Machine Learning Practice (Part 2)

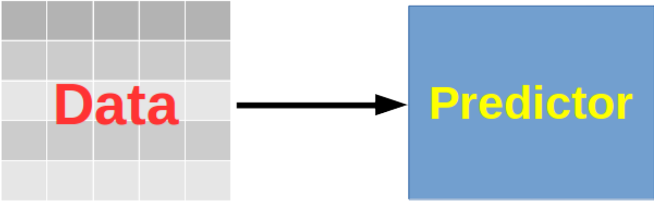
Zhirong Yang

Outline

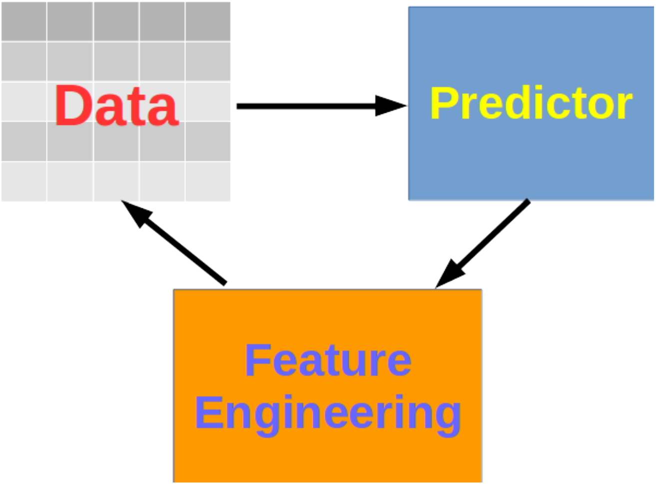
- Feature engineering
- Model interpretation
- Automatic machine learning

At the beginning

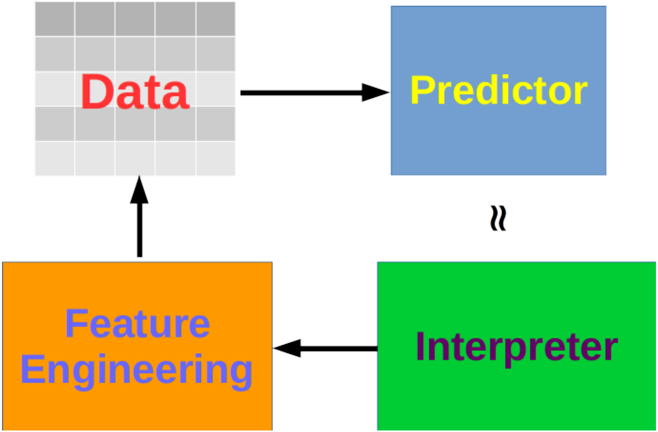
The data is in raw features



Feature engineering helps prediction



Feature engineering also helps interpretation



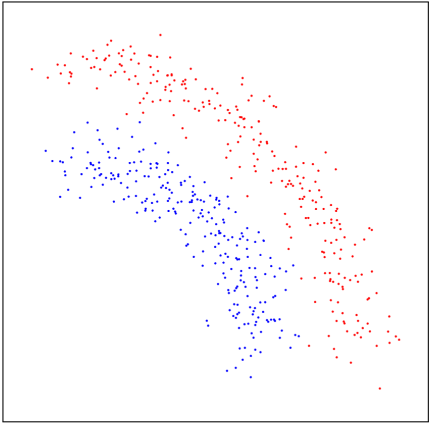
Feature Engineering

- Feature selection
  - Delete some existing columns
  - Can be done by using feature importance
- Feature generation (feature extraction)
  - Add some new columns
    - Generate new attributes using existing ones for each row
    - Generate new attributes across multiple rows

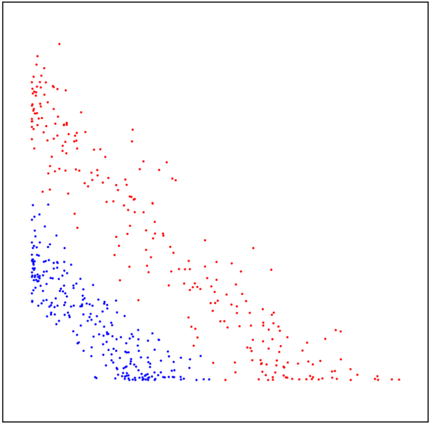
Feature Engineering

- Feature selection
  - Delete some existing columns
  - Can be done by using feature importance
- Feature generation (feature extraction)
  - Add some new columns
    - Generate new attributes using existing ones for each row
    - Generate new attributes across multiple rows
- Automatic feature engineering is an open problem!
  - The top conference ICLR mainly addresses this

Why new features can help? Example 1

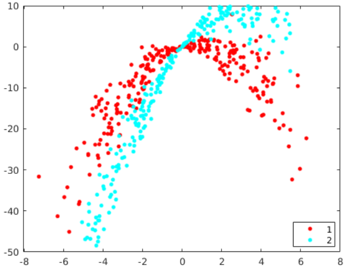


x_1 vs x_2

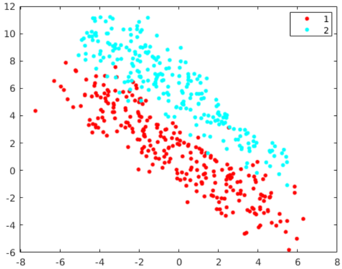


x_1^2 vs x_2^2

Why new features can help? Example 2



x_1 vs x_2



x_1 vs $\frac{x_2}{x_1}$

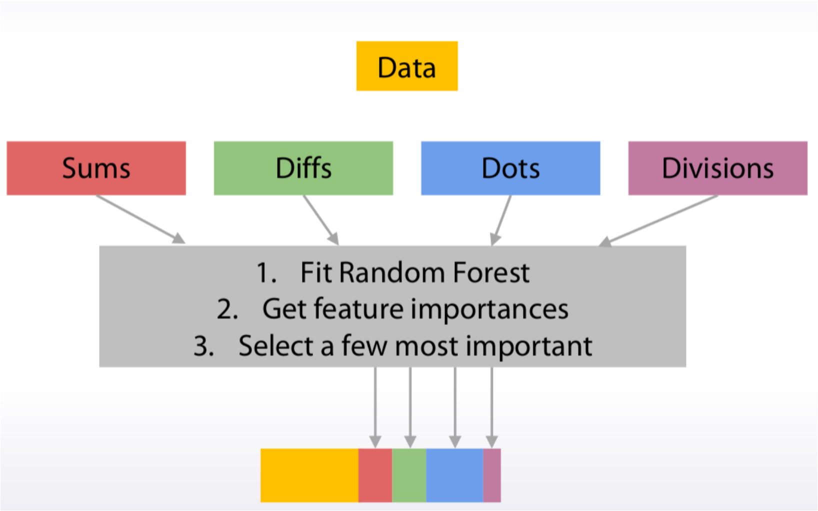
Row-wise feature generation

- Individual transform, e.g.
  - x^2
  - $\log(1 + x)$
- Pairwise transform, e.g.
  - $x + y$
  - $x - y$
  - $x. * y$
  - $x./y$
- Sometimes involve larger groups
  - mean, std, min, max, percentile, etc.

Practical notes

- We have a lot of possible interactions — d^2 for d features.
- Even more if use several types in interactions
- Need to reduce number of features
  - Group existing features before generation
  - Feature selection after generation

Example of interaction generation pipeline



High-order linear interactions

- Principal Component Analysis (PCA)
  - Finds uncorrelated components
- Canonical Correlation Analysis (CCA)
  - Finds correlated components across multiple sources

Principal Component Analysis (PCA)

- We have seen PCA in the Unsupervised Learning lecture
- Do PCA in Python

  Wrong way

```
pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.fit_transform(X_test)
```

  Right way

```
X_all = np.concatenate([X_train, X_test])
pca.fit(X_all)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

Canonical Correlation Analysis (CCA)

- Multiple aspects of the same data object
  - e.g. article (text, author), webpage (text, hyperlinks), TV news (vision, speech, scripts), song (sound, lyrics)

Canonical Correlation Analysis (CCA)

- Multiple aspects of the same data object
  - e.g. article (text, author), webpage (text, hyperlinks), TV news (vision, speech, scripts), song (sound, lyrics)
- CCA seeks linear transforms such that correlation is maximized in the common subspace
  - X and Y are random (vector) variables

$$(a', b') = \underset{a,b}{\operatorname{argmax}} \operatorname{corr}\left(a^T X, b^T Y\right)$$

Canonical Correlation Analysis (CCA)

- PCA is implemented by eigendecomposition

$$\Sigma w = \lambda w$$

- CCA is implemented by generalized eigendecomposition

$$\begin{pmatrix} 0 & \Sigma_{XY} \\ \Sigma_{YX} & 0 \end{pmatrix} \begin{pmatrix} w_X \\ w_Y \end{pmatrix} = \lambda \begin{pmatrix} \Sigma_{XX} & 0 \\ 0 & \Sigma_{YY} \end{pmatrix} \begin{pmatrix} w_X \\ w_Y \end{pmatrix}$$

Canonical Correlation Analysis (CCA)

- PCA is implemented by eigendecomposition

$$\Sigma w = \lambda w$$

- CCA is implemented by generalized eigendecomposition

$$\begin{pmatrix} 0 & \Sigma_{XY} \\ \Sigma_{YX} & 0 \end{pmatrix} \begin{pmatrix} w_X \\ w_Y \end{pmatrix} = \lambda \begin{pmatrix} \Sigma_{XX} & 0 \\ 0 & \Sigma_{YY} \end{pmatrix} \begin{pmatrix} w_X \\ w_Y \end{pmatrix}$$

- In Scikit-Learn, example:

```
from sklearn.cross_decomposition import CCA
X = [[0., 0., 1.], [1.,0.,0.], [2.,2.,2.], [3.,5.,4.]]
Y = [[0.1, -0.2], [0.9, 1.1], [6.2, 5.9], [11.9, 12.3]]
cca = CCA(n_components=1)
cca.fit(X, Y)
X_c, Y_c = cca.transform(X, Y)
```

Cross-row feature generation: example

Original data

	User_id	Page_id	Ad_price	Ad_position
0	4	6	165.977125	Bottom_right
1	4	6	34.5395640	Bottom_right
2	4	6	29.1963786	Bottom_left
3	4	6	79.4373729	Bottom_left
4	4	6	290.534595	Bottom_right
5	4	6	314.412660	Bottom_right
6	4	6	138.9007639	Bottom_right
7	4	6	107.4711914	Bottom_right
8	4	6	242.1089786	Bottom_left
9	4	7	27.16719836	Bottom_left
10	4	7	413.5421978	Bottom_right

Cross-row feature generation: example

Augmented data

	User_id	Page_id	Ad_price	Ad_position	Max_price	min_price	Min_price _position
0	4	6	95.874252	Bottom_right	474.63772	73.711548	Bottom_left
1	4	6	215.751007	Bottom_right	474.63772	73.711548	Bottom_left
2	4	6	474.637726	Bottom_left	474.63772	73.711548	Bottom_left
3	4	6	73.711548	Bottom_left	474.63772	73.711548	Bottom_left
4	4	6	79.288841	Bottom_right	474.63772	73.711548	Bottom_left
5	4	6	271.391785	Bottom_right	474.63772	73.711548	Bottom_left
6	4	6	296.529053	Bottom_right	474.63772	73.711548	Bottom_left
7	4	6	96.030029	Bottom_right	474.63772	73.711548	Bottom_left
8	4	6	130.175064	Bottom_left	474.63772	73.711548	Bottom_left
9	4	7	35.465202	Bottom_left	121.54219	35.465202	Bottom_left
10	4	7	121.542191	Bottom_right	121.54219	35.465202	Bottom_left

Cross-row feature generation: example

In [22]:
```
gb = df.groupby(['user_id','page_id'],as_index=False).agg(
     {'ad_price':{'max_price':np.max, 'min_price':np.min}})
gb.columns = ['user_id','page_id','min_price','max_price']
gb
```

Out[22]:

	user_id	page_id	min_price	max_price
0	4	6	73.711548	474.637726
1	4	7	35.465202	121.542191

In [23]:
```
df = pd.merge(df,gb,how='left',on=['user_id','page_id'])
```

More features

- How many pages user visited
- Standard deviation of prices
- Most visited page
- etc.

Local features

- Many datasets contain time or spatial information
- Various temporal and/or spatial features
  - Define windows (neighborhood) at various scales
  - Calculate statistics within the windows (neighborhoods)

Example: KNN features for `Springleaf` data set

- For every point, find 2000 nearest neighbors using Bray-Curtis metric
$$d(u, v) = \sum |u_i - v_i| \,/\, \sum |u_i + v_i|$$
- Calculate various features from those 2000 neighbors
  - Mean target of nearest 5, 10, 15, 500, 2000 neighbors
  - Mean distance to 10 closest neighbors
  - Mean distance to 10 closest neighbors with target 1
  - Mean distance to 10 closest neighbors with target 0

Automatic Machine Learning (AutoML)

Current ML pipeline

Repeat the following

- Clean & preprocess the data
- Select / engineer better features
- Select a model family
- Set the hyperparameters
- Construct ensembles of models
- ...

auto-sklearn example

```python
import autosklearn.classification as ac
cls = ac.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

auto-sklearn example

```python
import autosklearn.classification as ac
cls = ac.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

- AutoML greatly facilitates applying ML
  - No need to select among algorithms
  - No need to tune hyperparameters
  - etc.

auto-sklearn example

```
import autosklearn.classification as ac
cls = ac.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

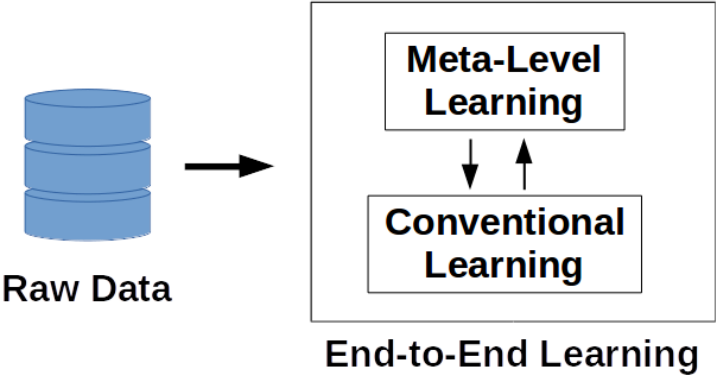
- AutoML greatly facilitates applying ML
  - No need to select among algorithms
  - No need to tune hyperparameters
  - etc.
- AutoML is the right form of ML

auto-sklearn example

```python
import autosklearn.classification as ac
cls = ac.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

- AutoML greatly facilitates applying ML
  - No need to select among algorithms
  - No need to tune hyperparameters
  - etc.
- AutoML is the right form of ML
- It will come to wide practice soon
- See AutoML.org for more information

AutoML: end-to-end learning



Current AutoML

- Hyperparameter Optimization
  - AutoML as Hyperparameter Optimization
  - Blackbox Optimization
  - Surrogate Optimization
- Neural Architecture Search

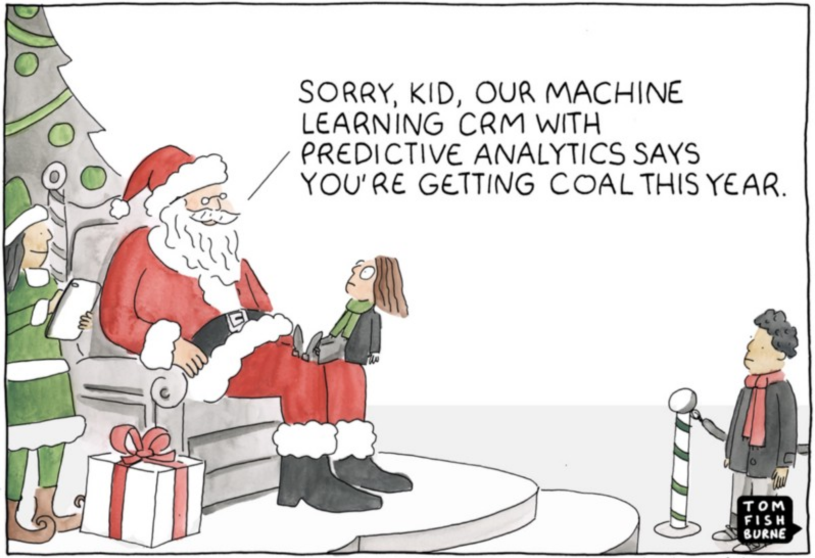
Some selected AutoML libraries

- auto-sklearn
- TPOT
- H2O
- AutoKeras
- Top AutoML Python libraries in 2022
- 10 Python Libraries For Automated Machine Learning That You Should Think To Use in 2023

A student solution using H2O in TDT05 2020

Model Interpretation

image source: www.explainxkcd.com


SORRY, KID, OUR MACHINE LEARNING CRM WITH PREDICTIVE ANALYTICS SAYS YOU'RE GETTING COAL THIS YEAR.
TOM FISH BURNE

Interpretability

Interpretability is the degree to which a human can

- understand the cause of a decision
- consistently predict the model's result.

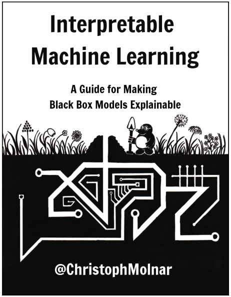
Interpretation is important to

- help in identifying the causes/factors
- find meaning in the world
- increase social acceptance
- enable debugging and auditing

Explanation

- An explanation usually relates the feature values of **an instance** to its model prediction in a humanly understandable way.
- An explanation is the answer to a why-question
  - Why did not the treatment work on the patient?
  - Why was my loan rejected?
  - Why have we not been contacted by alien life yet?

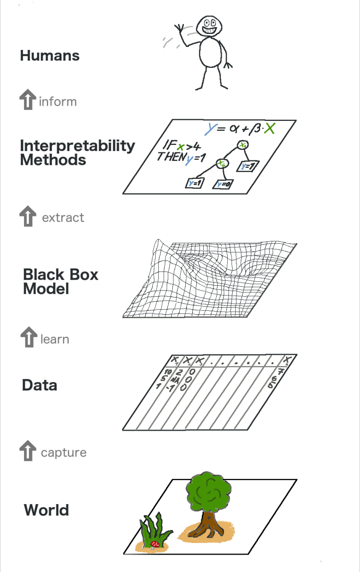
Additional Reading



https://christophm.github.io/interpretable-ml-book/

- Taxonomy of Interpretability Methods (Section 3.2)
- Scope of Interpretability (Section 3.3)
- Evaluation of Interpretability (Section 3.4)
- What is a Good explanation (Section 3.6.2)

Big picture of interpretable machine learning



Intrinsically interpretable models

In most cases, non-ML experts only understand the following

- Linear combination
  - Linear regression

$$y \approx \hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

  - Logistic regression (for classification)

$$y \approx \hat{y} = \mathrm{logit}(w_1 x_1 + w_2 x_2 + \cdots + w_d x_d)$$

  - Generalized linear regression

$$y \approx \hat{y} = f(w_1 x_1 + w_2 x_2 + \cdots + w_d x_d)$$

- IF-ELSE
  - decision tree
  - decision rule
- K-Nearest-Neighbors

Model-agnostic interpretation methods

- Direct relationship analysis
  - Partial Dependence Plot
  - Feature importance
- Surrogate methods
  - Global surrogate
  - Local surrogate

Partial Dependence Plot (PDP)

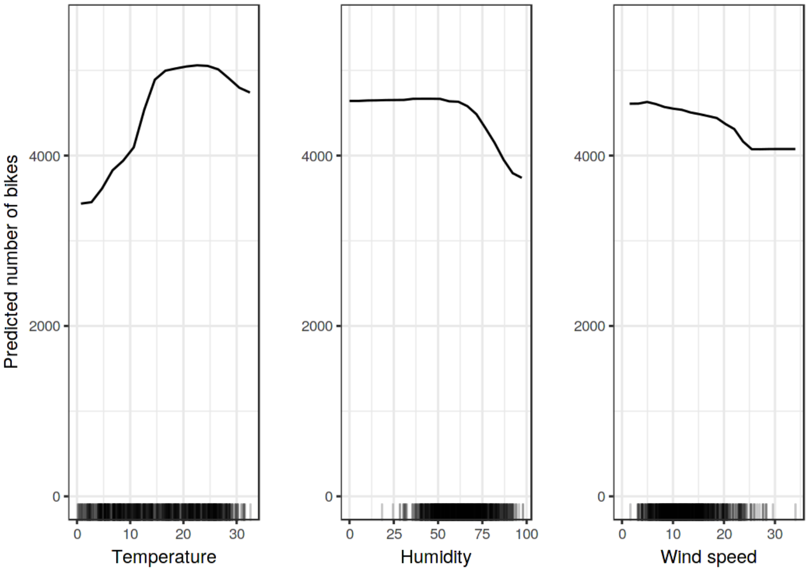
- PDP shows marginal effect which one or two features have on the predicted outcome of a machine learning model
- Definition

$$PD(x_S) = \hat{f}_{x_S}(x_S) = \mathbb{E}_{x_C}\left[\hat{f}(x_S, x_C)\right] = \int \hat{f}(x_S, x_C)\, d\mathbb{P}(x_C),$$

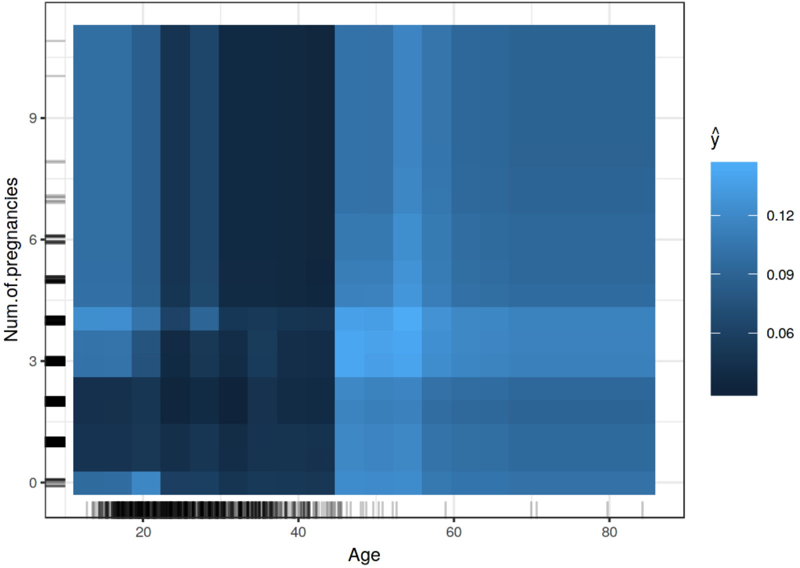
  where $\hat{f}$ is the predictor, x_S are the features to examine, and x_C are the other features.
- In practice

$$\hat{f}_{x_S}(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}\left(x_S, x_C^{(i)}\right)$$

PDP example for one feature (for bike data set)



PDP example for two features



PDP in sklearn

https://scikit-learn.org/stable/modules/partial_dependence.html

Feature Importance

- Feature importance is a major way for interpretation
- Linear model: |coefficient|
- Tree/forest: number of times for splitting

Feature Importance

- Feature importance is a major way for interpretation
- Linear model: $|\text{coefficient}|$
- Tree/forest: number of times for splitting
- Feature importance for black-box models?

Feature Importance

- Feature importance is a major way for interpretation
- Linear model: |coefficient|
- Tree/forest: number of times for splitting
- Feature importance for black-box models?
- Permutation Feature Importance
  - A feature is “important” if shuffling its values increases the model error
  - A feature is “unimportant” if shuffling its values leaves the model error unchanged

Permutation feature importance algorithm

Input: Trained model f, feature matrix X, target vector y, error measure $L(y, f)$.

1. Estimate the original model error $e^{\text{orig}} = L(y, f(X))$ (e.g. mean squared error)
2. For each feature $j = 1, ..., p$ do:
   - Generate feature matrix X^{perm} by permuting feature j in the data X. This breaks the association between feature j and true outcome y.
   - Estimate error $e^{\text{perm}} = L(y, f(X^{\text{perm}}))$ based on the predictions of the permuted data.
   - Calculate permutation feature importance $\text{FI}^j = e^{\text{perm}}/e^{\text{orig}}$. Alternatively, the difference can be used: $\text{FI}^j = e^{\text{perm}} - e^{\text{orig}}$
3. Sort features by descending FI.

Permutation feature importance algorithm (in practice)

- An economic implementation
  1. divide the data set in two halves
  2. swap the values of feature j of the two halves
- If you want a more accurate estimate (expensive)
  1. pair the instances
  2. swap the values of feature j of each pair
  3. resulting $n(n-1)$ estimates of the permutation error

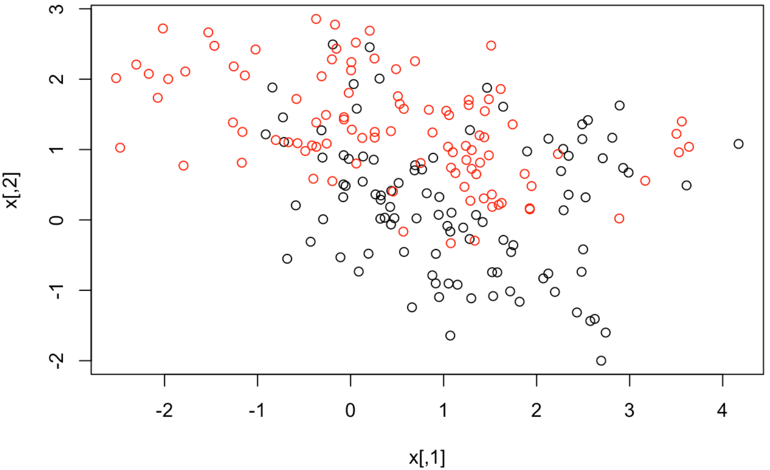
Permutation feature importance in sklearn

https://scikit-learn.org/stable/modules/permutation_importance.html

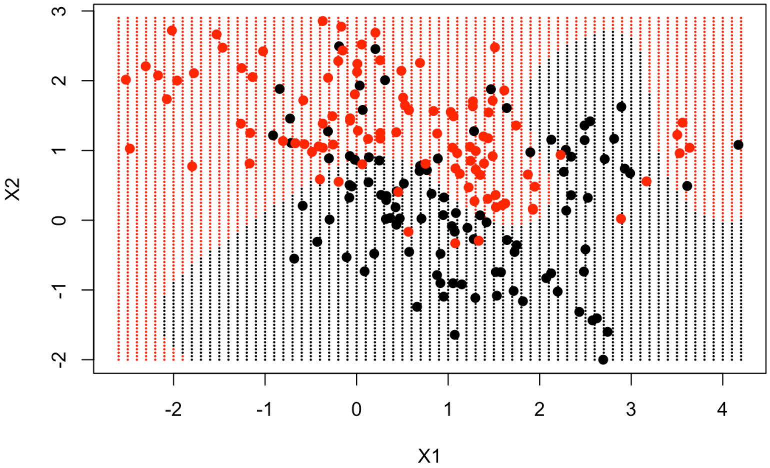
Global surrogate

- A global surrogate model is an interpretable model that is trained to approximate the predictions of a black box model.
- Steps to obtain a surrogate model:
  1. For a dataset X, get the predictions of the black box model.
  2. Select an interpretable model type (linear model, decision tree, etc).
  3. Train the interpretable model on the dataset X and its predictions. This is called the surrogate model.
  4. Measure how well the surrogate model replicates the predictions of the black box model.
  5. Interpret the surrogate model.

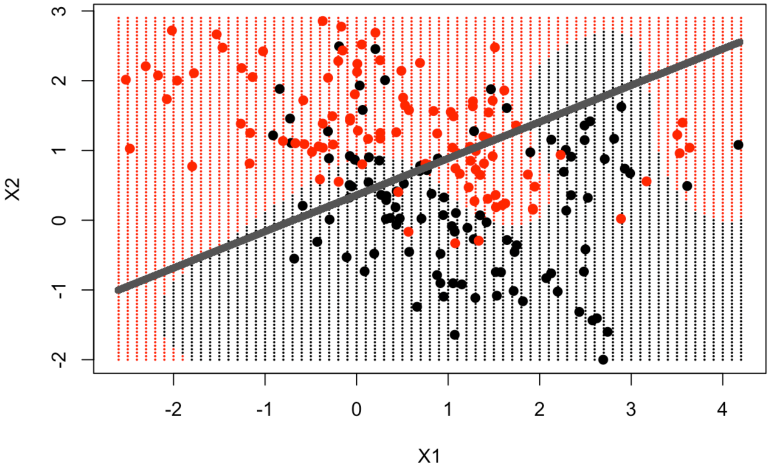
Global surrogate example



Global surrogate example



Global surrogate example



Local surrogate

- global surrogate
  - explain all instances as a whole
  - e.g. linear or decision tree
  - intuitive and straightforward
  - but tend to underfit
- local surrogate
  - explain individual instances
  - e.g. locally linear or a local decision tree

LIME objective

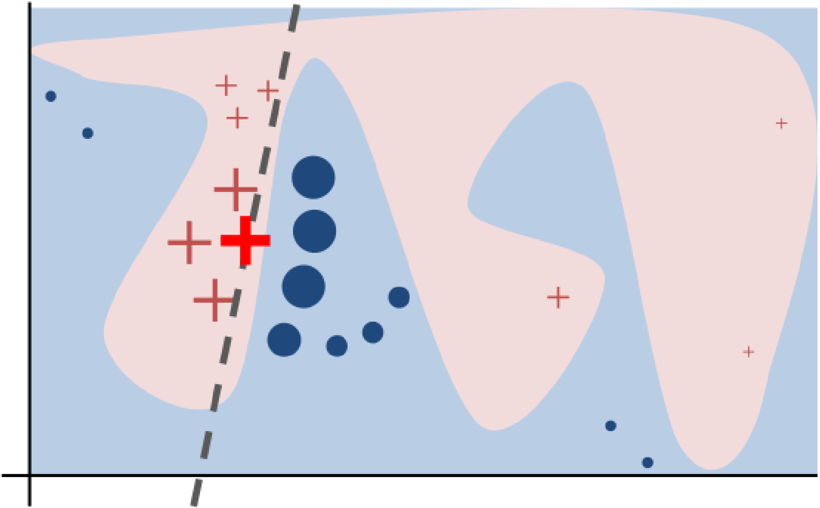
$$\text{explanation}(x) = \arg\min_{g \in G} L\left(f, g, \pi_x\right) + \Omega(g)$$

- x an instance
- L a loss function
- f the black-box model
- g the interpretable surrogate model
- π_x proximity measure defining "local"
- $\Omega(g)$ complexity of g

LIME recipe

1. Select x for which you want to have an explanation of its black box prediction.
2. Perturb the dataset and get the black box predictions for these new points.
3. Weigh the new samples according to their proximity to the instance of interest.
4. Train an interpretable model on the weighted and perturbed dataset.
5. Explain the prediction by interpreting the local model

Locally linear surrogate



Specify complexity Ω

- A good g is 1) close to f and 2) with low complexity
  - For linear g: a few coefficients
  - For tree g: a few levels and regular leaf values

Specify complexity Ω

- A good g is 1) close to f and 2) with low complexity
  - For linear g: a few coefficients
  - For tree g: a few levels and regular leaf values
- LIME originally uses locally linear g
  - i.e. locally, use the black-box predictions as supervised labels to fit a linear model with a few coefficients

K-LASSO

- LASSO = least absolute shrinkage and selection operator
- LASSO is one of the most widely used sparse linear model
- LASSO learning objective

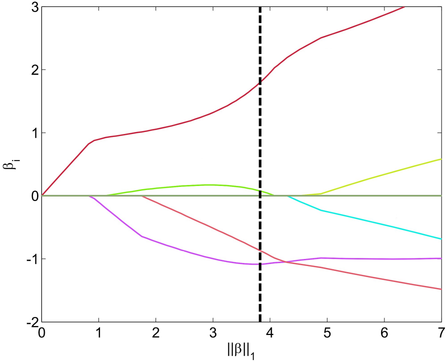
$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \leq \rho$$

- A smaller ρ results in a sparser model (more zero coefficients)
- K-LASSO is LASSO with exactly K nonzero coefficients

LASSO regularization path

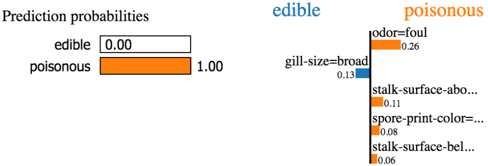
$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \leq \rho$$

- ρ starts from 0 and steadily increases
- record each coefficient (as curves)
- K-LASSO chooses the best β with only K nonzero β_i



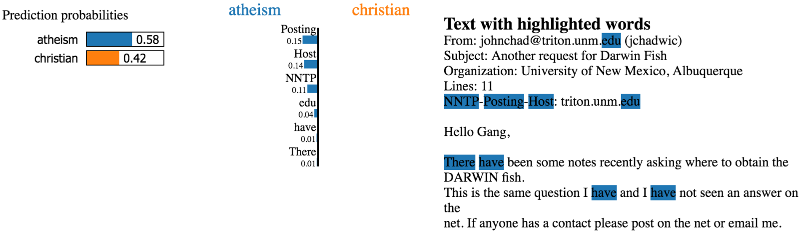
LIME example (tabular data)

Presenting the K-LASSO model



Feature	Value
odor=foul	True
gill-size=broad	True
stalk-surface-above-ring=silky	True
spore-print-color=chocolate	True
stalk-surface-below-ring=silky	True

LIME example (text data)



LIME presentation (image data)

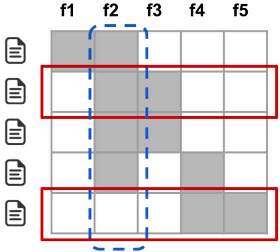
Explaining prediction of 'Cat' in pros and cons

Tutorial of LIME usage

Additional reading: selecting instances for interpretation

- Explaining a single instance reveals one aspect (with a few features) of the model
- Better to understand the model from different aspects
- Given a budget (e.g. patience), which instances to show?
- Submodular pick—try to cover as many important features as possible

Require: Instances X, Budget B
  for all $x_i \in X$ **do**
    $\mathcal{W}_i \leftarrow \mathbf{explain}(x_i, x'_i)$
  end for
  for $j \in \{1 \ldots d'\}$ **do**
    $I_j \leftarrow \sqrt{\sum_{i=1}^{n} |\mathcal{W}_{ij}|}$
  end for
  $V \leftarrow \{\}$
  while $|V| < B$ **do**
    $V \leftarrow V \cup \text{argmax}_i\, c(V \cup \{i\}, \mathcal{W}, I)$
  end while
  return V



I_j: importance of feature j

Coverage function $c(V, \mathcal{W}, I) = \sum_{j=1}^{d'} \mathbb{1}_{[\exists i \in V : |\mathcal{W}_{ij}| > 0]} I_j$

Additional reading: remaining challenges in LIME

- How to specify "local"?
- How to go beyond raw features?

Define "local"

- Typical choice is a Guassian kernel $w_t \propto \exp\left(-\frac{\|x-x_t\|^2}{2\sigma^2}\right)$
- How to set σ is an open problem
  - In LIME software $\sigma = 0.75\sqrt{\#\text{features}}$
  - The above default setting may not work; use with caution!
- By default Euclidean distance is used for $\|x - x_t\|$
  - Sometimes works, but not always!
  - For some data, no simple distance function works (e.g. for natural images)

Define "local"

- Typical choice is a Guassian kernel $w_t \propto \exp\left(-\frac{\|x-x_t\|^2}{2\sigma^2}\right)$
- How to set σ is an open problem
  - In LIME software $\sigma = 0.75\sqrt{\#\text{features}}$
  - The above default setting may not work; use with caution!
- By default Euclidean distance is used for $\|x - x_t\|$
  - Sometimes works, but not always!
  - For some data, no simple distance function works (e.g. for natural images)

This is the major weak point in LIME!

Ambiguity in "local"