prediction

- Measuring he feature importance to a single prediction