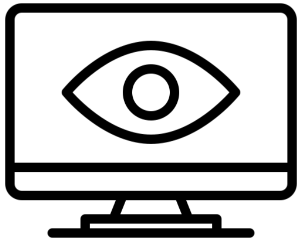
Machine Learning and Computer Vision



Jianbo Jiao

School of Computer Science

[Programming for Data Science]

Outline

- Machine Learning
  - Basic concepts
  - Machine learning in Python
    - SciPy
    - scikit-learn
- Computer Vision
  - Basic concepts
  - Python packages (scikit-image, OpenCV)

SciPy

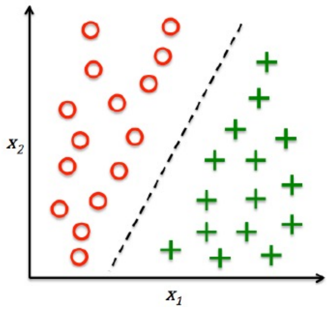
scikit learn

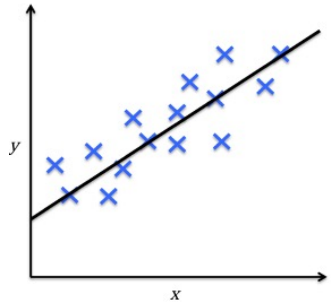
scikit-image
image processing in python

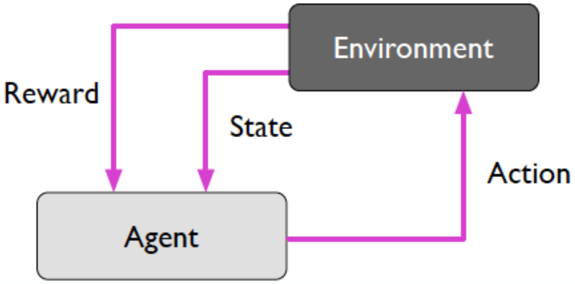
OpenCV

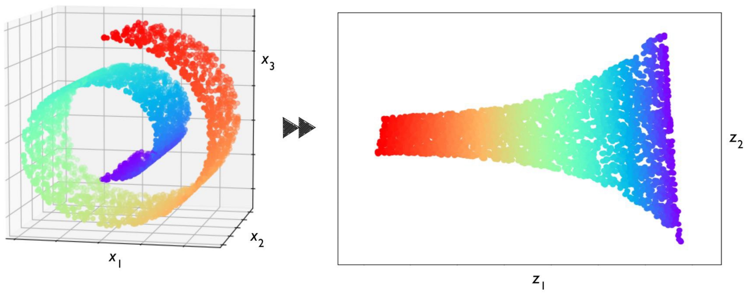
Machine learning – categories

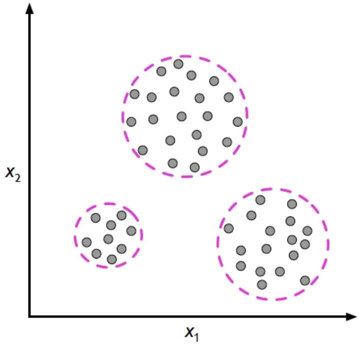
- Supervised learning
- Unsupervised learning
- Reinforcement learning







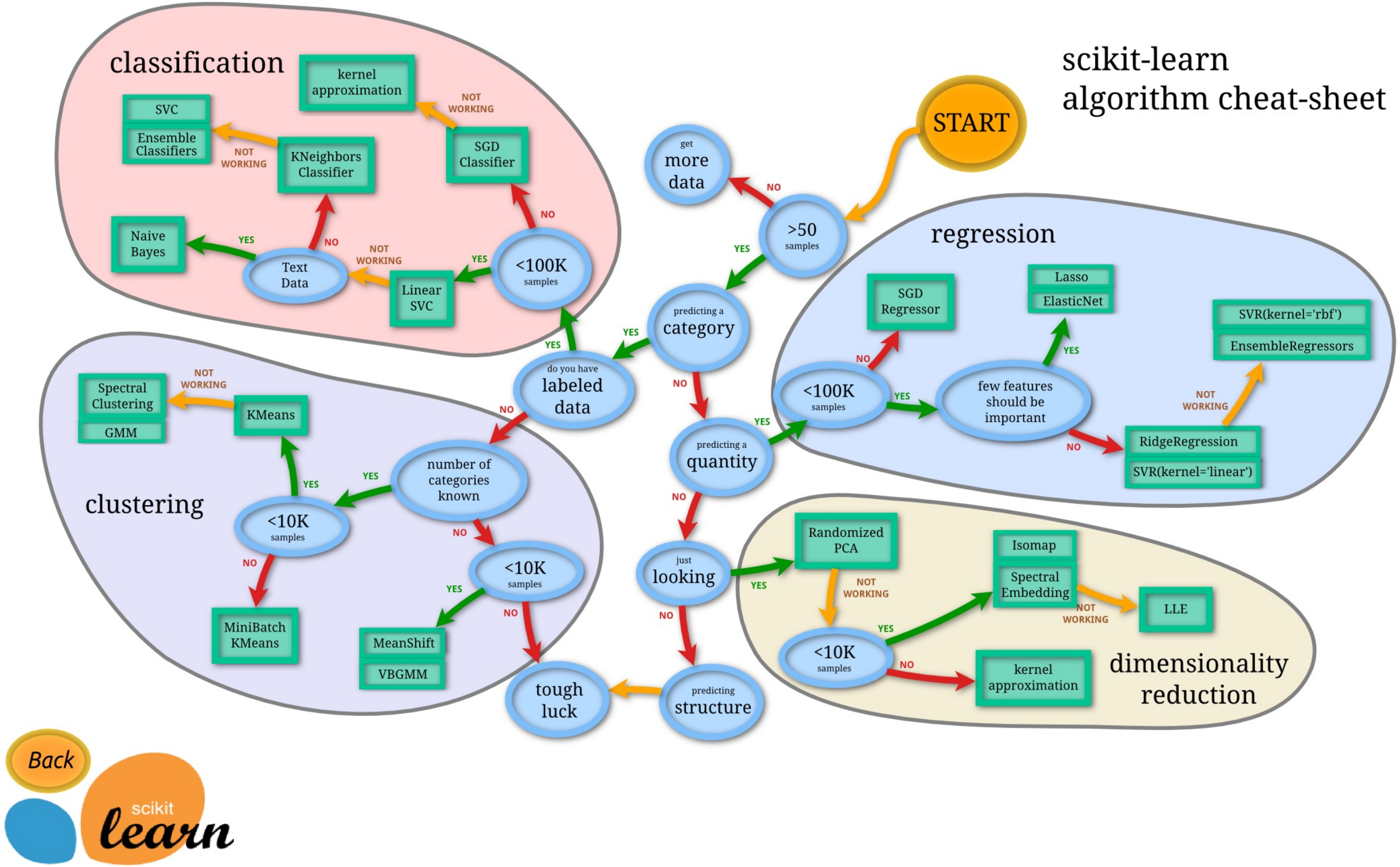




Machine learning in Python

Scikit-learn

- The best known machine learning package in Python.
- Provides efficient versions of a large number of common algorithms.
- Characterized by a clean, uniform, and streamlined API.
- Built on Numpy, Scipy and matplotlib.
- Open source and commercially usable.
- Official website: https://scikit-learn.org/stable/index.html



https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-learn

Installing the latest release

Operating System Windows macOS L

Packager pip conda

Use pip virtualenv

Install Python 3 using homebrew (`brew instal`
Then run:

```
$ pip install -U scikit-learn
```

In order to check your installation you can use

```
$ python -m pip show scikit-learn  # to s
$ python -m pip freeze  # to see all pack
$ python -c "import sklearn; sklearn.show
```

Installing the latest release

Operating System Windows macOS Linux

Packager pip conda

Install conda using the Anaconda or miniconda installers or the miniforge installers (no administrator permission required for any of those).
Then run:

```
$ conda create -n sklearn-env -c conda-forge scikit-learn
$ conda activate sklearn-env
```

In order to check your installation you can use

```
$ conda list scikit-learn  # to see which scikit-learn version is installed
$ conda list  # to see all packages installed in the active conda environment
$ python -c "import sklearn; sklearn.show_versions()"
```

Scikit-learn

- 1. Read and arrange data into a feature matrix (X) and target vector (Y).
- 2. Choose a class of model by importing the appropriate estimator class from sklearn.
- 3. Choose model hyperparameters by instantiating this class with desired values.
- 4. Fit the model to your data by calling the fit() method of the model instance.
- 5. Evaluate model: calculate performance metrics.
- 6. Apply the model to new data

- For supervised learning, predict labels using predict() method.
- For unsupervised learning, transform or infer properties of data using transform() or predict().

Scikit-learn

Scikit-learn doesn’t like categorical features in strings or text features.

- Categorical features
- Text features
- Image features (scikit-image)

Scikit-learn

Country	Age	Salary
India	44	72000
US	34	65000
Japan	46	98000
US	35	45000
Japan	23	34000

Country	Age	Salary
0	44	72000
2	34	65000
1	46	98000
2	35	45000
1	23	34000

labelEncoder can do this

0	1	2	Age	Salary
1	0	0	44	72000
0	0	1	34	65000
0	1	0	46	98000
0	0	1	35	45000
0	1	0	23	34000

DictVectorizer and OneHotEncoder can do this

- Convert categorial features to numbers.
  - DictVectorizer
  - OneHotEncoder
- They both create additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature.

Scikit-learn – Text Features

- Bag of words (BoW)
  - Take each snippet of text, count the occurrences of each word within it, and put the results in a table.
  - More frequent word gets a higher value.

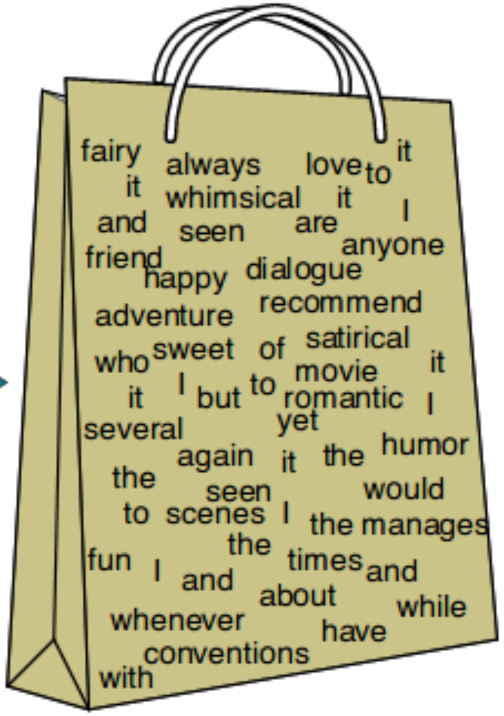
- Term frequency-inverse document frequency (TF-IDF)
  - Penalize some frequently occurring words across documents without useful or discriminatory information (e.g. and, the).
  - Gives a higher value to the word that is rare in all the documents combined but frequent in a single document.
  - TF-IDF often gives better results when working with ML algorithms.

Scikit-learn – Text Features

- Bag of words (BoW)

```python
from sklearn.feature_extraction.text import CountVectorizer
```

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

https://koushik1102.medium.com/nlp-bag-of-words-and-tf-idf-explained-fd1f49dce7c4

Scikit-learn – Text Features

- Term frequency-inverse document frequency (TF-IDF)

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0

Scikit-learn – Text Features

- Term frequency-inverse document frequency (TF-IDF)

```python
from sklearn.feature_extraction.text import TfidfVectorizer
```

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

$$TF_{ij} = \frac{f_{ij}}{n_j} \qquad (1)$$

Where f_{ij} is the frequency of term i in document j. n_j is the total number of words in document j.

$$IDF_i = 1 + log(\frac{N}{c_i}) \qquad (2)$$

Where N is the total number of documents in the corpus. c_i is the number of documents that contain word i.

$$w_{ij} = TF_{ij} \times IDF_i \qquad (3)$$

Where w_{ij} is the TF-IDF score of term i in document j.

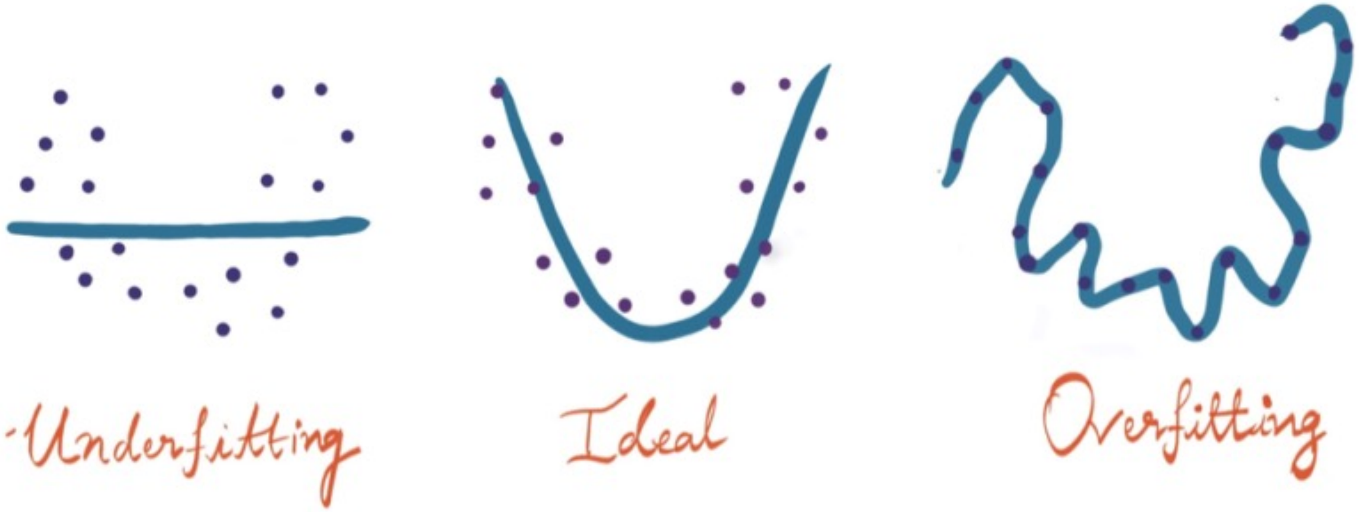
https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0

Scikit-learn

- 1. Read and arrange data into a feature matrix (X) and target vector (Y).
- **2. Choose a class of model by importing the appropriate estimator class from sklearn.**
- **3. Choose model hyperparameters by instantiating this class with desired values.**
- 4. Fit the model to your data by calling the fit() method of the model instance.
- 5. Evaluate model: calculate performance metrics.
- 6. Apply the model to new data

- For supervised learning, predict labels using predict() method.
- For unsupervised learning, transform or infer properties of data using transform() or predict().

Scikit-learn



Scikit-learn

Never train and evaluate the model on the same data!

Goal is to estimate likely performance of a model on **out-of-sample data.**

We want a model that **generalizes** well to unseen data, for example, by using an independent test set.

Scikit-learn

X_train

X_test

feature 1	feature 2	response
1	2	2
3	4	12
5	6	30
7	8	56
9	10	90

y_train

y_test

```python
# STEP 1: split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=4)
```

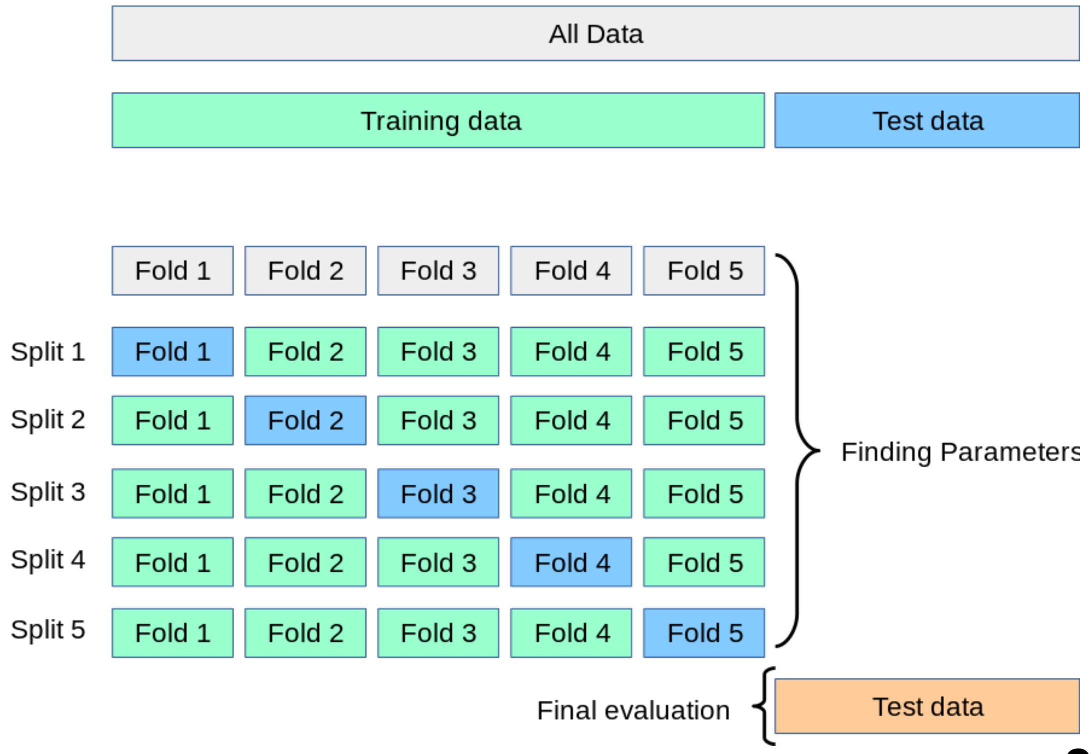
Scikit-learn– K-Fold Cross Validation (CV)

1. Split the dataset into K **equal** partitions (or "folds").
2. Use fold 1 as the **testing set** and the union of the other folds as the **training set**.
3. Calculate **testing accuracy**.
4. Repeat steps 2 and 3 K times, using a **different fold** as the testing set each time.
5. Use the **average testing accuracy** as the estimate of out-of-sample accuracy.

- Make use of all data for testing.

Scikit-learn– K-Fold Cross Validation (CV)

1. K can be any number, but K=10 is generally recommended
2. For classification problems, stratified sampling is recommended for creating the folds
   - Each response class should be represented with equal proportions in each of the K folds
   - scikit-learn's cross_val_score function does this by default
3. There're many CV variations. Refer to the sklearn website.



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

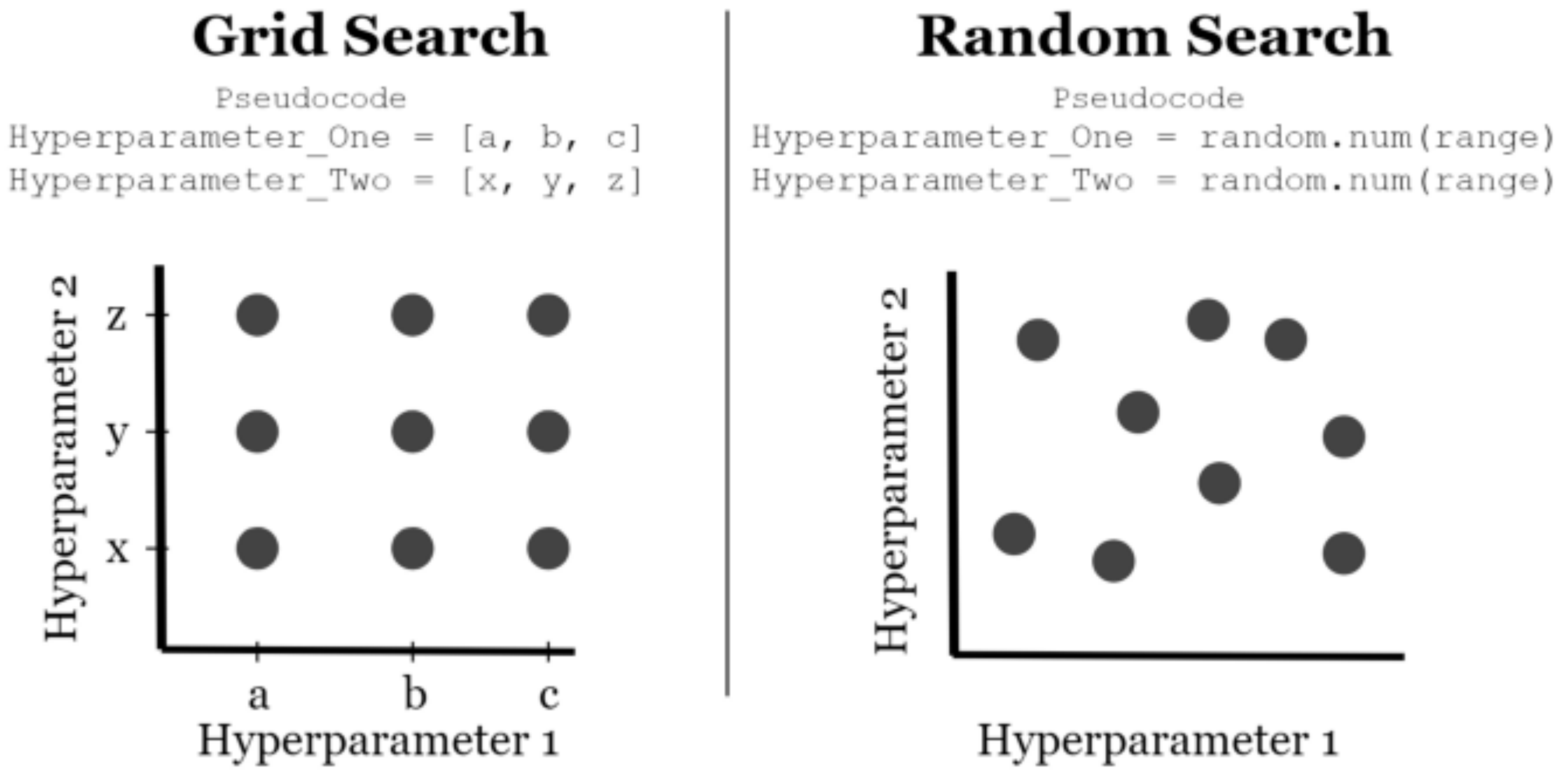
Scikit-learn– Exhaustive Grid Search

- Allows you to define a **grid of parameters** that will be **searched** using K-fold cross-validation.
- Test all the parameter combinations and return you the mean performance of each combination or the best setting over K-fold CV.
- In sklearn,
  from sklearn.model_selection import GridSearchCV
- Searching through all combinations could be computationally expensive/infeasible.

Scikit-learn– Randomized Search

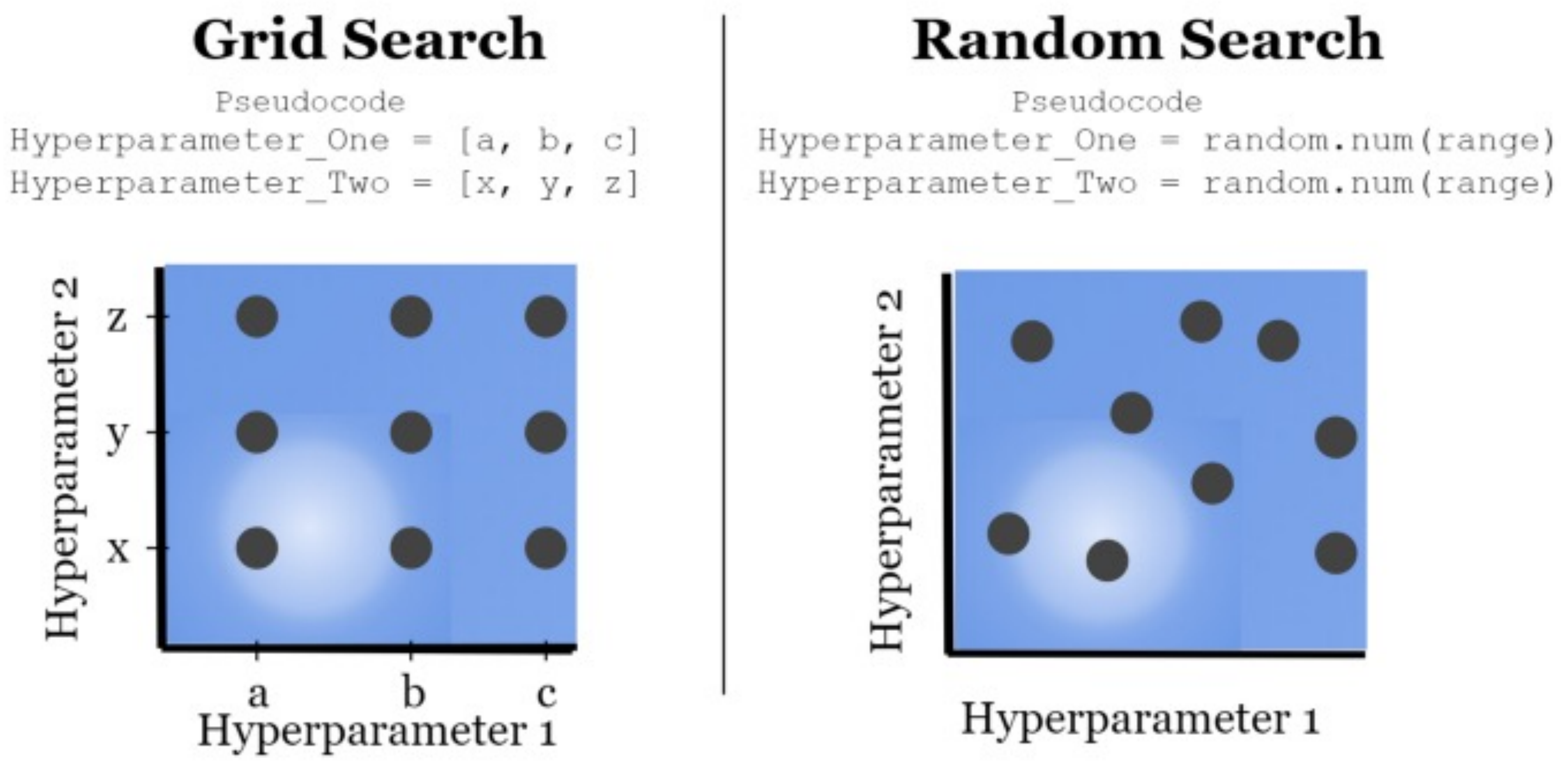
- Random Search sets up a grid of hyperparameter values and selects **random combinations** to train the model and score. This allows you to explicitly control the number of parameter combinations that are attempted.
- The random selection is based on a distribution for continuous parameters.
- The combinations are tested through K-fold cross-validation.
- In sklearn,
  from sklearn.model_selection import RandomizedSearchCV
- The chances of finding the optimal parameter are comparatively higher in random search because of the random search pattern.

Scikit-learn– ?? Search



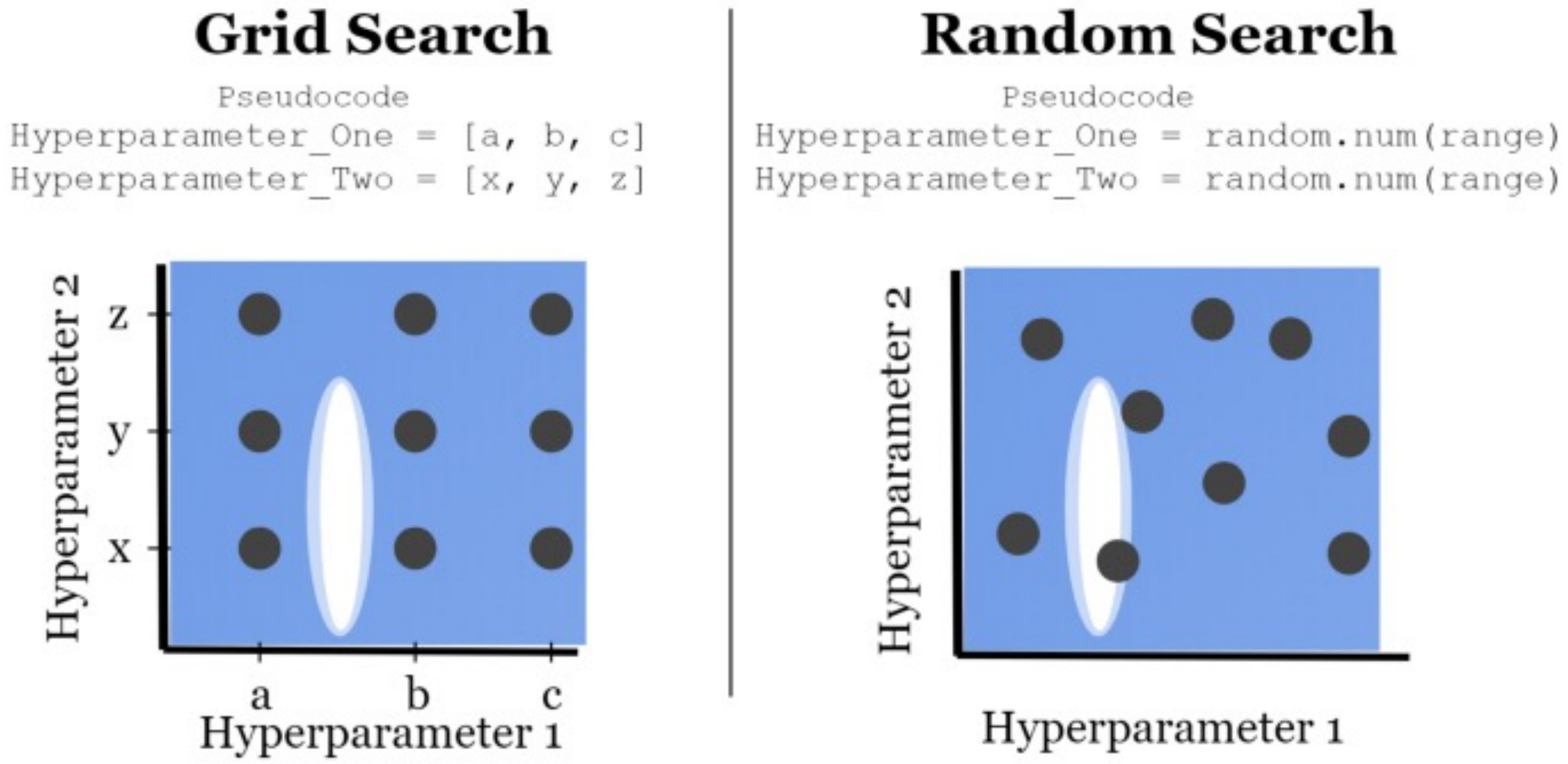
https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35

Scikit-learn– ?? Search



https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35

Scikit-learn– ?? Search



https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35

Scikit-learn

- 1. Read and arrange data into a feature matrix (X) and target vector (Y).
- 2. Choose a class of model by importing the appropriate estimator class from sklearn.
- 3. Choose model hyperparameters by instantiating this class with desired values.
- **4. Fit the model to your data by calling the fit() method of the model instance.**
- 5. Evaluate model: calculate performance metrics.
- 6. Apply the model to new data

- For supervised learning, predict labels using predict() method.
- For unsupervised learning, transform or infer properties of data using transform() or predict().

Scikit-learn

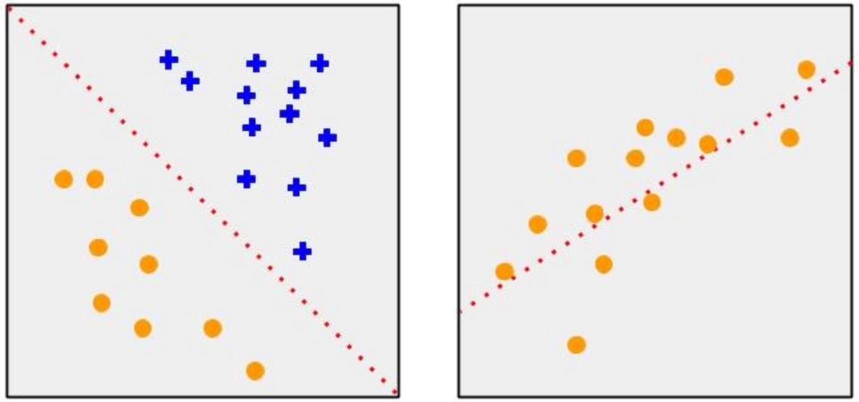
- 1. Read and arrange data into a feature matrix (X) and target vector (Y).
- 2. Choose a class of model by importing the appropriate estimator class from sklearn.
- 3. Choose model hyperparameters by instantiating this class with desired values.
- 4. Fit the model to your data by calling the fit() method of the model instance.
- **5. Evaluate model: calculate performance metrics.**
- 6. Apply the model to new data

- For supervised learning, predict labels using predict() method.
- For unsupervised learning, transform or infer properties of data using transform() or predict().

Scikit-learn

- **Regression problems:**
  - metrics which measure the distance between the model and the data
  - Mean Absolute Error, Mean Squared Error

- **Classification problems:**
  - 0-1 loss: correct 1; incorrect 0.
  - Classification accuracy
  - Applicable to both binary and multiclass.



Classification Regression

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

Scikit-learn

Scoring	Function	Comment
Classification		
'accuracy'	metrics.accuracy_score	
'balanced_accuracy'	metrics.balanced_accuracy_score	
'top_k_accuracy'	metrics.top_k_accuracy_score	
'average_precision'	metrics.average_precision_score	
'neg_brier_score'	metrics.brier_score_loss	
'f1'	metrics.f1_score	for binary targets
'f1_micro'	metrics.f1_score	micro-averaged
'f1_macro'	metrics.f1_score	macro-averaged
'f1_weighted'	metrics.f1_score	weighted average
'f1_samples'	metrics.f1_score	by multilabel sample
'neg_log_loss'	metrics.log_loss	requires `predict_proba` support
'precision' etc.	metrics.precision_score	suffixes apply as with 'f1'
'recall' etc.	metrics.recall_score	suffixes apply as with 'f1'
'jaccard' etc.	metrics.jaccard_score	suffixes apply as with 'f1'
'roc_auc'	metrics.roc_auc_score	
'roc_auc_ovr'	metrics.roc_auc_score	
'roc_auc_ovo'	metrics.roc_auc_score	
'roc_auc_ovr_weighted'	metrics.roc_auc_score	
'roc_auc_ovo_weighted'	metrics.roc_auc_score	
Clustering		
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score	
'adjusted_rand_score'	metrics.adjusted_rand_score	
'completeness_score'	metrics.completeness_score	
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score	
'homogeneity_score'	metrics.homogeneity_score	
'mutual_info_score'	metrics.mutual_info_score	
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score	
'rand_score'	metrics.rand_score	
'v_measure_score'	metrics.v_measure_score	
Regression		
'explained_variance'	metrics.explained_variance_score	
'max_error'	metrics.max_error	
'neg_mean_absolute_error'	metrics.mean_absolute_error	
'neg_mean_squared_error'	metrics.mean_squared_error	
'neg_root_mean_squared_error'	metrics.mean_squared_error	
'neg_mean_squared_log_error'	metrics.mean_squared_log_error	
'neg_median_absolute_error'	metrics.median_absolute_error	
'r2'	metrics.r2_score	
'neg_mean_poisson_deviance'	metrics.mean_poisson_deviance	
'neg_mean_gamma_deviance'	metrics.mean_gamma_deviance	
'neg_mean_absolute_percentage_error'	metrics.mean_absolute_percentage_error	
'd2_absolute_error_score'	metrics.d2_absolute_error_score	
'd2_pinball_score'	metrics.d2_pinball_score	
'd2_tweedie_score'	metrics.d2_tweedie_score	

Summary

- *Machine learning*
  - *Basics*
  - *SciPy*
  - *Scikit-learn*