Applied Machine Learning for Business Analytics

Lecture 2: Machine Learning Practices

Lecturer: Zhao Rui

Logistics

Agenda

1. Feature Engineering
2. Model Selection: Cross-Validation
3. Hyper-parameter Selection
4. Data Leakage

1. Feature Engineering

Recall that computers only understand numbers

What is Feature Engineering

- Feature engineering:
  - Extract features to use in your model
  - How to represent examples by the feature vectors?

Feature Engineering

- Core Question:
  - What properties of x **might be** relevant for predicting y?

A “Real” Machine Learning Task

- Example Task: Predict y, whether a string x is an email address
  - x: “diszr@nus.edu.sg” **y:1**
  - x: “nusmsba” **y:0**
  - x: “@trump” **y:0**
- Question: What properties of x might be relevant for predicting y?
- Feature extractor: Given input x, output a set of (feature name, feature value) pairs

“nus@gmail.com” → A **fixed-length** vector

Feature Engineering

- Question: What properties of x might be relevant for predicting y?
- Feature extractor: Given input x, output a set of (feature name, feature value) pairs

"nus@gmail.com"

feature extractor

from top to down

Length > 10	1
Length < 50	1
contain "@"	1
endwith "com"	1
endwith "sg"	0
length between @ and .	5
fraction of alpha	0.85

Discretization

ML models

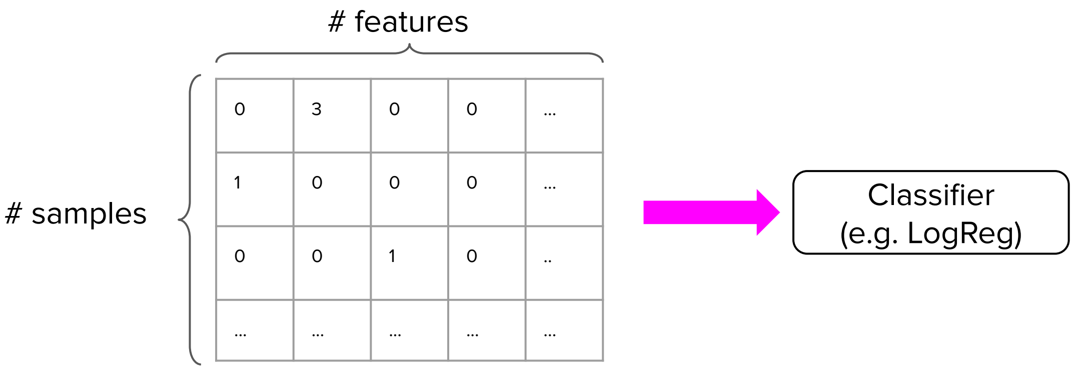
Outputs

Can we use length directly?

Engineered Features

- For text data: BoW Models

Stopword removal	I have a dog. He’s sleeping.
Lemmatization	I have dog. He’s sleeping.
Contraction	I have dog. He’s sleep.
Punctuation	I have dog. He is sleep.
Lowercase	I have dog He is sleep
N-gram	i have dog he is sleep

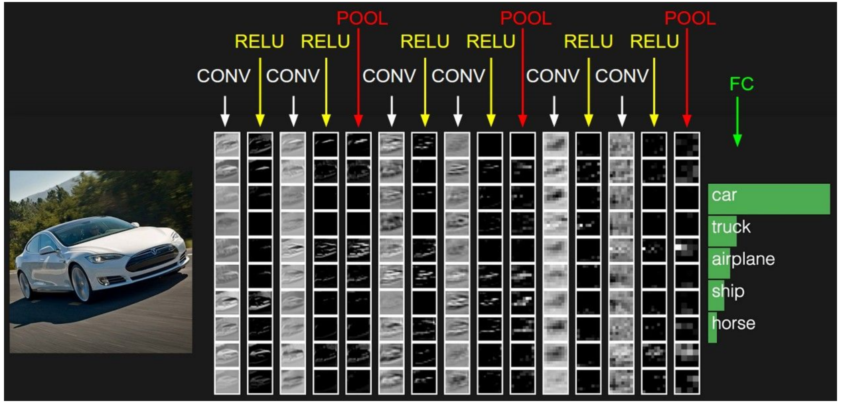


Features

I	you	have	dog	cat	he	she	is	they	sleep	I, have	have, dog	good, dog	...
1	0	1	1	0	1	0	1	0	1	1	1	0	...

Representation Learning

- Using Deep Learning Approach:
    - CNN, RNN, Attention Models
    - Learn representations from text, image, video, audio signals



http://cs231n.github.io/convolutional-networks/

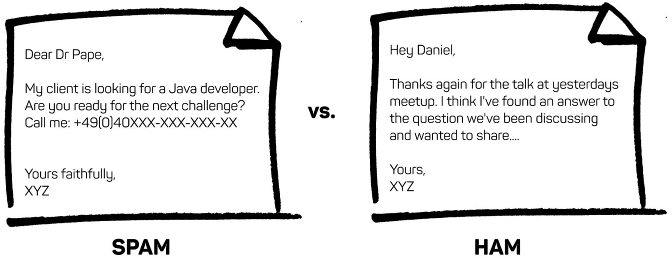
Feature Engineering

- In papers, deep learning papers promise no more feature engineering
    - We are still very far from that point
    - Deep learning are not the first choice in industry for many applications

Spam Classification

Except BoW Features:

- Post repetitiveness
- Language detection, typos, abnormal punctuations, ratio uppercase/lowercase
- IP, other users from the same IP
- Blacklisted links
- Targeted users
- ...



Feature engineering

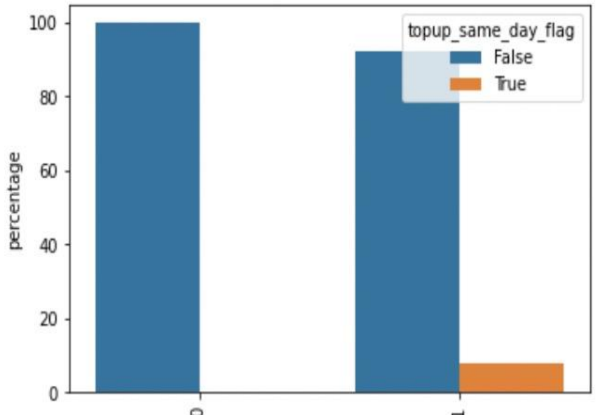
- For complex tasks, number of features can go up to millions or billions!
- Lots of ML production work involves coming up with new features
    - Fraudsters come up with new techniques very fast, so need to come up with new features very fast to counter
- Often require subject matter expertise
- Good Habits: Know your data
    - Visualize: Plot Histograms, Rank Most to least common value
    - Debug: Duplicate examples? Missing Values? Outliers? Data Agrees with dashboards? Training and Validation data similar?
    - Monitor: Feature quantiles

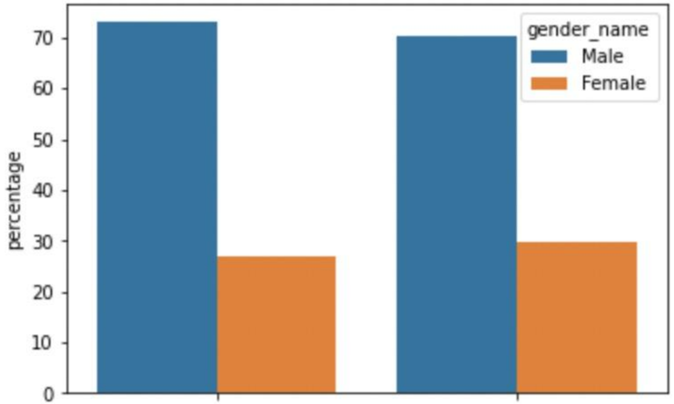
Several feature engineering tips

1. EDA
2. Handling missing values
3. Scaling
4. Discretization
5. Categorical features
6. Feature crossing

EDA for Discrete Features

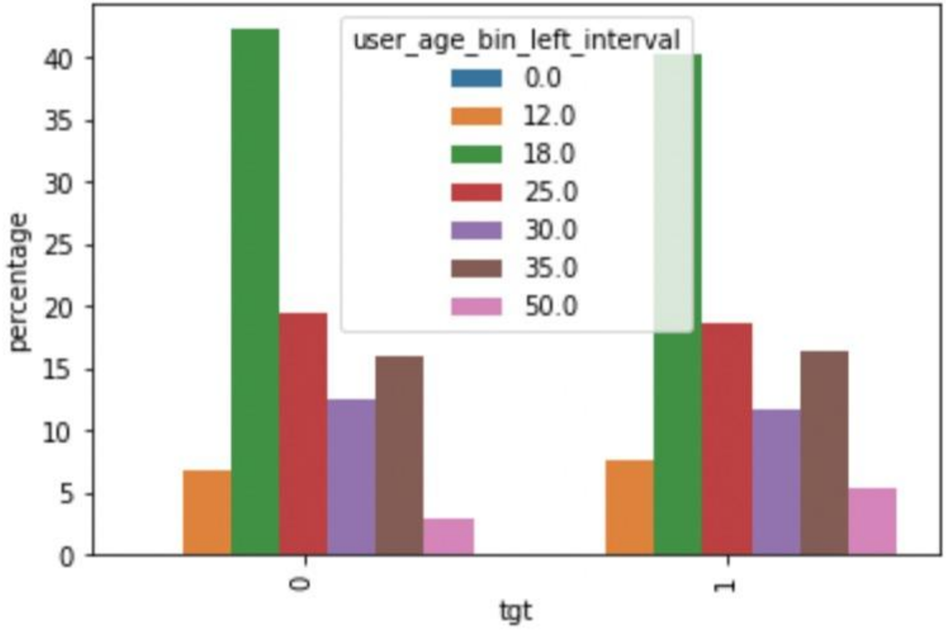
- Predict Tranx. Probabilities for Onboarding Users





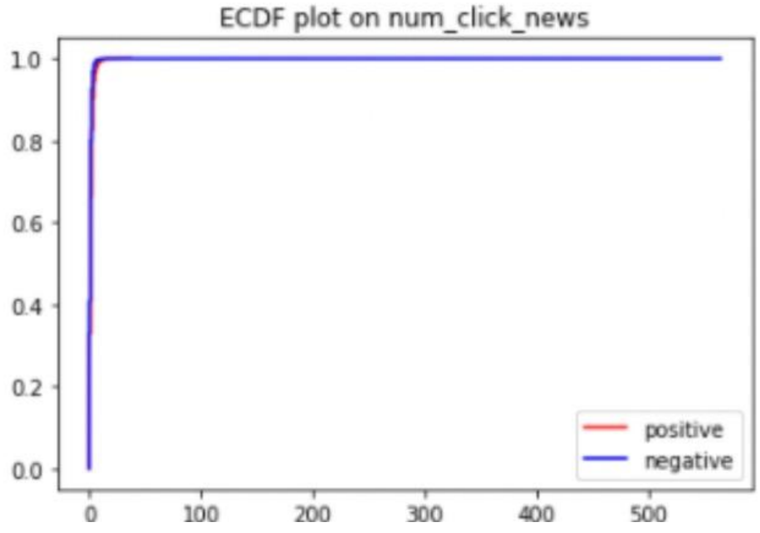
EDA for Continuous Features

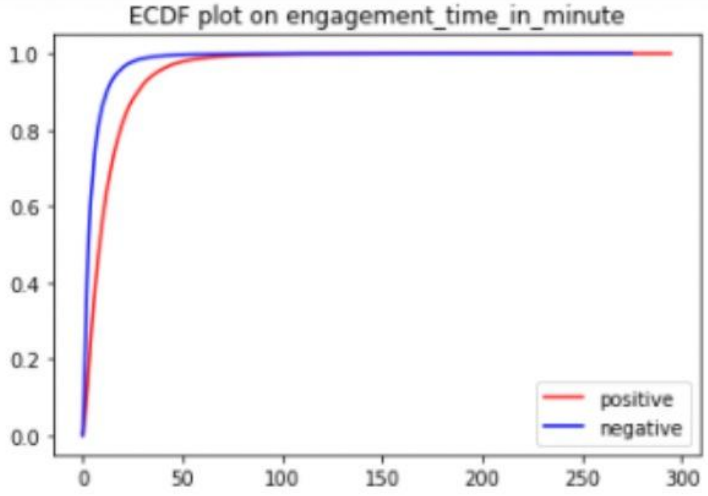
User Age Bin (legends = left interval limit, unit in years)



EDA for Continuous Features

- ECDF: An estimator of the cumulative distribution function





Why Data Goes Missing

- Data missing has different reasons
  - Missing at random (MAR)
  - Missing not at random (MNAR)
  - Missing completely at random (MCAR)

Source: https://en.wikipedia.org/wiki/Missing_data

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No

MAR

Missing at random – the missing data is related to another observed variable

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No

MNAR

Missing not at random – the data missing is related to the value itself

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B	($350,0000?)	Engineer	Yes
5	35	B	($350,0000?)	Doctor	Yes
6		A	50,000	Teacher	No

MCAR

Missing completely at random – there is no pattern to which values are missing

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No

Handling missing values

- Deletion – removing data with missing entries
- Imputation – filling missing fields with certain values

Handling missing values

- Deletion
    - Column deletion – remove columns with too many missing entries
        - drawbacks – even if half the values are missing, the remaining data still potentially useful information for predictions
        - e.g. even if over half the column for ‘Marital status’ is missing, marital status is still highly correlated with house purchasing
    - Row deletion

Marital status
Married
Single
Single

Handling missing values

- Row deletion
  - Good for: data missing completely at random (MCAR) and few values missing

ID	Age	Gender	Annual income	Job	Buy?
1	39	A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B	75,000	Engineer	Yes
5	35	B	35,000	Doctor	Yes
6	32	A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values

- Row deletion
    - Bad when many examples have missing fields

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values

- Row deletion
  - Bad for: missing data at random (MAR)
  - Can potentially bias data – we've accidentally removed all examples with gender 'A'

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B		Engineer	Yes
5	35	B		Doctor	Yes
6		A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

Handling missing values

- Row deletion
  - Bad for: missing values are not at random (MNAR)
  - Missing information is information itself

ID	Age	Gender	Annual income	Job	Buy?
1		A	150,000	Engineer	No
2	27	B	50,000	Teacher	No
3		A	100,000		Yes
4	40	B	($350,000?)	Engineer	Yes
5	35	B	($350,000?)	Doctor	Yes
6		A	50,000	Teacher	No
7	33	B	60,000	Teacher	No
8	20	B	10,000	Student	No

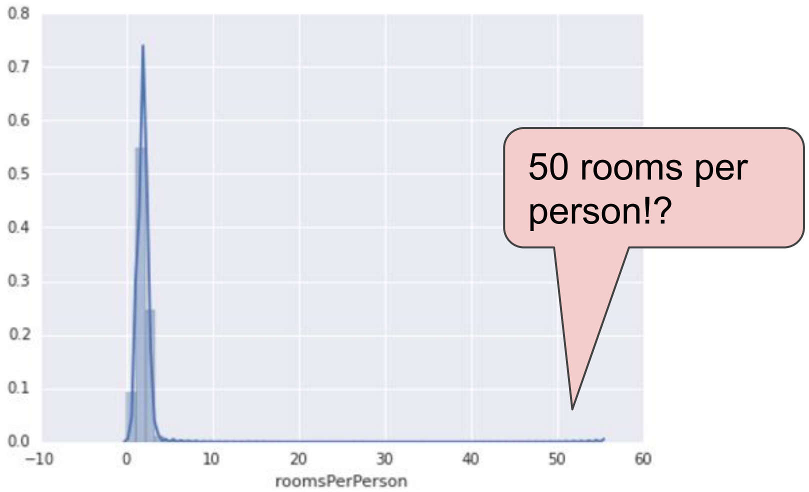
Imputation

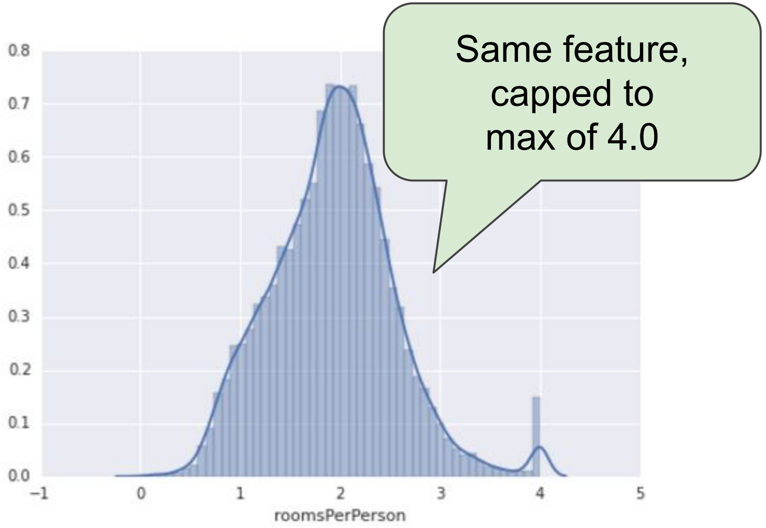
- Fill missing fields with certain values
  - Defaults
    - E.g. 0, or the empty string, etc.
  - Statistical measures – mean, median, mode
    - e.g. if a day in July is missing its temperature value, fill it with the median temperature in July

Scaling

Distribution should not have crazy outliers

Ideally all features transformed to a similar range, like (-1, 1) or (0, 5).





Types of scaling

scaling type	**use case**
min/max normalization	Any -- no assumptions about variables
z-score normalization	When variables follow a normal distribution
log scaling	When variables follow an exponential distribution

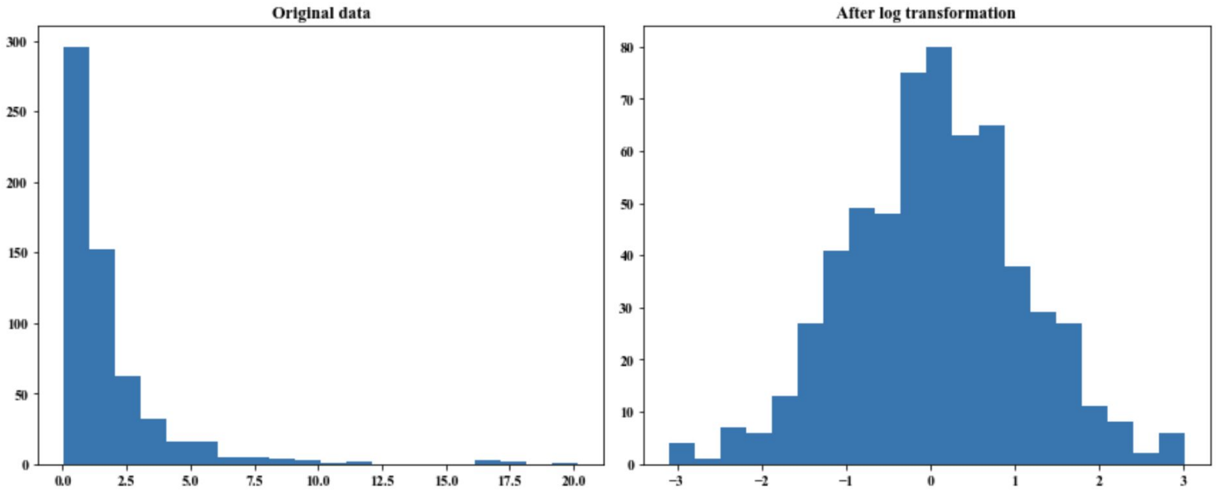
Feature Scaling

- Common Scaling Methods:
  - Min-max Scaler: $\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$
  - Z-score transformation: $\hat{x} = \frac{x - x_{mean}}{\sigma}$

- Which machine learning models require feature scaling?

Log scaling

- Help with skewed data
- Often gives performance gain



Potential Data Leakage

- scaling might lead to **data leakage**
- scaling variables requires “global” statistics

Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
    - Incorporate knowledge/expertise about each variable by constructing specific buckets
- Examples
    - Income
        - Lower income: x < $35,000
        - Middle income: $35,000 <= x < $100,000
        - High income: x >= $100,000
    - Age
        - Minors: x < 18
        - College: 18 <= x < 22
        - Young adult: 22 <= x < 30
        - 30 <= x < 40
        - 40 <= x < 65
        - Seniors: x >= 65

Encoding Categorical Features

- Example: you want to build a recommendation system for Amazon
    - There are over 2 million brands that we need to recommend
    - It could be used as part of items features
    - Let us try one-hot encoding

Encoding Categorical Features

- one-hot encoding
- How to address unseen brands when the model is deployed

Encoding Categorical Features

- one-hot encoding!
- encode unseen brands with “UNKNOWN”

Did we solve the unseen problem?

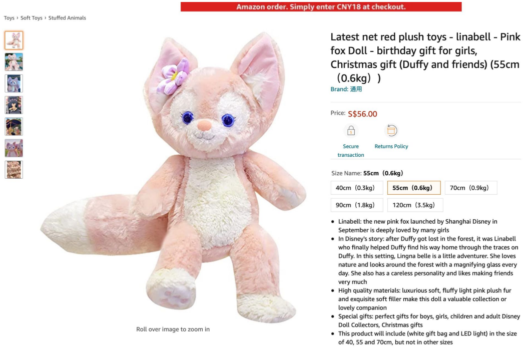
Encoding Categorical Features

- one-hot encoding!
- encode unseen brands with “UNKNOWN”
- Group low frequent 1% of brands and newcomers into “UNKNOWN” category

Did we solve the unseen problem?

Encoding Categorical Features

- one-hot encoding!
- encode unseen brands with “UNKNOWN”
- Group low frequent 1% of brands and newcomers into “UNKNOWN” category
- Problem – this treats all new brands the same as unpopular brands on the platform



What if Disney Linabell come to Amazon as a new brand

Encoding New Categories

1. The question is: how to implement a robust method of handling the potential new brands? (Out of Vocabulary Problems)
2. Two popular methods:
   a. Represent each category with its attribute
      i. For example, to represent a brand, use features: category, yearly revenue, company size, etc..
      ii. The similar trick could be found in BERT to address OOV words.
   b. Hashing trick
      i. Using a hash function to hash categories to different indexes

Feature Crossing

- Combine two or more features to create a new feature

Marriage	Single	Married	Single	Single	Married
Children	0	2	1	0	1
Marriage & children	Single, 0	Married, 2	Single, 1	Single, 0	Married, 1

Feature Crossing

- Helps models learn non-linear relationships between variables
- Warning – feature crossing can blow up your feature space
    - e.g. Feature A and B both have 100 categories → Feature A x B will have 10,000 categories
    - Need even more data to learn this new feature space
    - Blowing up feature space can increase risk of overfitting

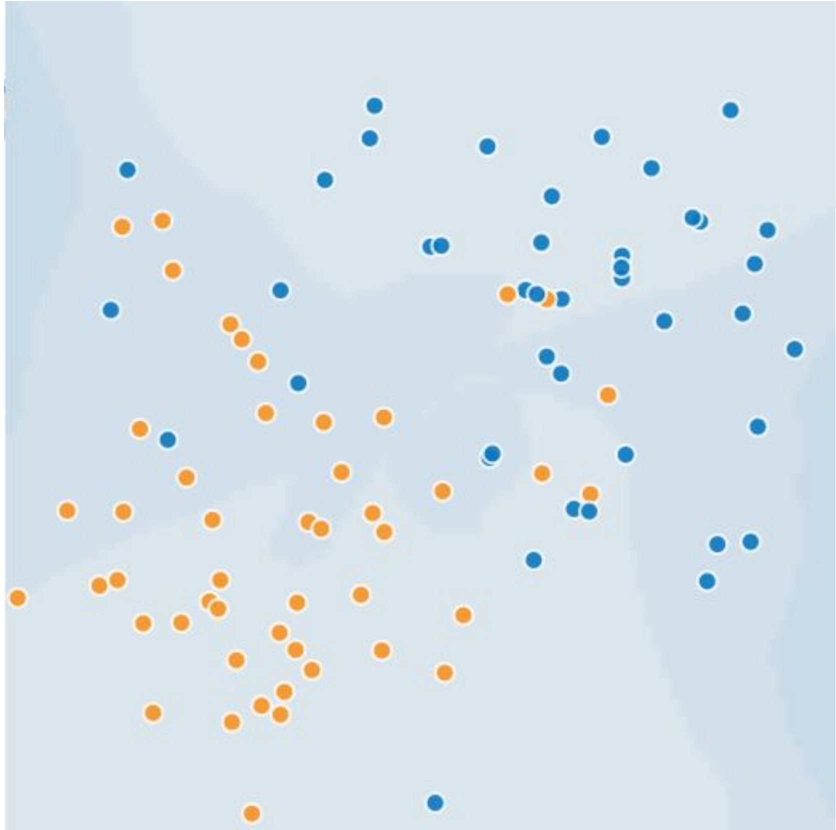
Feature crossing is widely used in recommendation system (CTR prediction)

1. https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html
2. https://www.ijcai.org/proceedings/2017/0239.pdf

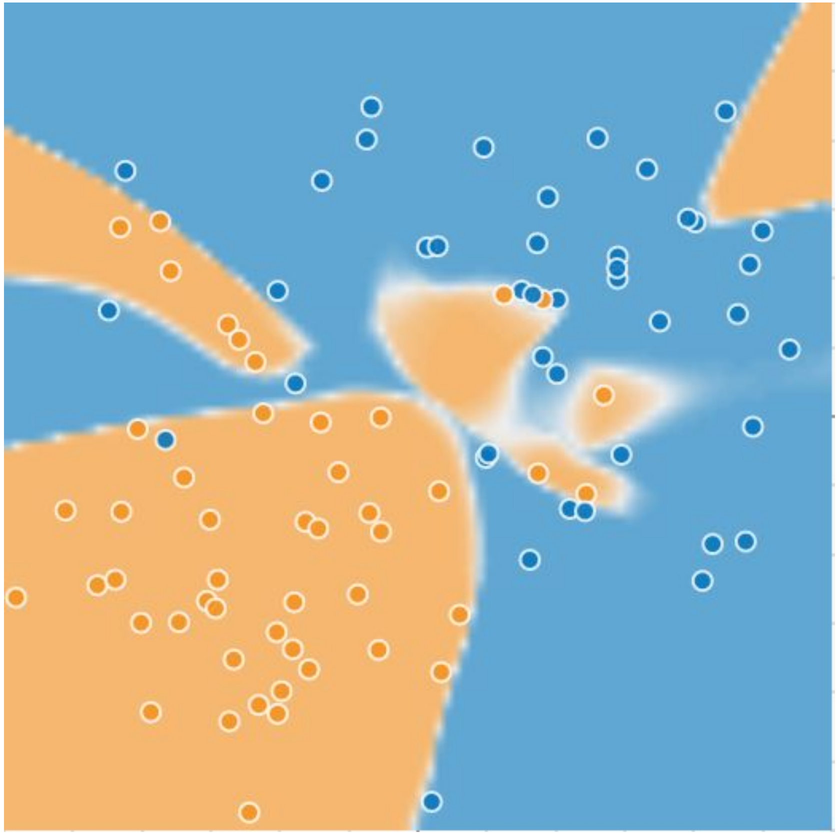
2. Cross-Validation

Which measure should we look for model evaluation?

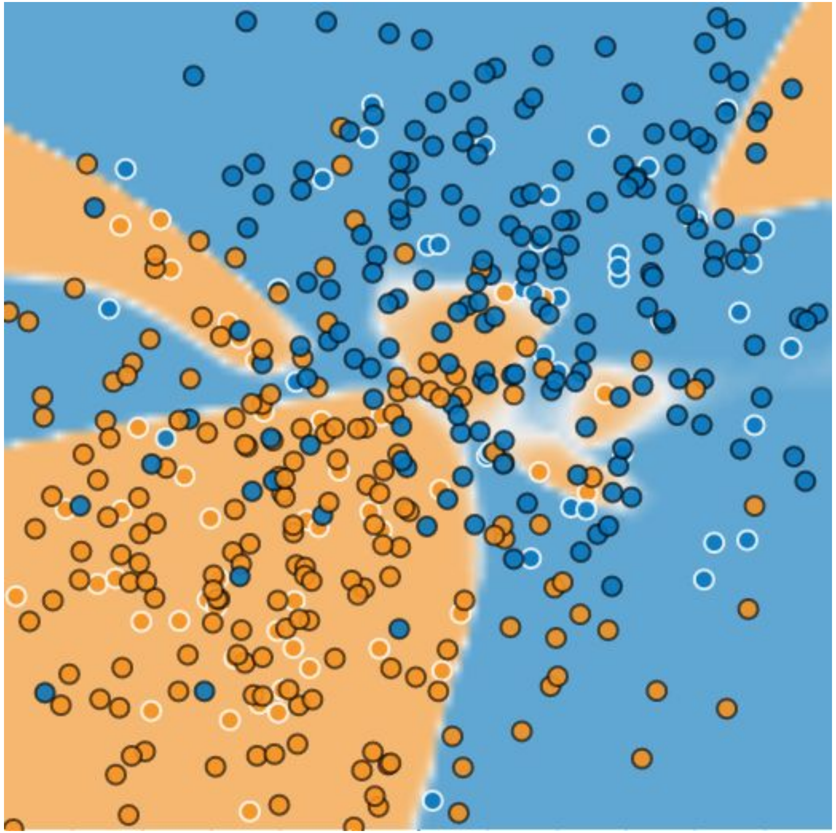
Let's try to train a model for this problem



How about this model?



More data



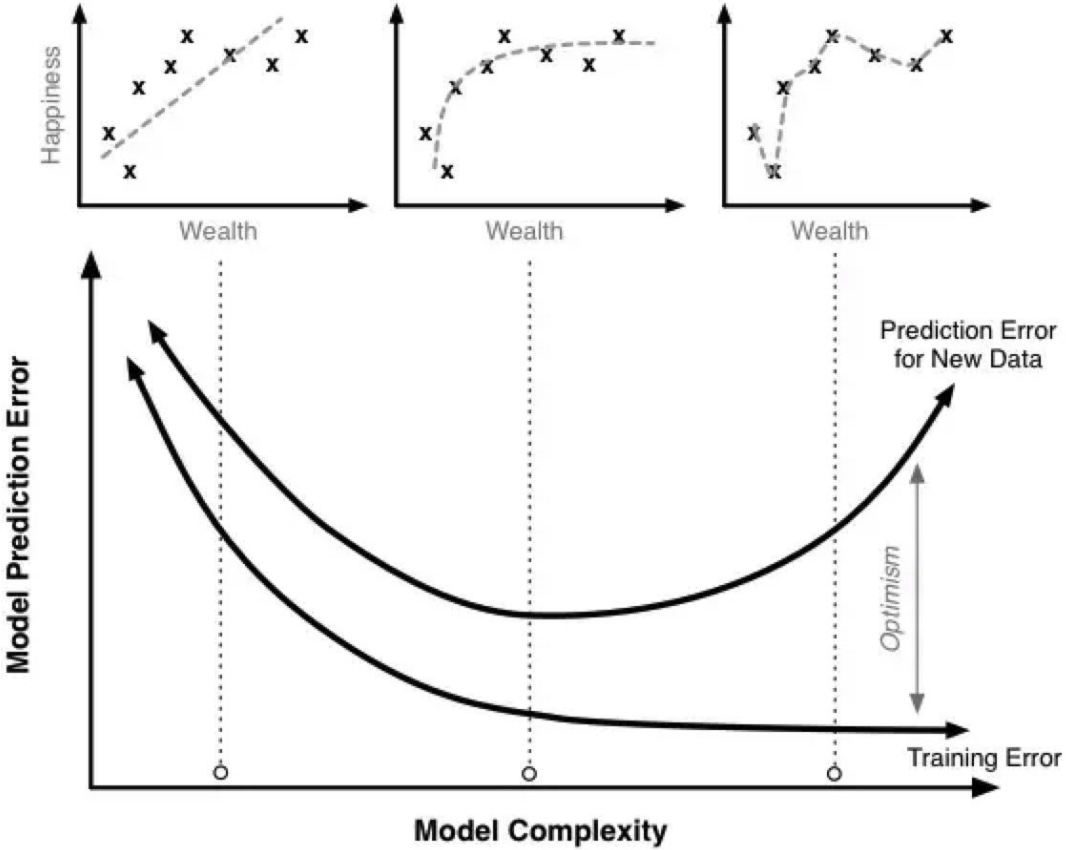
Which measure should we look for model evaluation?

Training performance is not suitable

Generalization

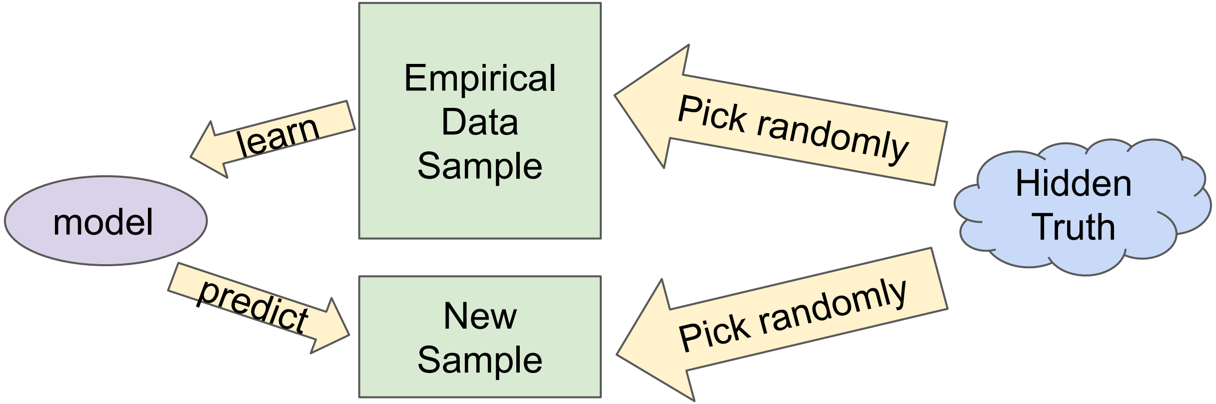
- In ML, a model is used to fit the data
- Once trained, the model is applied upon new data
- Generalization is the prediction capability of the model on live/new data

Model Complexity



The Big Picture

- Goal: predict well on new data drawn from (hidden) true distribution.
- Problem: we don't see the truth.
  - We only get to sample from it.
- If model *h* fits our current sample well, how can we trust it will predict well on other new samples?



Is the model overfitting?

- Intuition: Occam's Razor principle
  - The less complex a model is, the more likely that a good empirical result is not just due to the peculiarities of our samples.

- Theoretically:
  - Interesting field: generalization theory
  - Based on ideas of measuring model simplicity / complexity

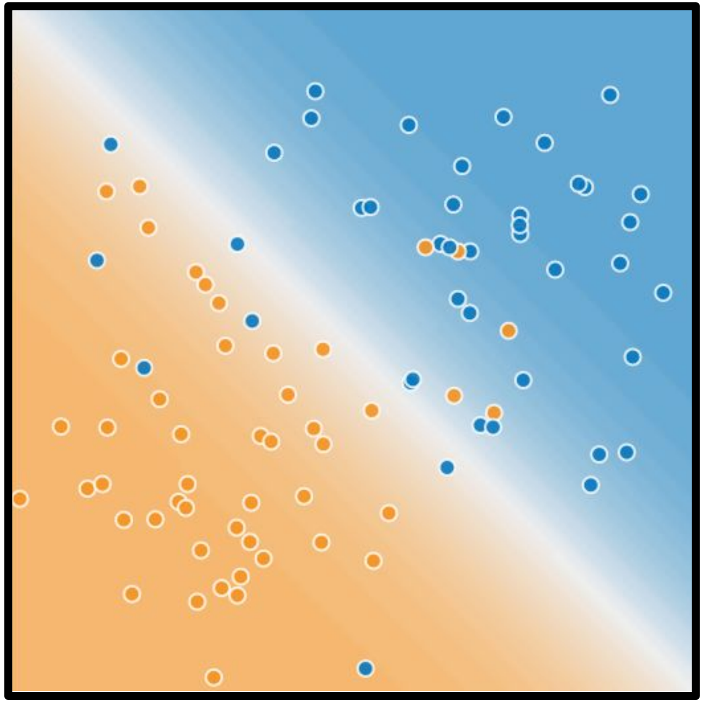
Is the model overfitting?

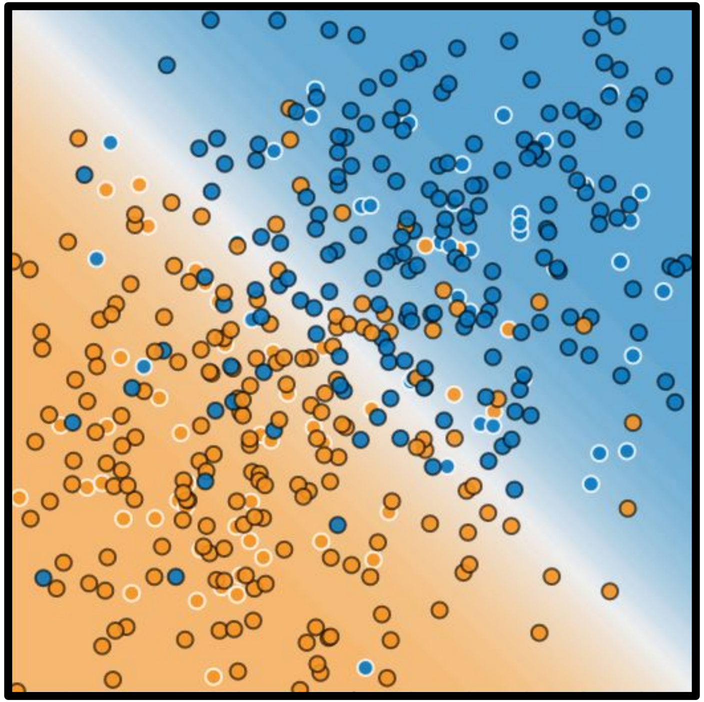
- Empirically:
  - Key point: will our model be good on new samples?
  - Evaluate: get new samples of data (test set)
  - If test set is large enough and we do not cheat by using test set over and over, the good performance on test set can be a useful indicator of model's generalization capability

Training/Test Splitting

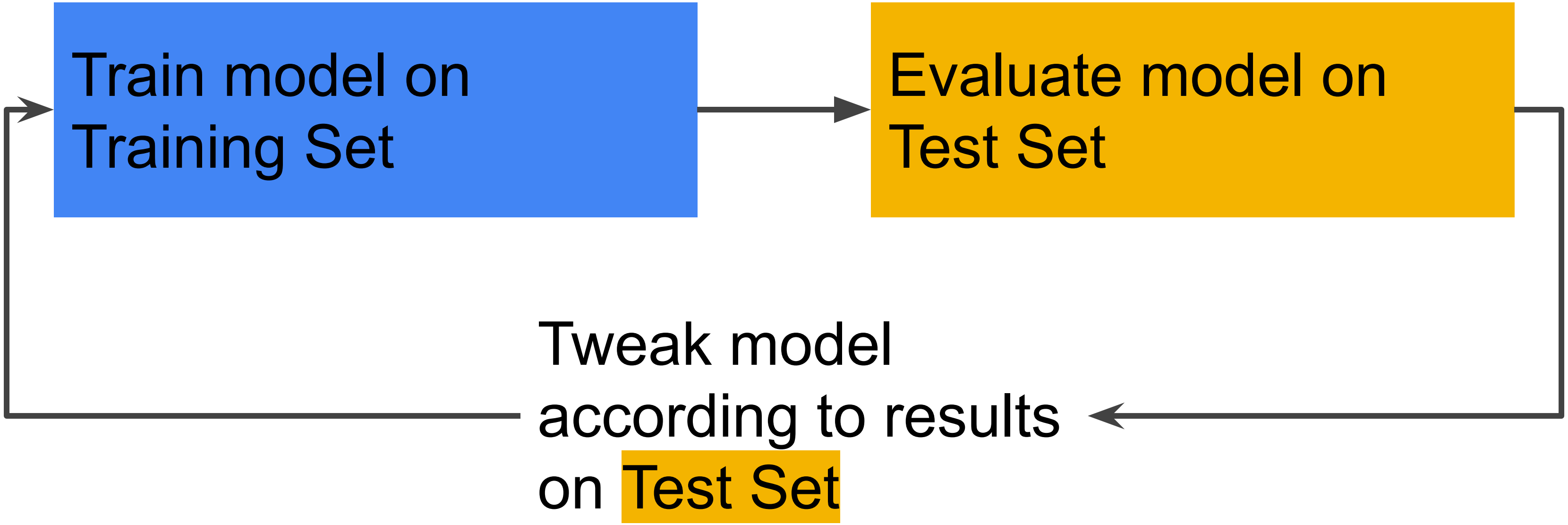
- If models do much better on the training set than the testing set, then models are likely overfitting.
- How do we divide?
    - Randomization for splitting
    - Larger training data size -> better model
    - Larger testing data size -> more confident in model's evaluation
    - One practical rule: 10-15% left for testing, the rest for training

Training vs Test





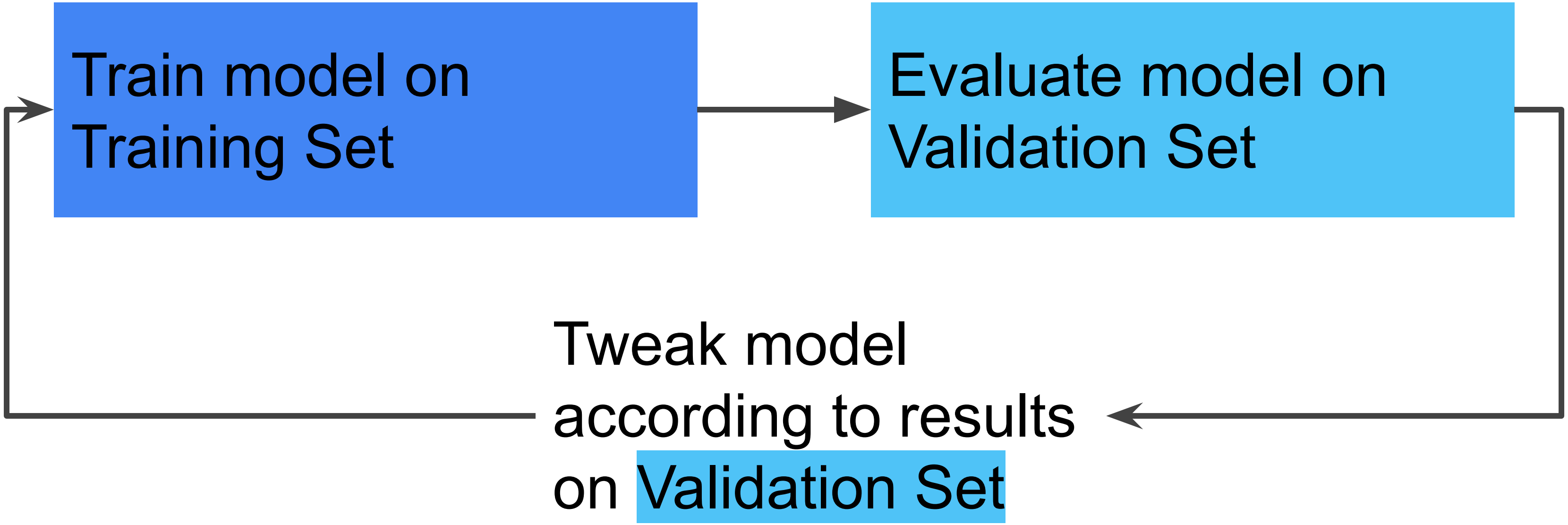
How about this workflow?



Pick model that does best on Test Set.

Partition Data Sets

Better Workflow: Use a validation set



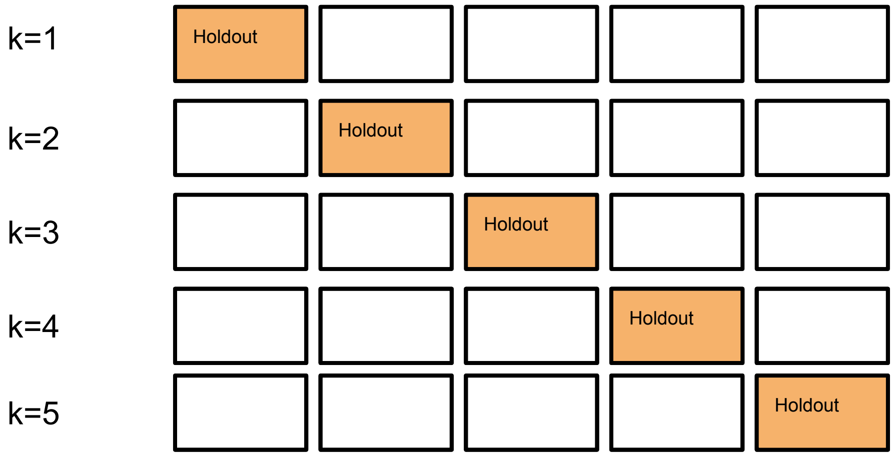
Pick model that does best on Validation Set
Confirm results on Test Set

Cross-Validation

- If we have a small dataset: CV can be used
- Idea is simple but smart:
  - Use your initial training data to generate multiple mini train-test splits. Use these splits to evaluate your model
  - K is a hyper-parameters. K is equal to the number of generated train-test splits.

Cross-Validation

- Partition data into k subsets, i.e., folds
- Iteratively train the model on k-1 folds while using the remaining fold as the test set (hold-out set)
- Compute the average performances over the K folds



CV

- Divide into three sets
  - Training set
  - Validation set
  - Test set
- Classic gotcha: only train the model on training data
  - Getting surprisingly low loss?
  - Check the whole procedure

How to detect overfitting

- After training/testing splitting, training loss is much less than testing loss.
- Start with a simple model as the benchmark
    - When add model complexity, you will have a reference point to see whether the additional complexity is worthy.

How to prevent overfitting

- Train with more data
  - Filter noisy data (outlier)
- Remove features
  - Remove irrelevant features
- Regularization
  - Control model complexity
  - Different machine learning models have their own regularization methods.

sklearn.linear_model.Ridge

class sklearn.linear_model.**Ridge**(*alpha=1.0, fit_intercept=True, normalize=False, copy_X=True, max_iter=None, tol=0.001, solver='auto', random_state=None*) [source]

Linear least squares with l2 regularization.

Minimizes the objective function:

```
||y - Xw||^2_2 + alpha * ||w||^2_2
```

This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. Also known as Ridge Regression or Tikhonov regularization. This estimator has built-in support for multi-variate regression (i.e., when y is a 2d-array of shape (n_samples, n_targets)).

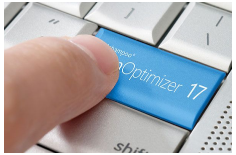
Read more in the User Guide.

Alpha is the controlling parameter, which is also hyperparameter

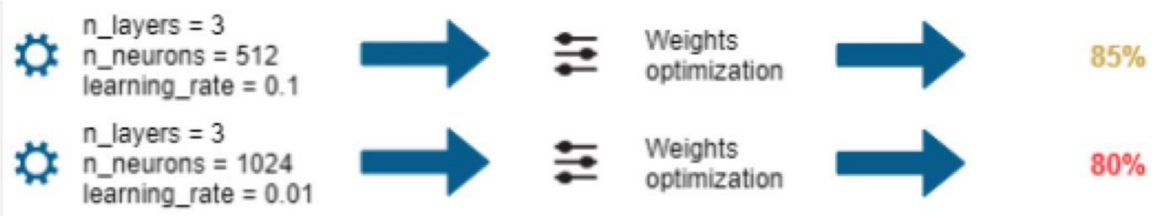
3. Hyperparameter Optimization

Hyperparameters

- Machine learning algorithms usually have two kinds of weights:
  - ***Parameters***: learned by data during training such as slope of linear regression, layer weights of neural networks
  - **Hyperparameters**: left to us to select beforehand such as K in KNN, number of layers in neural networks



Hyperparameters **Parameters** **Scores**



Hyperparameters

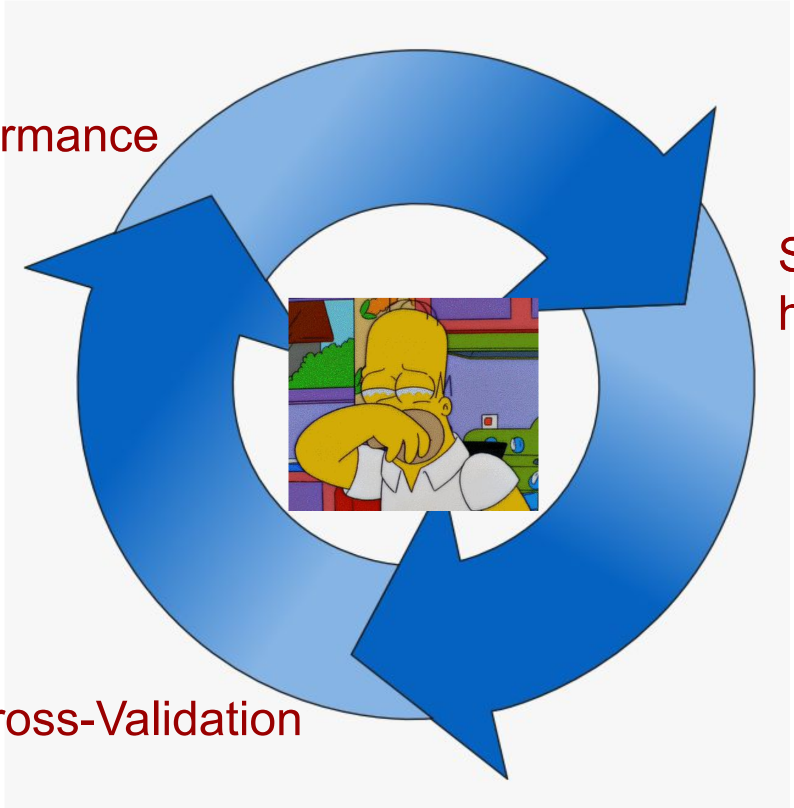
```
>>> from sklearn.linear_model import Ridge
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.randn(n_samples)
>>> X = rng.randn(n_samples, n_features)
>>> clf = Ridge(alpha=1.0)
>>> clf.fit(X, y)
Ridge()
```

Hyperparameters should be passed when you initialize the machine learning model **before training**

Searching is Iterative, then Expensive

Track the performance

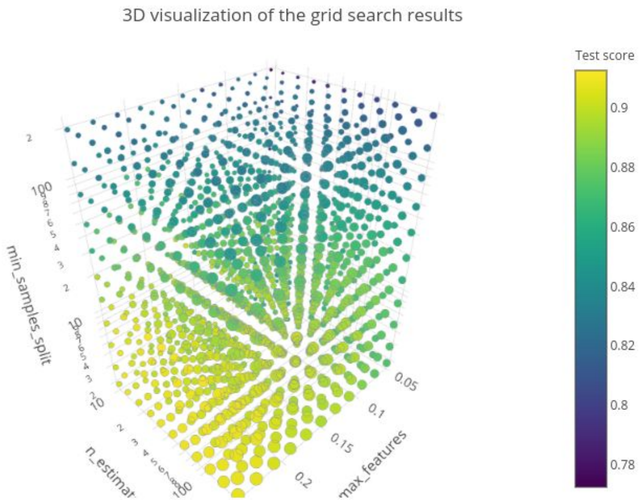
Select one potential hyperparameter set



Run Cross-Validation

Grid Search

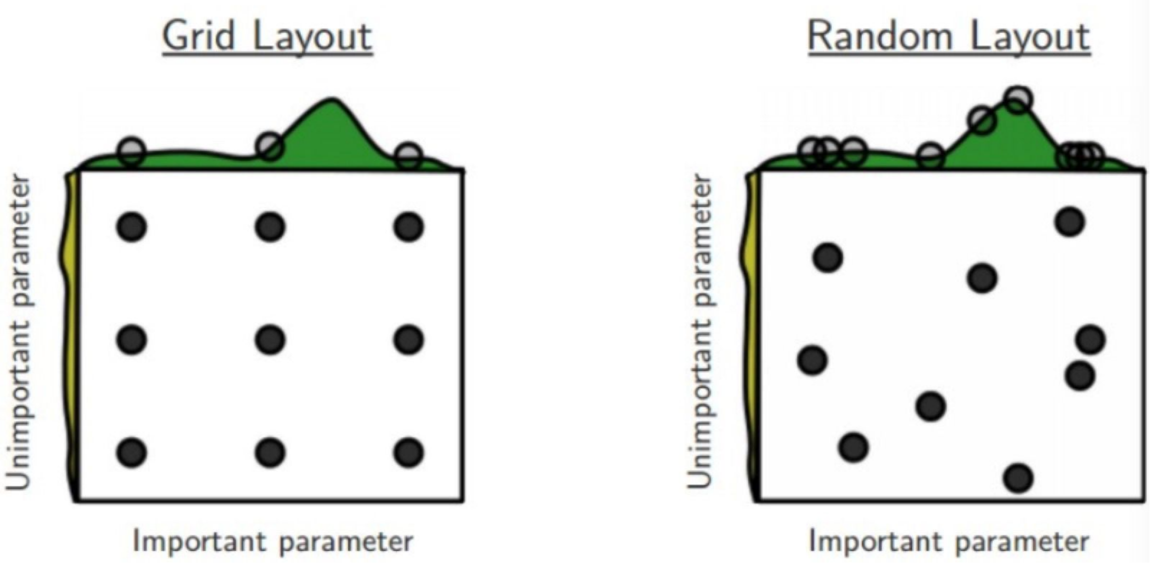
- Define a grid on n-dimensions, where each of these maps for an hyperparameter
- For each dimension, define the range of possible values
- Search for all combinations and select the best one



Inefficient !!!

Random Search

- Randomly pick the point from the configuration space
- The rest is the same as grid search



Good on high-dim spaces

From Bergstra and Bengio

Advanced Search Algorithms

- For grid and random search, the previous trials can not contribute to each new guess.
- Try to model the hyperparameter search as a machine learning task
  - Tree-structured Parzen Estimator
  - Gaussian Process
  - Other bayesian optimization methods

Main idea: based on the distribution of the previous results, decide which set of parameters should be explored firstly

4. Data Leakage

Data Leakage

- Training data leakage
  - Oversampling before splits
    - Training data may overlap with testing data
  - Prepare features on the entire data instead of just training data
    - Create vocab/preprocessing scaler from train+test data
  - Group leakage
    - A patient has 2 CT scans, 1 in train, 1 in test.
- **Feature leakage**
  - Some form of the label “slip” into the features
  - **This same information is not available during inference**

Feature Leakage Example I

- Detect Lung Cancer from CT Scans
- Collected from Hospital I
- Performs well on unseen data from I
- Performs poorly on new data from Hospital II

Date
Doctor note
Medical record
Scanner type
CT scan Image

Feature Leakage Example I

- Detect Lung Cancer from CT Scans
- Collected from Hospital I
- Performs well on unseen data from I
- Performs poorly on new data from Hospital II

Date
Doctor note
Medical record
Scanner type
CT scan Image

At hospital I, when doctors suspect that a patient has lung cancer, they send that patient to a higher-quality scanner

How to avoid leakage

- Check for duplication between train and valid/test splits
- Use only train splits for feature engineering (model training for sure)
- Train model on subset of features
  - If performance very high on subset, either high quality features or leakage!
- Monitor model performance as more features are added
  - If sudden increase, either high quality features or leakage!
- Check the correlation between feature and label
- Keep asking yourself during model development: can we use this information when the model is deployed for inference?