Artificial Intelligence

National Taipei University

Machine Learning: Supervised and Unsupervised Learning; The Theory of Learning and Ensemble Learning

1141AI04
MBA, IM, NTPU (M5276) (Fall 2025)
Tue 2, 3, 4 (9:10-12:00) (B3F17)

https://meet.google.com/paj-zhhj-mya

NVIDIA University Ambassador Certified Instructor
aws educate Cloud Ambassador 2020 Cohort
aws academy Accredited Educator
aws certified Cloud Practitioner
aws certified Solutions Architect Associate

Min-Yuh Day, Ph.D,
Professor and Director
Institute of Information Management, National Taipei University
https://web.ntpu.edu.tw/~myday

2025-10-07

Syllabus

Week	Date	Subject/Topics
1	2025/09/09	Introduction to Artificial Intelligence
2	2025/09/16	Artificial Intelligence and Intelligent Agents; Problem Solving
3	2025/09/23	Knowledge, Reasoning and Knowledge Representation; Uncertain Knowledge and Reasoning
4	2025/09/30	Case Study on Artificial Intelligence I
5	2025/10/07	Machine Learning: Supervised and Unsupervised Learning; The Theory of Learning and Ensemble Learning

Syllabus

Week	Date	Subject/Topics
6	2025/10/14	NVIDIA Fundamentals of Deep Learning I: Deep Learning; Neural Networks
7	2025/10/21	NVIDIA Fundamentals of Deep Learning II: Convolutional Neural Networks; Data Augmentation and Deployment
8	2025/10/28	Self-Learning
9	2025/11/04	Midterm Project Report
10	2025/11/11	NVIDIA Fundamentals of Deep Learning III: Pre-trained Models; Natural Language Processing

Syllabus

Week	Date	Subject/Topics
11	2025/11/18	Case Study on Artificial Intelligence II
12	2025/11/25	Computer Vision and Robotics
13	2025/12/02	Generative AI, Agentic AI, and Physical AI
14	2025/12/09	Philosophy and Ethics of AI and the Future of AI
15	2025/12/16	Final Project Report I
16	2025/12/23	Final Project Report II

Machine Learning:

Supervised and Unsupervised Learning; The Theory of Learning and Ensemble Learning

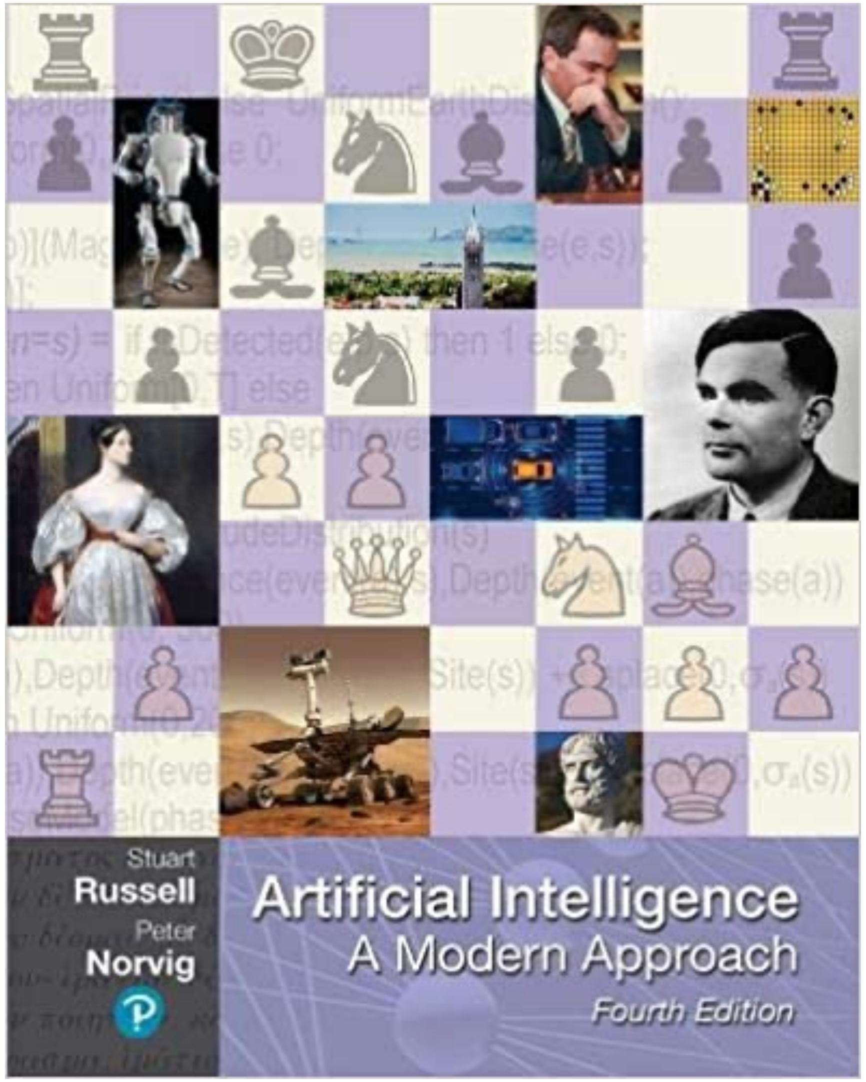
Outline

- Machine Learning
    - Supervised Learning
    - Unsupervised Learning
- The Theory of Learning
    - Computational Learning Theory
    - Probably Approximately Correct (PAC) Learning
- Ensemble Learning
    - Bagging: Random Forests (RF)
    - Boosting: Gradient Boosting, XGBoost, LightGBM, CatBoost
    - Stacking
    - Online learning
- Meta Learning: Learning to Learn

Stuart Russell and Peter Norvig (2020),

Artificial Intelligence: A Modern Approach,

4th Edition, Pearson



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

https://www.amazon.com/Artificial-Intelligence-A-Modern-Approach/dp/0134610997/

Artificial Intelligence: A Modern Approach

1. **Artificial Intelligence**
2. **Problem Solving**
3. **Knowledge and Reasoning**
4. **Uncertain Knowledge and Reasoning**
5. **Machine Learning**
6. **Communicating, Perceiving, and Acting**
7. **Philosophy and Ethics of AI**

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Artificial Intelligence:

Machine Learning

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Artificial Intelligence:
5. Machine Learning

- Learning from Examples
- Learning Probabilistic Models
- Deep Learning
- Reinforcement Learning

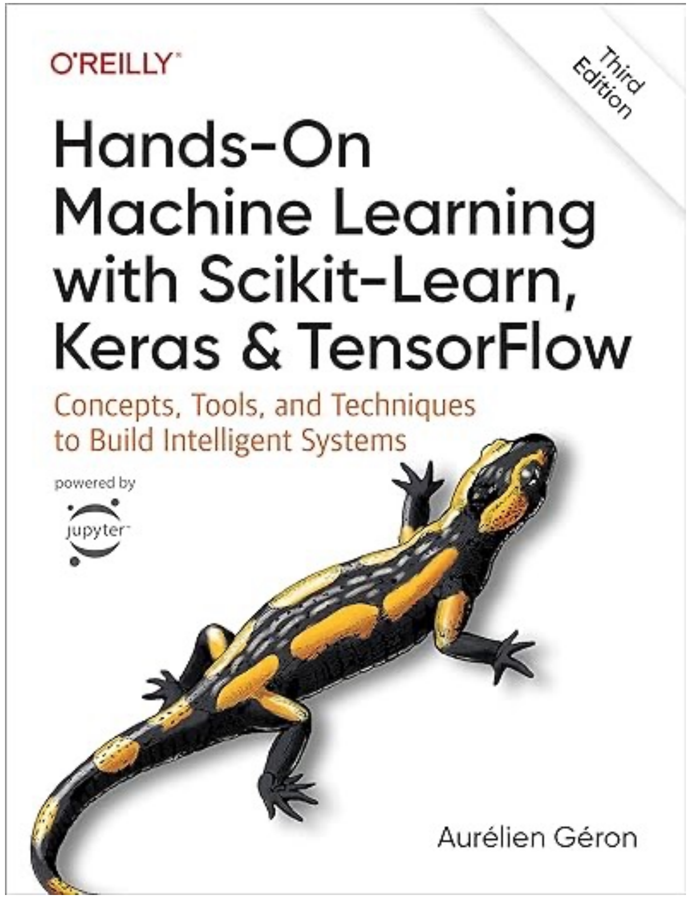
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Aurélien Géron (2022),

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow:

Concepts, Tools, and Techniques to Build Intelligent Systems,

3rd Edition, O’Reilly Media



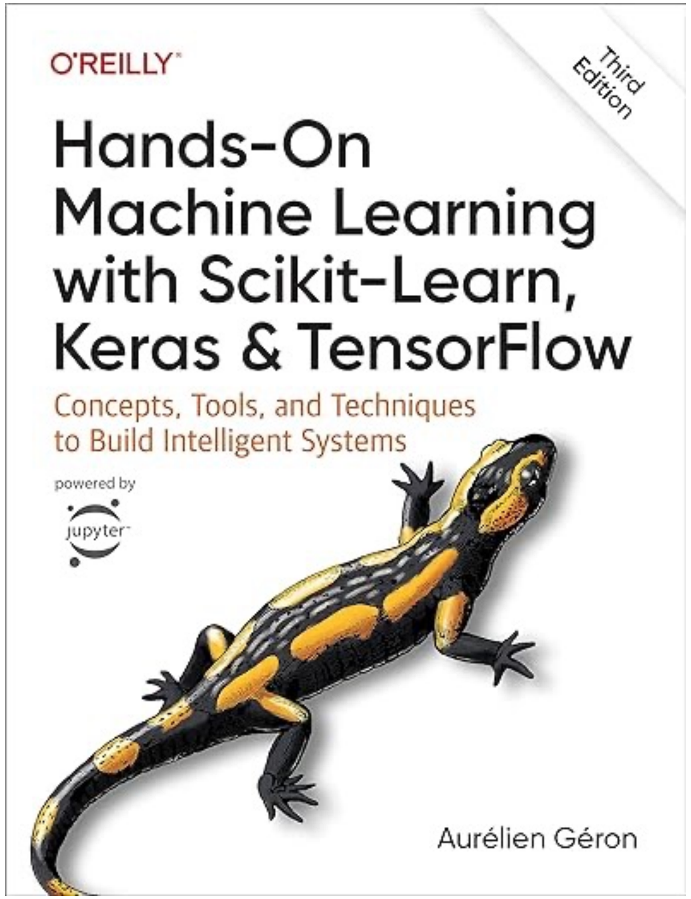
https://github.com/ageron/handson-ml3

Source: https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1098125975

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks

1.The Machine Learning landscape
2.End-to-end Machine Learning project
3.Classification
4.Training Models
5.Support Vector Machines
6.Decision Trees
7.Ensemble Learning and Random Forests
8.Dimensionality Reduction
9.Unsupervised Learning Techniques
10.Artificial Neural Nets with Keras
11.Training Deep Neural Networks
12.Custom Models and Training with TensorFlow
13.Loading and Preprocessing Data
14.Deep Computer Vision Using Convolutional Neural Networks
15.Processing Sequences Using RNNs and CNNs
16.Natural Language Processing with RNNs and Attention
17.Autoencoders, GANs, and Diffusion Models
18.Reinforcement Learning
19.Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml3

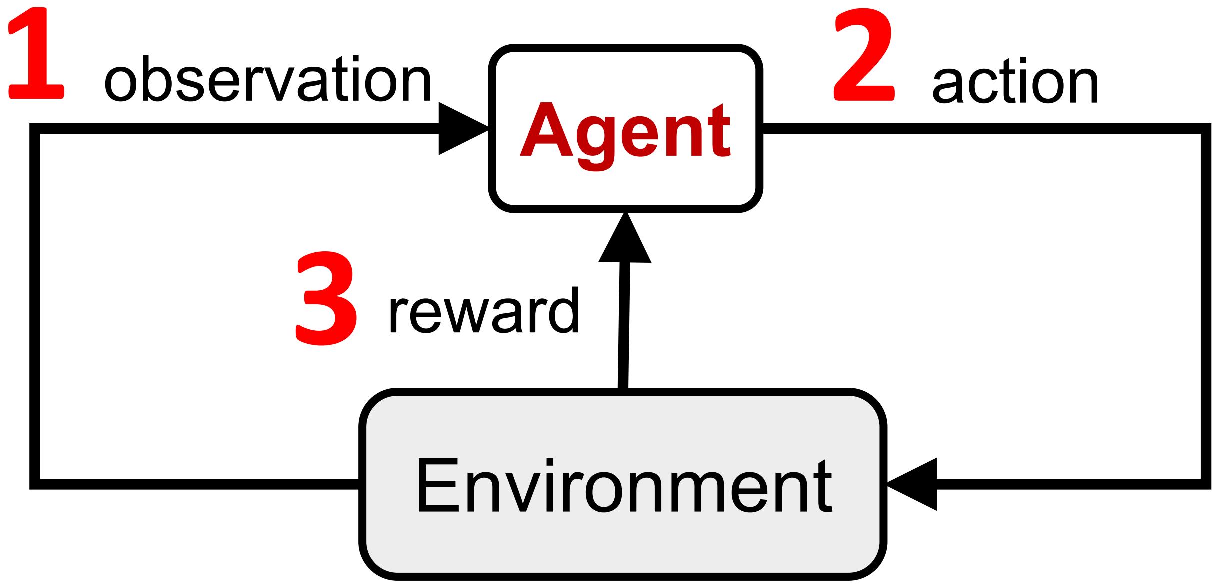
Reinforcement Learning (DL)

Agent

Environment

Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Reinforcement Learning (DL)



1 observation

Agent

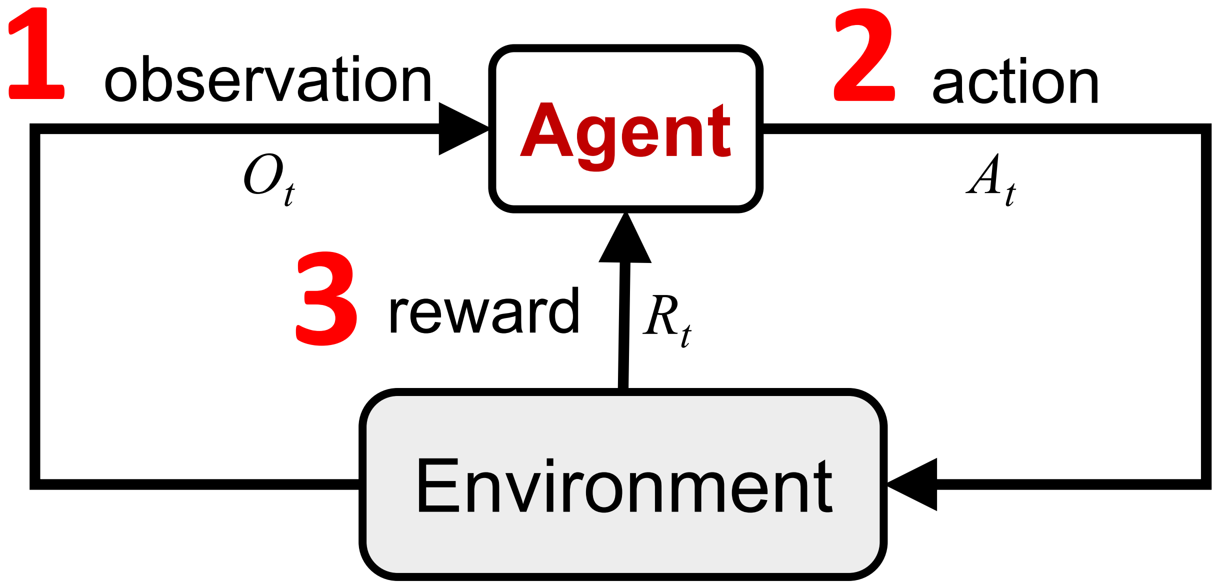
2 action

3 reward

Environment

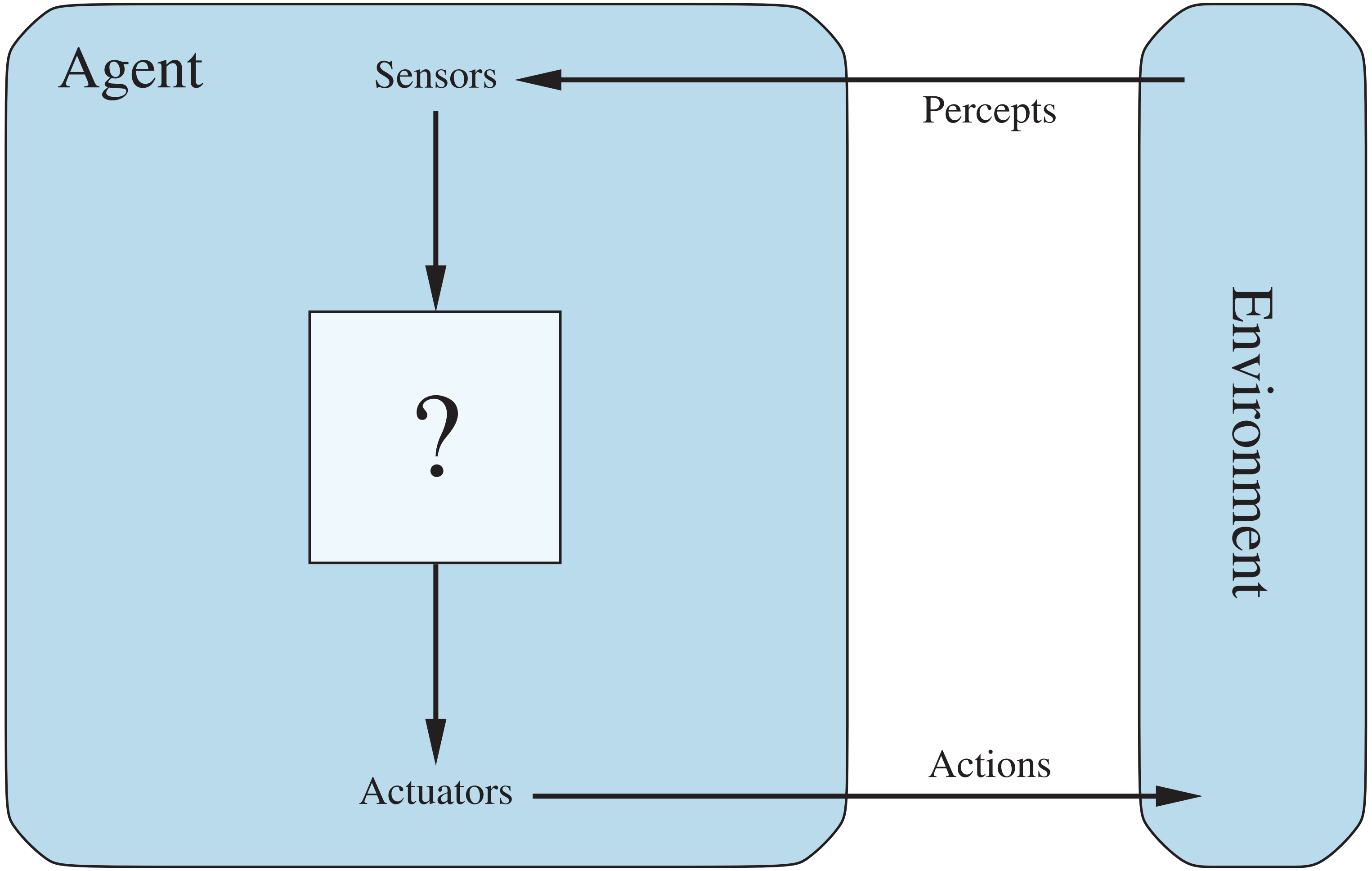
Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Reinforcement Learning (DL)



Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Agents interact with environments through sensors and actuators



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

x	y

Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

x	y
input	*Output* *label*

Iris flower data set

setosa

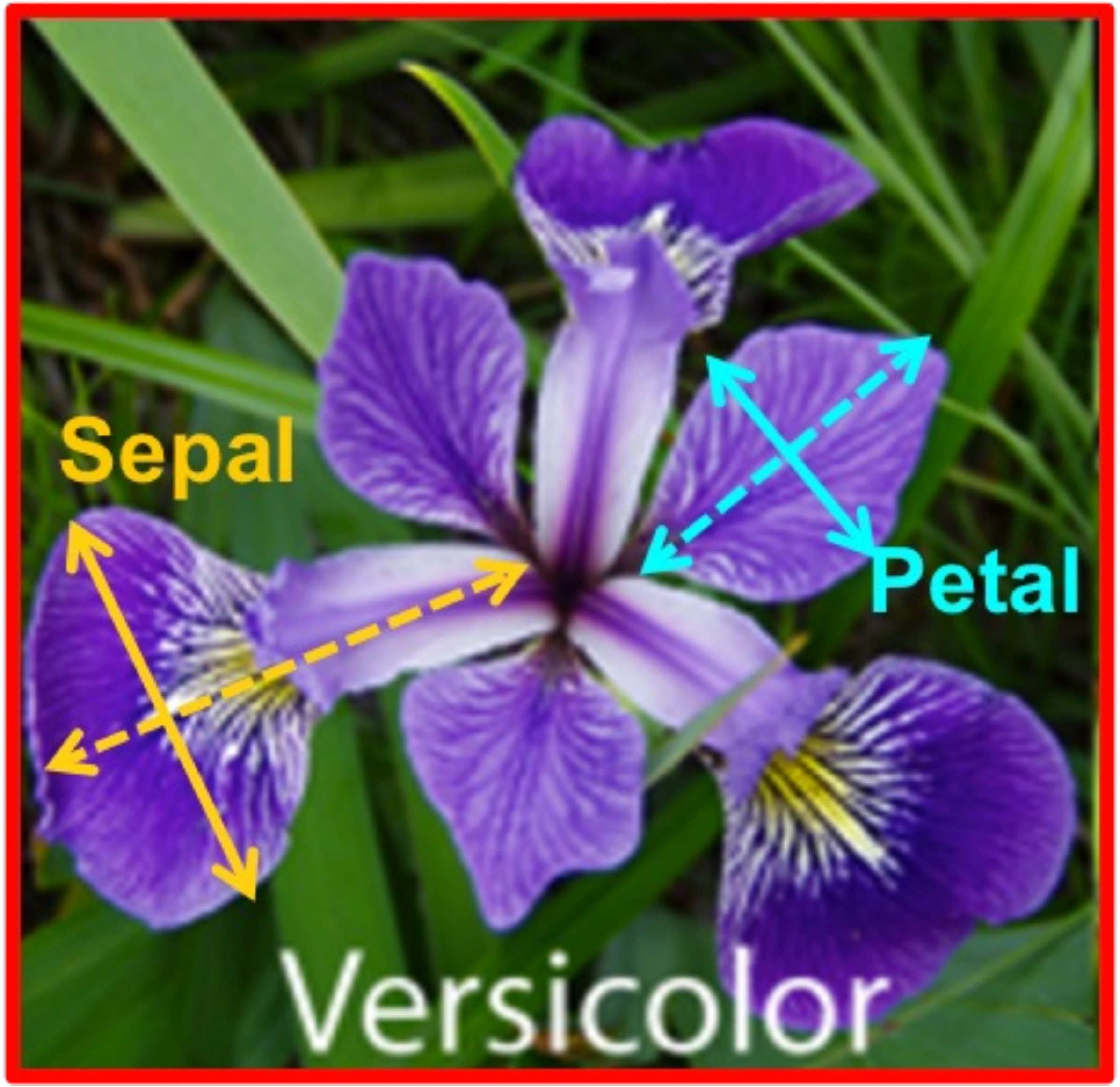
versicolor

virginica

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

Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

Iris Classfication



Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

iris.data

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
```

setosa

virginica

versicolor

Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

Machine Learning
Supervised Learning (Classification)
Learning from Examples

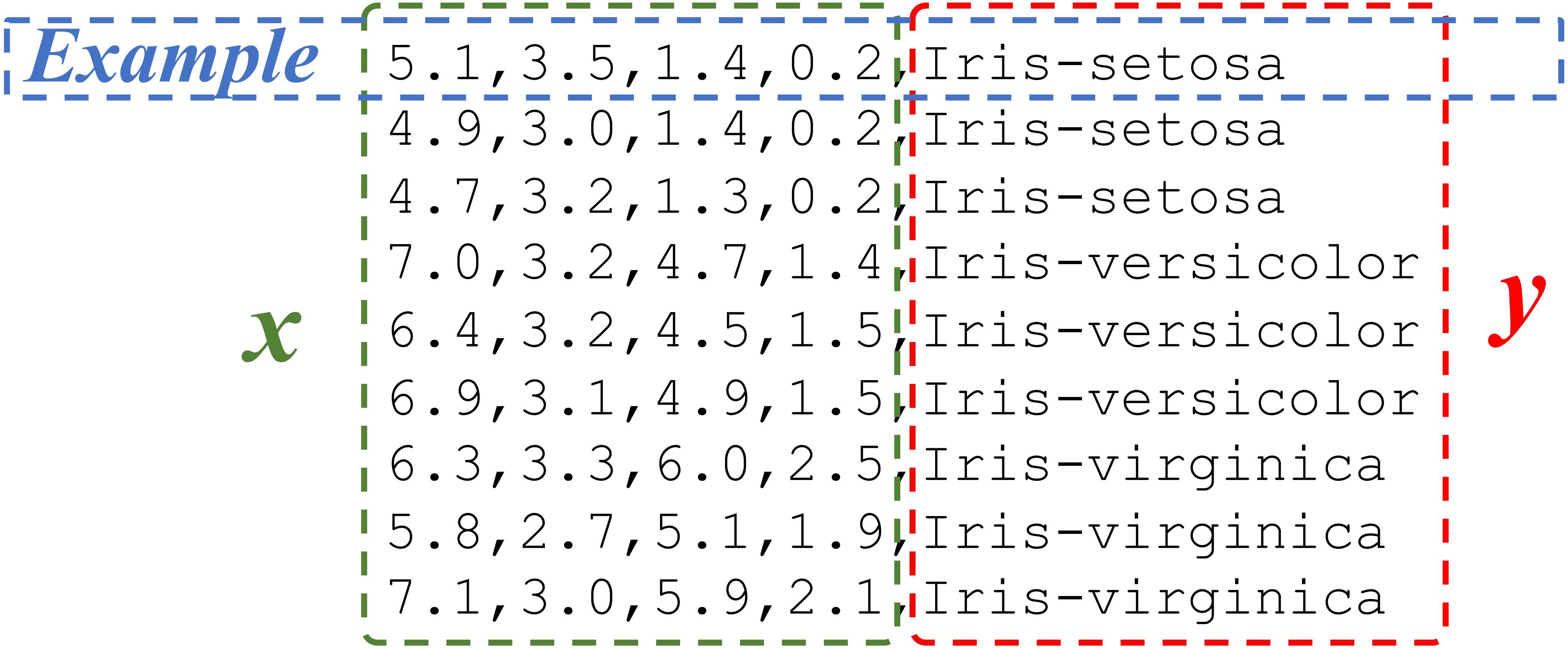
$$y = f(x)$$

Example `5.1,3.5,1.4,0.2,Iris-setosa`

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$



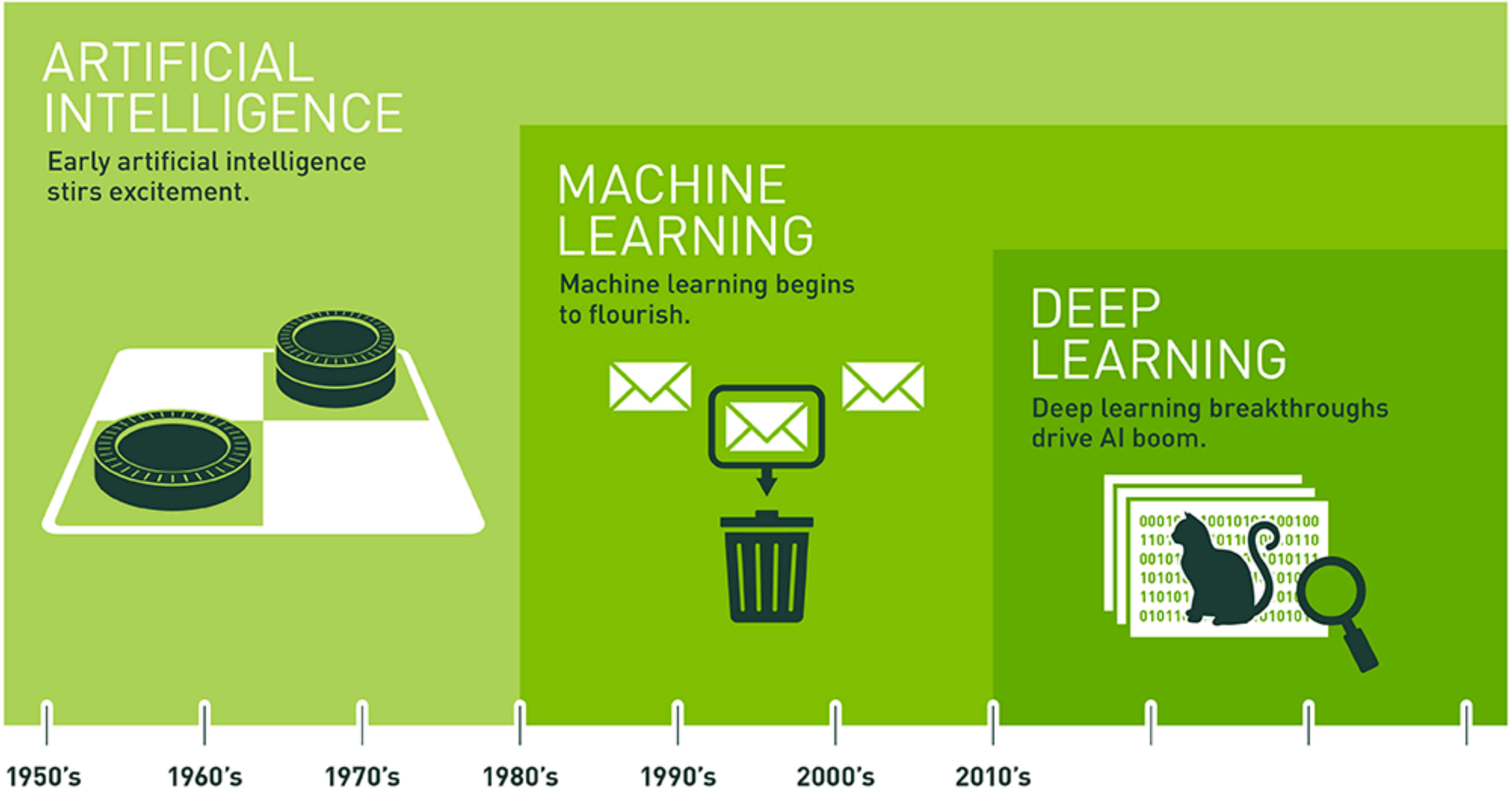
Example

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

x

y

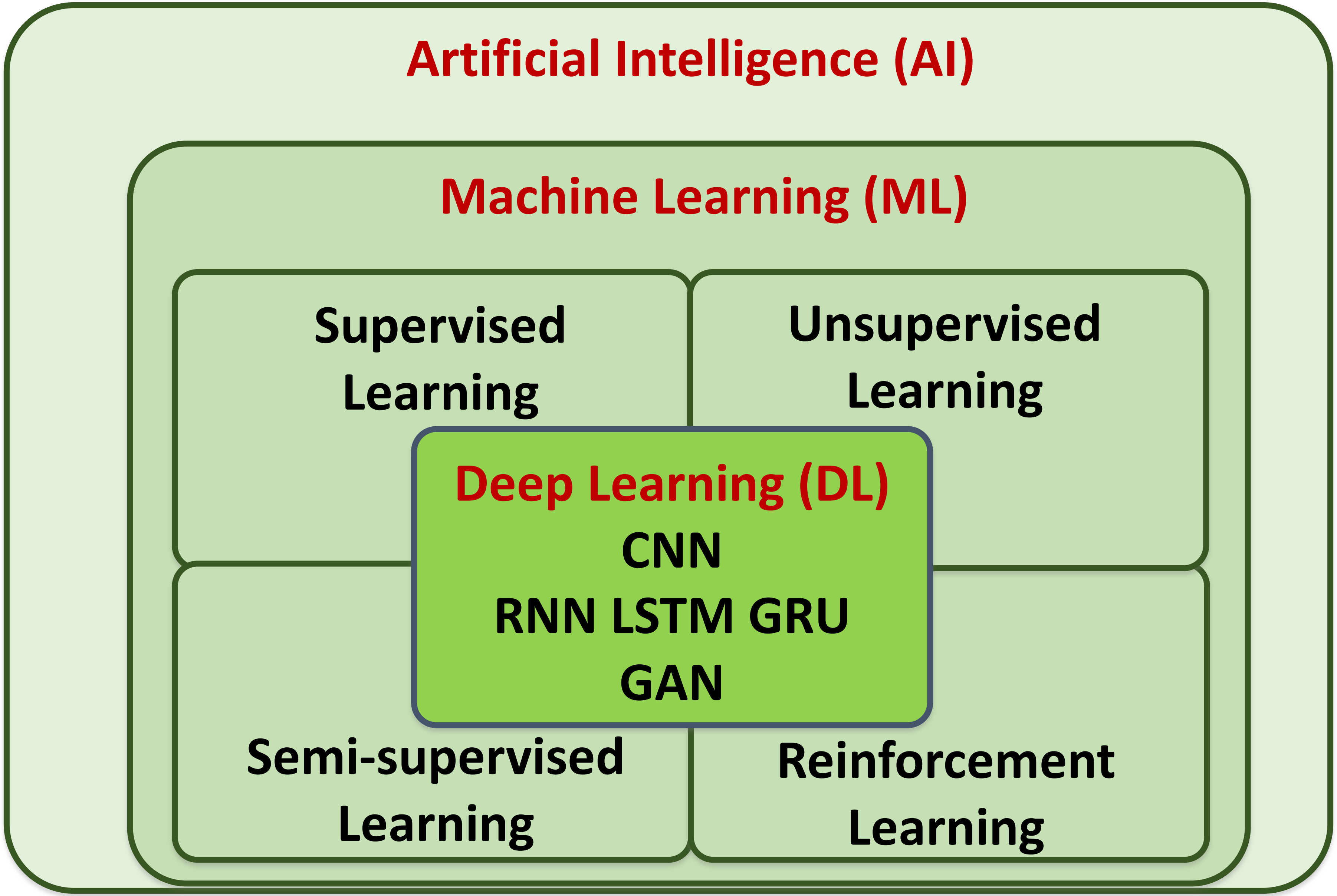
Artificial Intelligence
Machine Learning & Deep Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

AI, ML, DL



Artificial Intelligence (AI)

Machine Learning (ML)

Supervised Learning

Unsupervised Learning

Deep Learning (DL)

CNN

RNN LSTM GRU

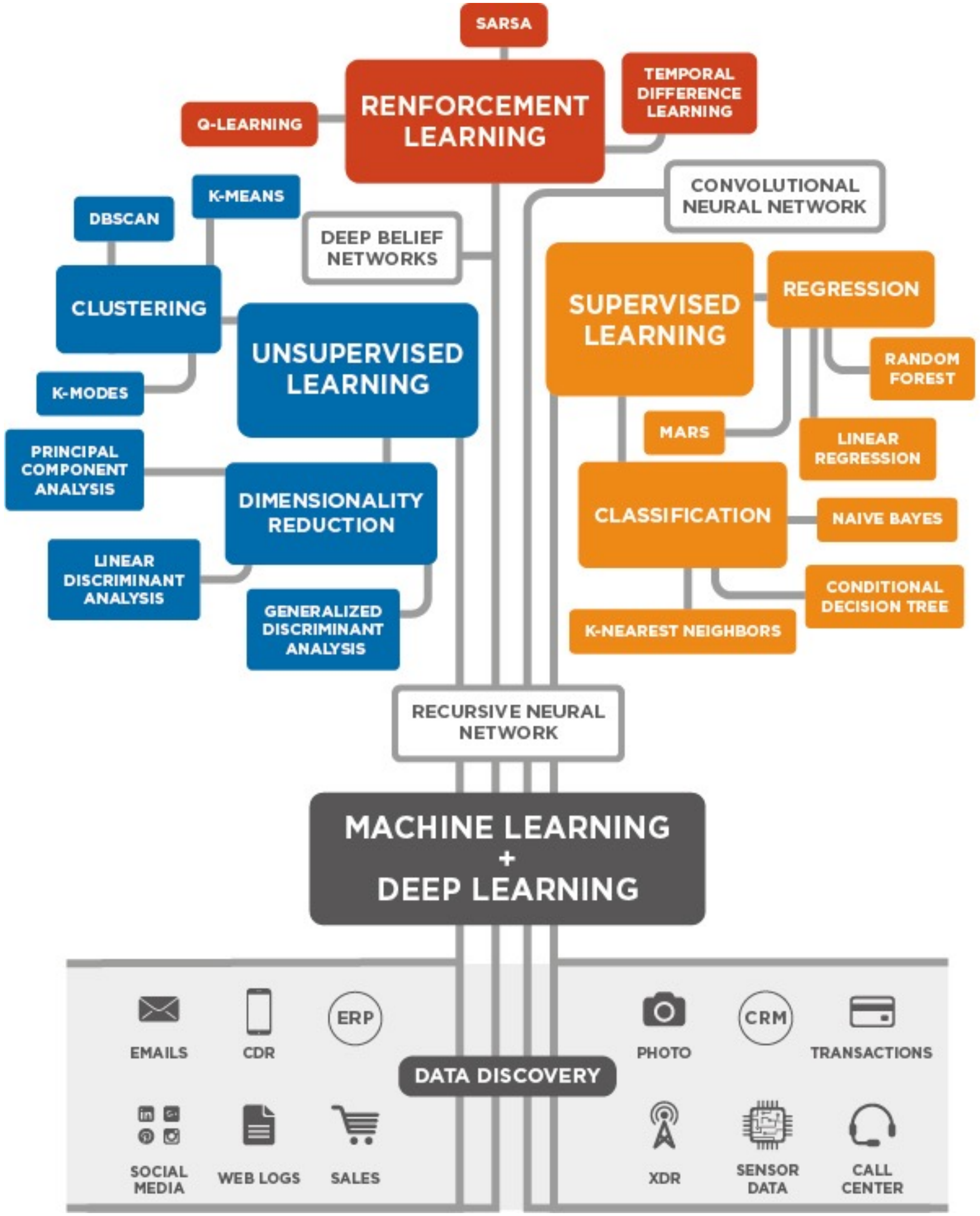
GAN

Semi-supervised Learning

Reinforcement Learning

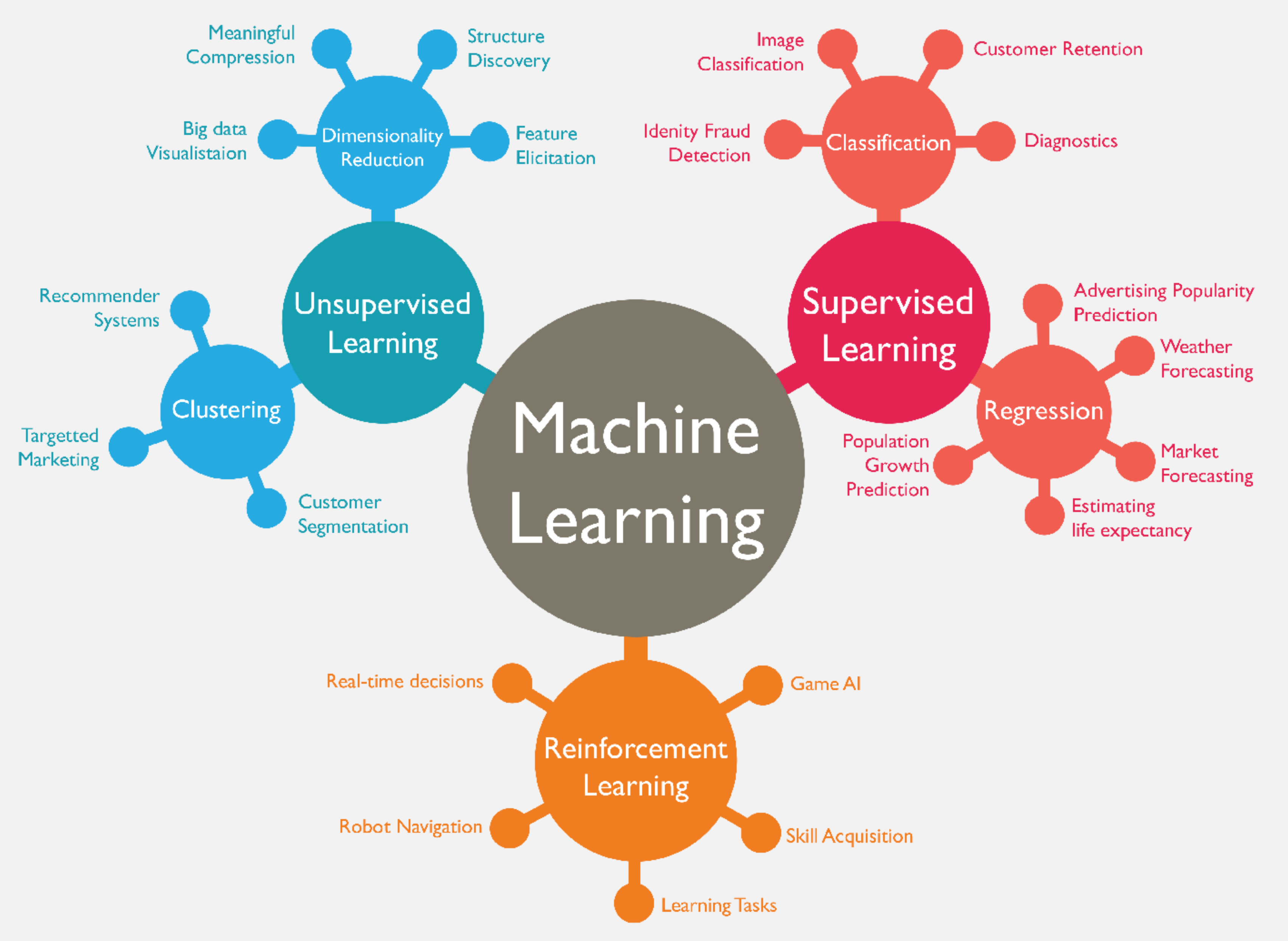
Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

3 Machine Learning Algorithms



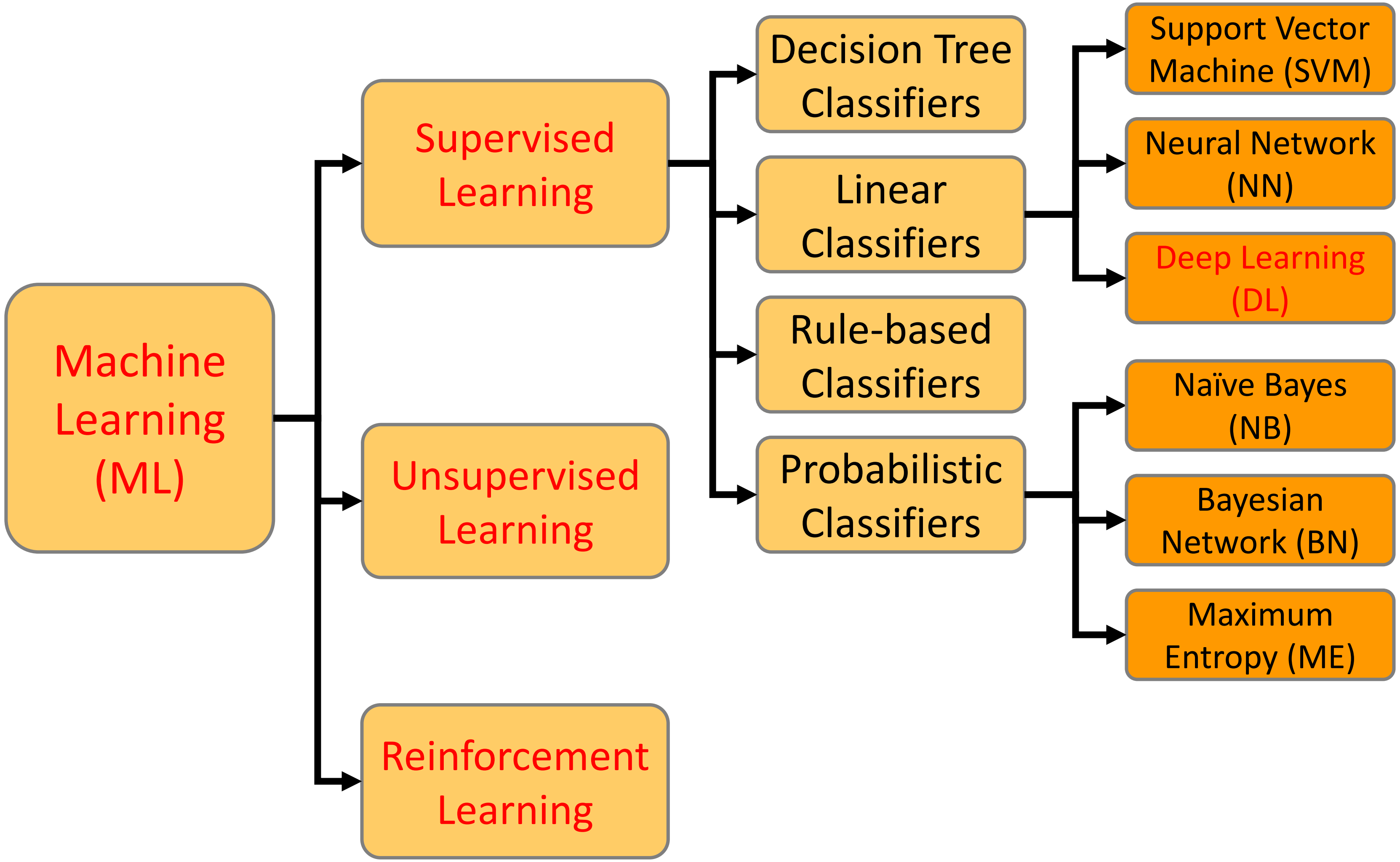
Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

Machine Learning (ML)



Source: https://www.mactores.com/services/aws-big-data-machine-learning-cognitive-services/

Machine Learning (ML) / Deep Learning (DL)



Machine Learning (ML)

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Decision Tree Classifiers

Linear Classifiers

Rule-based Classifiers

Probabilistic Classifiers

Support Vector Machine (SVM)

Neural Network (NN)

Deep Learning (DL)

Naïve Bayes (NB)

Bayesian Network (BN)

Maximum Entropy (ME)

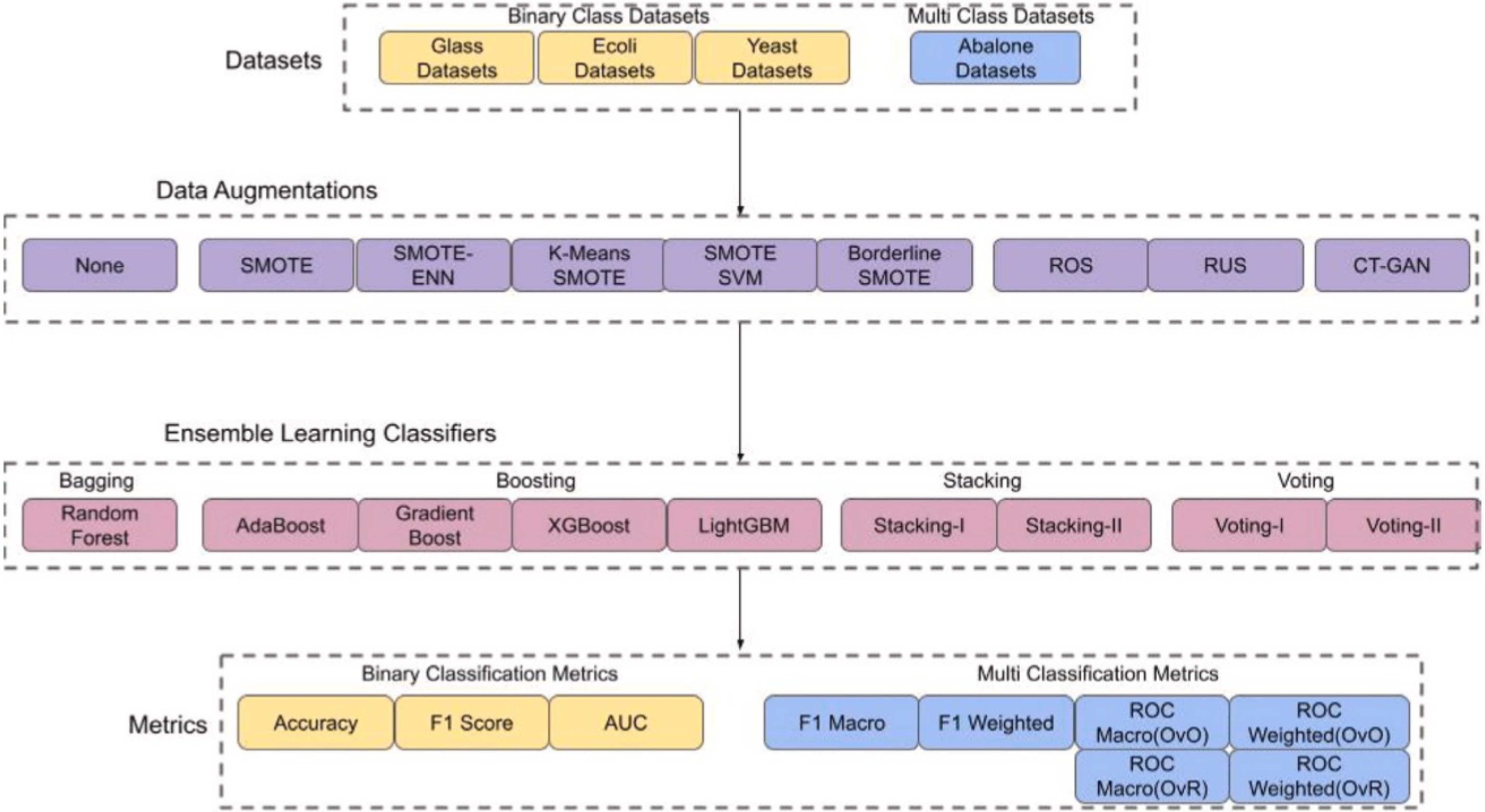
Source: Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero, and Enrique Herrera-Viedma (2015),
"Sentiment analysis: A review and comparative analysis of web services," Information Sciences, 311, pp. 18-38.

Machine Learning Models

- Deep Learning
- Association rules
- Decision tree
- Clustering
- Bayesian
- Kernel
- Ensemble
- Dimensionality reduction
- Regression Analysis
- Instance based

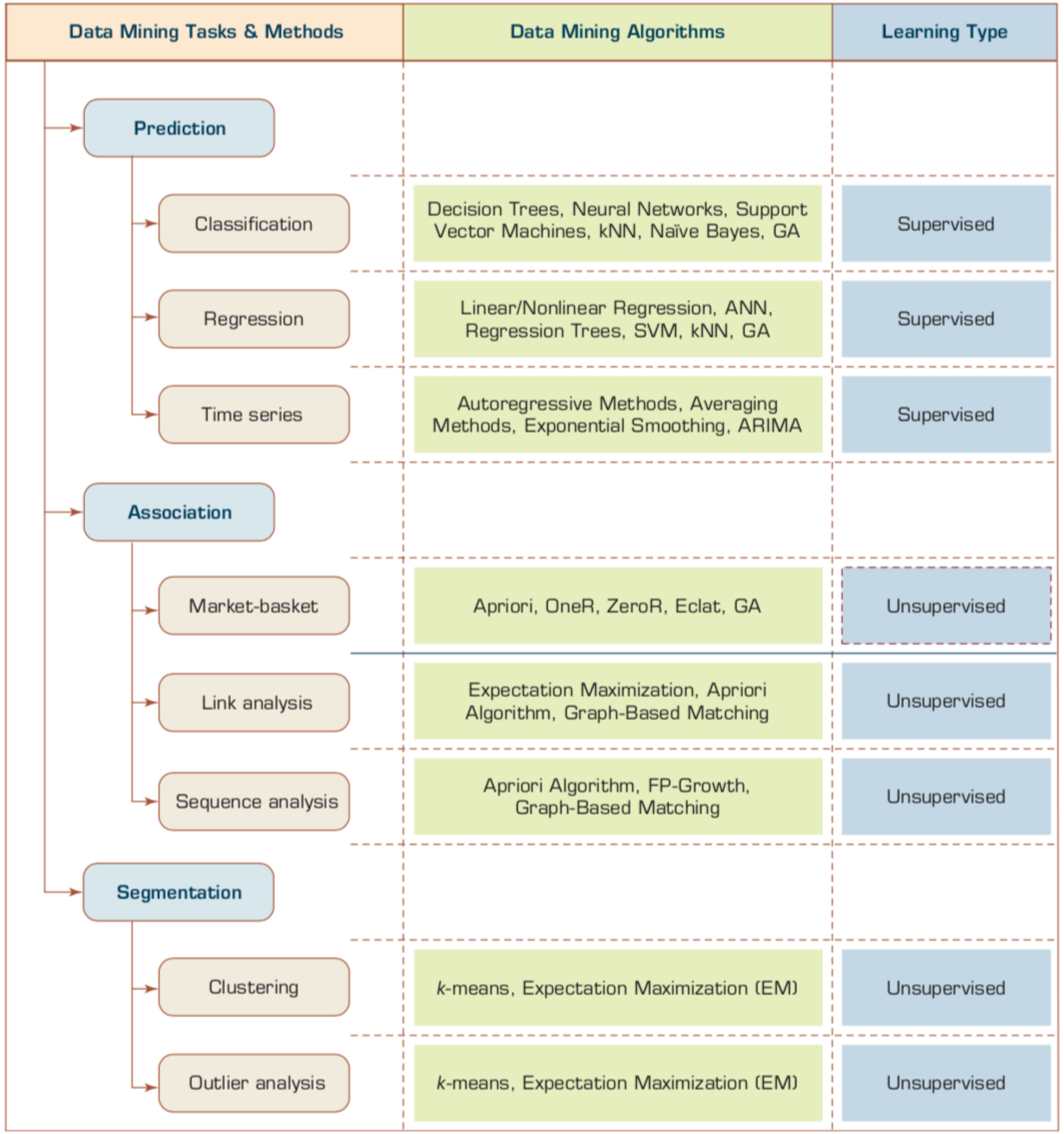
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Ensemble Learning and Data Augmentations (DA)



Source: Khan, A. A., Chaudhari, O., & Chandra, R. (2023). A review of ensemble learning and data augmentation models for class imbalanced problems: combination, implementation and evaluation. Expert Systems with Applications, 122778.

Machine Learning: Data Mining Tasks & Methods



Data Mining Tasks & Methods	Data Mining Algorithms	Learning Type
Prediction		
Classification	Decision Trees, Neural Networks, Support Vector Machines, kNN, Naïve Bayes, GA	Supervised
Regression	Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA	Supervised
Time series	Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA	Supervised
Association		
Market-basket	Apriori, OneR, ZeroR, Eclat, GA	Unsupervised
Link analysis	Expectation Maximization, Apriori Algorithm, Graph-Based Matching	Unsupervised
Sequence analysis	Apriori Algorithm, FP-Growth, Graph-Based Matching	Unsupervised
Segmentation		
Clustering	*k*-means, Expectation Maximization (EM)	Unsupervised
Outlier analysis	*k*-means, Expectation Maximization (EM)	Unsupervised

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Data Mining Methods

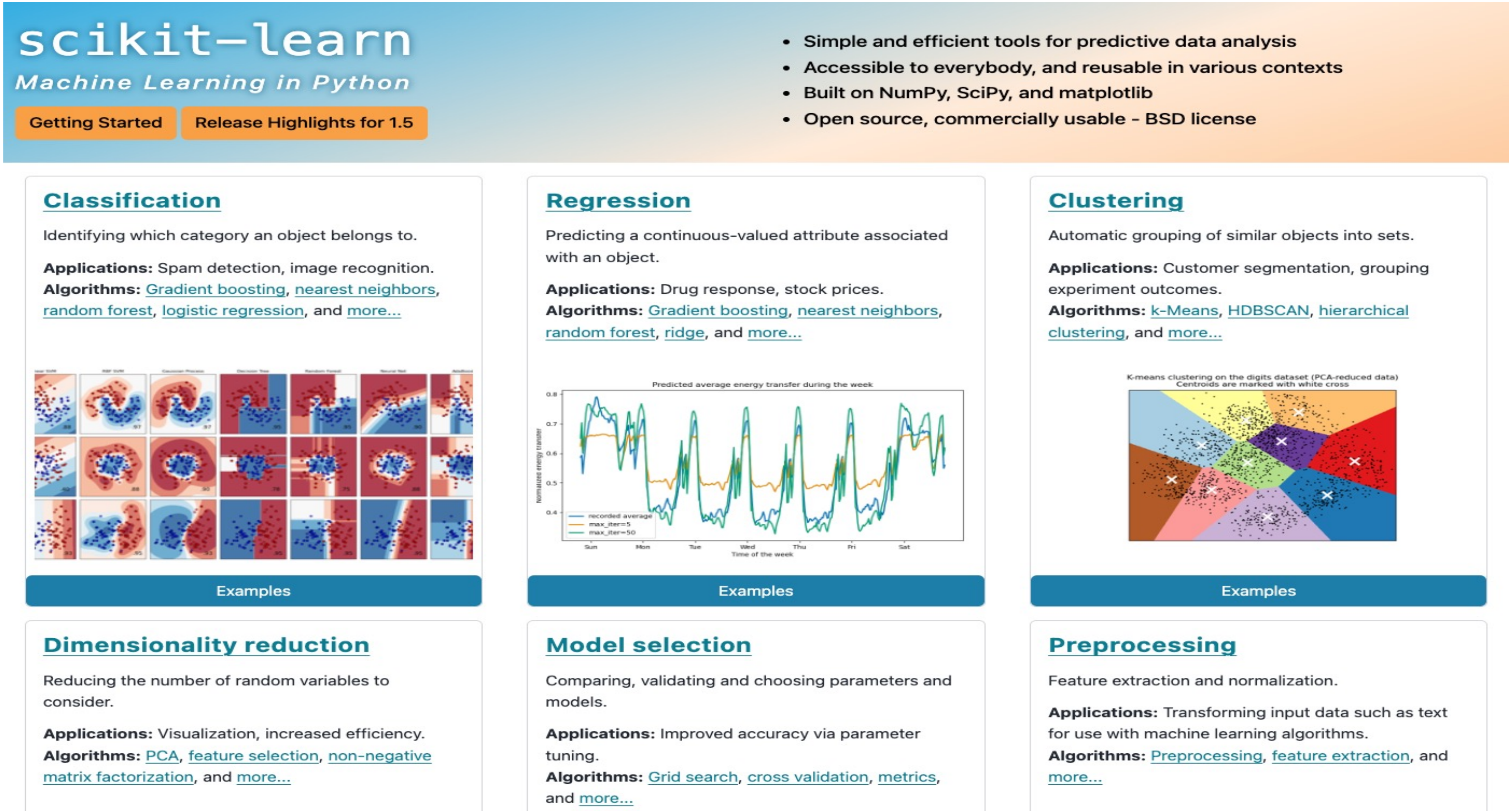
- **Supervised Learning**
  - **Classification**
    - Class Label Prediction
  - **Regression**
    - Numeric Value Prediction
- **Unsupervised Learning**
  - **Clustering**
  - **Association**

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Scikit-Learn

Machine Learning in Python

Scikit-Learn



scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency.
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning.
Algorithms: Grid search, cross validation, metrics, and more...

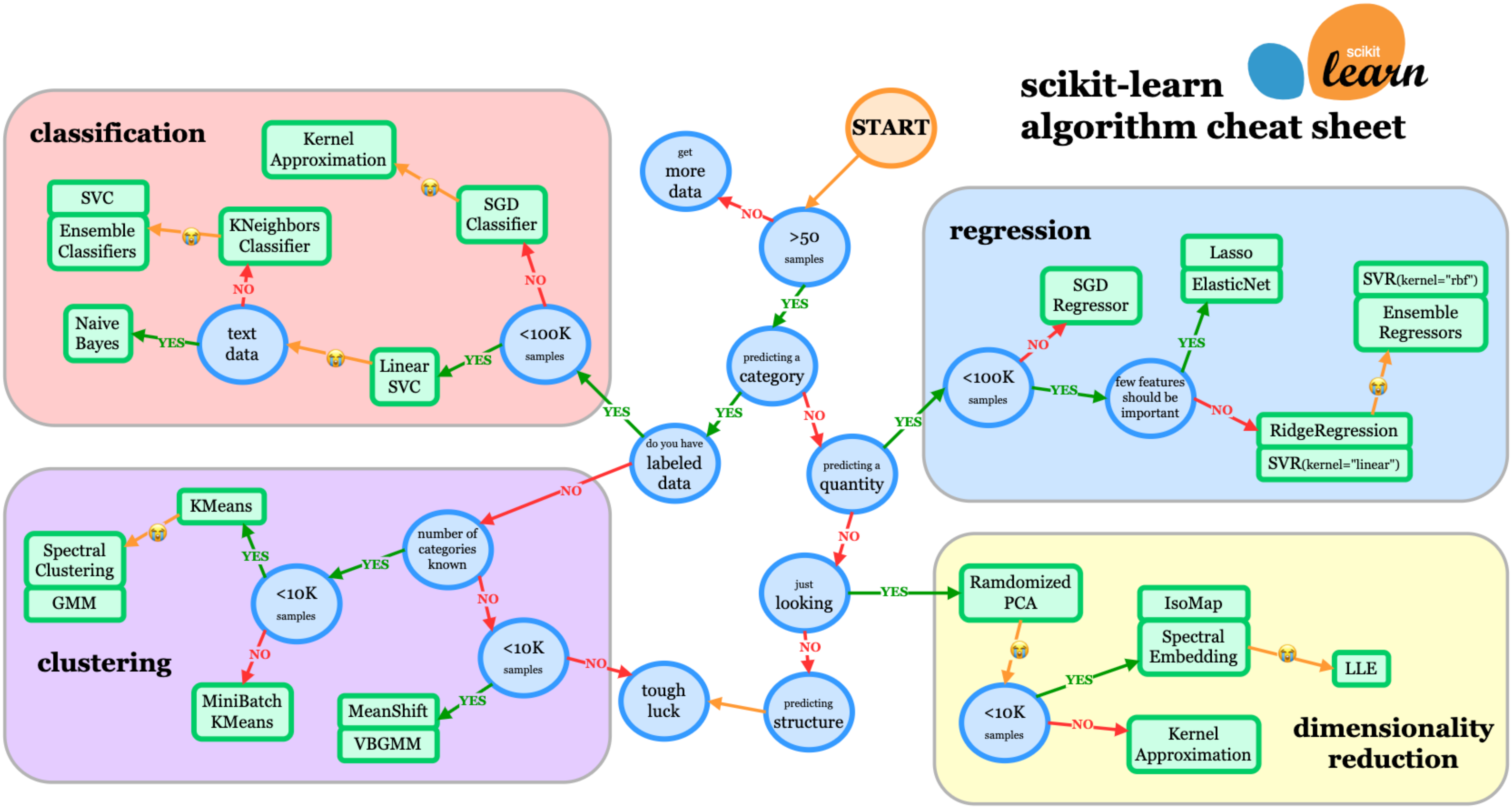
Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: Preprocessing, feature extraction, and more...

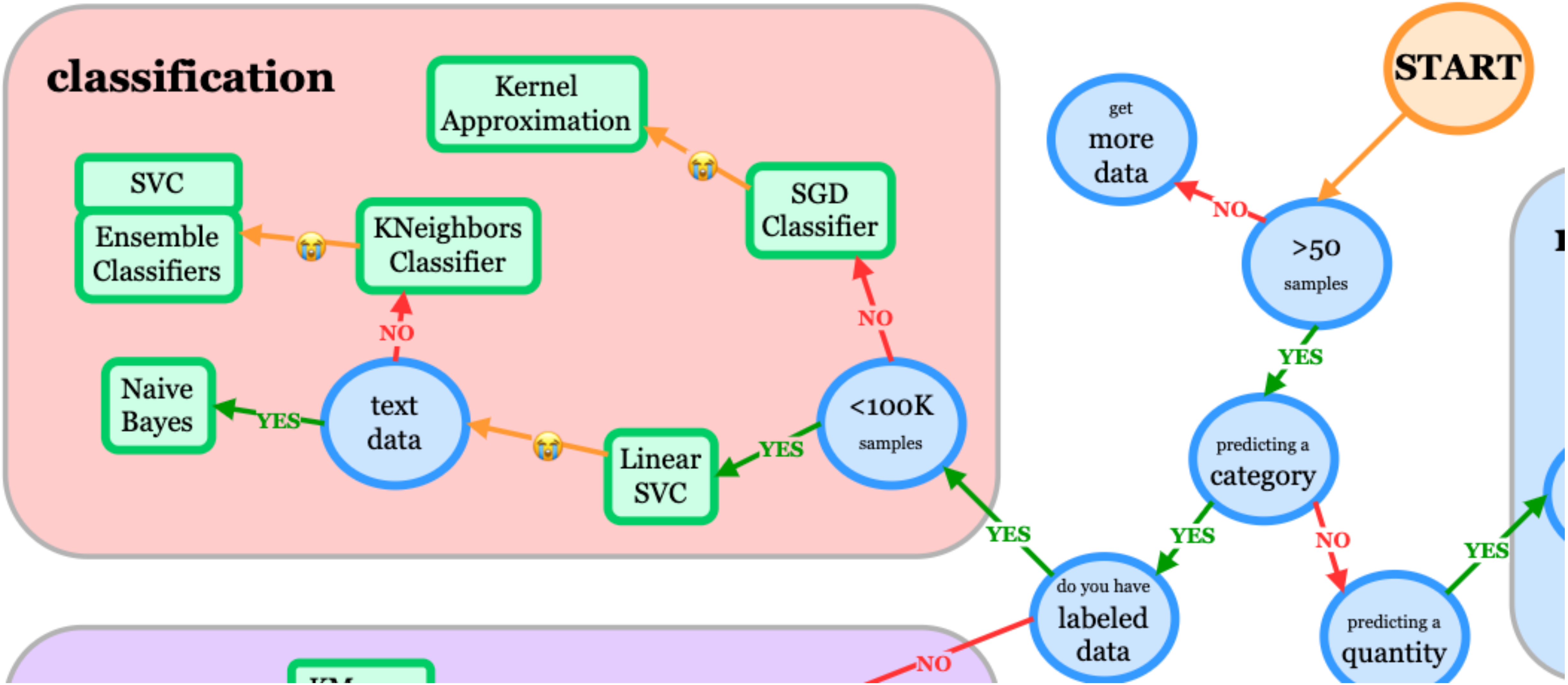
Source: http://scikit-learn.org/

Scikit-Learn Machine Learning Map



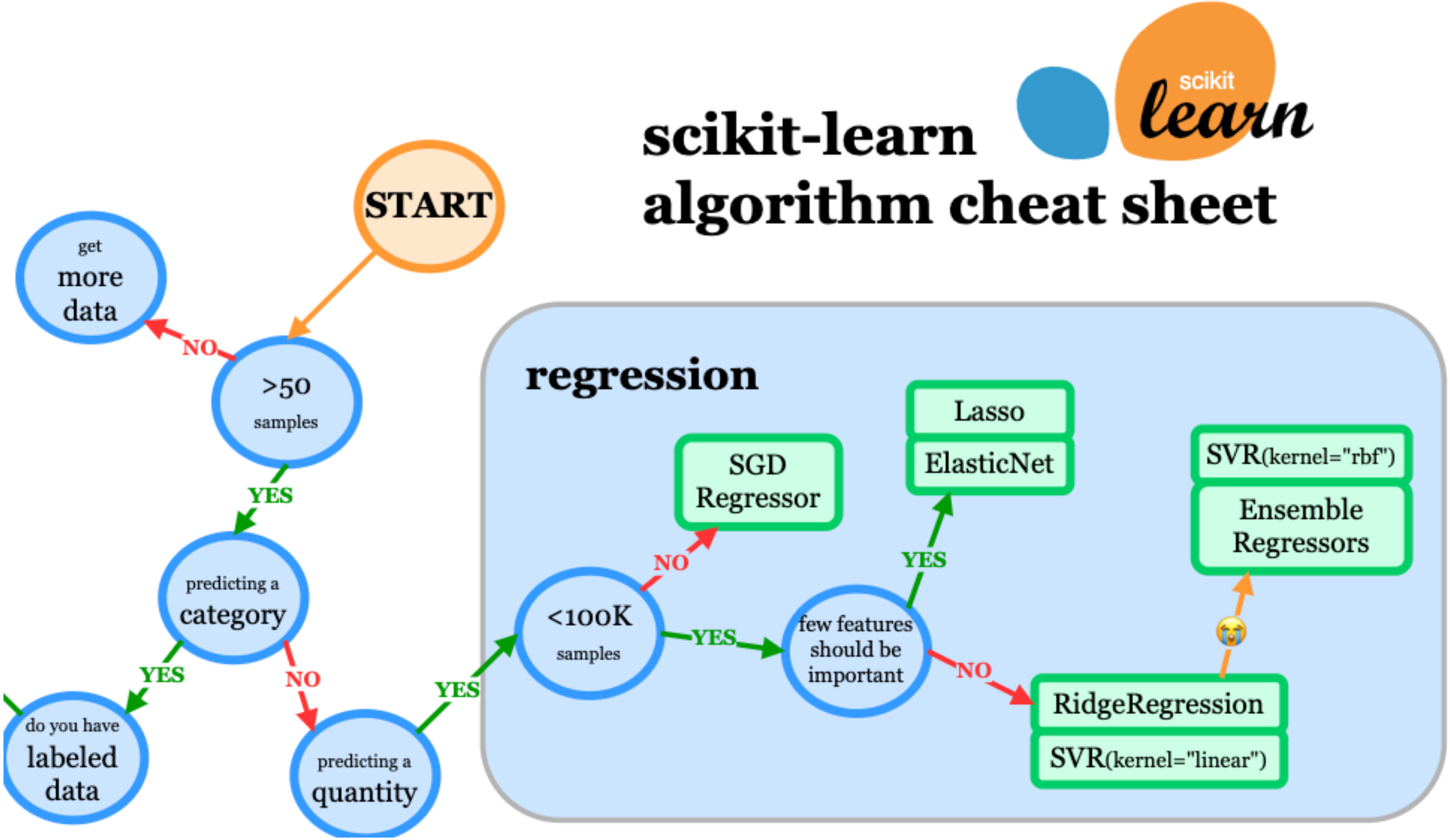
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



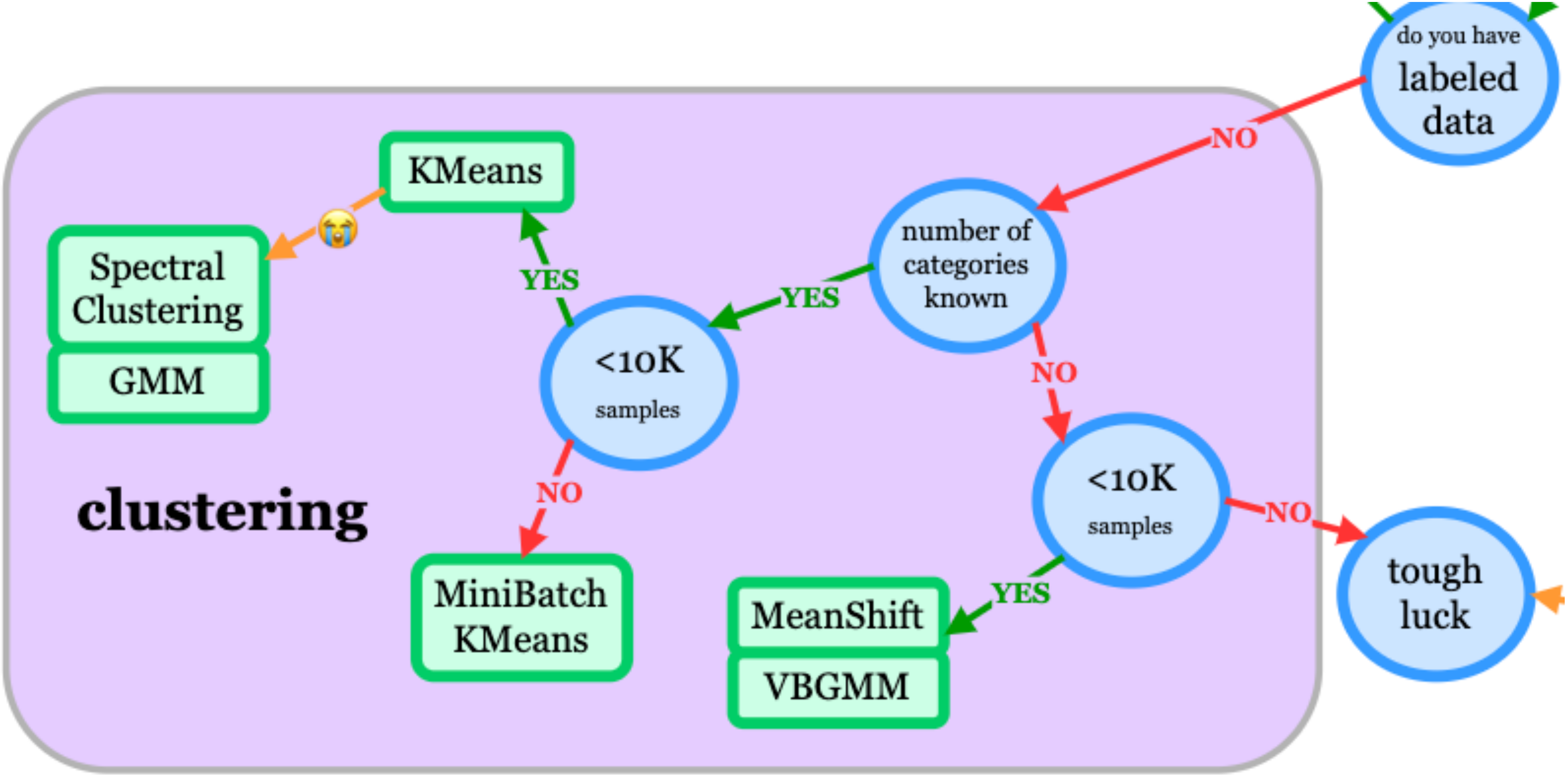
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



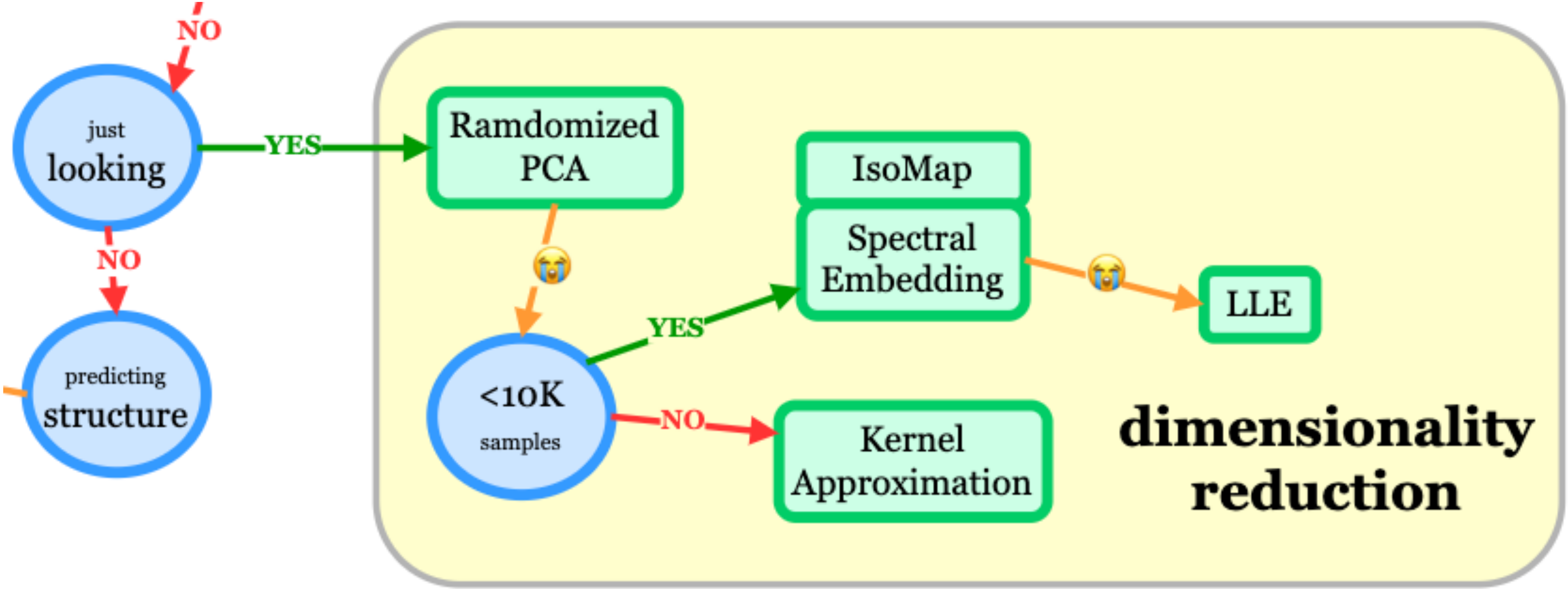
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



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

Scikit-Learn Machine Learning Map



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

Iris flower data set

setosa

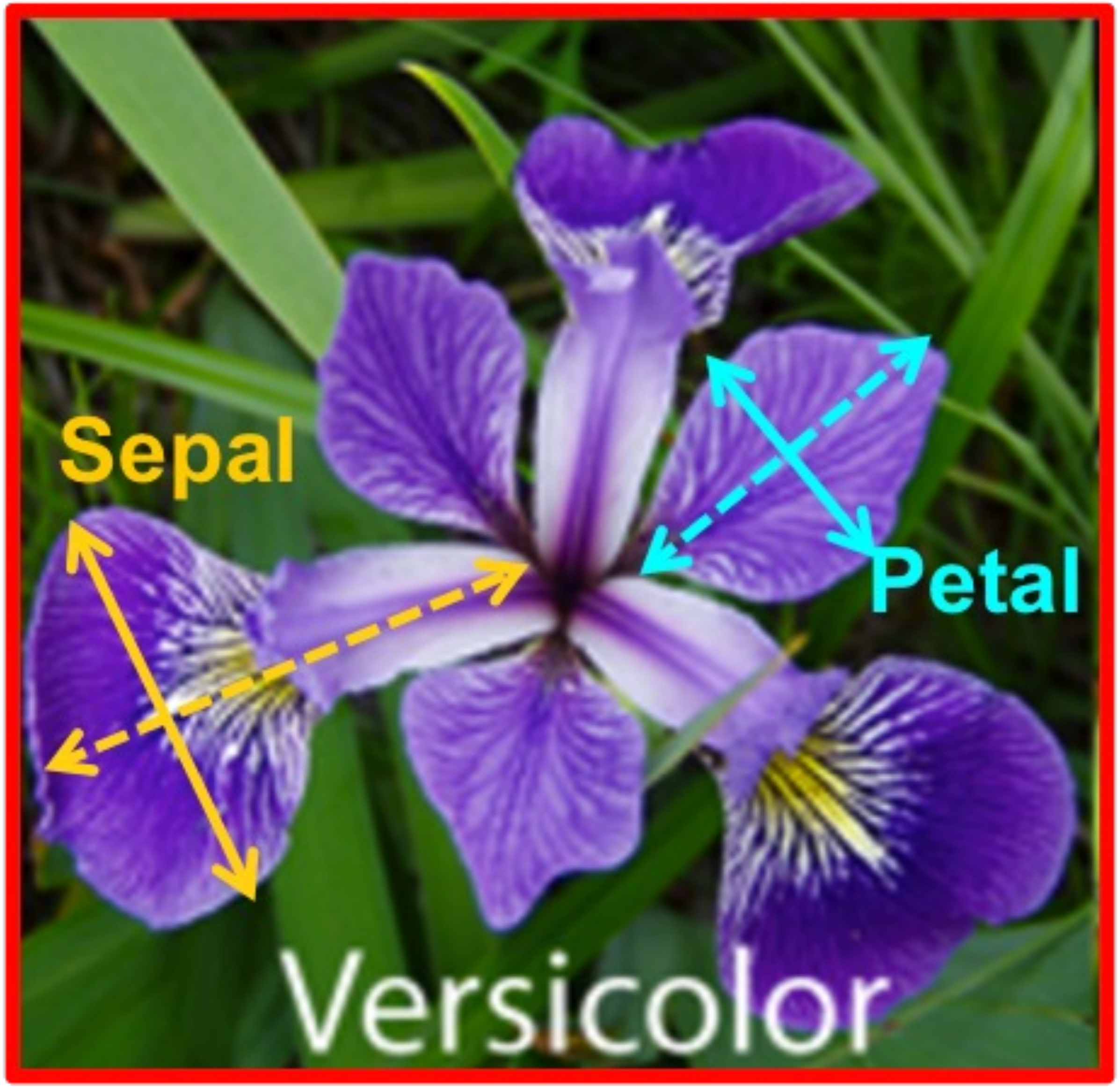
versicolor

virginica

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

Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

Iris Classfication



Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

iris.data

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
```

setosa

virginica

versicolor

Machine Learning
Supervised Learning (Classification)
Learning from Examples

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$

x

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

y

Machine Learning

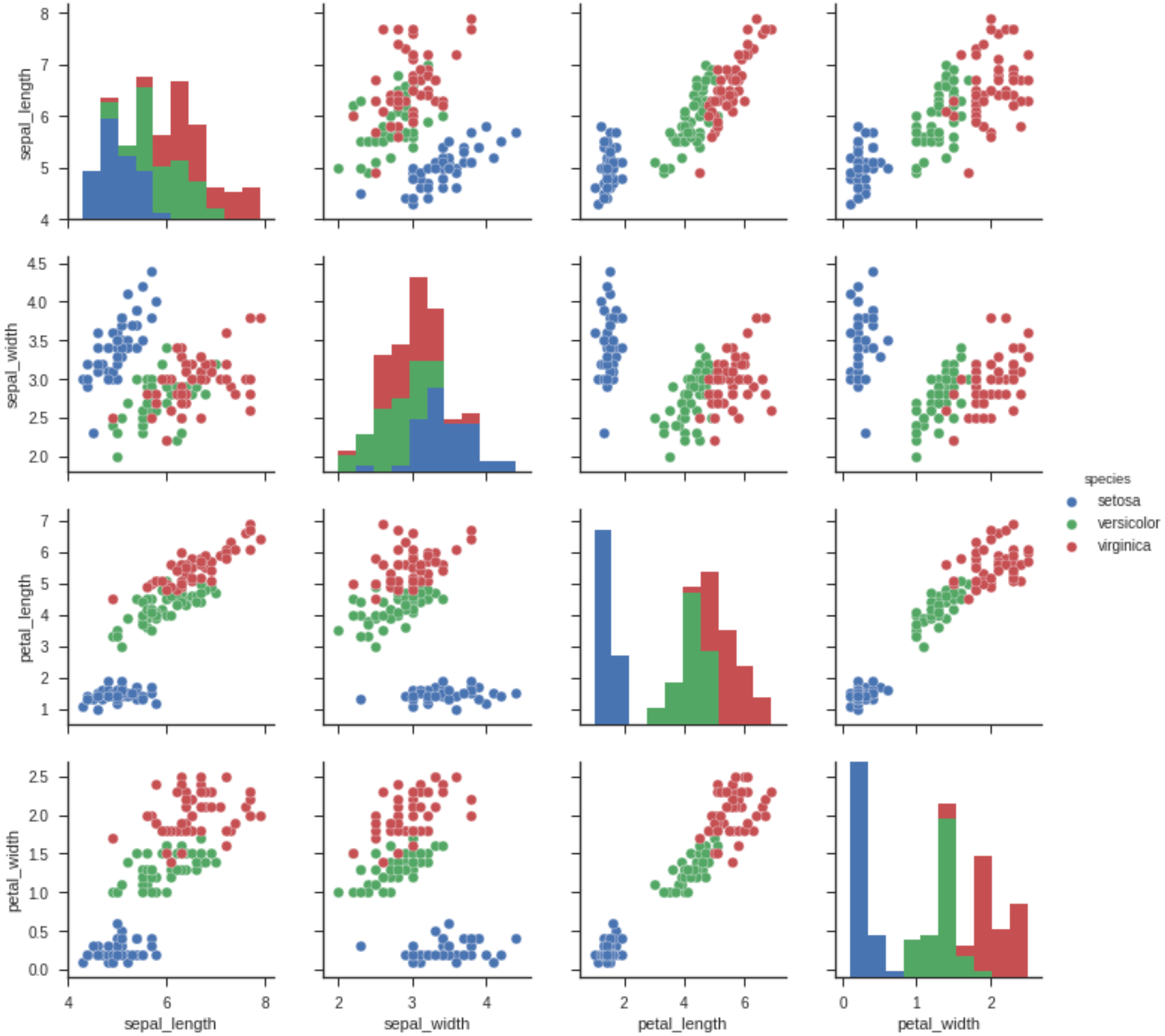
Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

$$\boxed{x} \quad \boxed{y}$$

Iris Data Visualization



Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

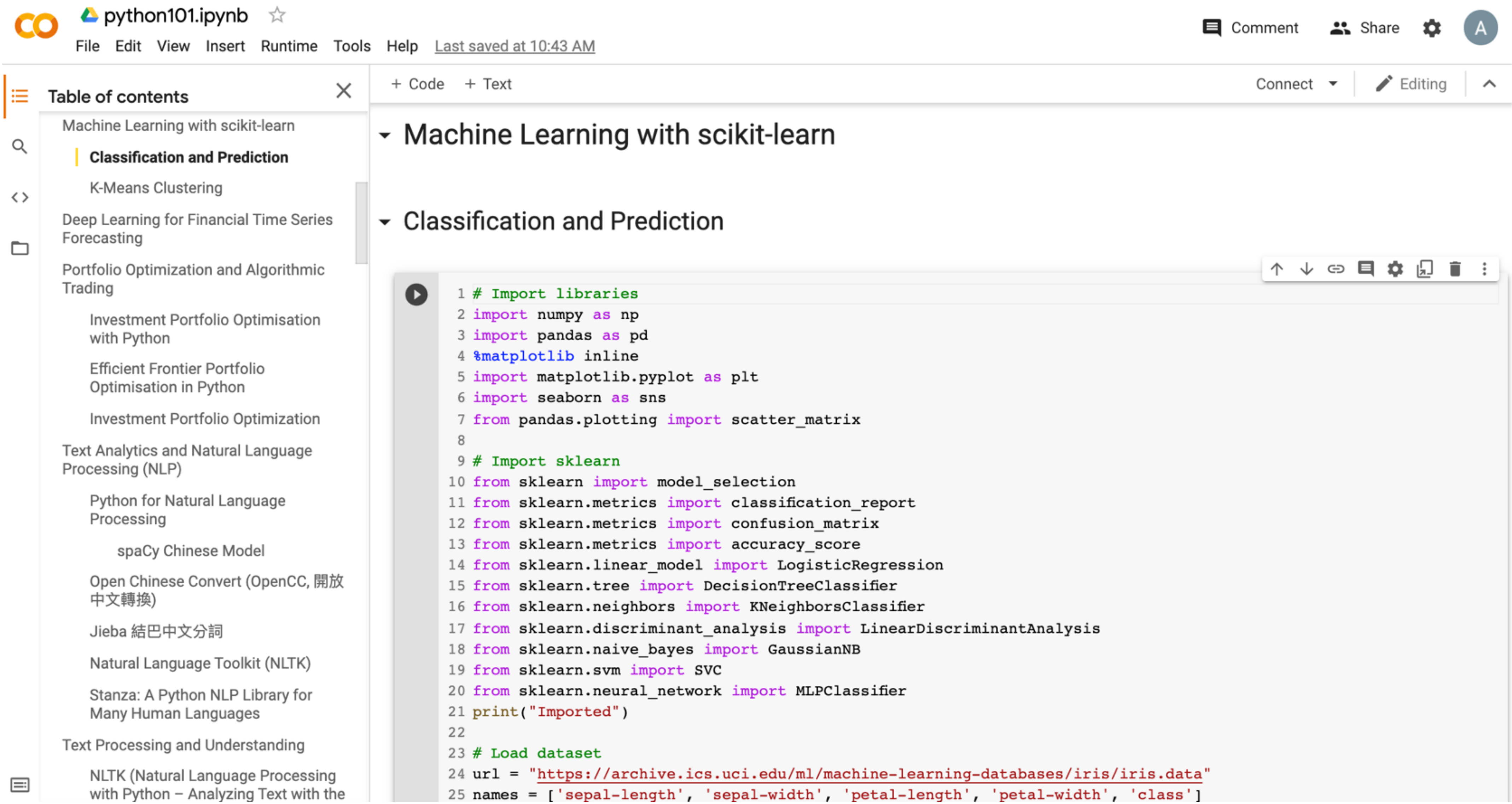


Table of contents

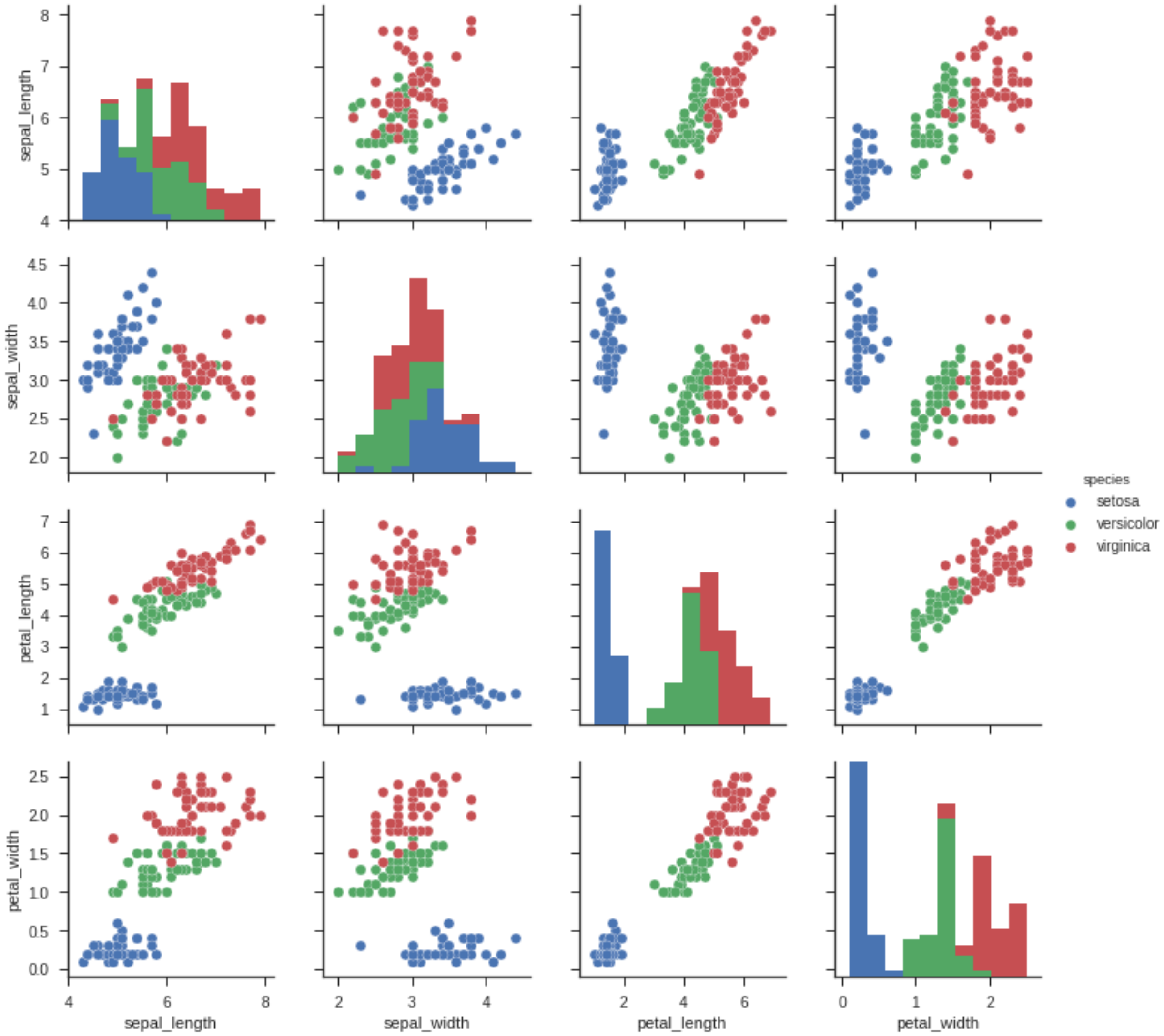
Machine Learning with scikit-learn

Classification and Prediction

```
# Import libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
```

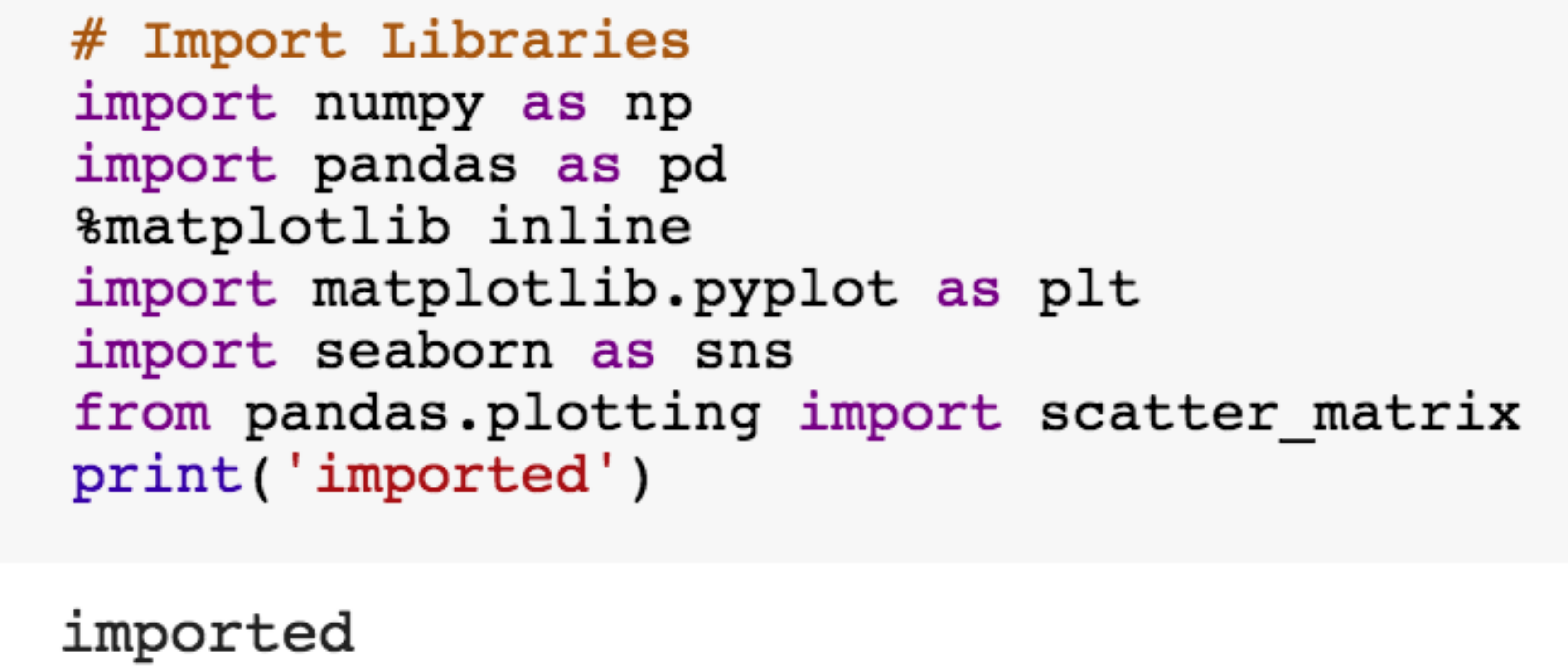
https://tinyurl.com/aintpupython101

```
import seaborn as sns
sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")
```



Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

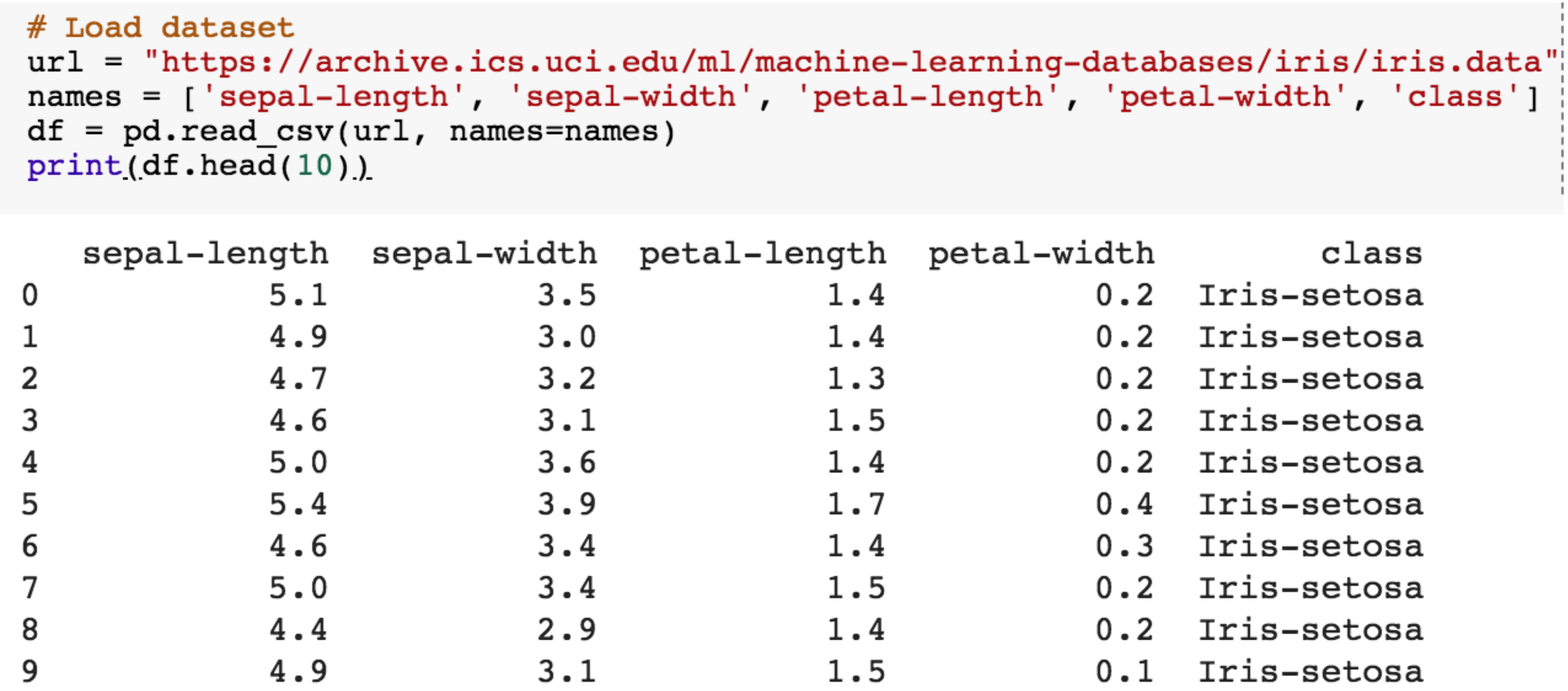
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
```



```
# Import Libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
print('imported')
```

```
imported
```

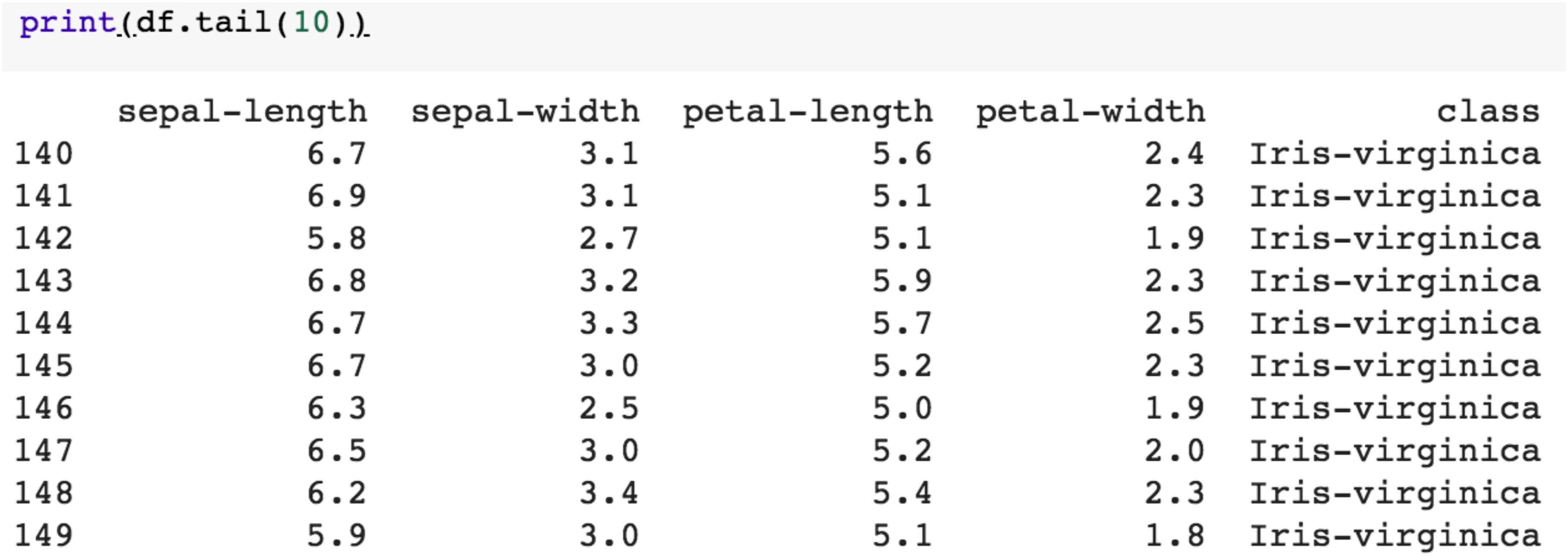
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```



```
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```

```
   sepal-length  sepal-width  petal-length  petal-width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
```

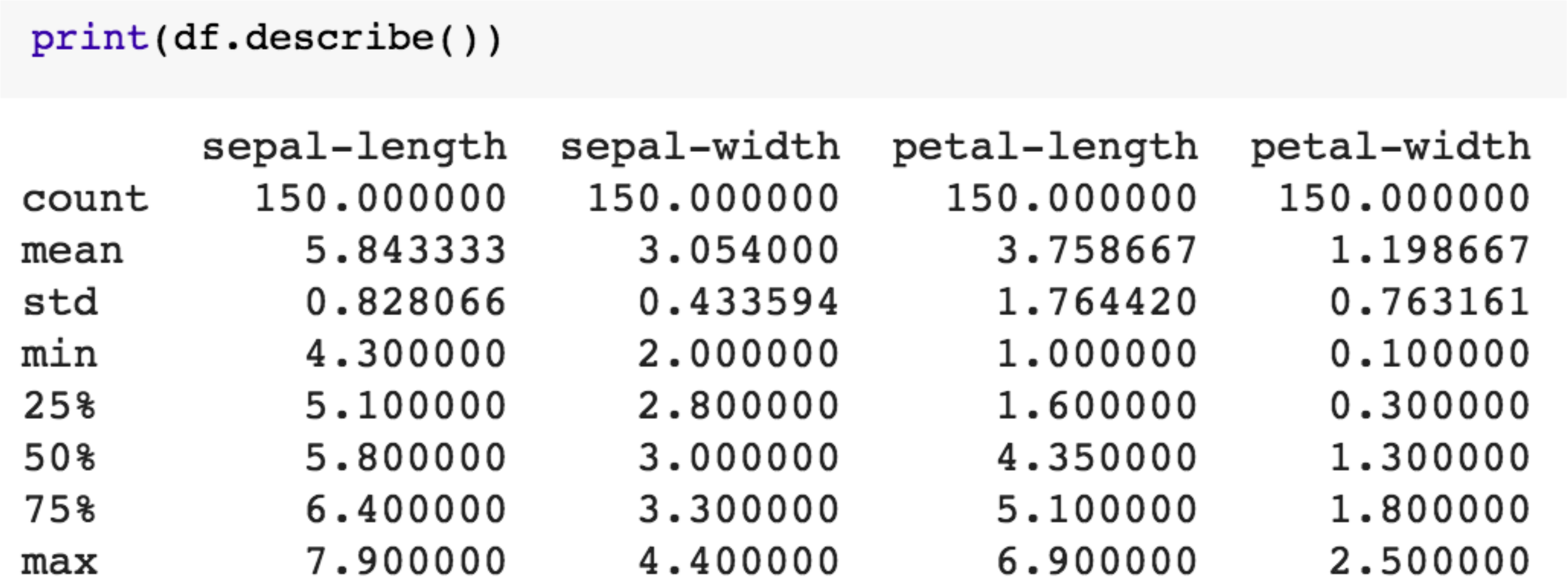
df.tail(10)



```
print(df.tail(10))
```

```
     sepal-length  sepal-width  petal-length  petal-width           class
140           6.7          3.1           5.6          2.4  Iris-virginica
141           6.9          3.1           5.1          2.3  Iris-virginica
142           5.8          2.7           5.1          1.9  Iris-virginica
143           6.8          3.2           5.9          2.3  Iris-virginica
144           6.7          3.3           5.7          2.5  Iris-virginica
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica
```

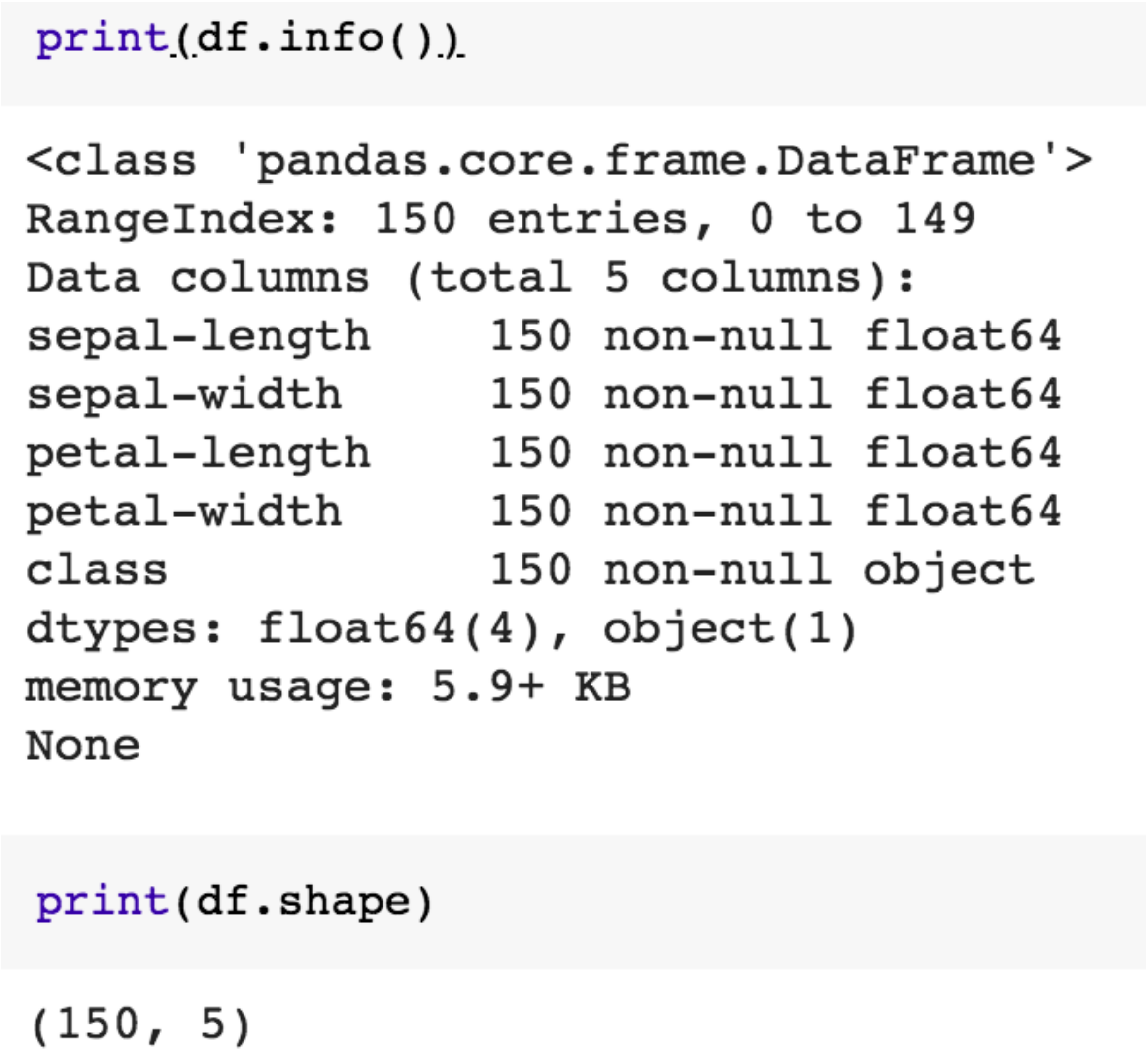
df.describe()



```
print(df.describe())
```

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
print(df.info())
print(df.shape)
```



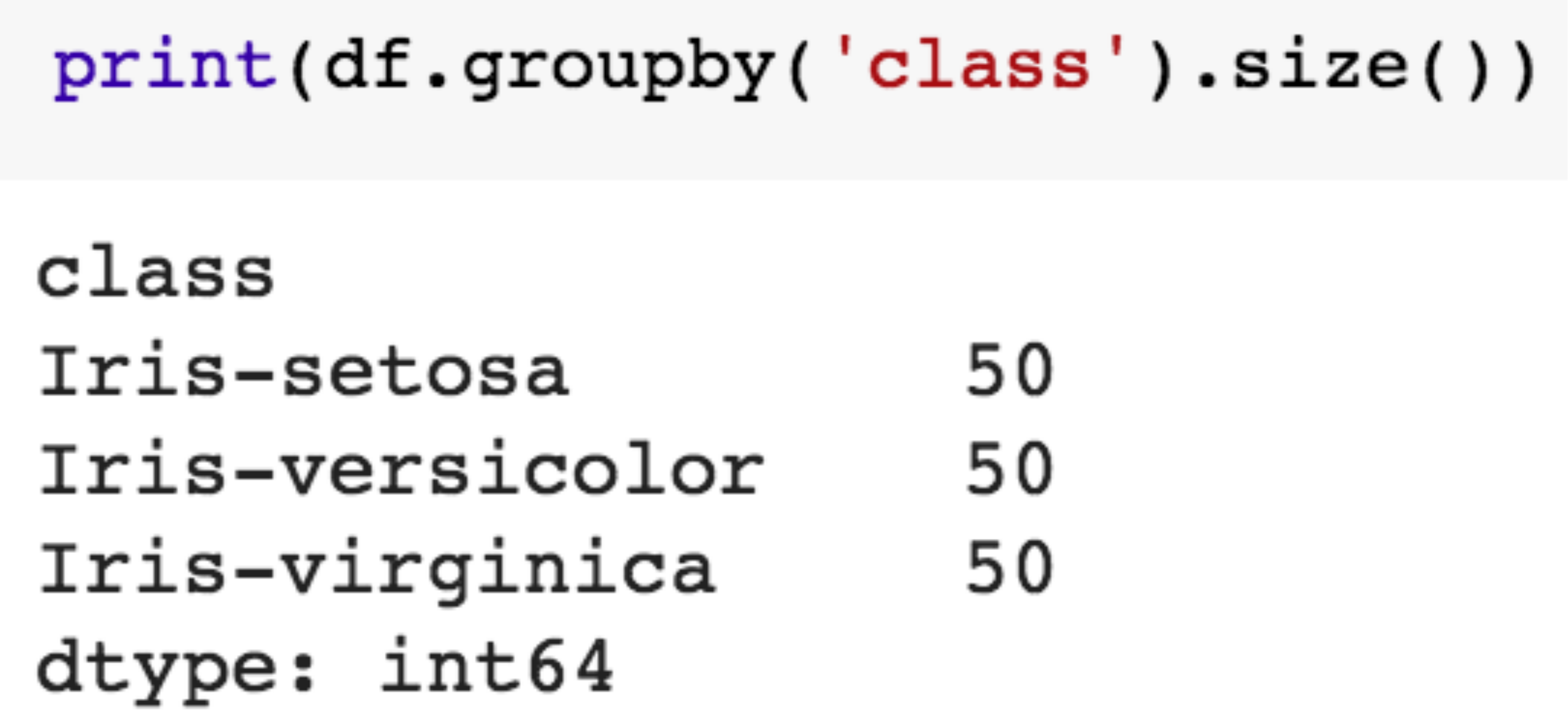
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal-length    150 non-null float64
sepal-width     150 non-null float64
petal-length    150 non-null float64
petal-width     150 non-null float64
class           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
None
```

```
print(df.shape)
```

```
(150, 5)
```

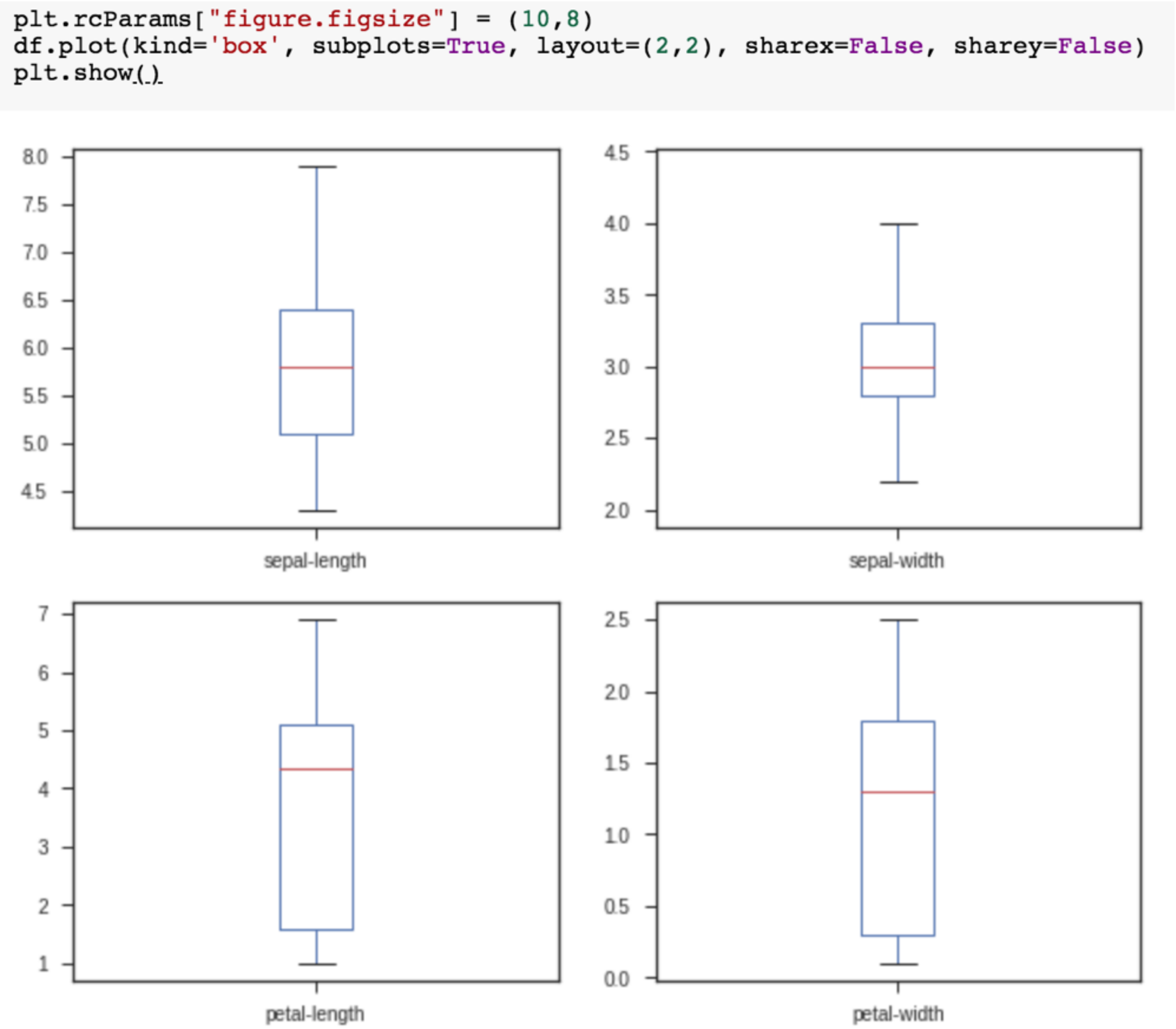
df.groupby('class').size()



```python
print(df.groupby('class').size())
```

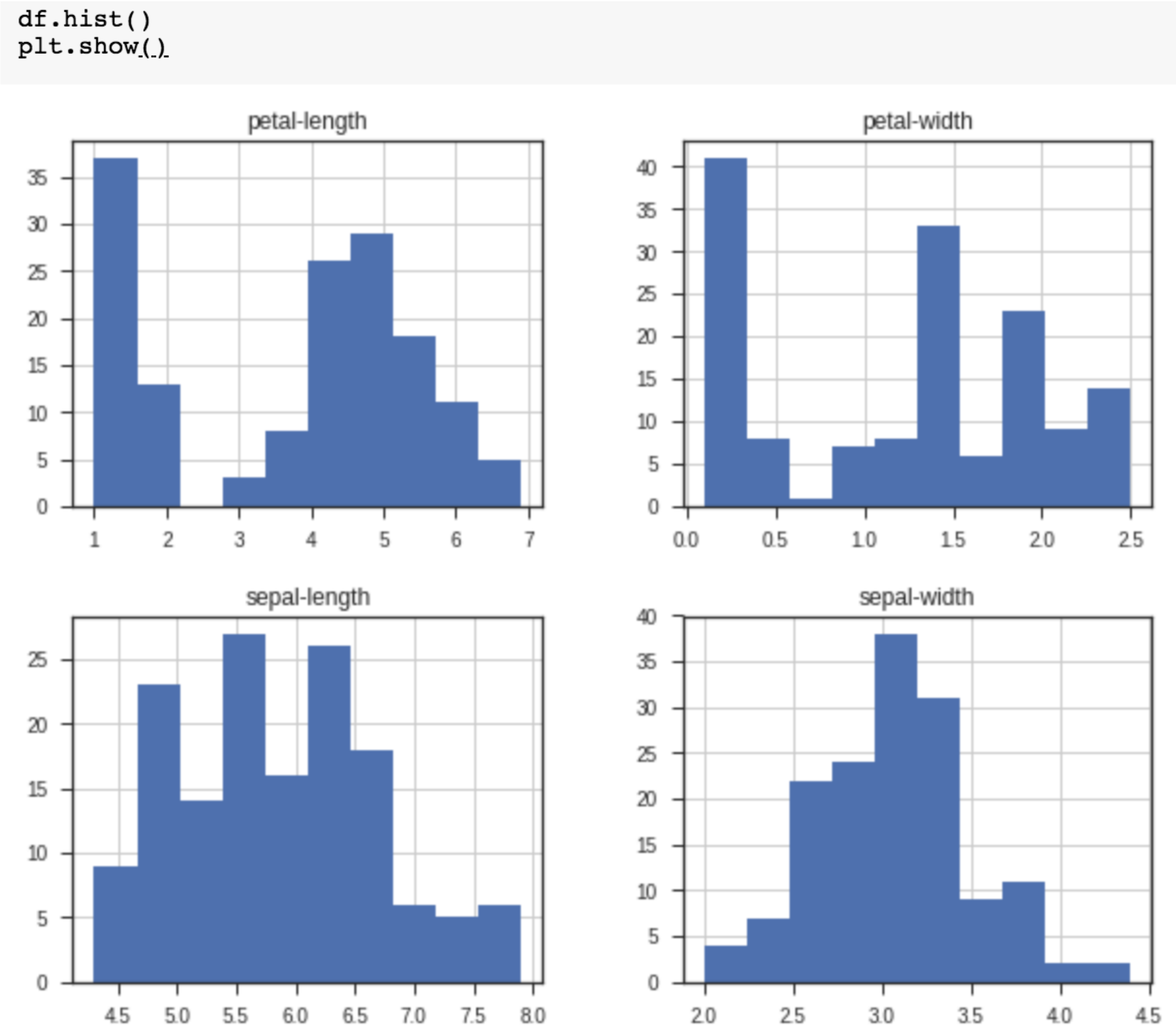
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```



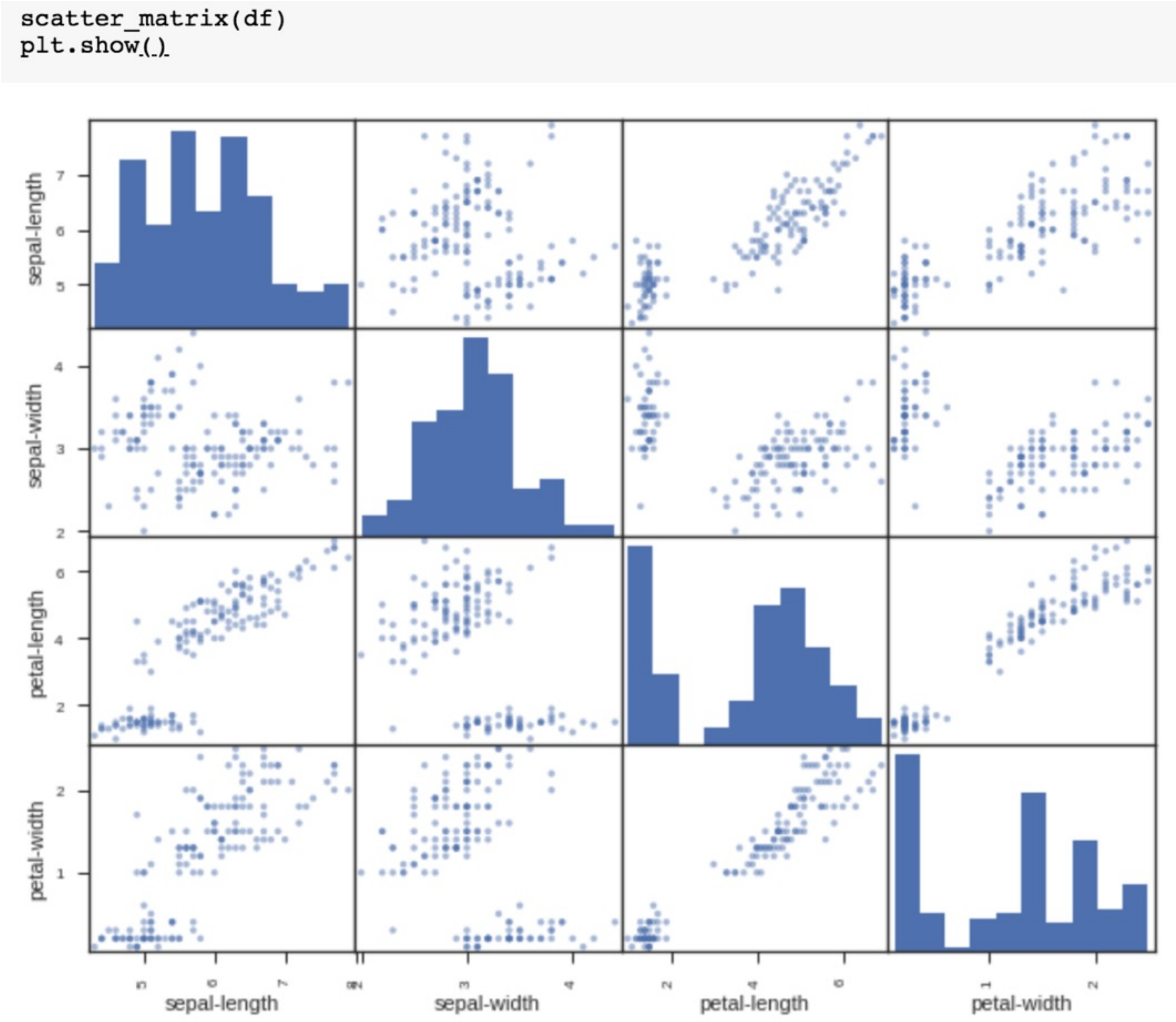
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

```
df.hist()
plt.show()
```



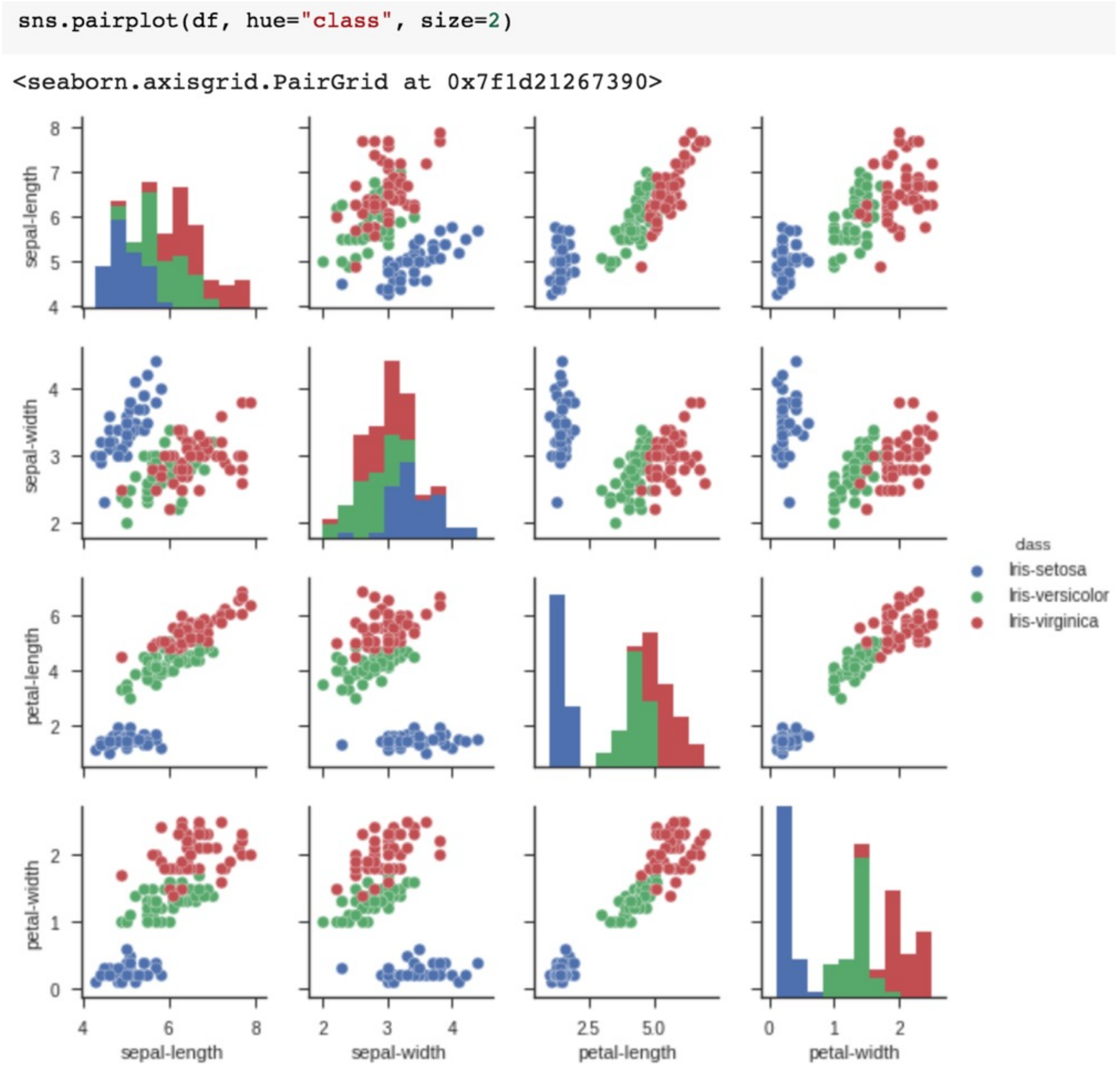
```
df.hist()
plt.show()
```

```
scatter_matrix(df)
plt.show()
```



```
scatter_matrix(df)
plt.show()
```

sns.pairplot(df, hue="class", size=2)



```
sns.pairplot(df, hue="class", size=2)
```

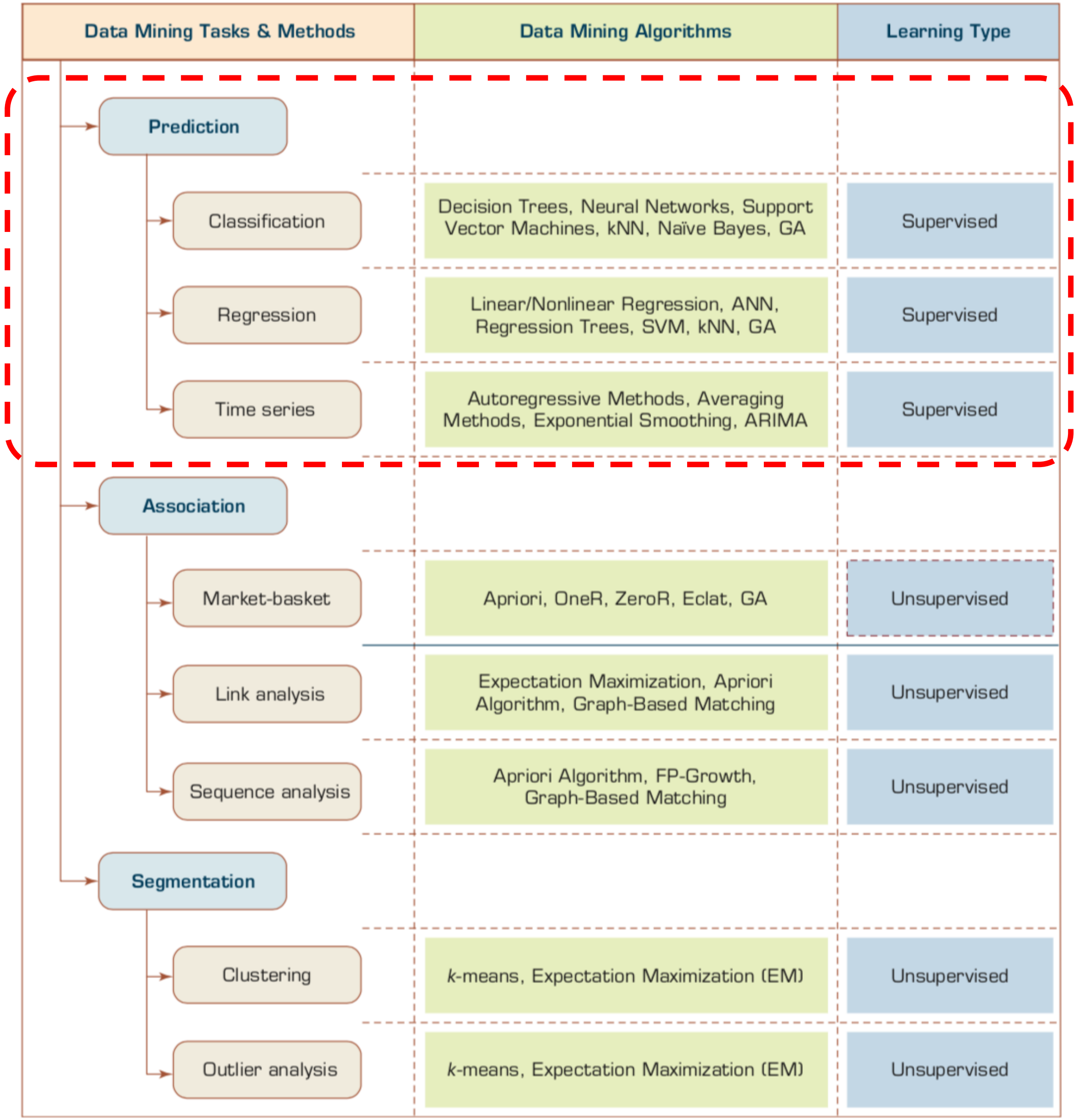
```
<seaborn.axisgrid.PairGrid at 0x7f1d21267390>
```

Machine Learning:
Supervised Learning:
Classification
and
Prediction

Machine Learning: Data Mining Tasks & Methods

Prediction
Classification

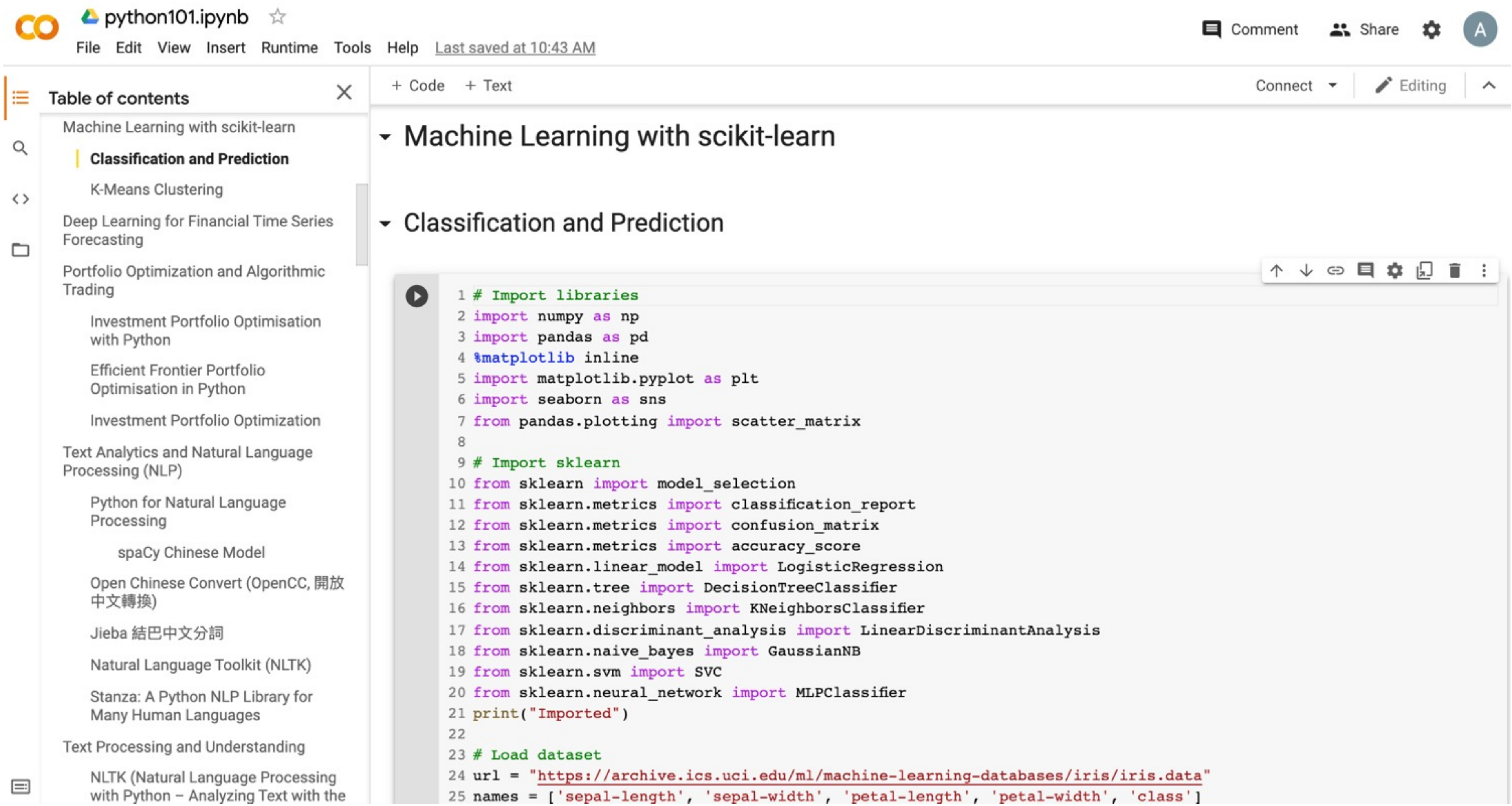
Supervised Learning:
Classification
and
Prediction



Data Mining Tasks & Methods	Data Mining Algorithms	Learning Type
Prediction		
Classification	Decision Trees, Neural Networks, Support Vector Machines, kNN, Naïve Bayes, GA	Supervised
Regression	Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA	Supervised
Time series	Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA	Supervised
Association		
Market-basket	Apriori, OneR, ZeroR, Eclat, GA	Unsupervised
Link analysis	Expectation Maximization, Apriori Algorithm, Graph-Based Matching	Unsupervised
Sequence analysis	Apriori Algorithm, FP-Growth, Graph-Based Matching	Unsupervised
Segmentation		
Clustering	*k*-means, Expectation Maximization (EM)	Unsupervised
Outlier analysis	*k*-means, Expectation Maximization (EM)	Unsupervised

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Machine Learning: Supervised Learning Classification and Prediction



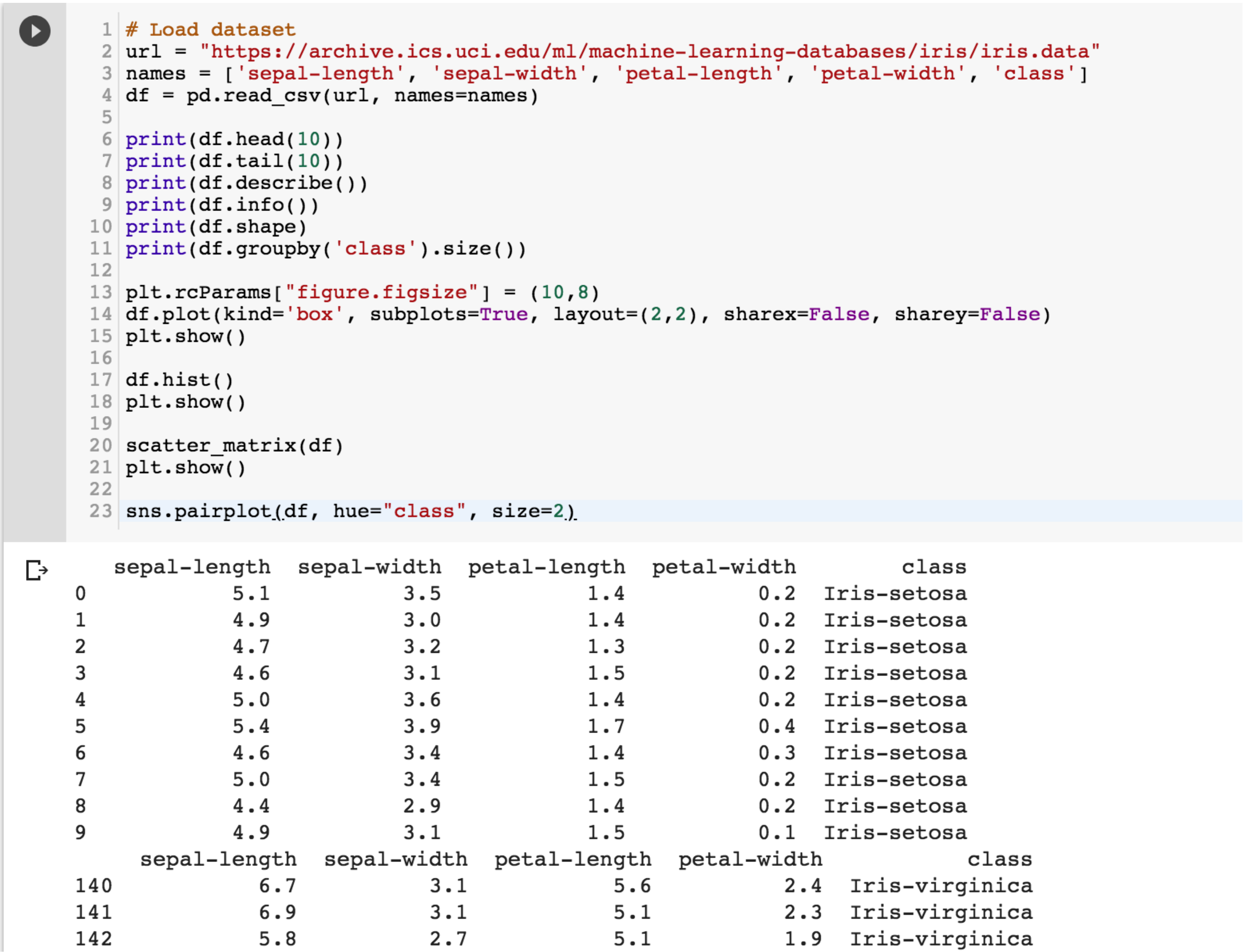
Machine Learning with scikit-learn

Classification and Prediction

```
# Import libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
```

https://tinyurl.com/aintpupython101

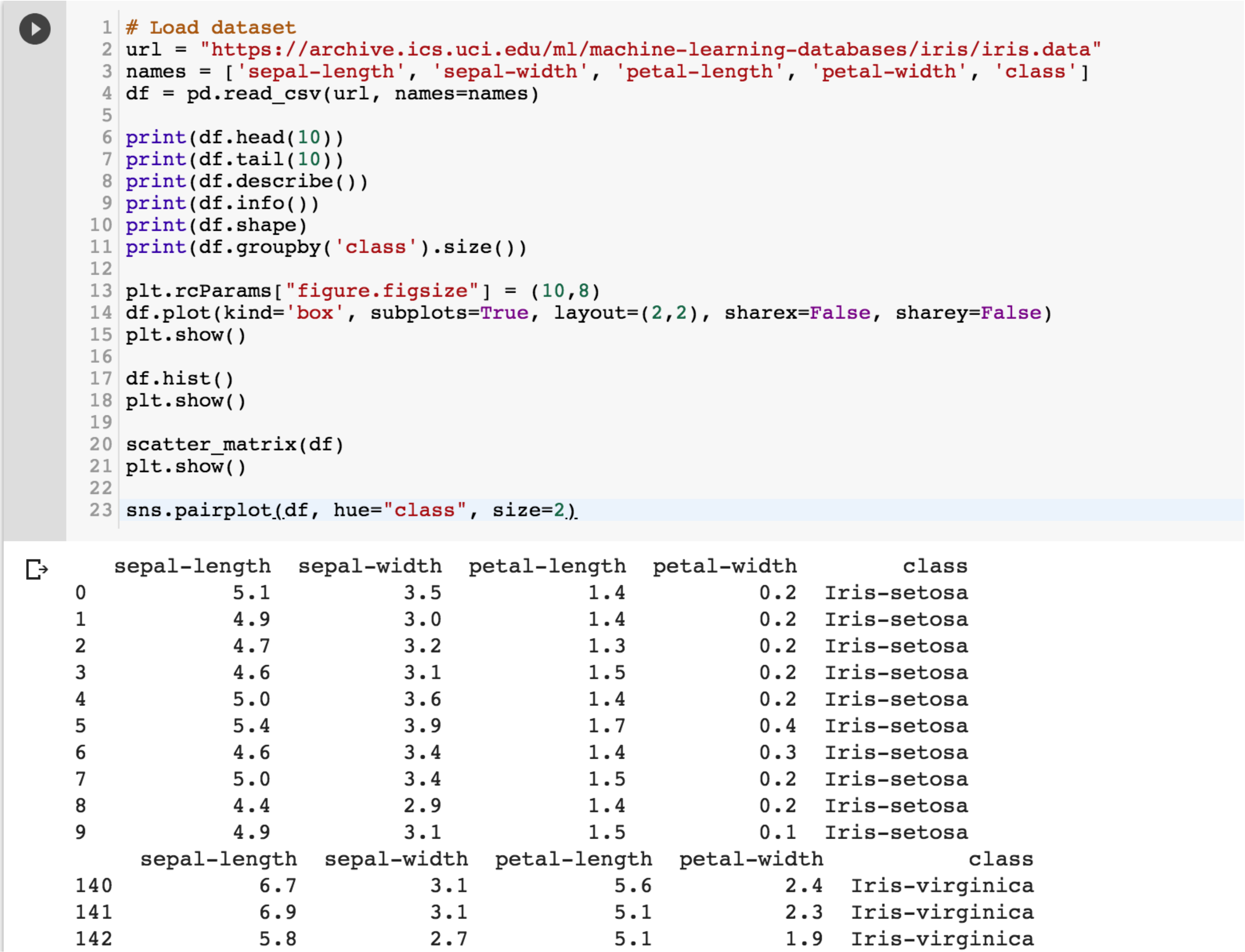
```python
# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")
```



```python
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2)
```

```
   sepal-length  sepal-width  petal-length  petal-width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
     sepal-length  sepal-width  petal-length  petal-width           class
140           6.7          3.1           5.6          2.4  Iris-virginica
141           6.9          3.1           5.1          2.3  Iris-virginica
142           5.8          2.7           5.1          1.9  Iris-virginica
```

https://tinyurl.com/aintpupython101

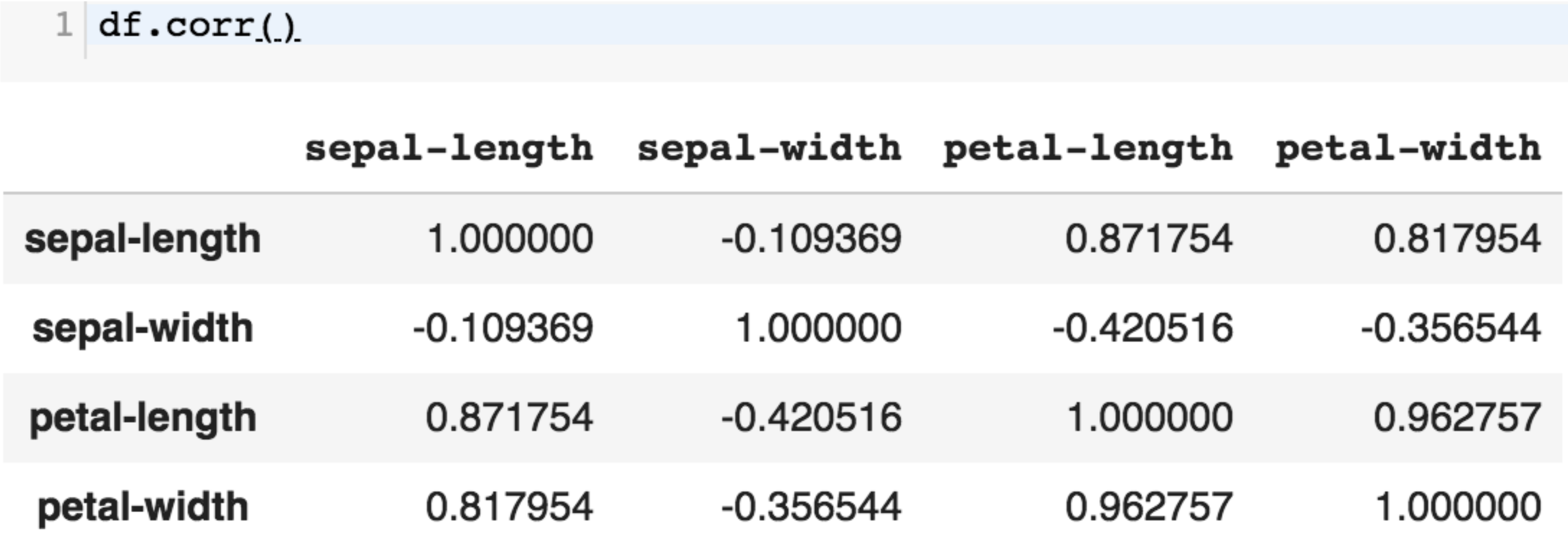


```
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2)
```

```
   sepal-length  sepal-width  petal-length  petal-width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
     sepal-length  sepal-width  petal-length  petal-width           class
140           6.7          3.1           5.6          2.4  Iris-virginica
141           6.9          3.1           5.1          2.3  Iris-virginica
142           5.8          2.7           5.1          1.9  Iris-virginica
```

https://tinyurl.com/aintpupython101

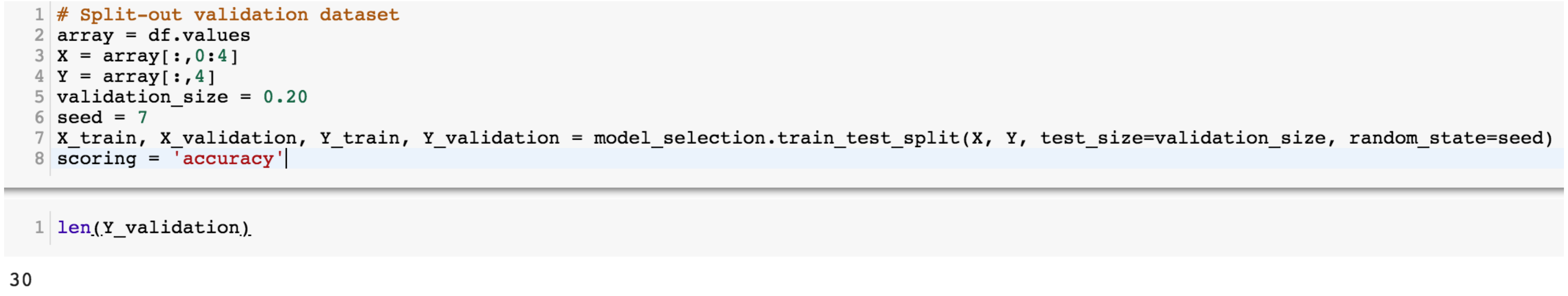
`df.corr()`



```
df.corr()
```

	sepal-length	sepal-width	petal-length	petal-width
sepal-length	1.000000	-0.109369	0.871754	0.817954
sepal-width	-0.109369	1.000000	-0.420516	-0.356544
petal-length	0.871754	-0.420516	1.000000	0.962757
petal-width	0.817954	-0.356544	0.962757	1.000000

```
# Split-out validation dataset
array = df.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y,
test_size=validation_size, random_state=seed)
scoring = 'accuracy'
```



```
# Split-out validation dataset
array = df.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=seed)
scoring = 'accuracy'
```

```
len(Y_validation)
```

```
30
```

```
# Models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA',
LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT',
DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
```

```
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10,
random_state=seed)
    cv_results =
model_selection.cross_val_score(model,
X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %.4f (%.4f)" % (name,
cv_results.mean(), cv_results.std())
    print(msg)
```

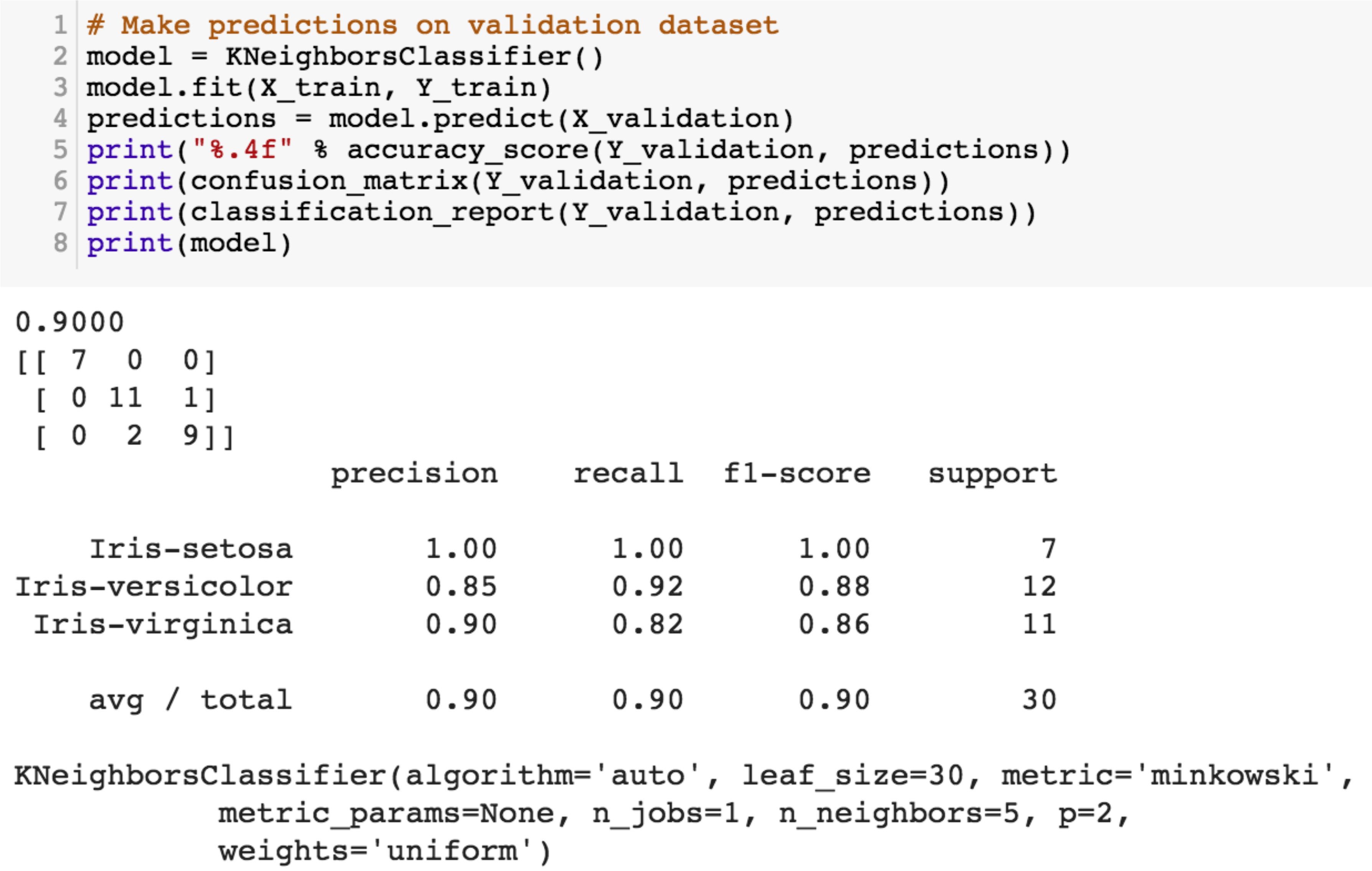
```python
# Models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %.4f (%.4f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.9667 (0.0408)
LDA: 0.9750 (0.0382)
KNN: 0.9833 (0.0333)
DT: 0.9750 (0.0382)
NB: 0.9750 (0.0534)
SVM: 0.9917 (0.0250)
```

https://tinyurl.com/aintpupython101

```
# Make predictions on validation dataset
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation,
predictions))
print(confusion_matrix(Y_validation,
predictions))
print(classification_report(Y_validation,
predictions))
print(model)
```

https://tinyurl.com/aintpupython101



```python
# Make predictions on validation dataset
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

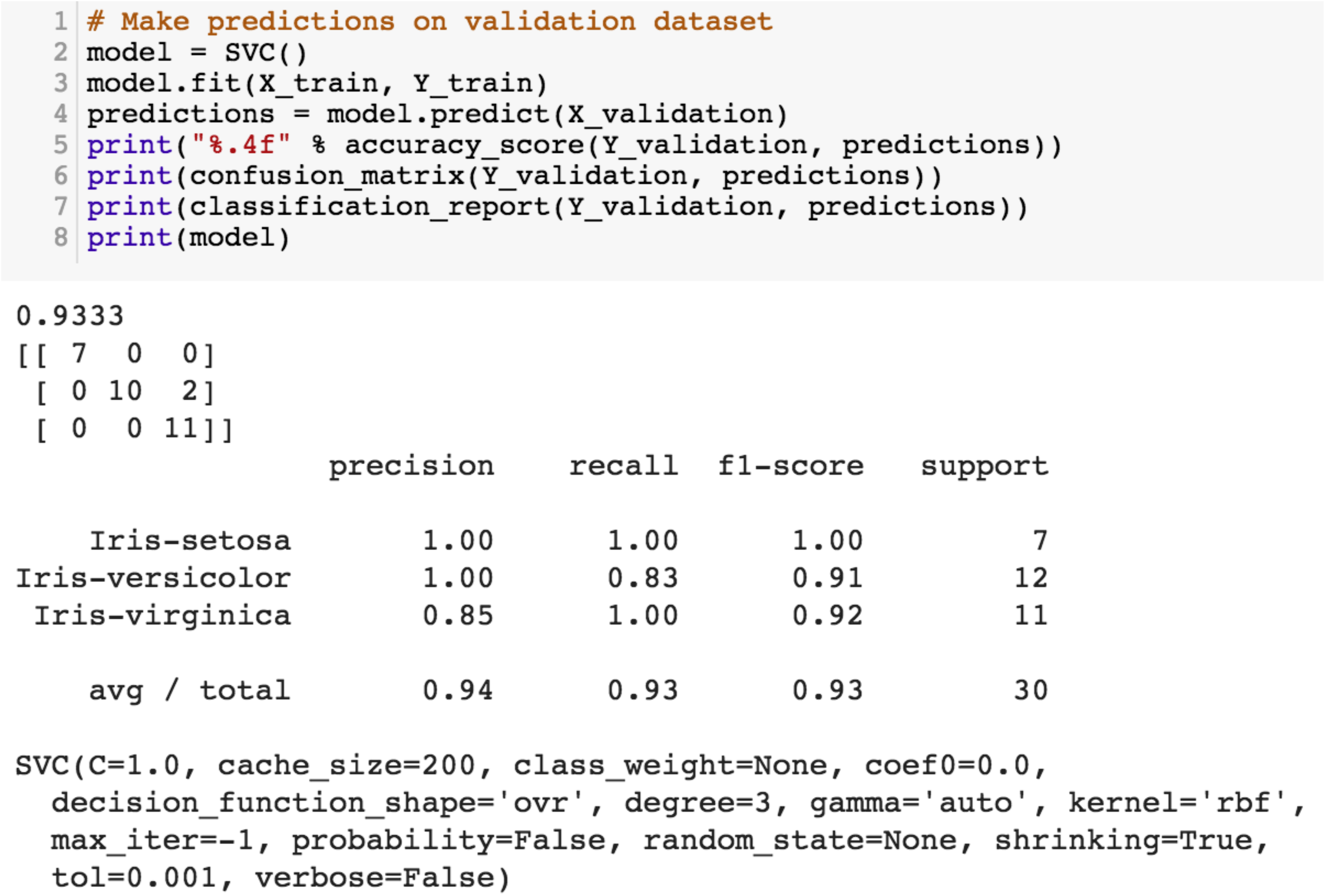
```
0.9000
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

    avg / total       0.90      0.90      0.90        30

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

https://tinyurl.com/aintpupython101

```
# Make predictions on validation dataset
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation,
predictions))
print(confusion_matrix(Y_validation,
predictions))
print(classification_report(Y_validation,
predictions))
print(model)
```

https://tinyurl.com/aintpupython101

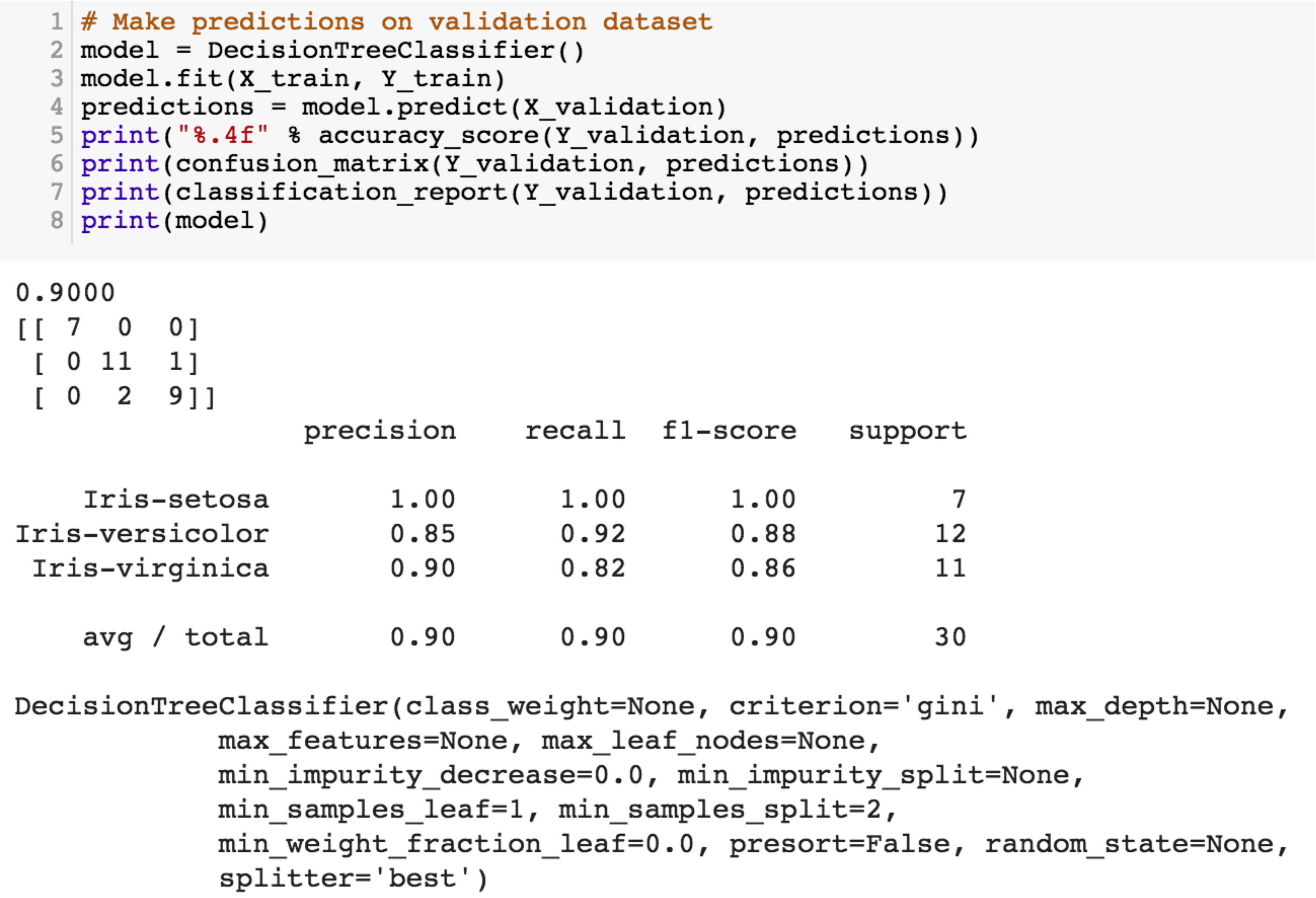
```
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```



```
# Make predictions on validation dataset
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9333
[[ 7  0  0]
 [ 0 10  2]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.83      0.91        12
 Iris-virginica       0.85      1.00      0.92        11

    avg / total       0.94      0.93      0.93        30

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

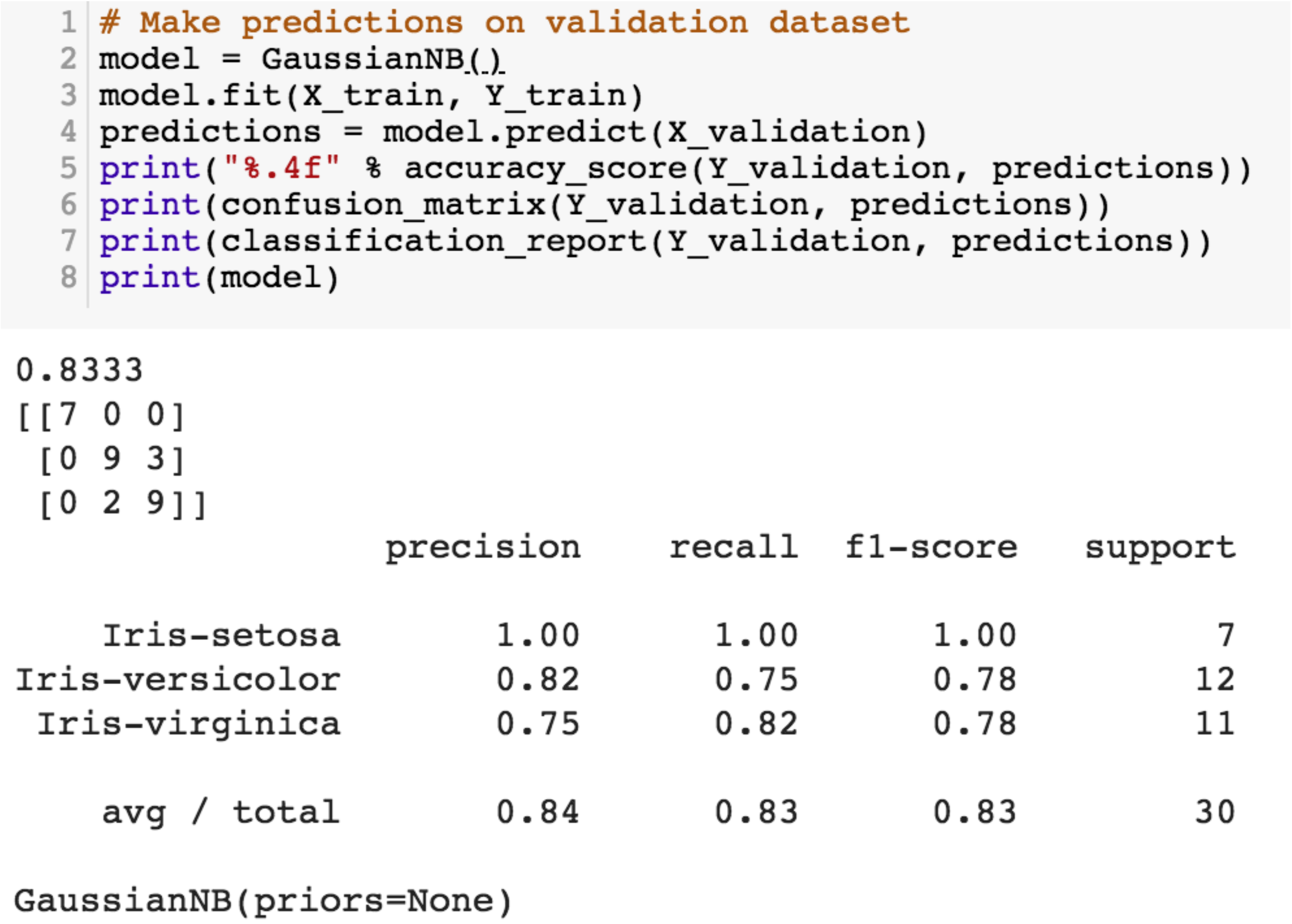
https://tinyurl.com/aintpupython101



```python
# Make predictions on validation dataset
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

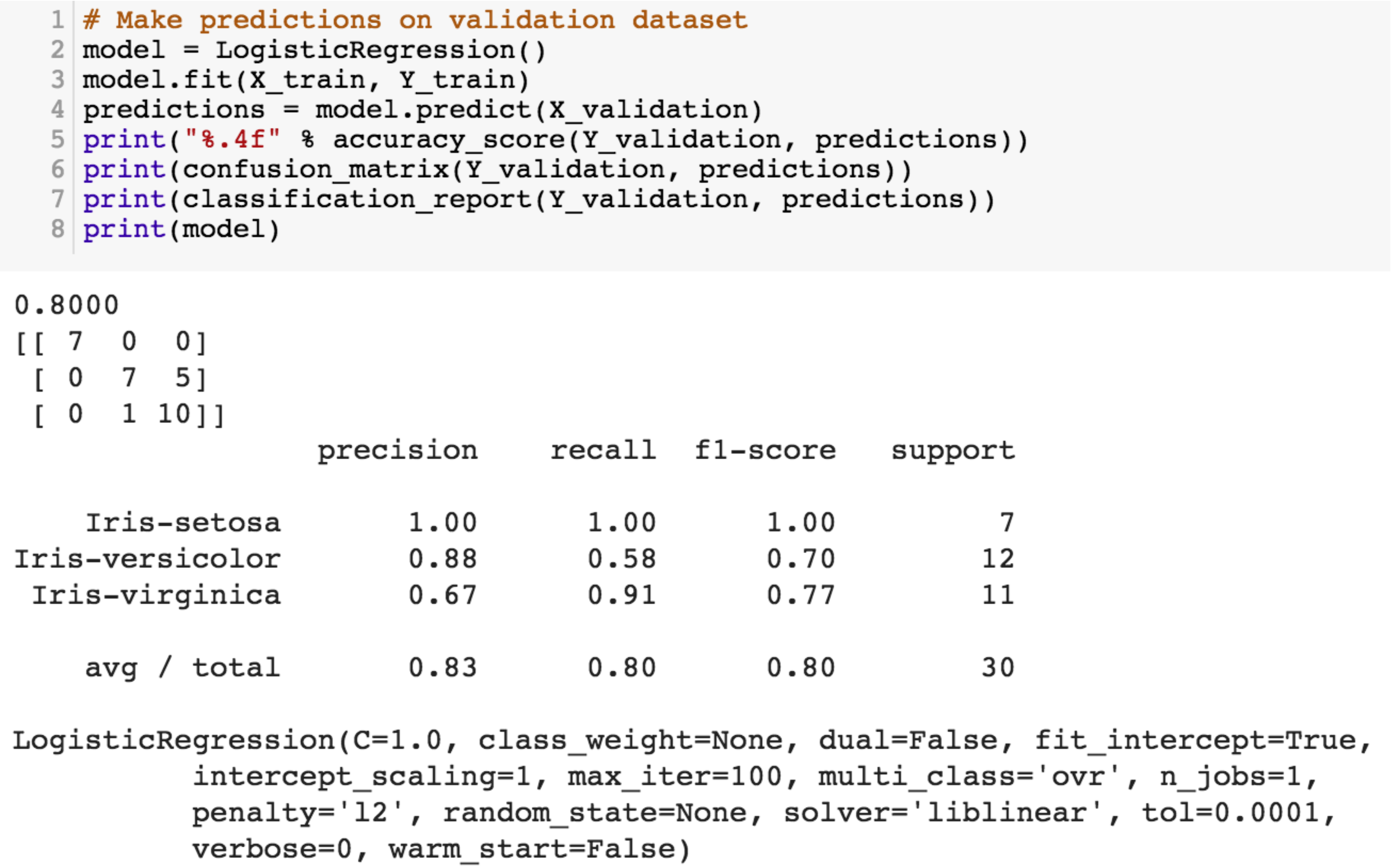
```
0.9000
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

    avg / total       0.90      0.90      0.90        30

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

https://tinyurl.com/aintpupython101



```python
# Make predictions on validation dataset
model = GaussianNB()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

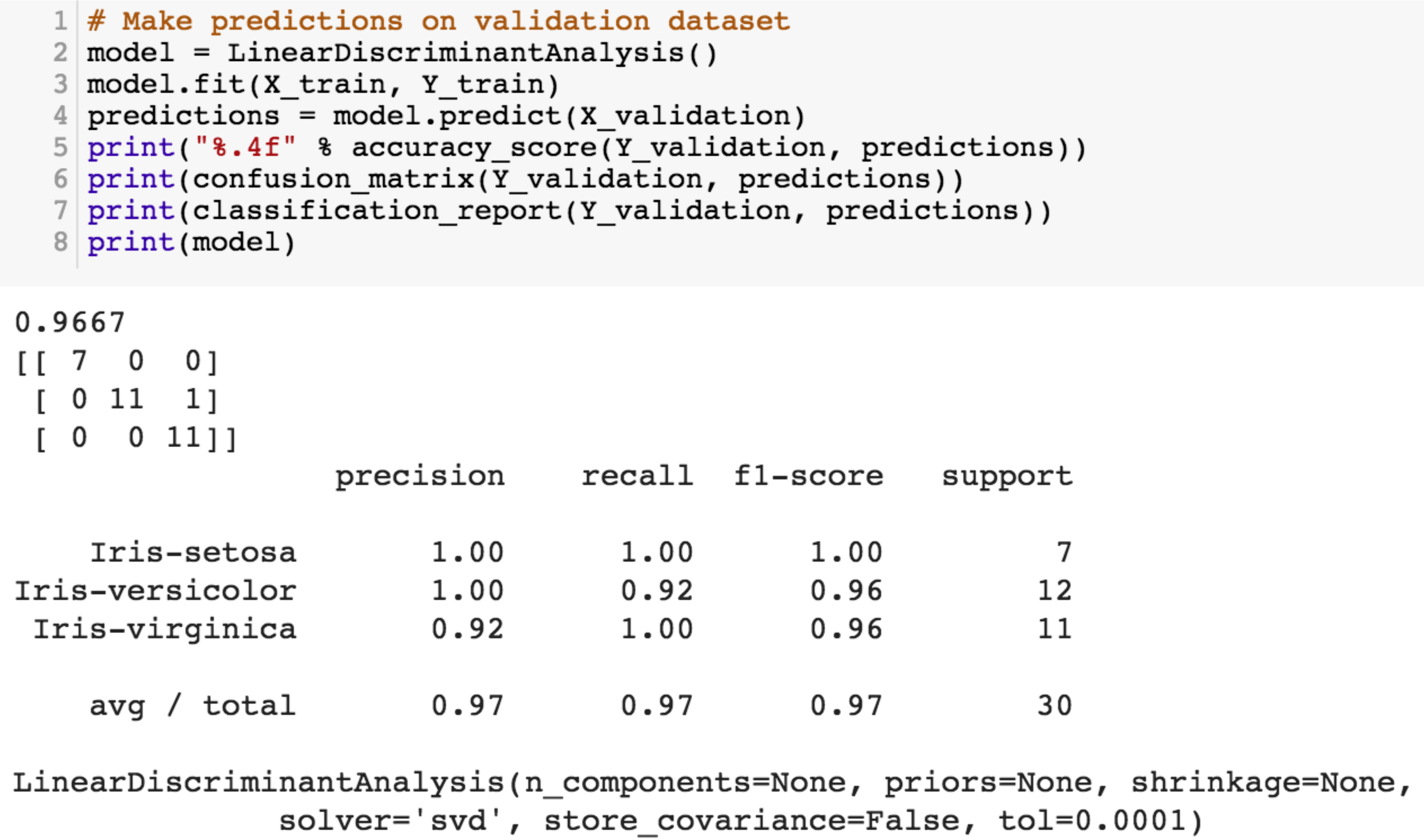
```
0.8333
[[7 0 0]
 [0 9 3]
 [0 2 9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.82      0.75      0.78        12
 Iris-virginica       0.75      0.82      0.78        11

    avg / total       0.84      0.83      0.83        30

GaussianNB(priors=None)
```



```python
# Make predictions on validation dataset
model = LogisticRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.8000
[[ 7  0  0]
 [ 0  7  5]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.88      0.58      0.70        12
 Iris-virginica       0.67      0.91      0.77        11

    avg / total       0.83      0.80      0.80        30

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

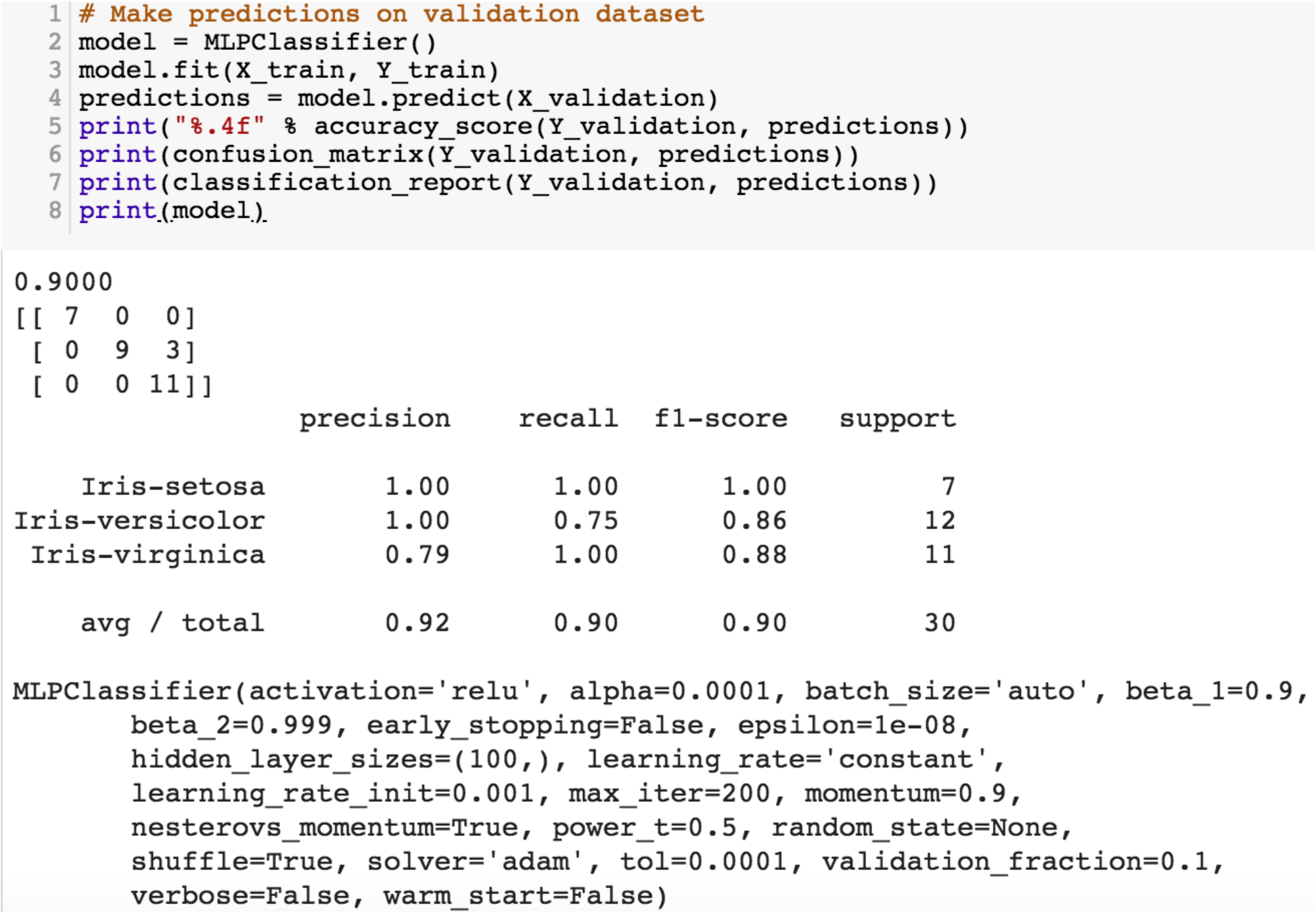
https://tinyurl.com/aintpupython101



```
# Make predictions on validation dataset
model = LinearDiscriminantAnalysis()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9667
[[ 7  0  0]
 [ 0 11  1]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.92      0.96        12
 Iris-virginica       0.92      1.00      0.96        11

    avg / total       0.97      0.97      0.97        30

LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
              solver='svd', store_covariance=False, tol=0.0001)
```

https://tinyurl.com/aintpupython101



```python
# Make predictions on validation dataset
model = MLPClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9000
[[ 7  0  0]
 [ 0  9  3]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.75      0.86        12
 Iris-virginica       0.79      1.00      0.88        11

    avg / total       0.92      0.90      0.90        30

MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

https://tinyurl.com/aintpupython101

Evaluation

(Accuracy of Classification Model)

Assessing the Classification Model

- Predictive accuracy
  - Hit rate
- Speed
  - Model building; predicting
- Robustness
- Scalability
- Interpretability
  - Transparency, explainability

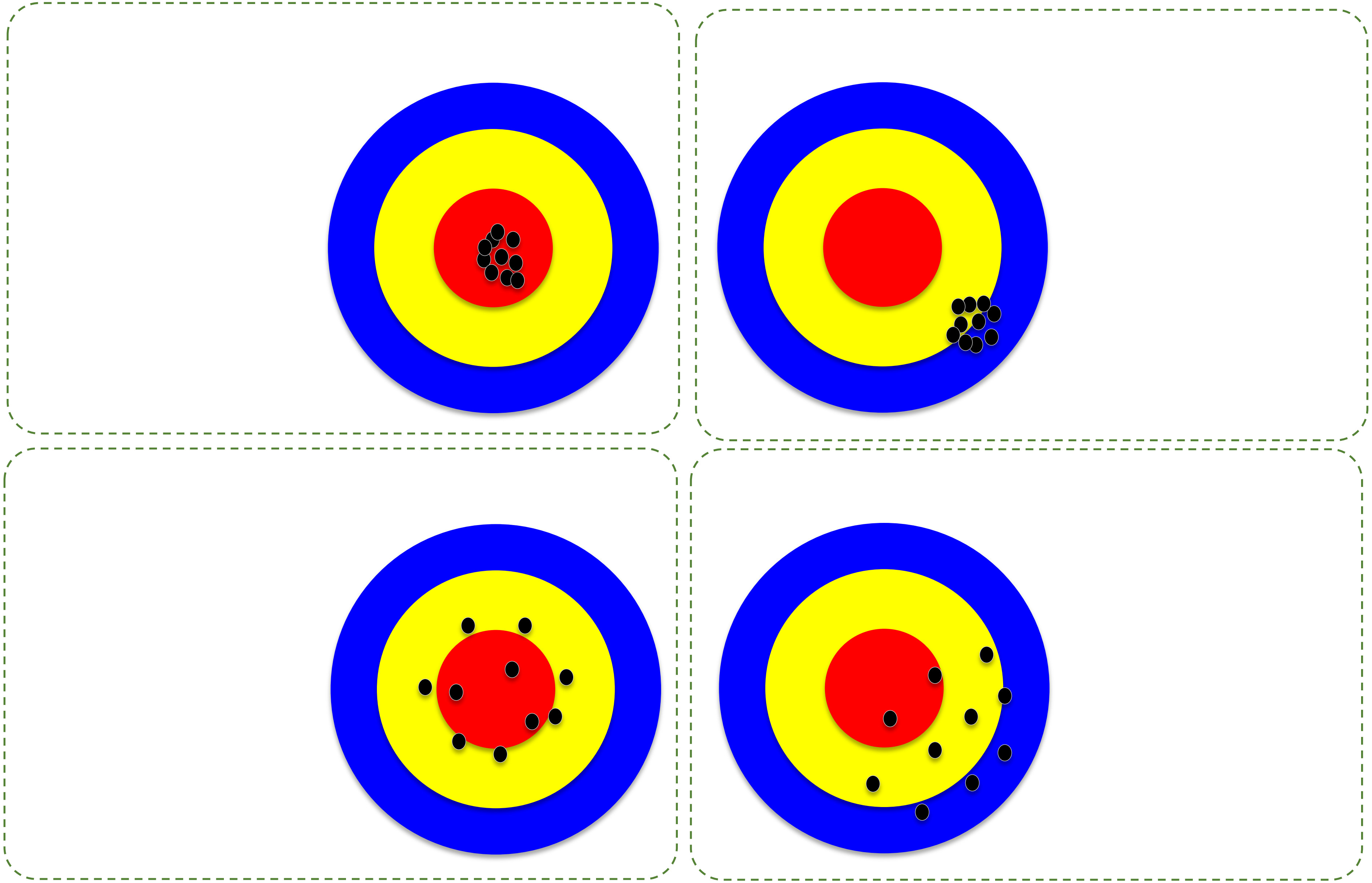
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Accuracy

Validity

Precision

Reliability



Accuracy vs. Precision

A

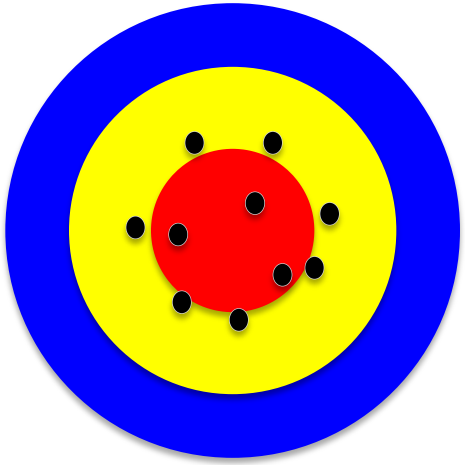
High Accuracy
High Precision

B

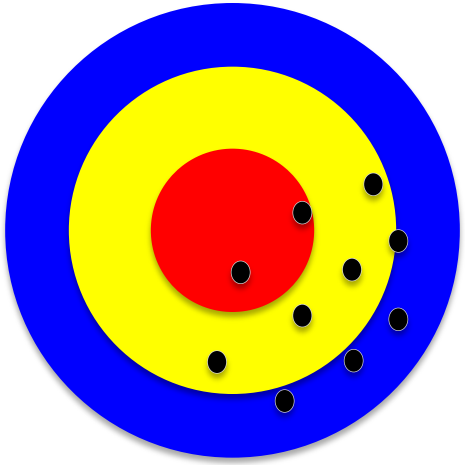
Low Accuracy
High Precision

C

High Accuracy
Low Precision



D



Low Accuracy
Low Precision

Accuracy vs. Precision

A

High Accuracy
High Precision

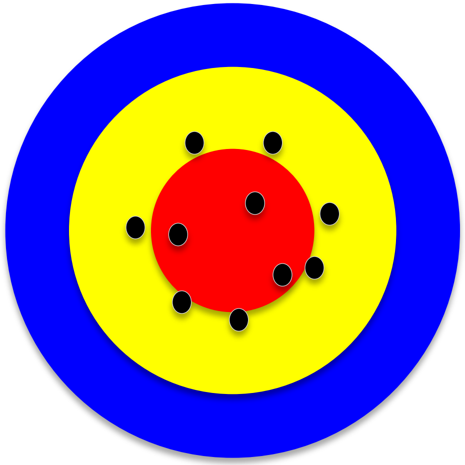
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

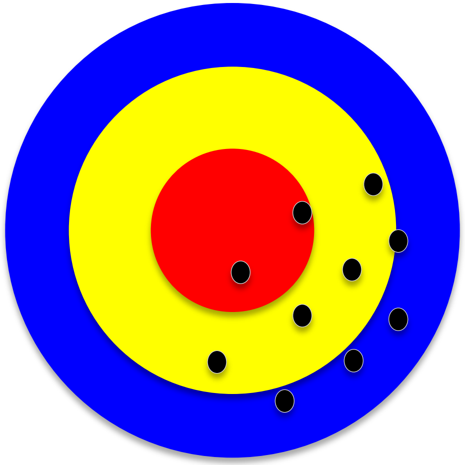
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Accuracy vs. Precision

A

High Accuracy
High Precision

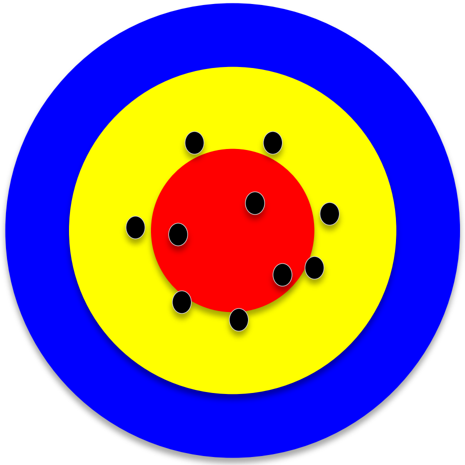
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

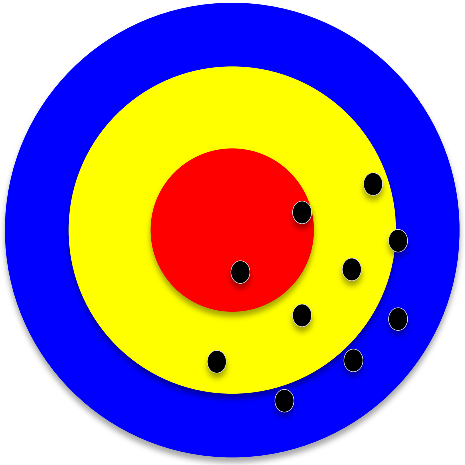
C



High Accuracy
Low Precision

High Validity
Low Reliability

D

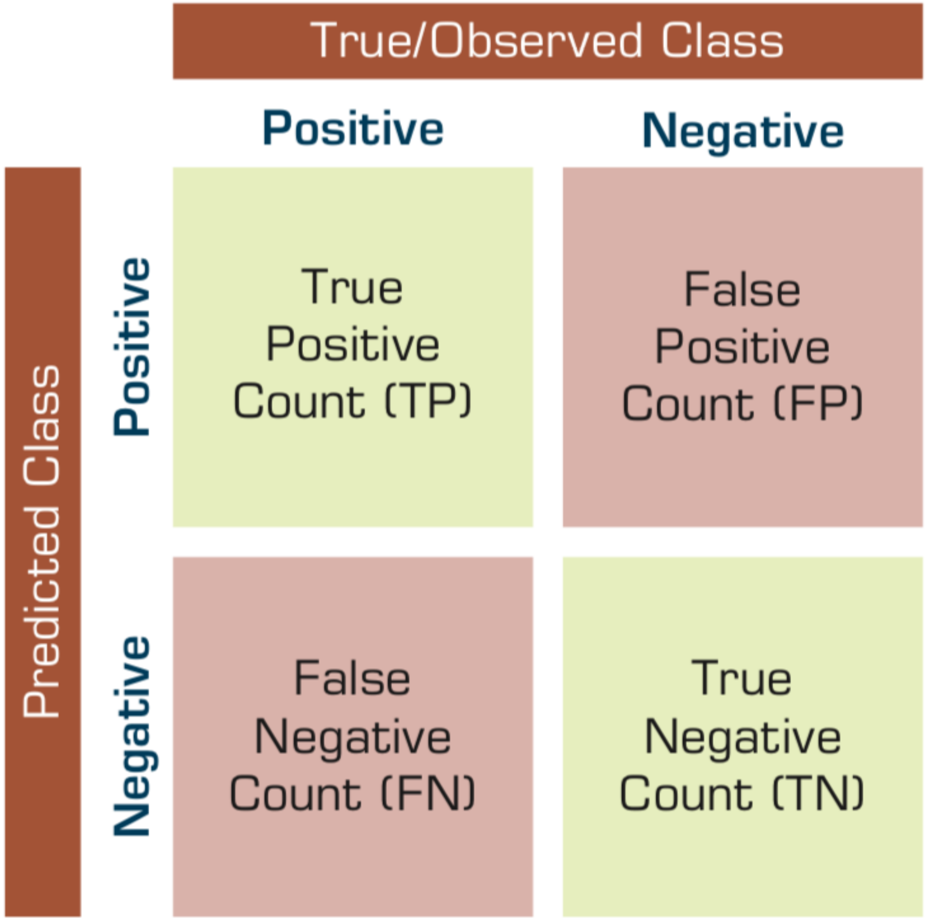


Low Accuracy
Low Precision

Low Validity
Low Reliability

Confusion Matrix

for Tabulation of Two-Class Classification Results



		True/Observed Class	
		Positive	**Negative**
Predicted Class	**Positive**	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

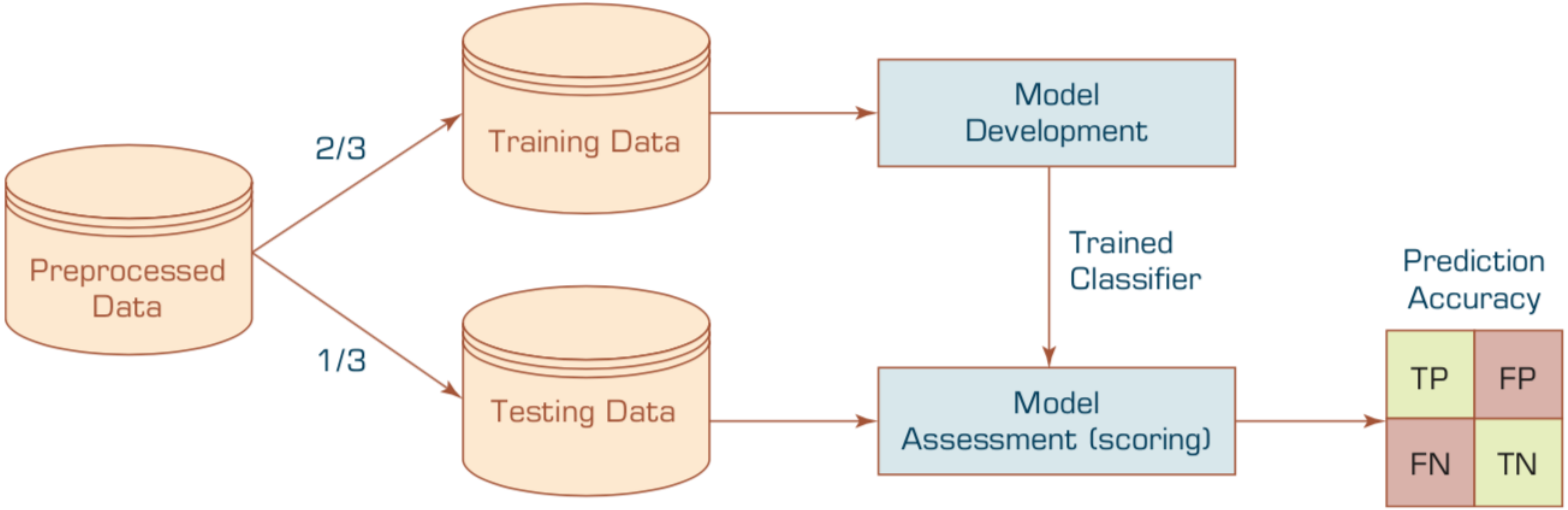
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Sensitivity =True Positive Rate

Specificity =True Negative Rate

Estimation Methodologies for Classification

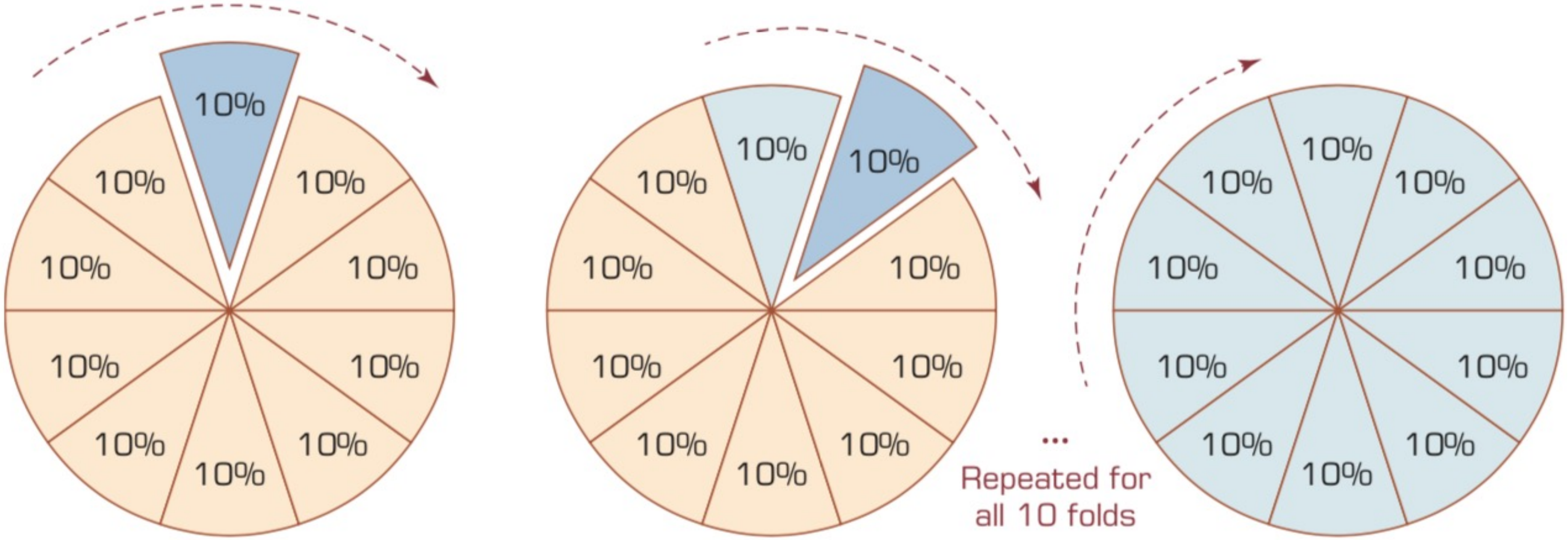
- Simple split (or holdout or test sample estimation)
  - Split the data into 2 mutually exclusive sets training (~70%) and testing (30%)



  - For ANN, the data is split into three sub-sets (training [~60%], validation [~20%], testing [~20%])

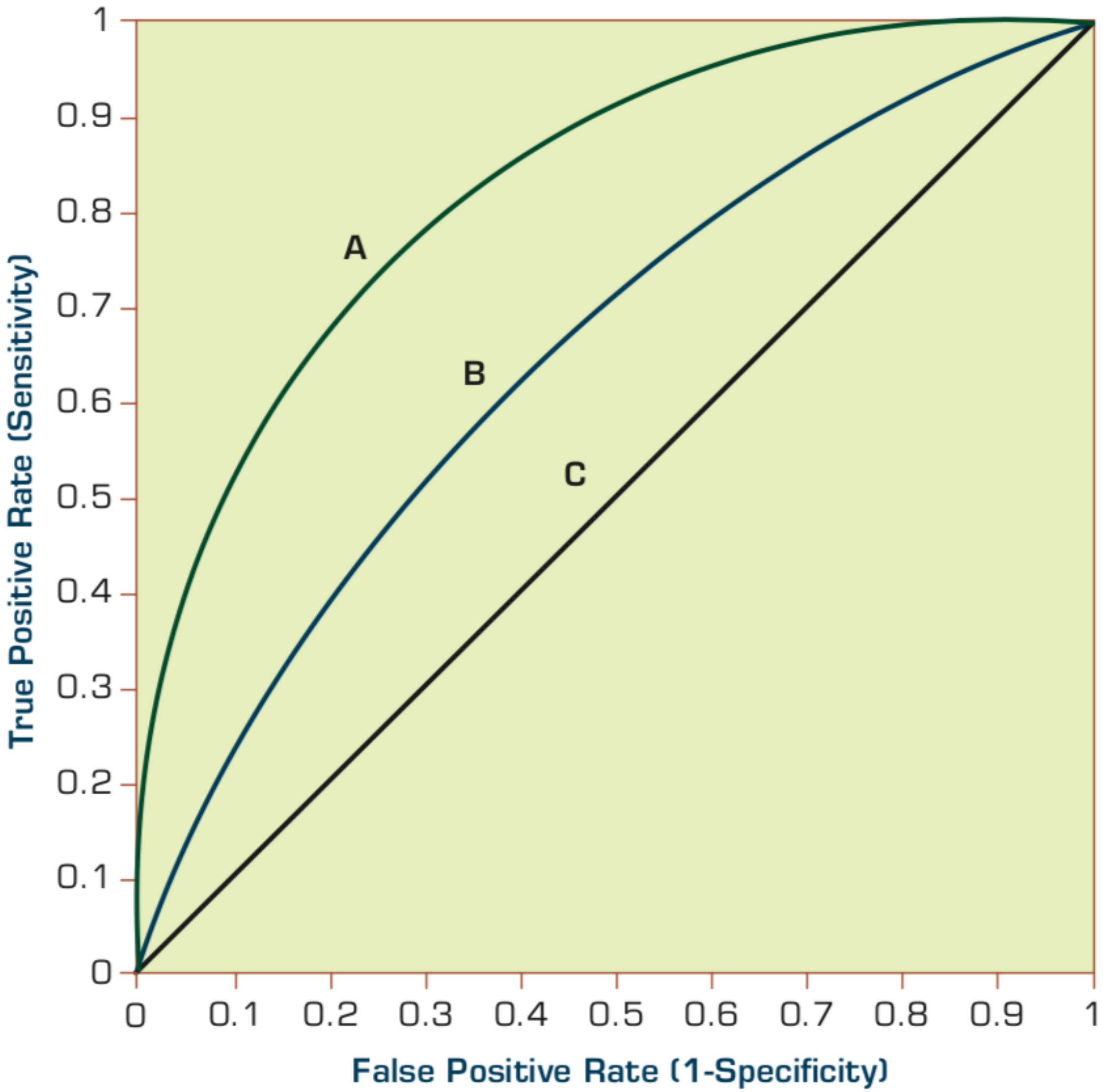
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

k-Fold Cross-Validation



Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Estimation Methodologies for Classification
Area under the ROC curve



Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

		True Class **(actual value)**		
		Positive	Negative	**total**
Predictive Class **(prediction outcome)**	Positive	True Positive (TP)	False Positive (FP)	P’
	Negative	False Negative (FN)	True Negative (TN)	N’
total		P	N	

$$True\ Positive\ Rate\ (\mathrm{Sensitivity}) = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate\ (\mathrm{Specificity}) = \frac{TN}{TN + FP}$$

$$False\ Positive\ Rate\ = \frac{FP}{FP + TN}$$

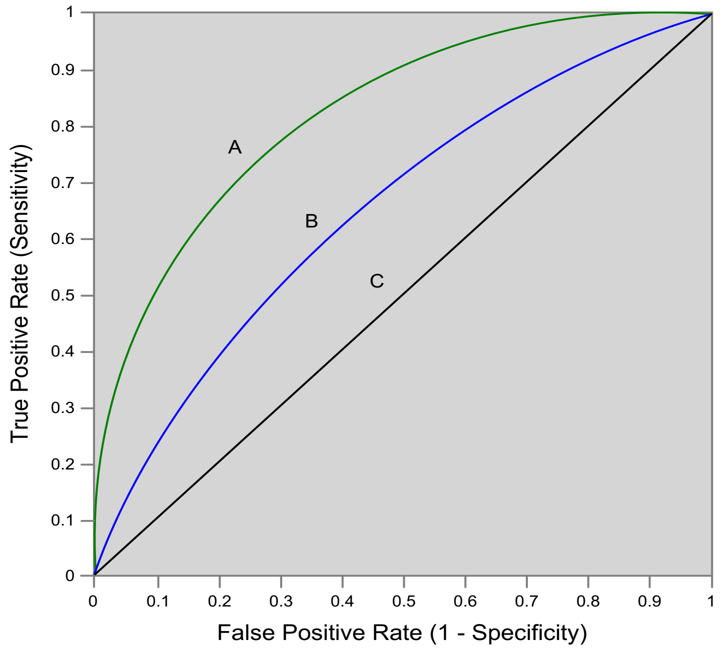
$$False\ Positive\ Rate\ (1\text{-}\mathrm{Specificity}) = \frac{FP}{FP + TN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

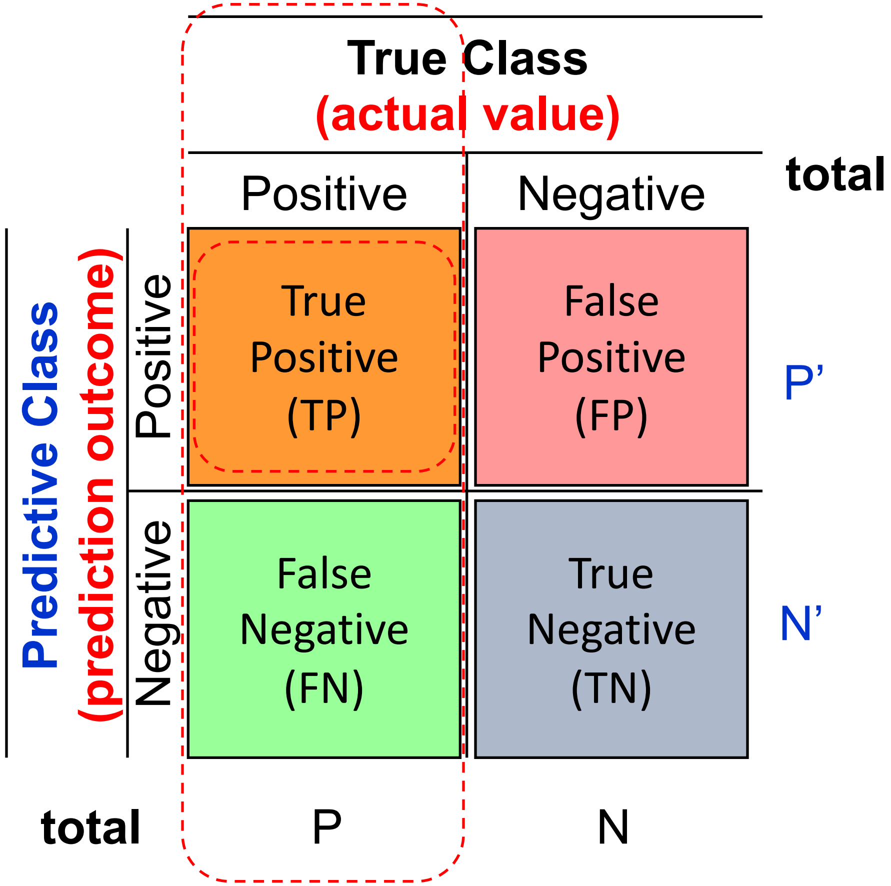
$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$



Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



		True Class (actual value)		**total**
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P’
	Negative	False Negative (FN)	True Negative (TN)	N’
total		P	N	

$$True\ Positive\ Rate\ (\text{Sensitivity}) = \frac{TP}{TP + FN}$$

Sensitivity

= True Positive Rate

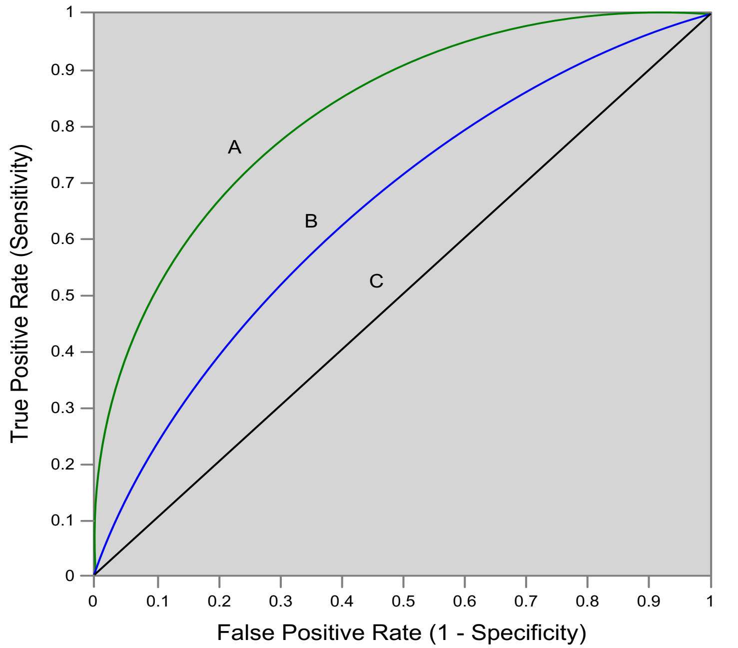
= Recall

= Hit rate

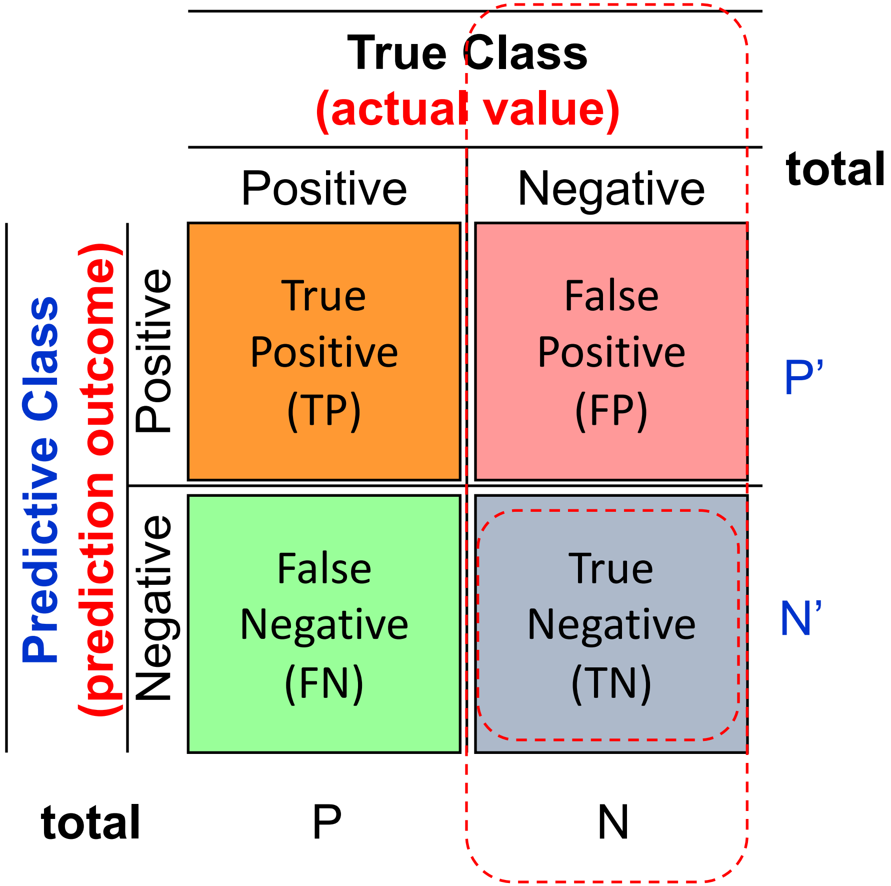
= TP / (TP + FN)

$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$



Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



		True Class (actual value)		
		Positive	Negative	total
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

Specificity

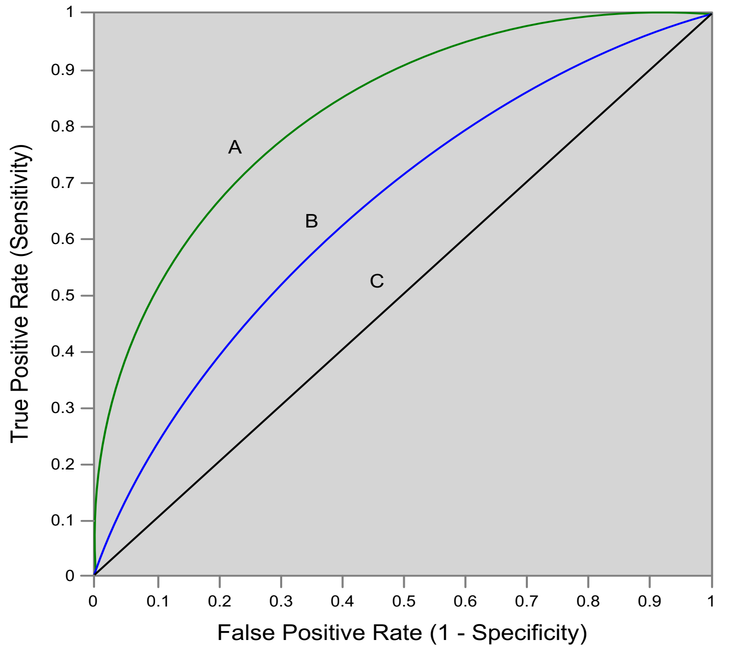
= True Negative Rate

= TN / N

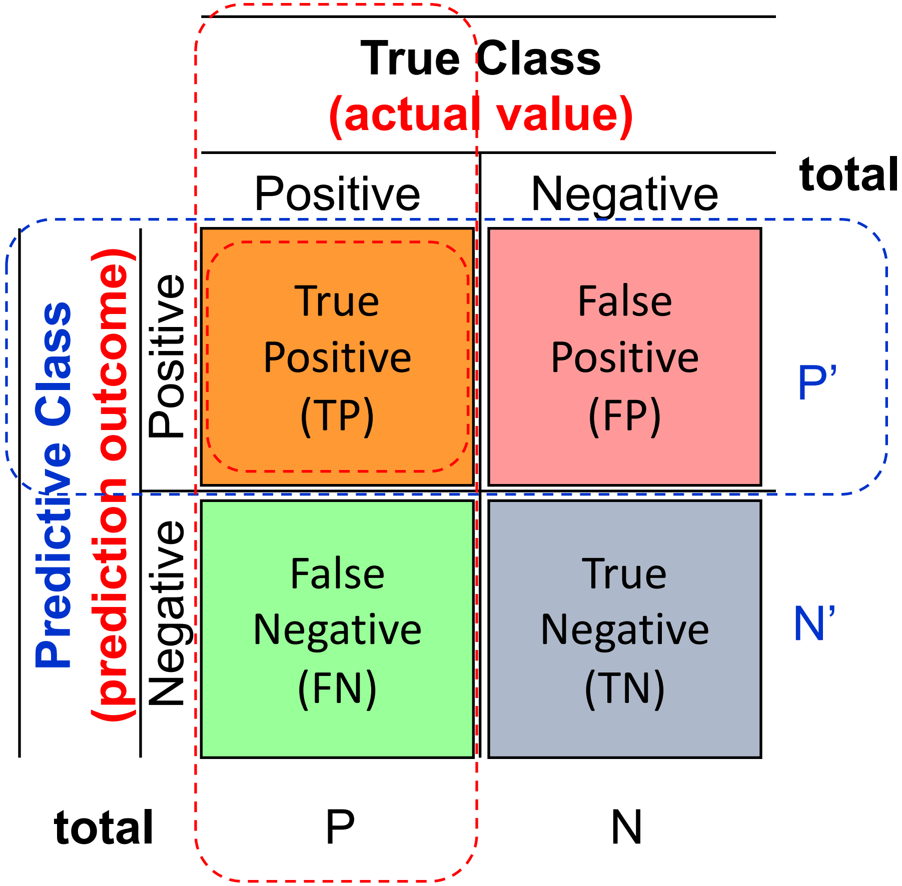
= TN / (TN+ FP)

$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

$$False\ Positive\ Rate\ (1\text{-}\text{Specificity}) = \frac{FP}{FP + TN}$$



Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



		True Class (actual value)		
		Positive	Negative	total
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P’
	Negative	False Negative (FN)	True Negative (TN)	N’
total		P	N	

Precision

= Positive Predictive Value (PPV)

$$Precision = \frac{TP}{TP + FP}$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

F1 score (F-score)(F-measure)

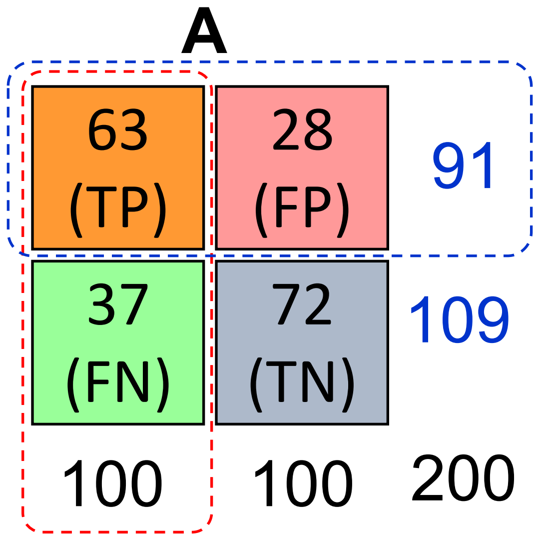
is the harmonic mean of precision and recall

= 2TP / (P + P’)

= 2TP / (2TP + FP + FN)

$$F = 2 * \frac{precision * recall}{precision + recall}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate
= TP / (TP + FN)

Specificity
= True Negative Rate
= TN / N
= TN / (TN + FP)

TPR = 0.63

$$Recall = \frac{TP}{TP+FN}$$

$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN+FP}$$

FPR = 0.28

$$False\ Positive\ Rate\ (1\text{-Specificity}) = \frac{FP}{FP+TN}$$

PPV = 0.69
=63/(63+28)
=63/91

$$Precision = \frac{TP}{TP+FP}$$

Precision
= Positive Predictive Value (PPV)

F1 = 0.66
= 2*(0.63*0.69)/(0.63+0.69)
= (2 * 63) /(100 + 91)
= (0.63 + 0.69) / 2 =1.32 / 2 =0.66

$$F = 2 * \frac{precision * recall}{precision + recall}$$

F1 score (F-score)
(F-measure)
is the harmonic mean of precision and recall
= 2TP / (P + P')
= 2TP / (2TP + FP + FN)

ACC = 0.68
= (63 + 72) / 200
= 135/200 = 67.5

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

A

63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

TPR = 0.63

FPR = 0.28

PPV = 0.69
=63/(63+28)
=63/91

F1 = 0.66
= 2*(0.63*0.69)/(0.63+0.69)
= (2 * 63) /(100 + 91)
= (0.63 + 0.69) / 2 =1.32 / 2 =0.66

ACC = 0.68
= (63 + 72) / 200
= 135/200 = 67.5

B

77 (TP)	77 (FP)	154
23 (FN)	23 (TN)	46
100	100	200

TPR = 0.77
FPR = 0.77
PPV = 0.50
F1 = 0.61
ACC = 0.50

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

Precision
= Positive Predictive Value (PPV)

$$Precision = \frac{TP}{TP + FP}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

C

24 (TP)	88 (FP)	112
76 (FN)	12 (TN)	88
100	100	200

TPR = 0.24
FPR = 0.88
PPV = 0.21
F1 = 0.22
ACC = 0.18

C’

76 (TP)	12 (FP)	88
24 (FN)	88 (TN)	112
100	100	200

TPR = 0.76
FPR = 0.12
PPV = 0.86
F1 = 0.81
ACC = 0.82

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

Precision
= Positive Predictive Value (PPV)

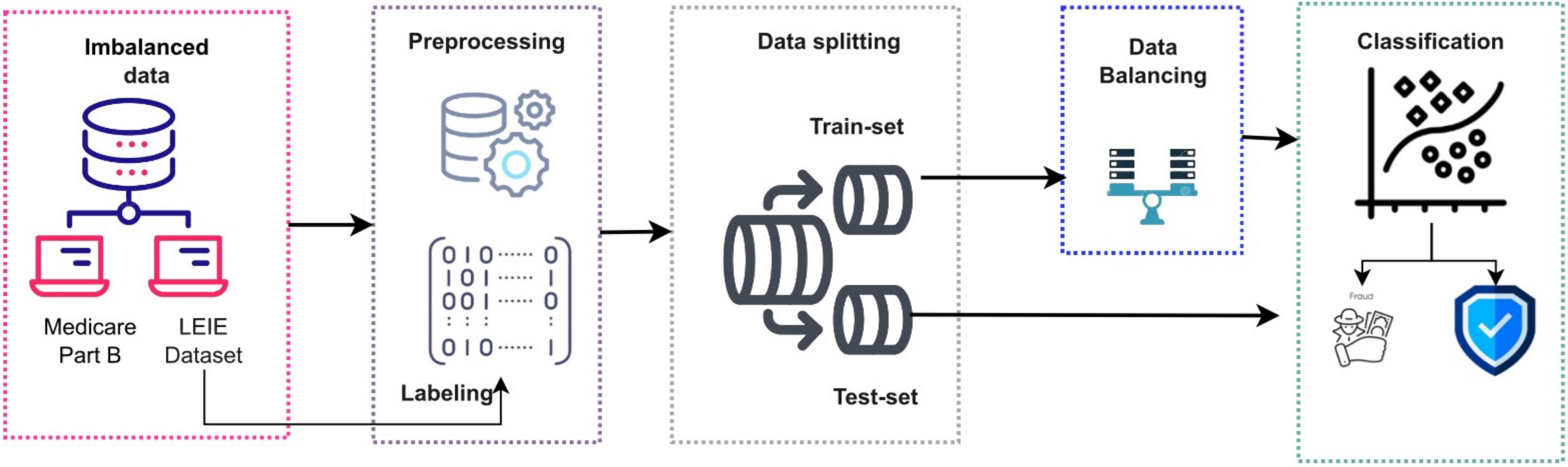
$$Precision = \frac{TP}{TP + FP}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN

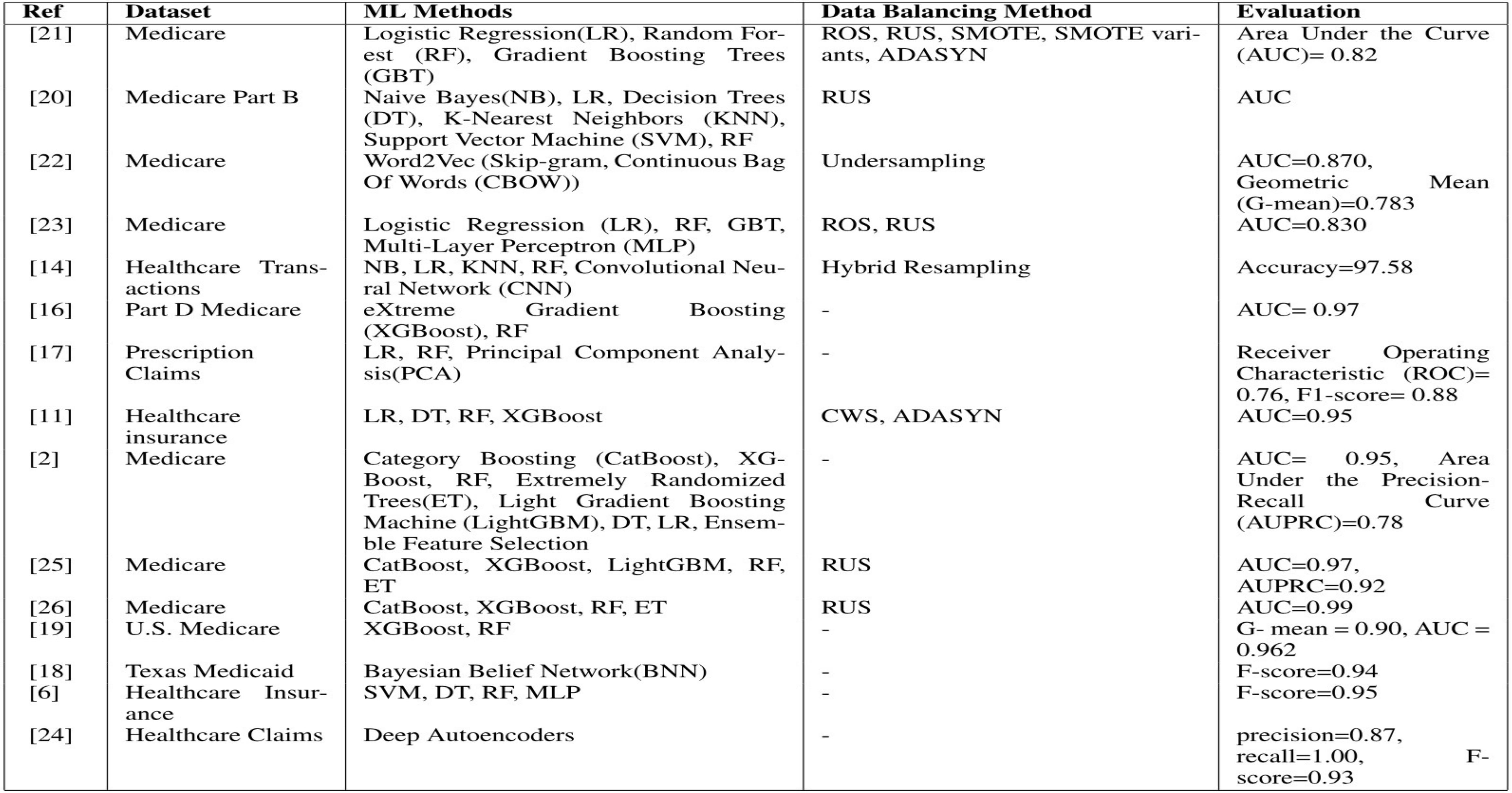
Architecture for healthcare fraud detection based on SMOTE-ENN

Synthetic Minority Over-sampling technique with Edited Nearest Neighbors (SMOTE-ENN)



Source: Bounab, R., Zarour, K., Guelib, B., & Khlifa, N. (2024). Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN. IEEE Access

Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance



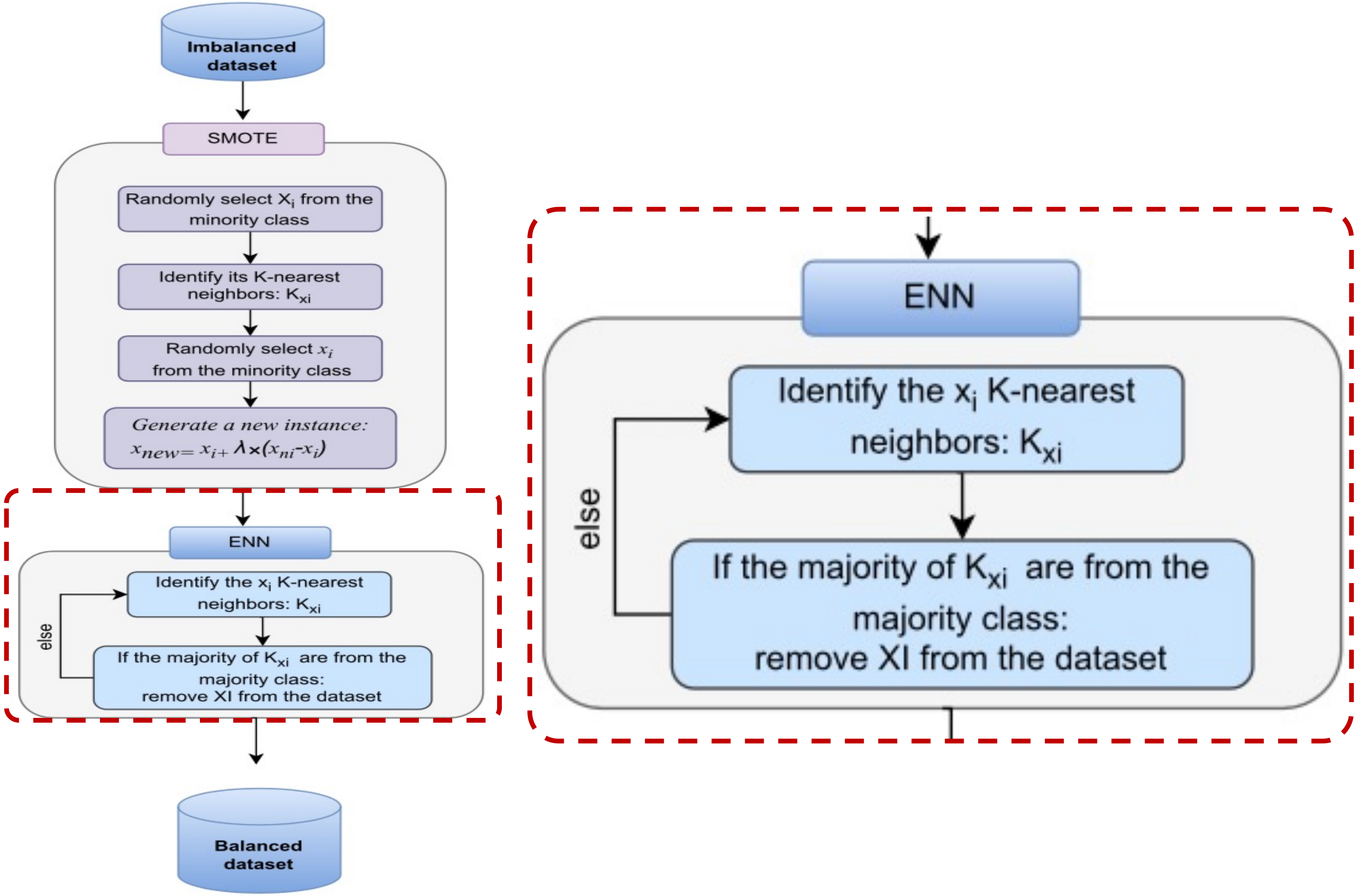
Ref	Dataset	ML Methods	Data Balancing Method	Evaluation
[21]	Medicare	Logistic Regression(LR), Random Forest (RF), Gradient Boosting Trees (GBT)	ROS, RUS, SMOTE, SMOTE variants, ADASYN	Area Under the Curve (AUC)= 0.82
[20]	Medicare Part B	Naive Bayes(NB), LR, Decision Trees (DT), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), RF	RUS	AUC
[22]	Medicare	Word2Vec (Skip-gram, Continuous Bag Of Words (CBOW))	Undersampling	AUC=0.870, Geometric Mean (G-mean)=0.783
[23]	Medicare	Logistic Regression (LR), RF, GBT, Multi-Layer Perceptron (MLP)	ROS, RUS	AUC=0.830
[14]	Healthcare Transactions	NB, LR, KNN, RF, Convolutional Neural Network (CNN)	Hybrid Resampling	Accuracy=97.58
[16]	Part D Medicare	eXtreme Gradient Boosting (XGBoost), RF	-	AUC= 0.97
[17]	Prescription Claims	LR, RF, Principal Component Analysis(PCA)	-	Receiver Operating Characteristic (ROC)= 0.76, F1-score= 0.88
[11]	Healthcare insurance	LR, DT, RF, XGBoost	CWS, ADASYN	AUC=0.95
[2]	Medicare	Category Boosting (CatBoost), XGBoost, RF, Extremely Randomized Trees(ET), Light Gradient Boosting Machine (LightGBM), DT, LR, Ensemble Feature Selection	-	AUC= 0.95, Area Under the Precision-Recall Curve (AUPRC)=0.78
[25]	Medicare	CatBoost, XGBoost, LightGBM, RF, ET	RUS	AUC=0.97, AUPRC=0.92
[26]	Medicare	CatBoost, XGBoost, RF, ET	RUS	AUC=0.99
[19]	U.S. Medicare	XGBoost, RF	-	G- mean = 0.90, AUC = 0.962
[18]	Texas Medicaid	Bayesian Belief Network(BNN)	-	F-score=0.94
[6]	Healthcare Insurance	SVM, DT, RF, MLP	-	F-score=0.95
[24]	Healthcare Claims	Deep Autoencoders	-	precision=0.87, recall=1.00, F-score=0.93

Source: Bounab, R., Zarour, K., Guelib, B., & Khlifa, N. (2024). Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN. IEEE Access

Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN

SMOTE-ENN process

Synthetic Minority Over-sampling technique with Edited Nearest Neighbors (SMOTE-ENN)

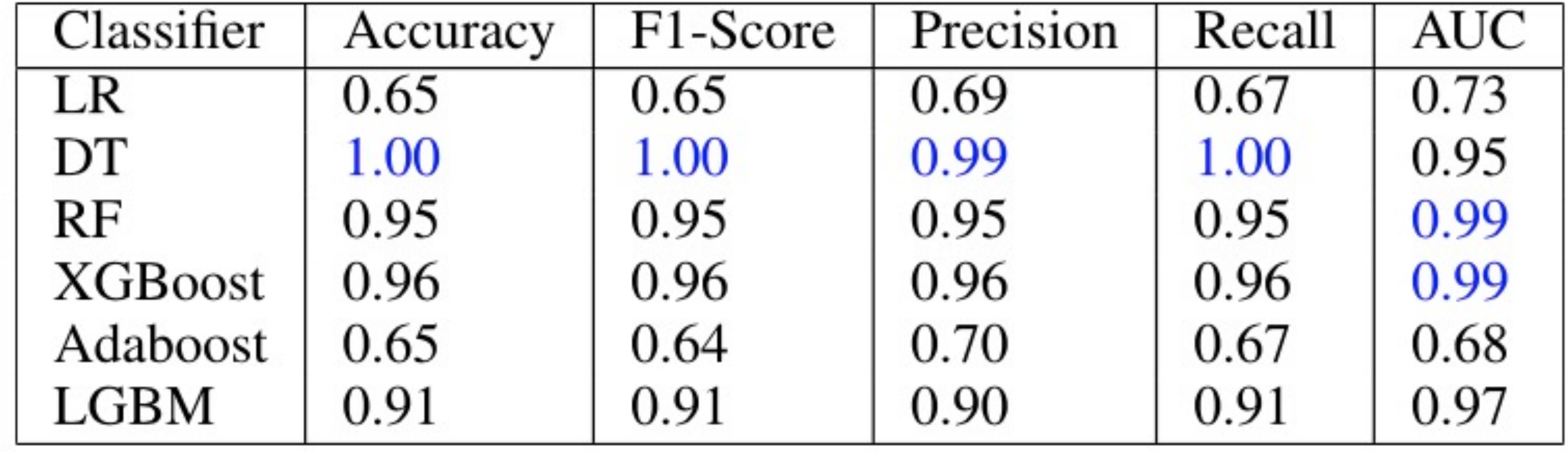


Source: Bounab, R., Zarour, K., Guelib, B., & Khlifa, N. (2024). Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN. IEEE Access

Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN

Synthetic Minority Over-sampling technique with Edited Nearest Neighbors (SMOTE-ENN)

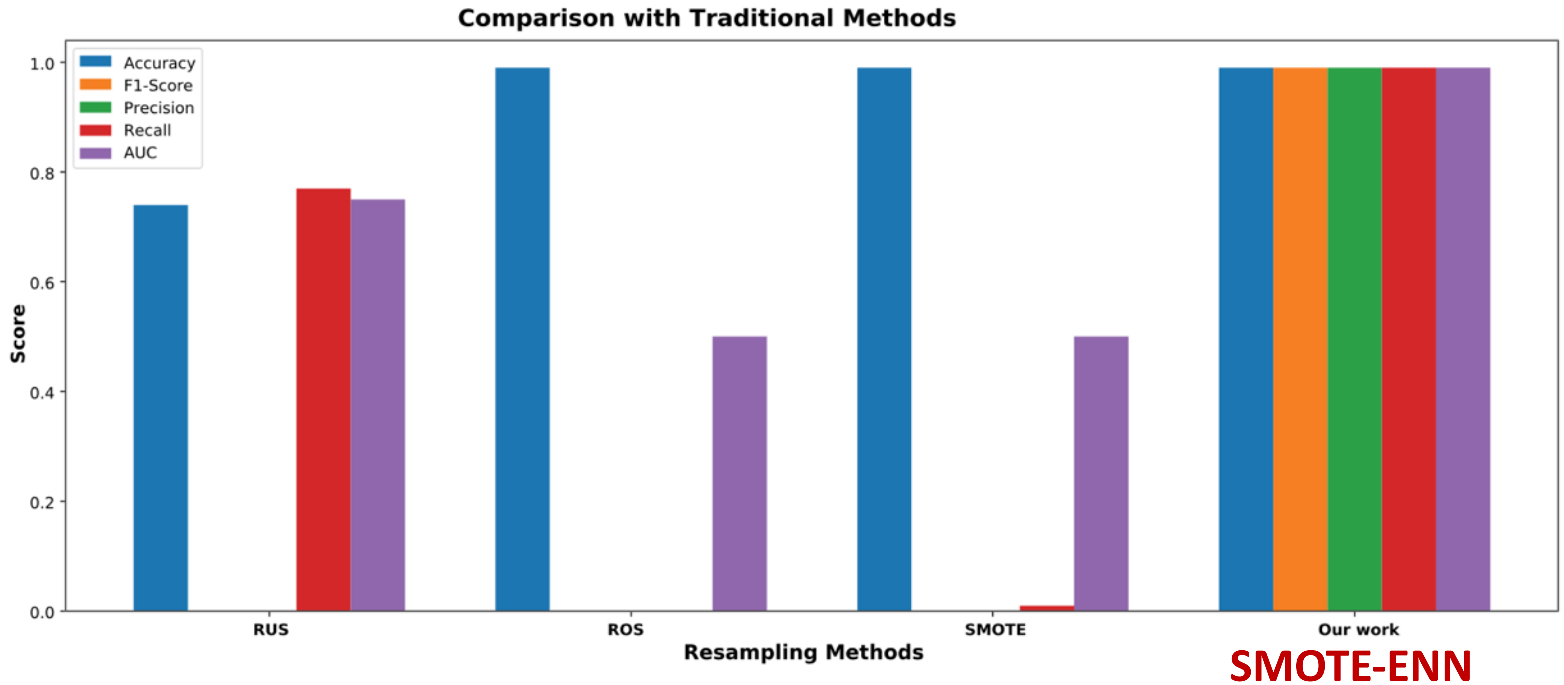
Classification results using SMOTE-ENN and cross-validation



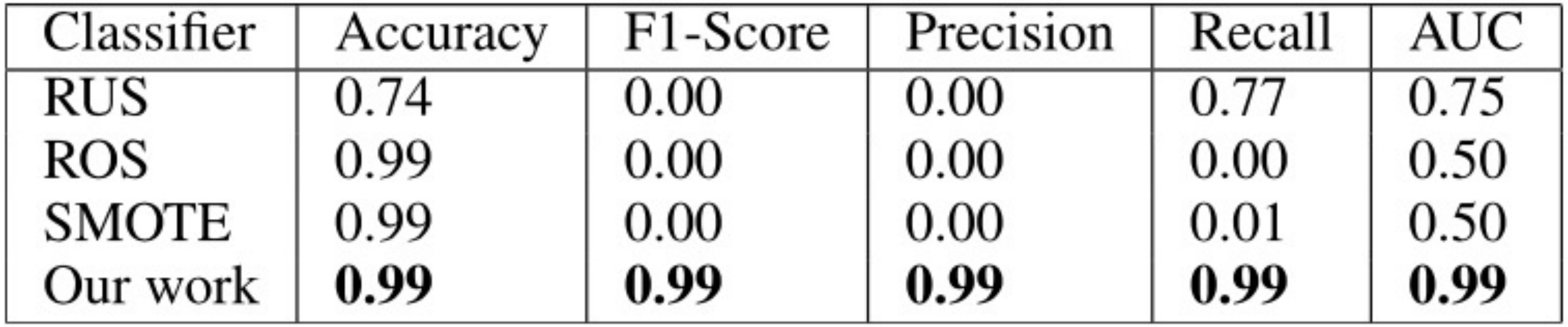
Classifier	Accuracy	F1-Score	Precision	Recall	AUC
LR	0.65	0.65	0.69	0.67	0.73
DT	1.00	1.00	0.99	1.00	0.95
RF	0.95	0.95	0.95	0.95	0.99
XGBoost	0.96	0.96	0.96	0.96	0.99
Adaboost	0.65	0.64	0.70	0.67	0.68
LGBM	0.91	0.91	0.90	0.91	0.97

Source: Bounab, R., Zarour, K., Guelib, B., & Khlifa, N. (2024). Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN. IEEE Access

Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN



SMOTE-ENN

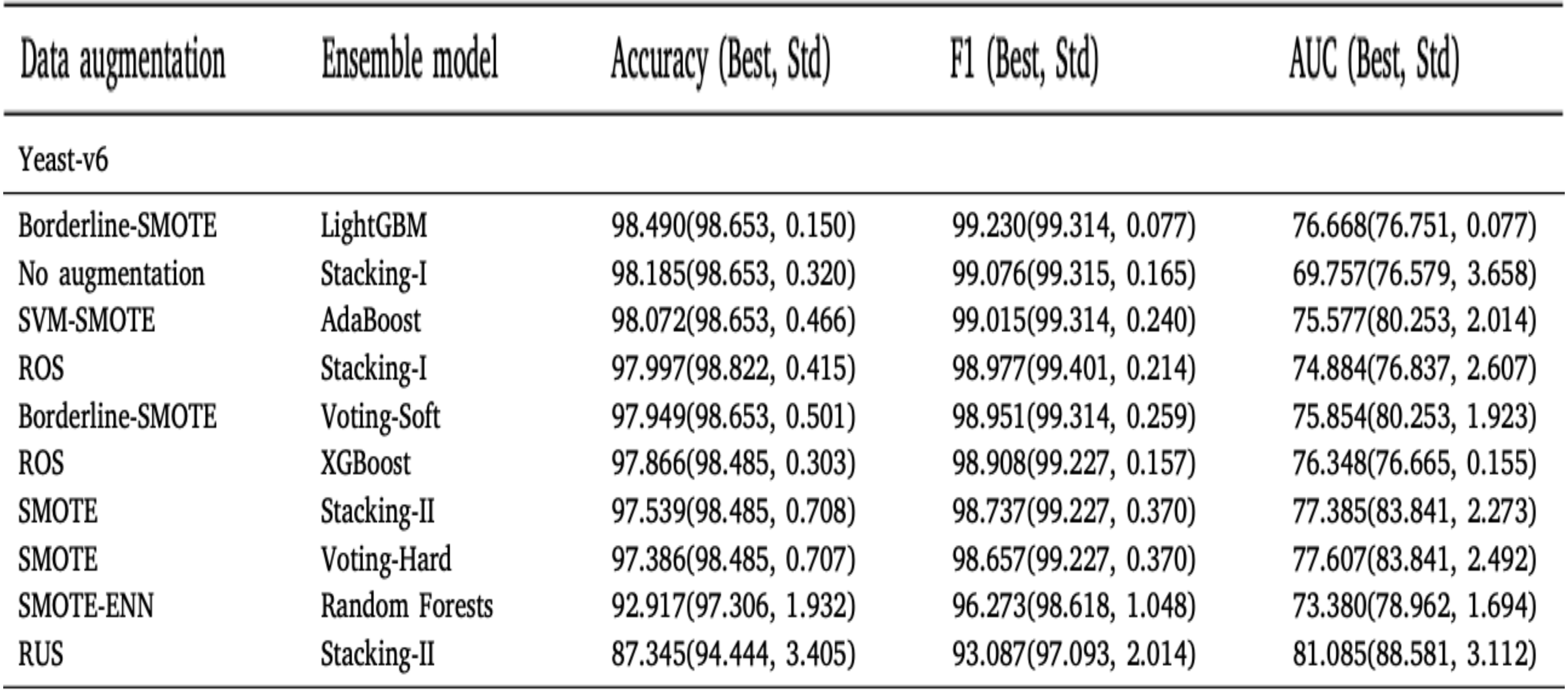


Classifier	Accuracy	F1-Score	Precision	Recall	AUC
RUS	0.74	0.00	0.00	0.77	0.75
ROS	0.99	0.00	0.00	0.00	0.50
SMOTE	0.99	0.00	0.00	0.01	0.50
Our work	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

SMOTE-ENN

Source: Bounab, R., Zarour, K., Guelib, B., & Khlifa, N. (2024). Enhancing Medicare Fraud Detection Through Machine Learning: Addressing Class Imbalance With SMOTE-ENN. IEEE Access

ML Evaluation of Imbalanced Dataset: Ensemble Learning and Data Augmentations (DA)



Data augmentation	Ensemble model	Accuracy (Best, Std)	F1 (Best, Std)	AUC (Best, Std)
Yeast-v6				
Borderline-SMOTE	LightGBM	98.490(98.653, 0.150)	99.230(99.314, 0.077)	76.668(76.751, 0.077)
No augmentation	Stacking-I	98.185(98.653, 0.320)	99.076(99.315, 0.165)	69.757(76.579, 3.658)
SVM-SMOTE	AdaBoost	98.072(98.653, 0.466)	99.015(99.314, 0.240)	75.577(80.253, 2.014)
ROS	Stacking-I	97.997(98.822, 0.415)	98.977(99.401, 0.214)	74.884(76.837, 2.607)
Borderline-SMOTE	Voting-Soft	97.949(98.653, 0.501)	98.951(99.314, 0.259)	75.854(80.253, 1.923)
ROS	XGBoost	97.866(98.485, 0.303)	98.908(99.227, 0.157)	76.348(76.665, 0.155)
SMOTE	Stacking-II	97.539(98.485, 0.708)	98.737(99.227, 0.370)	77.385(83.841, 2.273)
SMOTE	Voting-Hard	97.386(98.485, 0.707)	98.657(99.227, 0.370)	77.607(83.841, 2.492)
SMOTE-ENN	Random Forests	92.917(97.306, 1.932)	96.273(98.618, 1.048)	73.380(78.962, 1.694)
RUS	Stacking-II	87.345(94.444, 3.405)	93.087(97.093, 2.014)	81.085(88.581, 3.112)

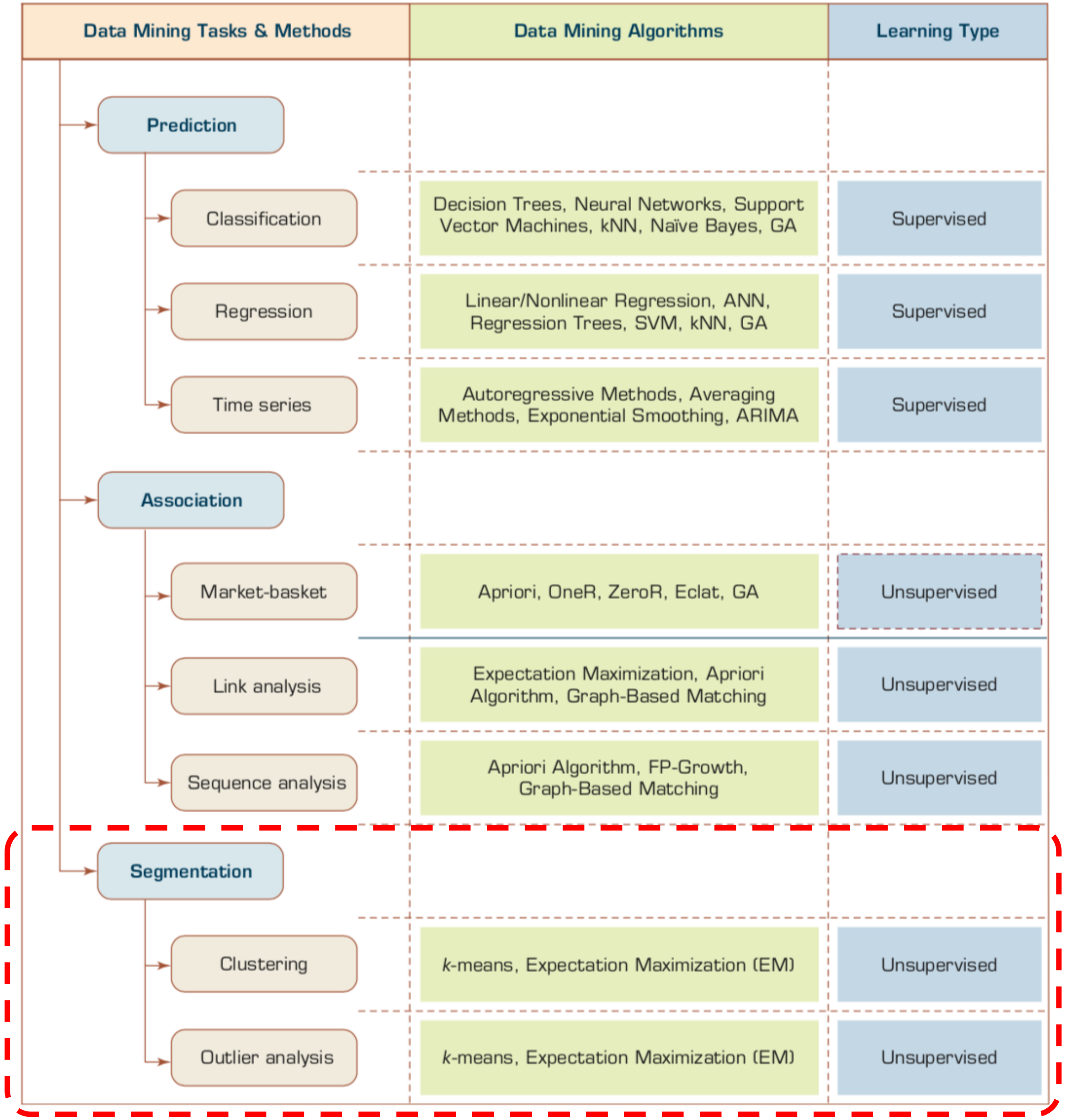
Source: Khan, A. A., Chaudhari, O., & Chandra, R. (2023). A review of ensemble learning and data augmentation models for class imbalanced problems: combination, implementation and evaluation. Expert Systems with Applications, 122778.

Machine Learning: Unsupervised Learning: Cluster Analysis, Market Segmentation

Machine Learning: Data Mining Tasks & Methods

Unsupervised Learning:
Cluster Analysis,
Market Segmentation

Segmentation



Data Mining Tasks & Methods	Data Mining Algorithms	Learning Type
Prediction		
Classification	Decision Trees, Neural Networks, Support Vector Machines, kNN, Naïve Bayes, GA	Supervised
Regression	Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA	Supervised
Time series	Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA	Supervised
Association		
Market-basket	Apriori, OneR, ZeroR, Eclat, GA	Unsupervised
Link analysis	Expectation Maximization, Apriori Algorithm, Graph-Based Matching	Unsupervised
Sequence analysis	Apriori Algorithm, FP-Growth, Graph-Based Matching	Unsupervised
Segmentation		
Clustering	*k*-means, Expectation Maximization (EM)	Unsupervised
Outlier analysis	*k*-means, Expectation Maximization (EM)	Unsupervised

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Example of Cluster Analysis

Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

K-Means Clustering

Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1	(3.67, 5.83)
m2	(6.75, 3.50)

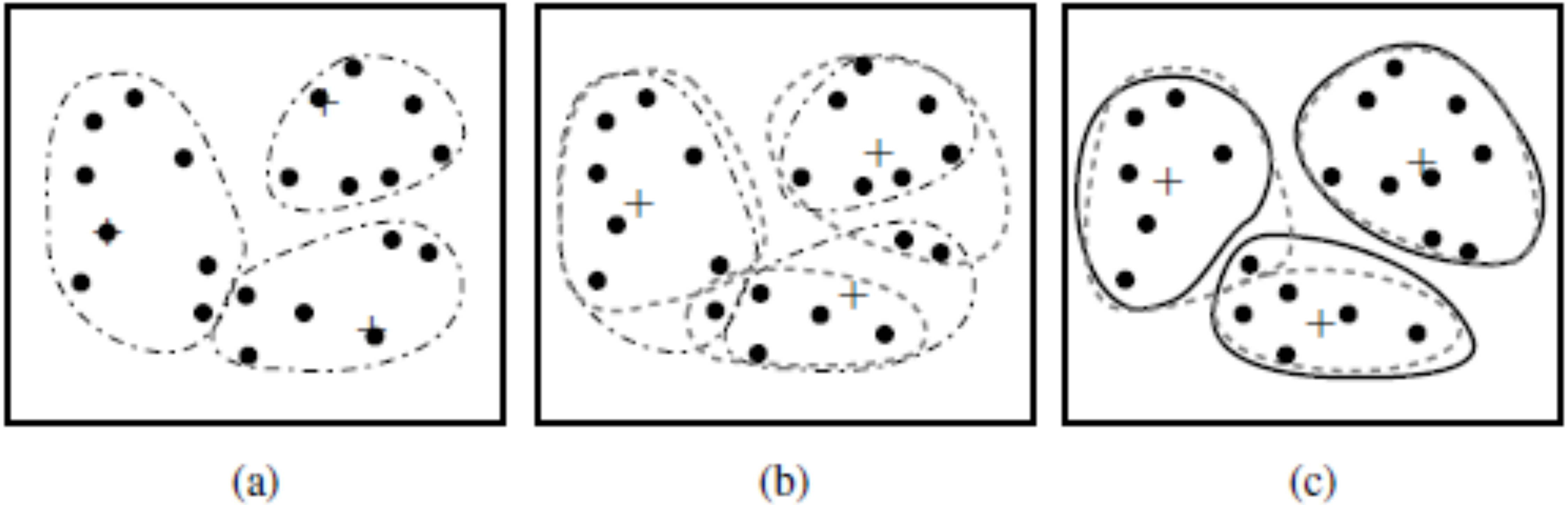
Cluster Analysis

Cluster Analysis

- Used for automatic identification of natural groupings of things
- Part of the machine-learning family
- Employ unsupervised learning
- Learns the clusters of things from past data, then assigns new instances
- There is not an output variable
- Also known as segmentation

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Cluster Analysis



Clustering of a set of objects based on the *k-means method.*
(The mean of each cluster is marked by a "+".)

Source: Han & Kamber (2006)

Cluster Analysis

- **Clustering results may be used to**
  - **Identify natural groupings of customers**
  - **Identify rules for assigning new cases to classes for targeting/diagnostic purposes**
  - **Provide characterization, definition, labeling of populations**
  - **Decrease the size and complexity of problems for other data mining methods**
  - **Identify outliers in a specific domain (e.g., rare-event detection)**

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

k-Means Clustering Algorithm

- *k* : pre-determined number of clusters
- Algorithm (Step 0: determine value of *k*)

Step 1: Randomly generate *k* random points as initial cluster centers

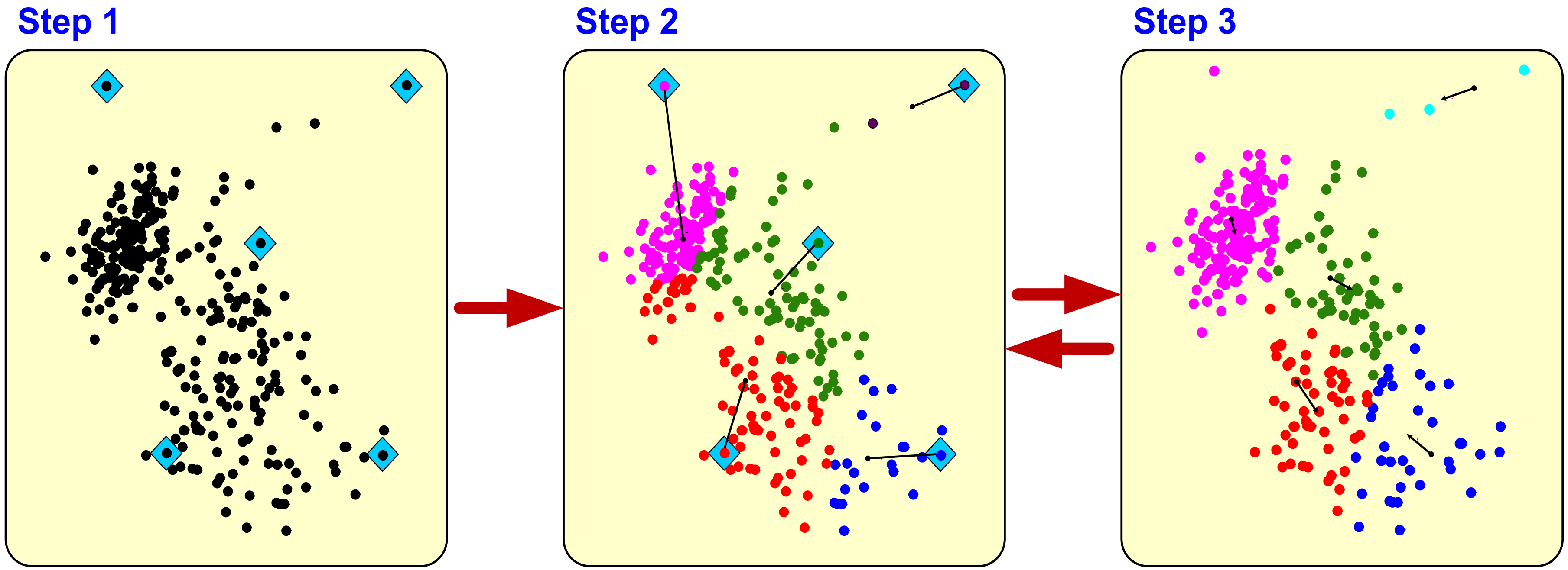
Step 2: Assign each point to the nearest cluster center

Step 3: Re-compute the new cluster centers

Repetition step: Repeat steps 2 and 3 until some convergence criterion is met (usually that the assignment of points to clusters becomes stable)

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Cluster Analysis for Data Mining - *k*-Means Clustering Algorithm



Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Similarity

Distance

Similarity and Dissimilarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects

- Some popular ones include: ***Minkowski distance***:

$$d(i,j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \ldots + |x_{i_p} - x_{j_p}|^q)}$$

where *i* = (*x*$_{i1}$, *x*$_{i2}$, …, *x*$_{ip}$) and *j* = (*x*$_{j1}$, *x*$_{j2}$, …, *x*$_{jp}$) are two *p*-dimensional data objects, and *q* is a positive integer

- If *q* = *1*, *d* is Manhattan distance

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

Source: Han & Kamber (2006)

Similarity and Dissimilarity Between Objects (Cont.)

- ***If q = 2, d*** **is Euclidean distance:**

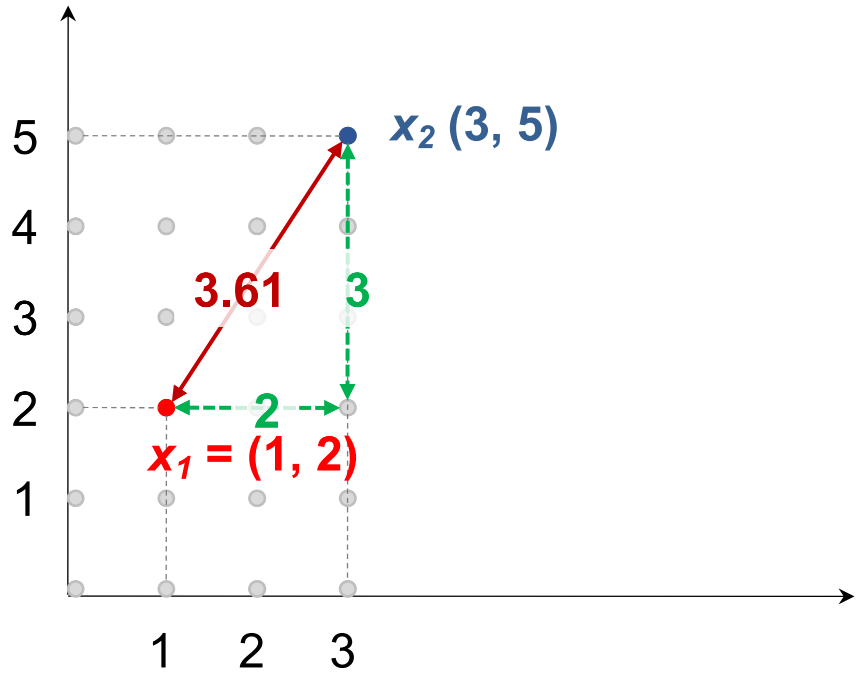
$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + ... + |x_{i_p} - x_{j_p}|^2)}$$

  - **Properties**
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$
- **Also, one can use weighted distance, parametric Pearson product moment correlation, or other disimilarity measures**

Source: Han & Kamber (2006)

Euclidean distance vs Manhattan distance

- **Distance of two point $x_1 = (1, 2)$ and x_2 $(3, 5)$**



Euclidean distance:

$= ((3-1)^2 + (5-2)^2)^{1/2}$

$= (2^2 + 3^2)^{1/2}$

$= (4 + 9)^{1/2}$

$= (13)^{1/2}$

$= 3.61$

Manhattan distance:

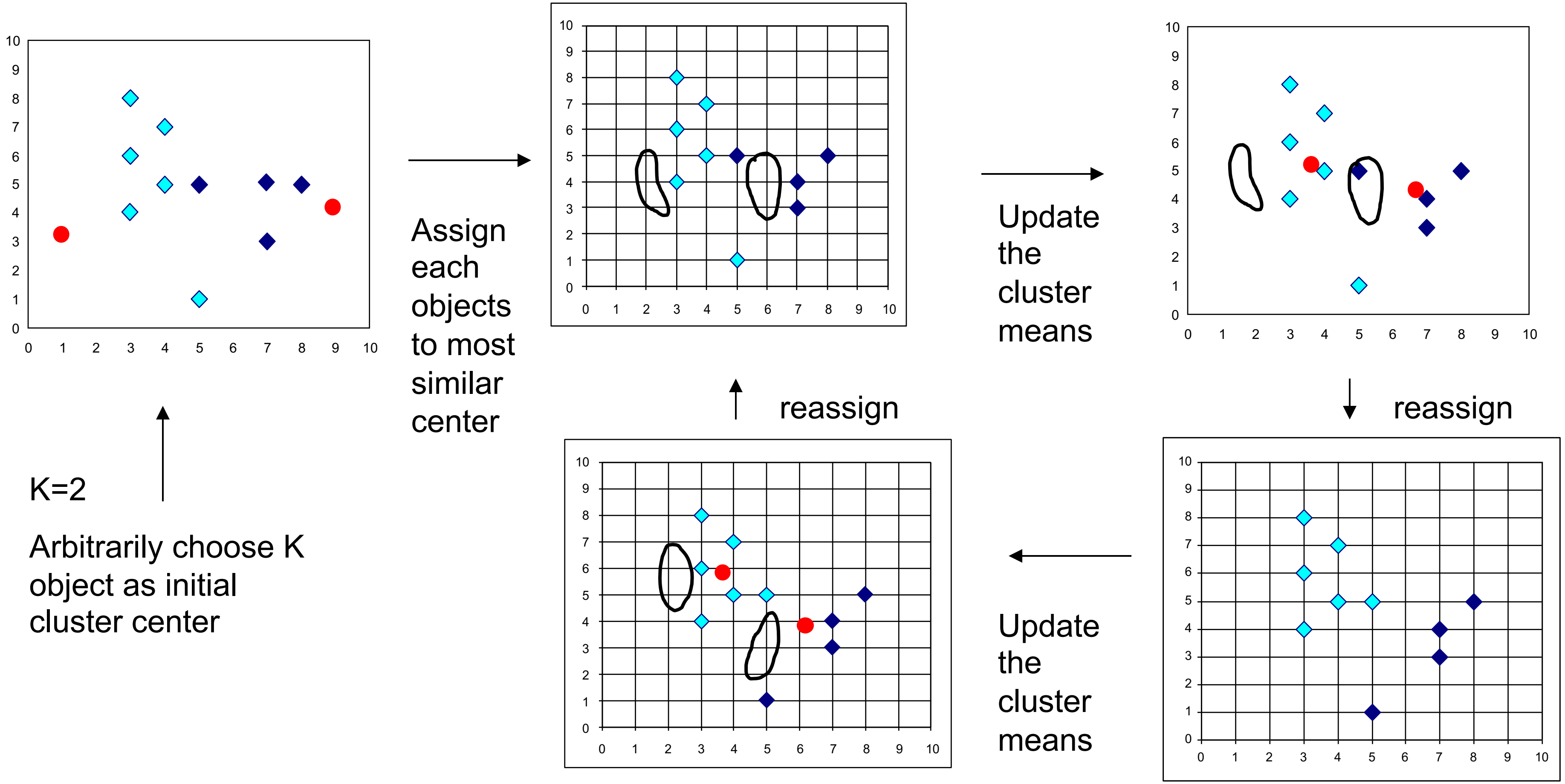
$= (3-1) + (5-2)$

$= 2 + 3$

$= 5$

The *K-Means* Clustering Method

- **Example**



K=2

Arbitrarily choose K object as initial cluster center

Assign each objects to most similar center

Update the cluster means

reassign

reassign

Update the cluster means

Source: Han & Kamber (2006)

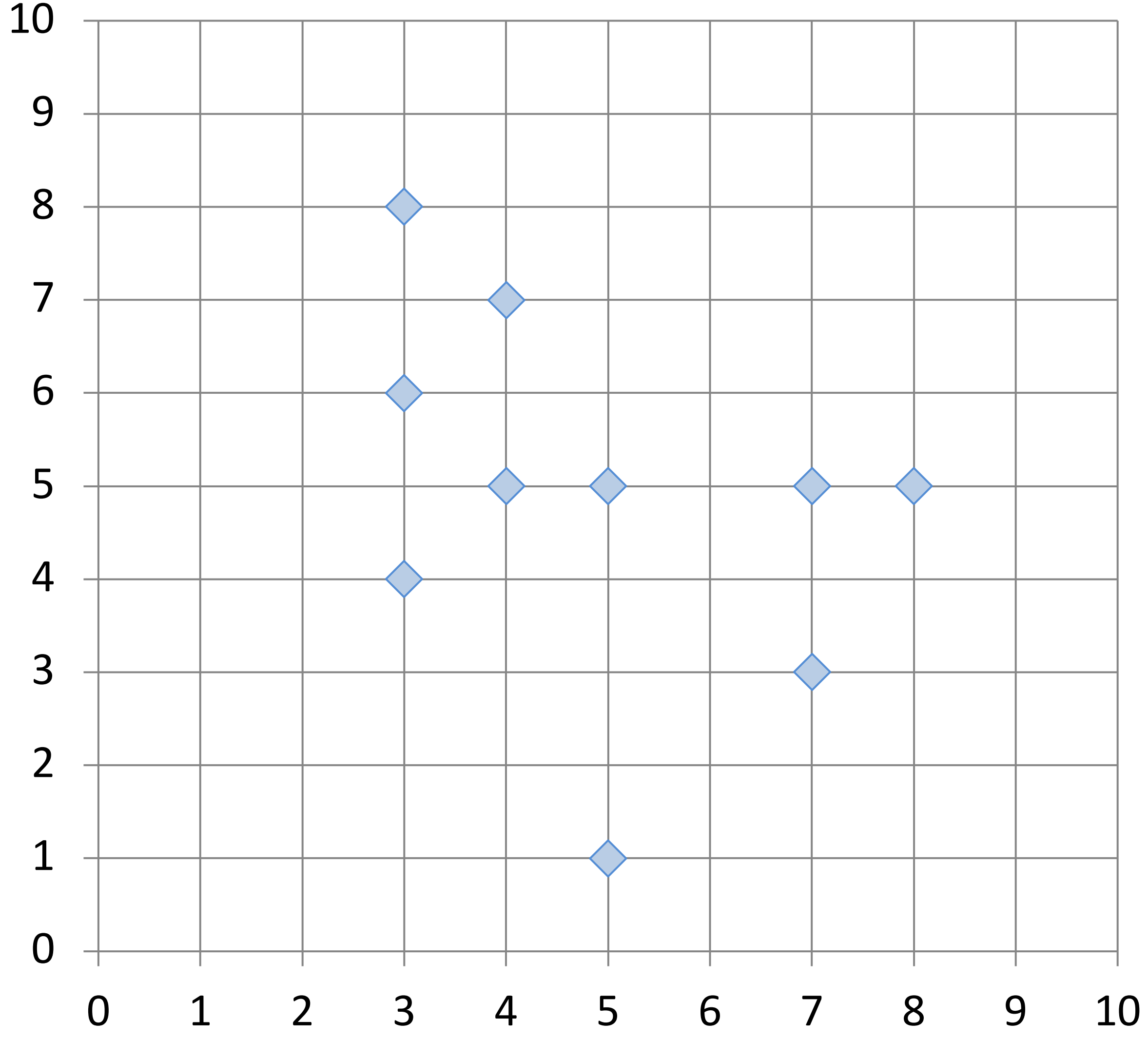
K-Means Clustering

Example of Cluster Analysis

Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

K-Means Clustering

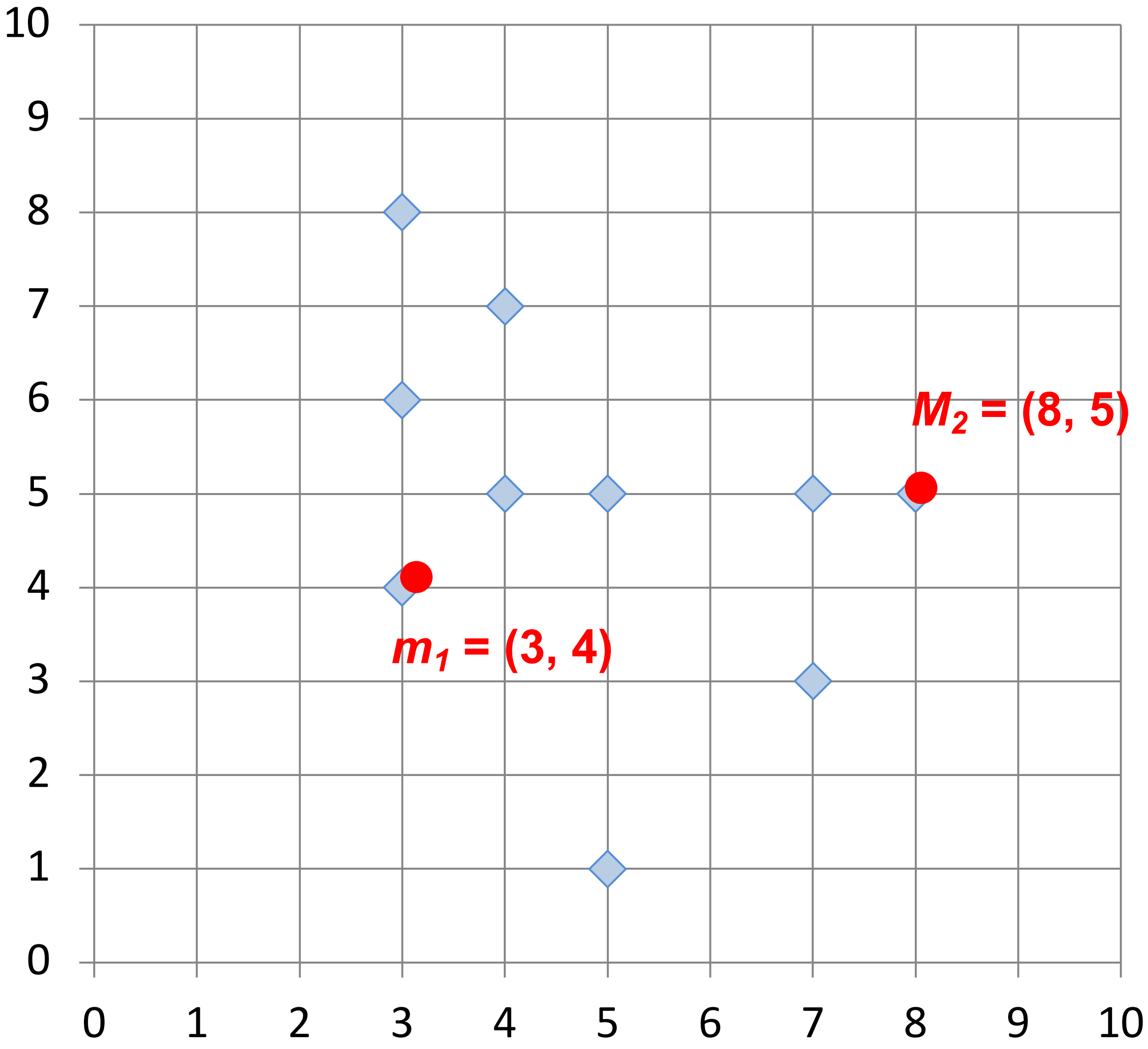
Step by Step



Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

K-Means Clustering

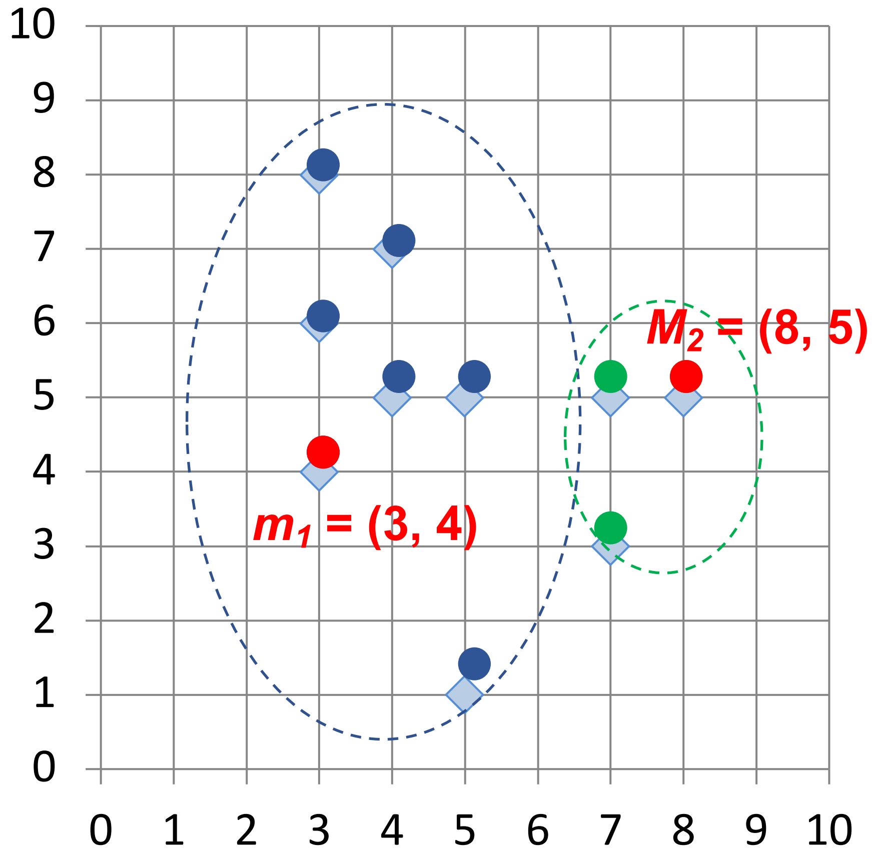
Step 1: K=2, Arbitrarily choose K object as initial cluster center



Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)
Initial	m1	(3, 4)
Initial	m2	(8, 5)

Step 2: Compute seed points as the centroids of the clusters of the current partition

Step 3: Assign each objects to most similar center



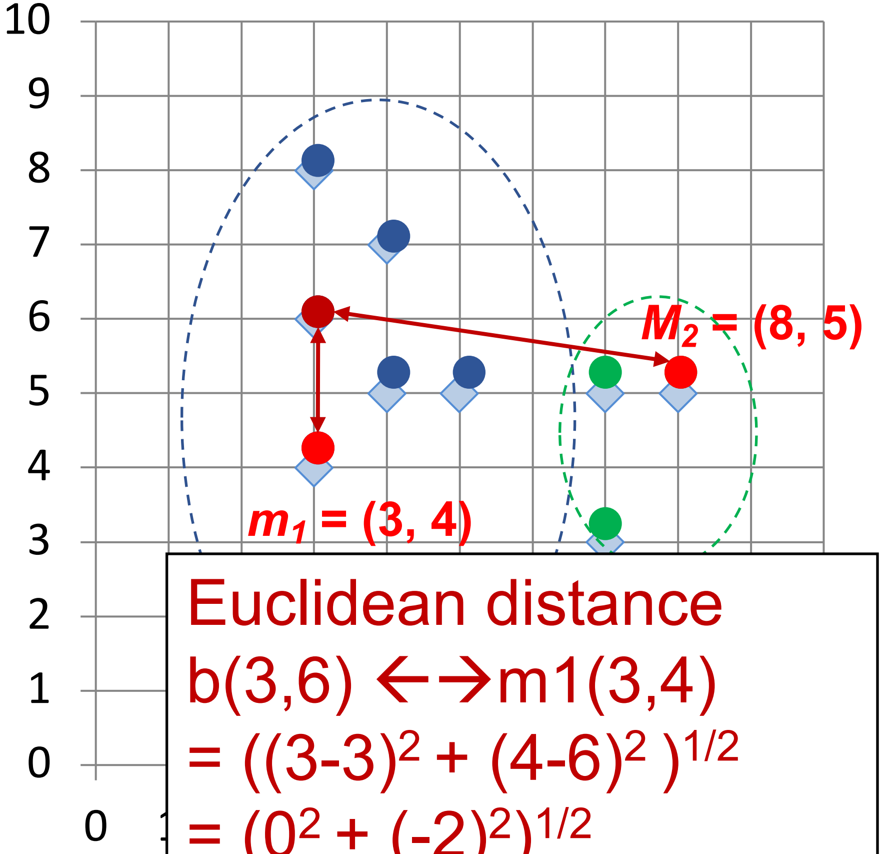
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	0.00	5.10	Cluster1
p02	b	(3, 6)	2.00	5.10	Cluster1
p03	c	(3, 8)	4.00	5.83	Cluster1
p04	d	(4, 5)	1.41	4.00	Cluster1
p05	e	(4, 7)	3.16	4.47	Cluster1
p06	f	(5, 1)	3.61	5.00	Cluster1
p07	g	(5, 5)	2.24	3.00	Cluster1
p08	h	(7, 3)	4.12	2.24	Cluster2
p09	i	(7, 5)	4.12	1.00	Cluster2
p10	j	(8, 5)	5.10	0.00	Cluster2

Initial m1 (3, 4)
Initial m2 (8, 5)

K-Means Clustering

Step 2: Compute seed points as the centroids of the clusters of the current partition

Step 3: Assign each objects to most similar center



K-

Euclidean distance

b(3,6) ←→ m1(3,4)

$= ((3-3)^2 + (4-6)^2)^{1/2}$

$= (0^2 + (-2)^2)^{1/2}$

$= (0 + 4)^{1/2}$

$= (4)^{1/2}$

= 2.00

Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	0.00	5.10	Cluster1
p02	b	(3, 6)	2.00	5.10	Cluster1
p03	c	(3, 8)	4.00	5.83	Cluster1
p04	d	(4, 5)	1.41	4.00	Cluster1
p05					ster1
p06					ster1
p07					ster1
p08					ster2
p09					ster2
p10					ster2

Euclidean distance

b(3,6) ←→ m2(8,5)

$= ((8-3)^2 + (5-6)^2)^{1/2}$

$= (5^2 + (-1)^2)^{1/2}$

$= (25 + 1)^{1/2}$

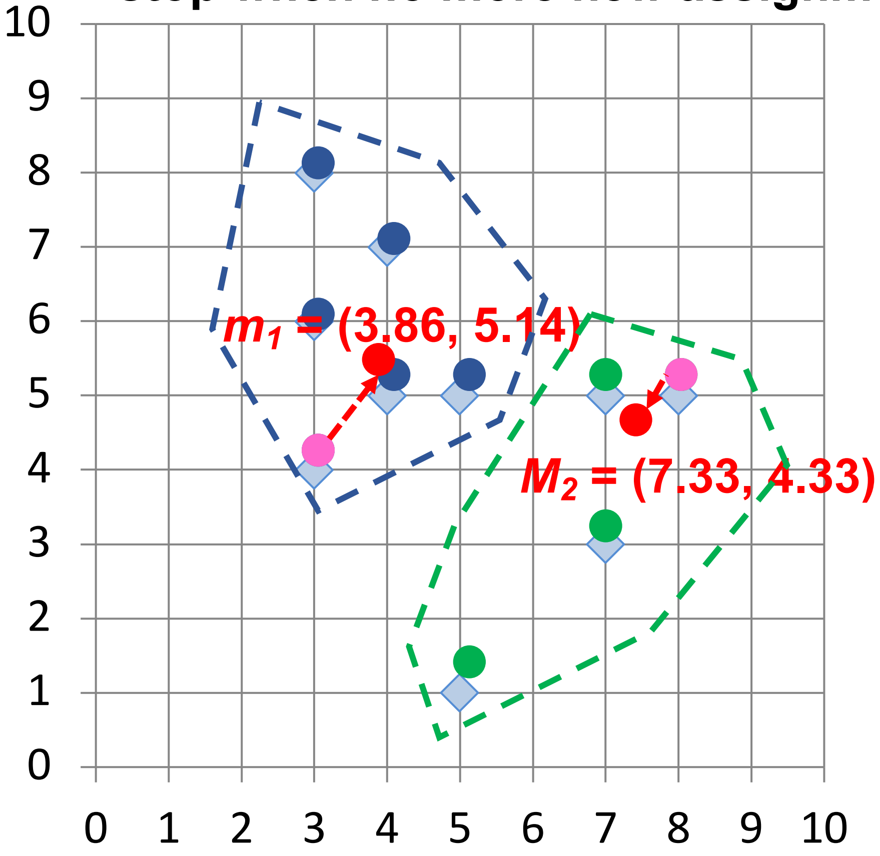
$= (26)^{1/2}$

= 5.10

Initial m1 (3, 4)

Initial m2 (8, 5)

Step 4: Update the cluster means,
Repeat Step 2, 3,
stop when no more new assignment

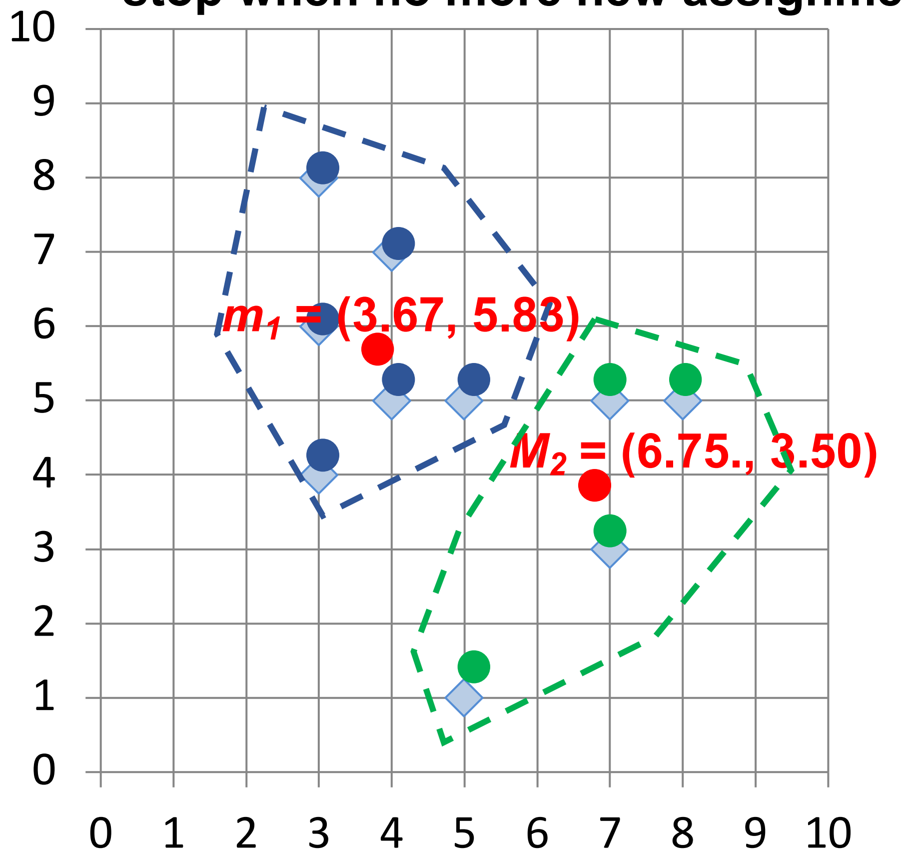


Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.43	4.34	Cluster1
p02	b	(3, 6)	1.22	4.64	Cluster1
p03	c	(3, 8)	2.99	5.68	Cluster1
p04	d	(4, 5)	0.20	3.40	Cluster1
p05	e	(4, 7)	1.87	4.27	Cluster1
p06	f	(5, 1)	4.29	4.06	Cluster2
p07	g	(5, 5)	1.15	2.42	Cluster1
p08	h	(7, 3)	3.80	1.37	Cluster2
p09	i	(7, 5)	3.14	0.75	Cluster2
p10	j	(8, 5)	4.14	0.95	Cluster2

m1	(3.86, 5.14)
m2	(7.33, 4.33)

K-Means Clustering

Step 4: Update the cluster means,
Repeat Step 2, 3,
stop when no more new assignment

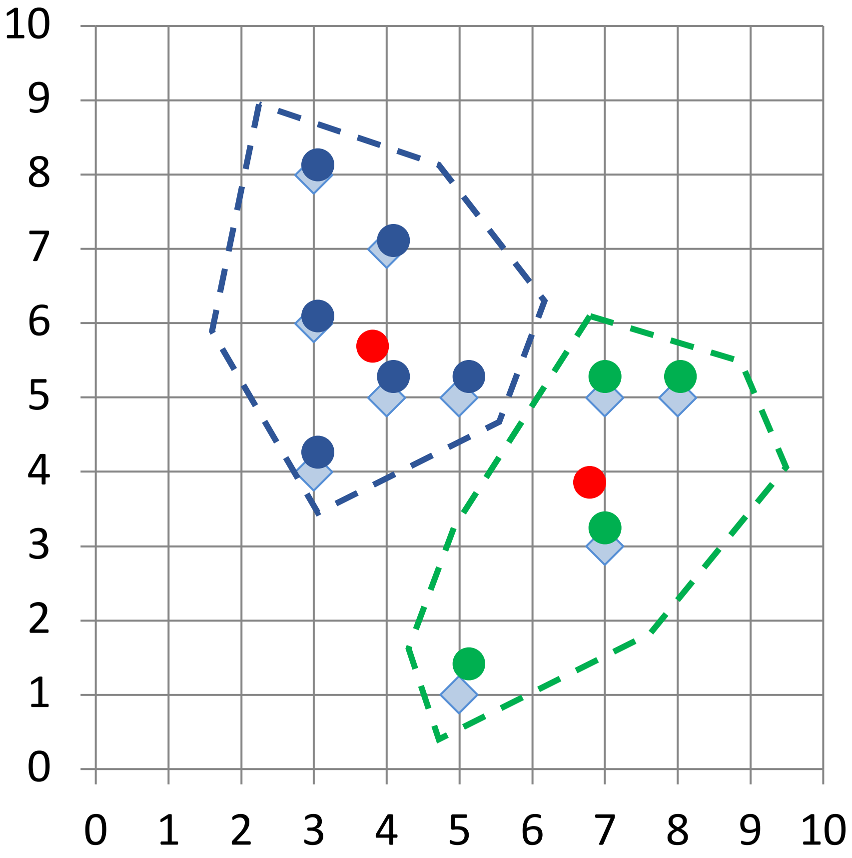


Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1	(3.67, 5.83)
m2	(6.75, 3.50)

K-Means Clustering

stop when no more new assignment



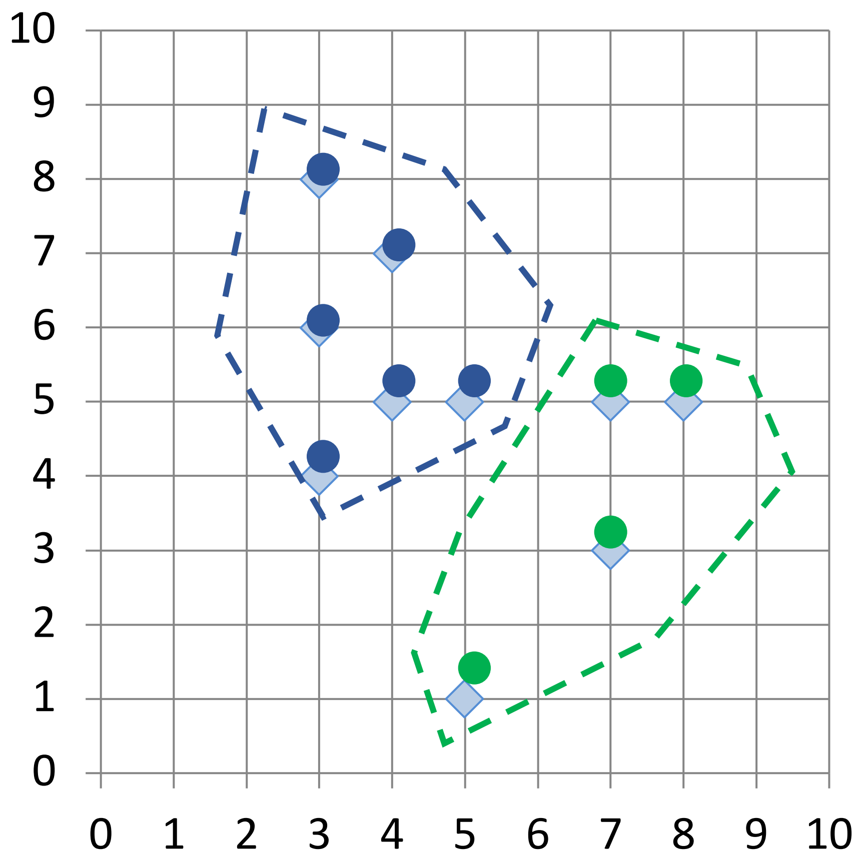
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1	(3.67, 5.83)
m2	(6.75, 3.50)

K-Means Clustering

K-Means Clustering (*K*=2, two clusters)

stop when no more new assignment



Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1	(3.67, 5.83)
m2	(6.75, 3.50)

K-Means Clustering

K-Means Clustering

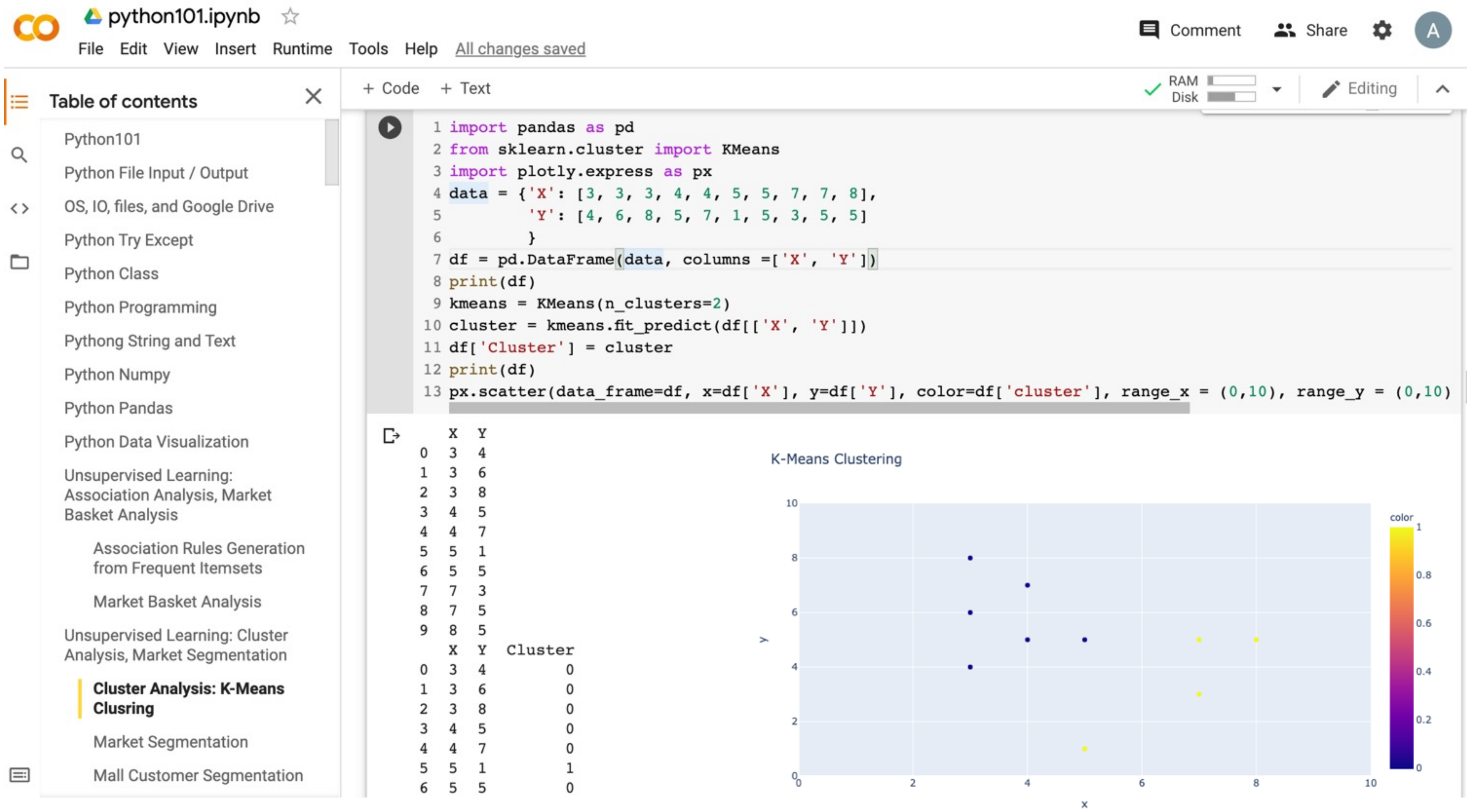
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1	(3.67, 5.83)
m2	(6.75, 3.50)

Machine Learning: Unsupervised Learning: Cluster Analysis
K-Means Clustering

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



```python
import pandas as pd
from sklearn.cluster import KMeans
import plotly.express as px
data = {'X': [3, 3, 3, 4, 4, 5, 5, 7, 7, 8],
        'Y': [4, 6, 8, 5, 7, 1, 5, 3, 5, 5]
       }
df = pd.DataFrame(data, columns =['X', 'Y'])
print(df)
kmeans = KMeans(n_clusters=2)
cluster = kmeans.fit_predict(df[['X', 'Y']])
df['Cluster'] = cluster
print(df)
px.scatter(data_frame=df, x=df['X'], y=df['Y'], color=df['cluster'], range_x = (0,10), range_y = (0,10)
```

```
   X  Y
0  3  4
1  3  6
2  3  8
3  4  5
4  4  7
5  5  1
6  5  5
7  7  3
8  7  5
9  8  5
   X  Y  Cluster
0  3  4        0
1  3  6        0
2  3  8        0
3  4  5        0
4  4  7        0
5  5  1        1
6  5  5        0
```

https://tinyurl.com/aintpupython101

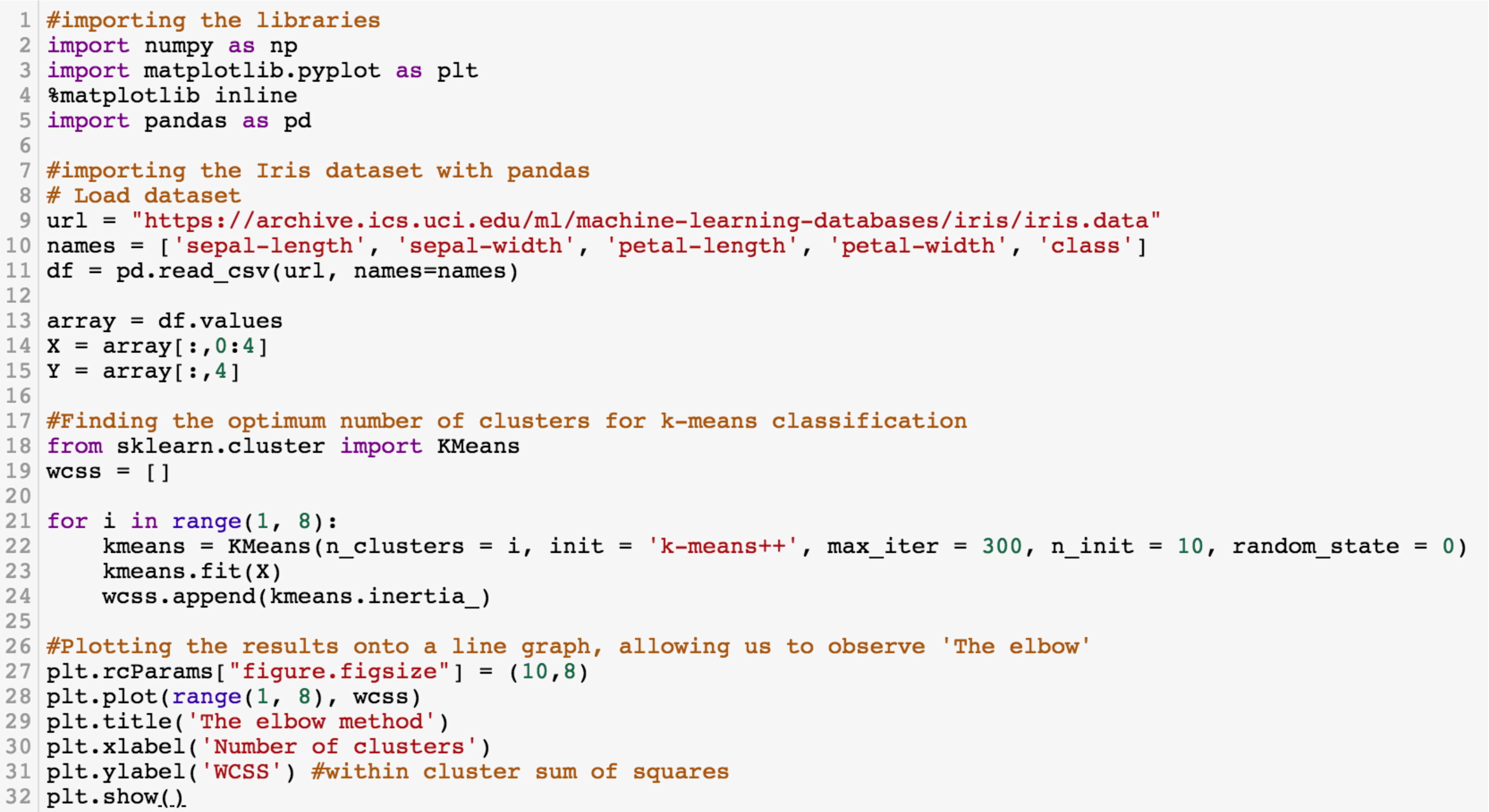
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
cluster = kmeans.fit_predict(df[['X', 'Y']])
```

```
import pandas as pd
from sklearn.cluster import KMeans
import plotly.express as px
data = {'X': [3, 3, 3, 4, 4, 5, 5, 7, 7, 8],
        'Y': [4, 6, 8, 5, 7, 1, 5, 3, 5, 5]
        }
df = pd.DataFrame(data, columns =['X', 'Y'])
print(df)
kmeans = KMeans(n_clusters=2)
cluster = kmeans.fit_predict(df[['X', 'Y']])
df['Cluster'] = cluster
print(df)
px.scatter(data_frame=df, x=df['X'], y=df['Y'], color=df['cluster'], range_x = (0,10), range_y = (0,10), title='K-Means Clustering')
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
cluster = kmeans.fit_predict(df[['X', 'Y']])
```

```
import pandas as pd
from sklearn.cluster import KMeans
import plotly.express as px
data = {'X': [3, 3, 3, 4, 4, 5, 5, 7, 7, 8],
'Y': [4, 6, 8, 5, 7, 1, 5, 3, 5, 5]
}
df = pd.DataFrame(data, columns =['X', 'Y'])
print(df)
kmeans = KMeans(n_clusters=2)
cluster = kmeans.fit_predict(df[['X', 'Y']])
df['Cluster'] = cluster
print(df)
px.scatter(data_frame=df, x=df['X'], y=df['Y'],
color=df['cluster'], range_x = (0,10), range_y = (0,10),
title='K-Means Clustering')
```

K-Means Clustering



```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd

#importing the Iris dataset with pandas
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

array = df.values
X = array[:,0:4]
Y = array[:,4]

#Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 8):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#Plotting the results onto a line graph, allowing us to observe 'The elbow'
plt.rcParams["figure.figsize"] = (10,8)
plt.plot(range(1, 8), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```

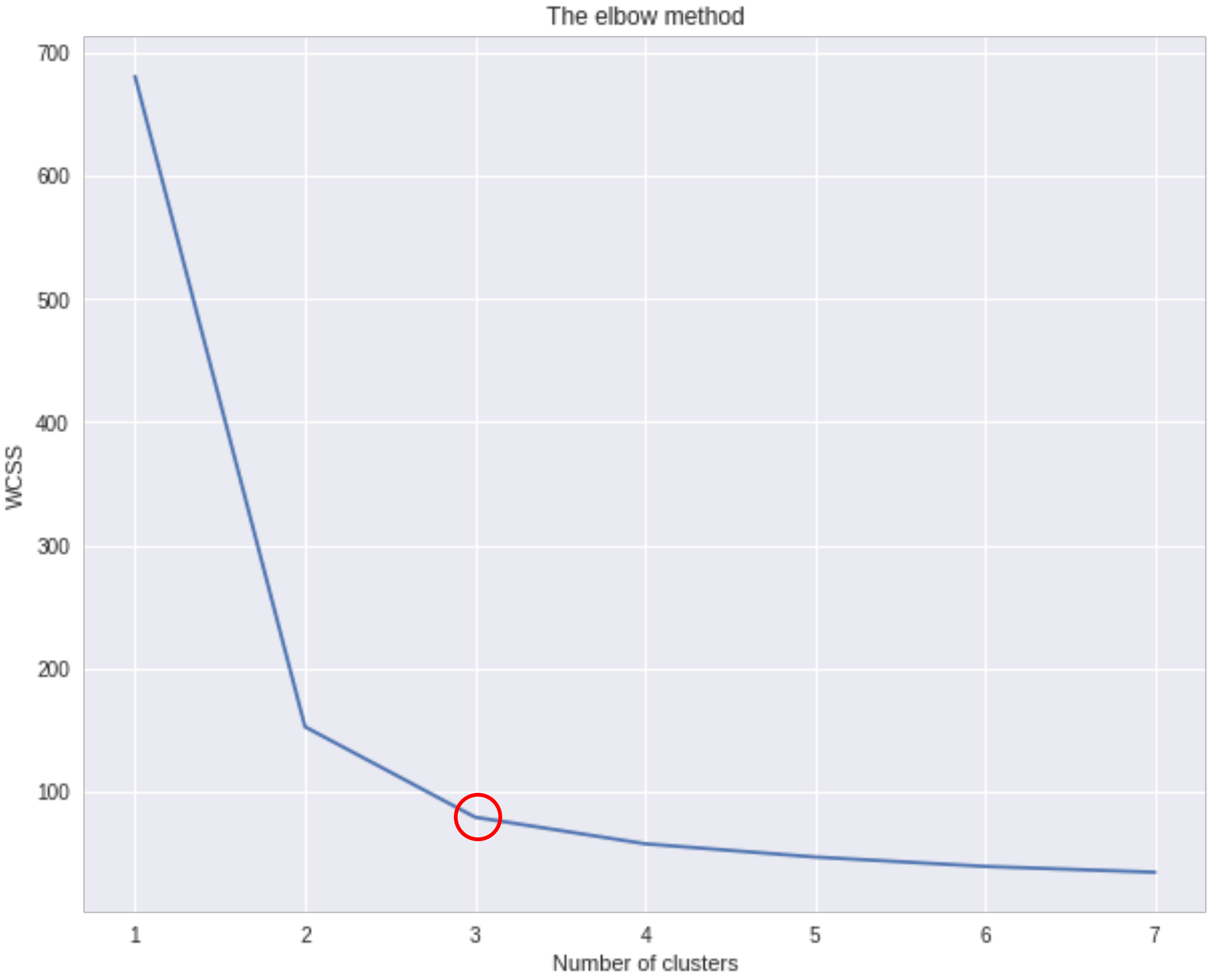
https://tinyurl.com/aintpupython101

```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd

#importing the Iris dataset with pandas
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-
learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width',
'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

array = df.values
X = array[:,0:4]
Y = array[:,4]
```

```
#Finding the optimum number of clusters for k-means
classification
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 8):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#Plotting the results onto a line graph, allowing us to
observe 'The elbow'
plt.rcParams["figure.figsize"] = (10,8)
plt.plot(range(1, 8), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```

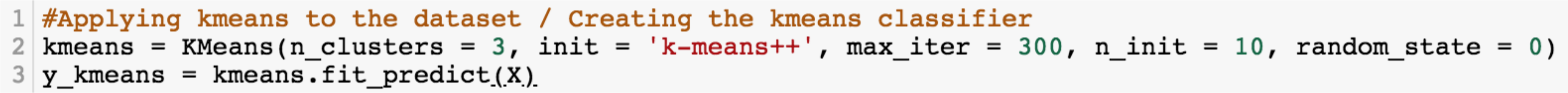
K-Means Clustering

The elbow method (*k=3*)



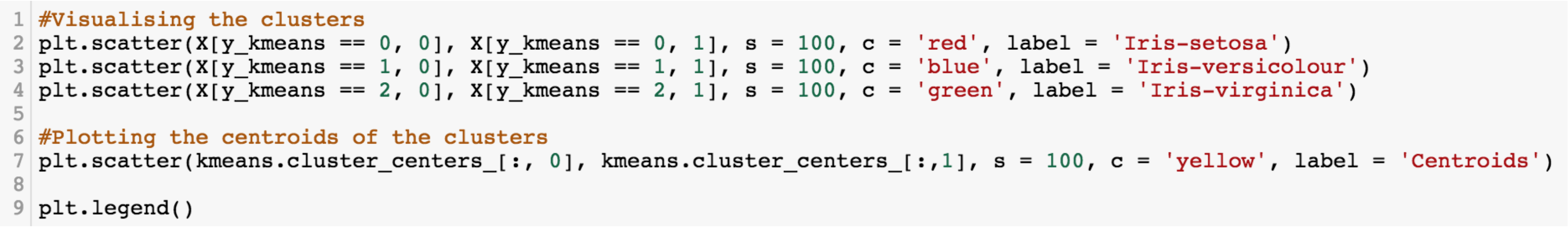
https://tinyurl.com/aintpupython101

```
kmeans = KMeans(n_clusters = 3,
init = 'k-means++', max_iter = 300,
n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
```



```
#Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
```

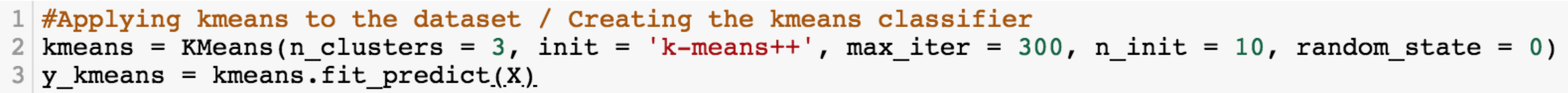
```
#Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100,
c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100,
c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100,
c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label =
'Centroids')

plt.legend()
```



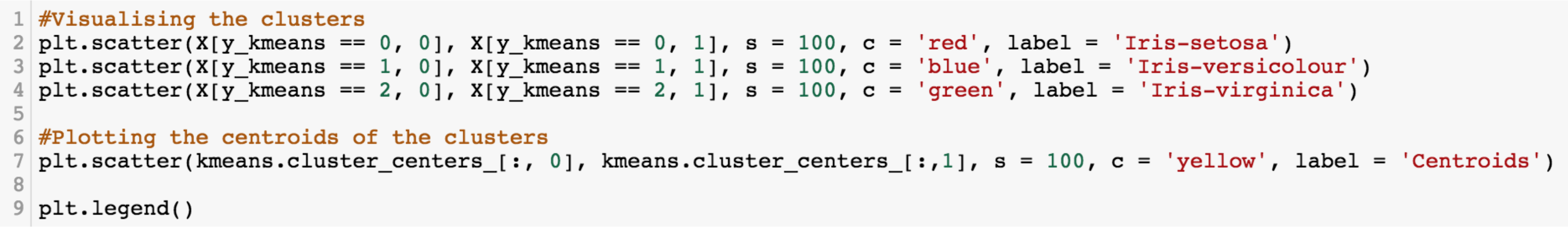
```
#Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

https://tinyurl.com/aintpupython101

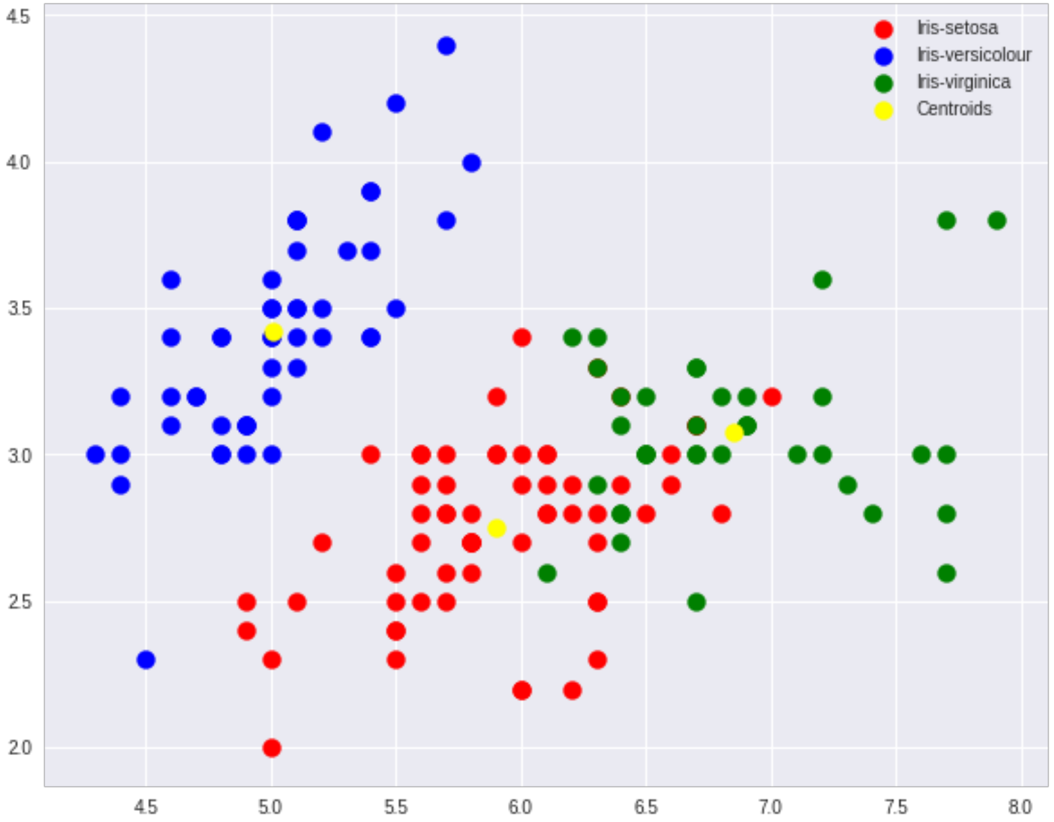
K-Means Clustering



```
#Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
```

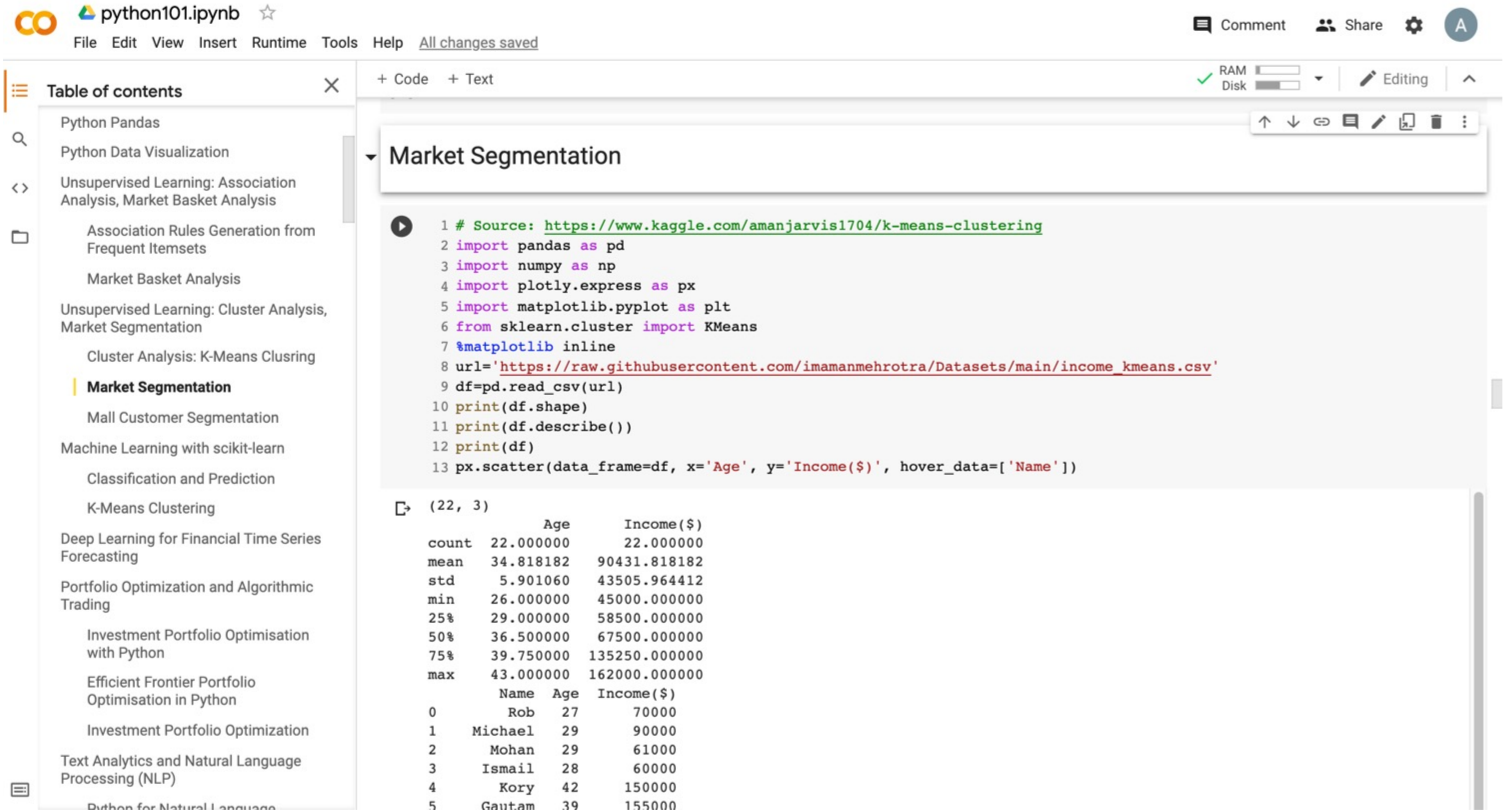


```
#Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```



https://tinyurl.com/aintpupython101

Market Segmentation



Market Segmentation

```python
# Source: https://www.kaggle.com/amanjarvis1704/k-means-clustering
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
%matplotlib inline
url='https://raw.githubusercontent.com/imamanmehrotra/Datasets/main/income_kmeans.csv'
df=pd.read_csv(url)
print(df.shape)
print(df.describe())
print(df)
px.scatter(data_frame=df, x='Age', y='Income($)', hover_data=['Name'])
```

```
(22, 3)
             Age       Income($)
count  22.000000       22.000000
mean   34.818182    90431.818182
std     5.901060    43505.964412
min    26.000000    45000.000000
25%    29.000000    58500.000000
50%    36.500000    67500.000000
75%    39.750000   135250.000000
max    43.000000   162000.000000
        Name  Age  Income($)
0        Rob   27      70000
1    Michael   29      90000
2      Mohan   29      61000
3     Ismail   28      60000
4       Kory   42     150000
5     Gautam   39     155000
```

https://tinyurl.com/aintpupython101

Mall Customer Segmentation

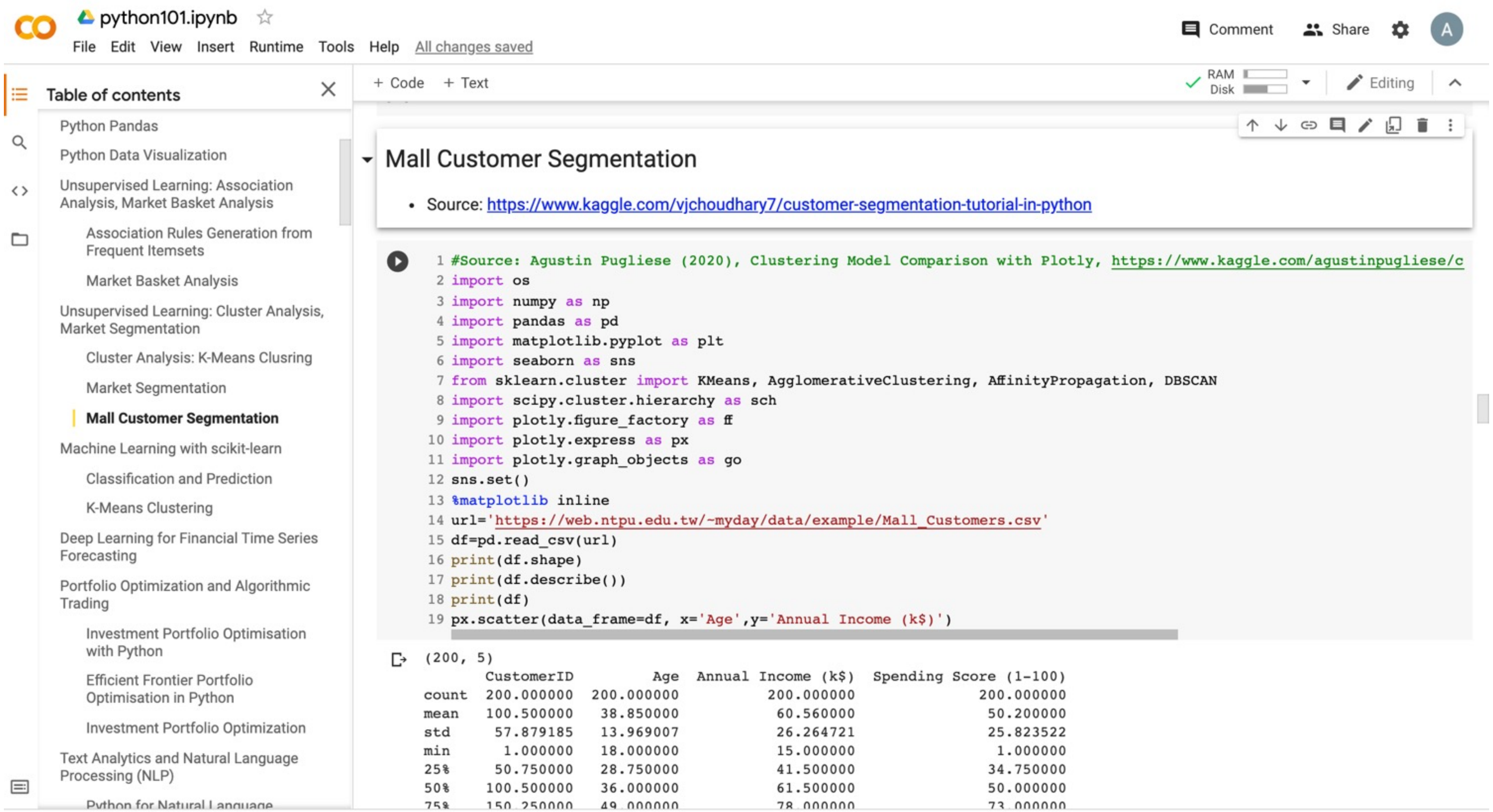


Table of contents

Mall Customer Segmentation

- Source: https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python

```python
#Source: Agustin Pugliese (2020), Clustering Model Comparison with Plotly, https://www.kaggle.com/agustinpugliese/c
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, AgglomerativeClustering, AffinityPropagation, DBSCAN
import scipy.cluster.hierarchy as sch
import plotly.figure_factory as ff
import plotly.express as px
import plotly.graph_objects as go
sns.set()
%matplotlib inline
url='https://web.ntpu.edu.tw/~myday/data/example/Mall_Customers.csv'
df=pd.read_csv(url)
print(df.shape)
print(df.describe())
print(df)
px.scatter(data_frame=df, x='Age',y='Annual Income (k$)')
```

```
(200, 5)
       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
50%    100.500000   36.000000           61.500000               50.000000
75%    150.250000   49.000000           78.000000               73.000000
```

https://tinyurl.com/aintpupython101

Mall Customer Segmentation

- Python Pandas
- Python Data Visualization
- Unsupervised Learning: Association Analysis, Market Basket Analysis
  - Association Rules Generation from Frequent Itemsets
  - Market Basket Analysis
- Unsupervised Learning: Cluster Analysis, Market Segmentation
  - Cluster Analysis: K-Means Clusring
  - Market Segmentation
  - **Mall Customer Segmentation**
- Machine Learning with scikit-learn
  - Classification and Prediction
  - K-Means Clustering
- Deep Learning for Financial Time Series Forecasting
- Portfolio Optimization and Algorithmic Trading
  - Investment Portfolio Optimisation with Python
  - Efficient Frontier Portfolio Optimisation in Python
  - Investment Portfolio Optimization
- Text Analytics and Natural Language Processing (NLP)

```
fig = px.scatter(df2, x="Age", y = "Annual Income (k$)",size='Spending Score (1-100)', color="Gender")
fig.show()
```

https://tinyurl.com/aintpupython101

Mall Customer Segmentation

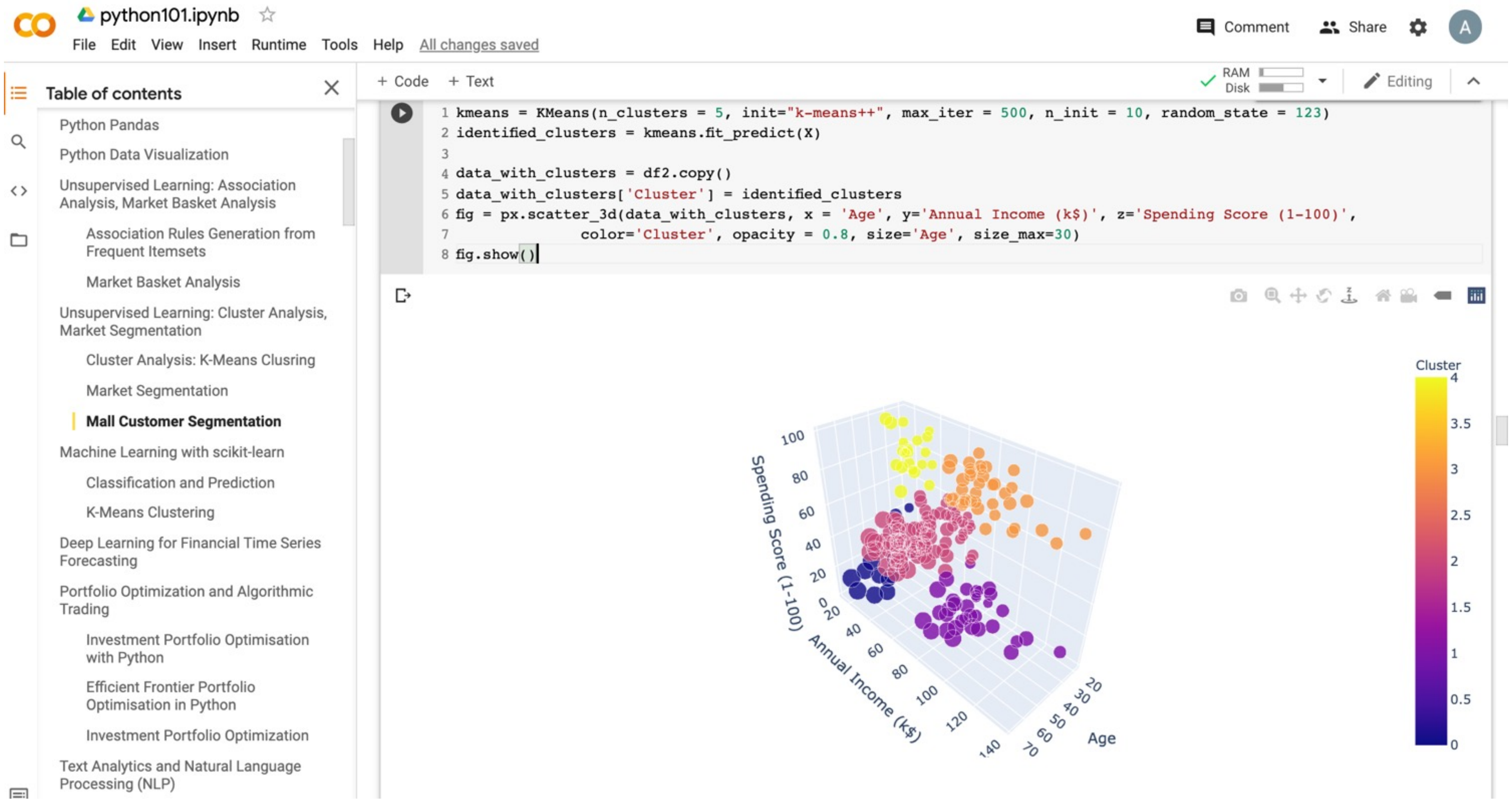


Table of contents

```python
kmeans = KMeans(n_clusters = 5, init="k-means++", max_iter = 500, n_init = 10, random_state = 123)
identified_clusters = kmeans.fit_predict(X)

data_with_clusters = df2.copy()
data_with_clusters['Cluster'] = identified_clusters
fig = px.scatter_3d(data_with_clusters, x = 'Age', y='Annual Income (k$)', z='Spending Score (1-100)',
              color='Cluster', opacity = 0.8, size='Age', size_max=30)
fig.show()
```

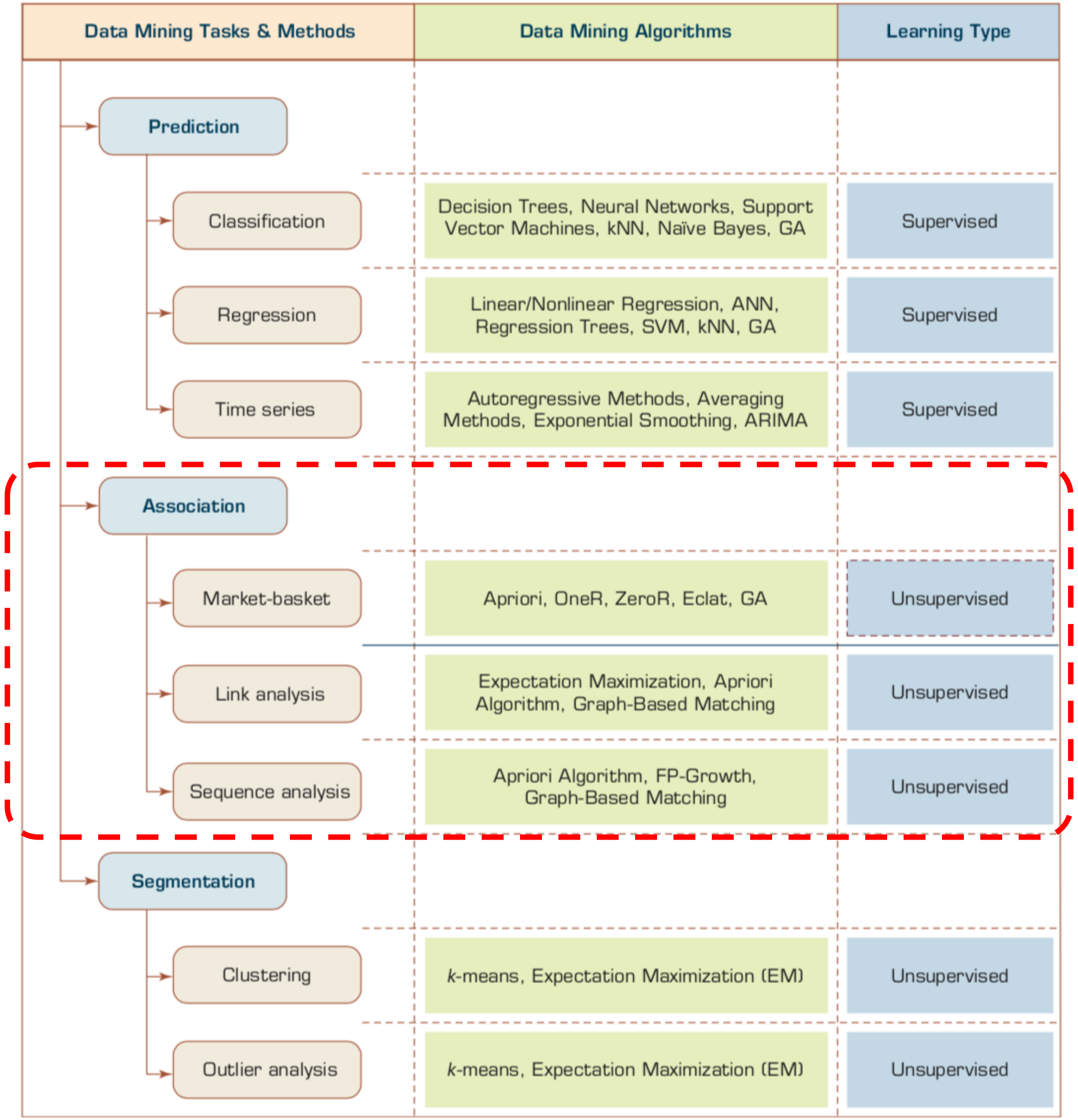
https://tinyurl.com/aintpupython101

Machine Learning: Unsupervised Learning: Association Analysis, Market Basket Analysis

Machine Learning: Data Mining Tasks & Methods

Unsupervised Learning:
Association Analysis
Market-basket

Association



Data Mining Tasks & Methods	Data Mining Algorithms	Learning Type
Prediction		
Classification	Decision Trees, Neural Networks, Support Vector Machines, kNN, Naïve Bayes, GA	Supervised
Regression	Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA	Supervised
Time series	Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA	Supervised
Association		
Market-basket	Apriori, OneR, ZeroR, Eclat, GA	Unsupervised
Link analysis	Expectation Maximization, Apriori Algorithm, Graph-Based Matching	Unsupervised
Sequence analysis	Apriori Algorithm, FP-Growth, Graph-Based Matching	Unsupervised
Segmentation		
Clustering	*k*-means, Expectation Maximization (EM)	Unsupervised
Outlier analysis	*k*-means, Expectation Maximization (EM)	Unsupervised

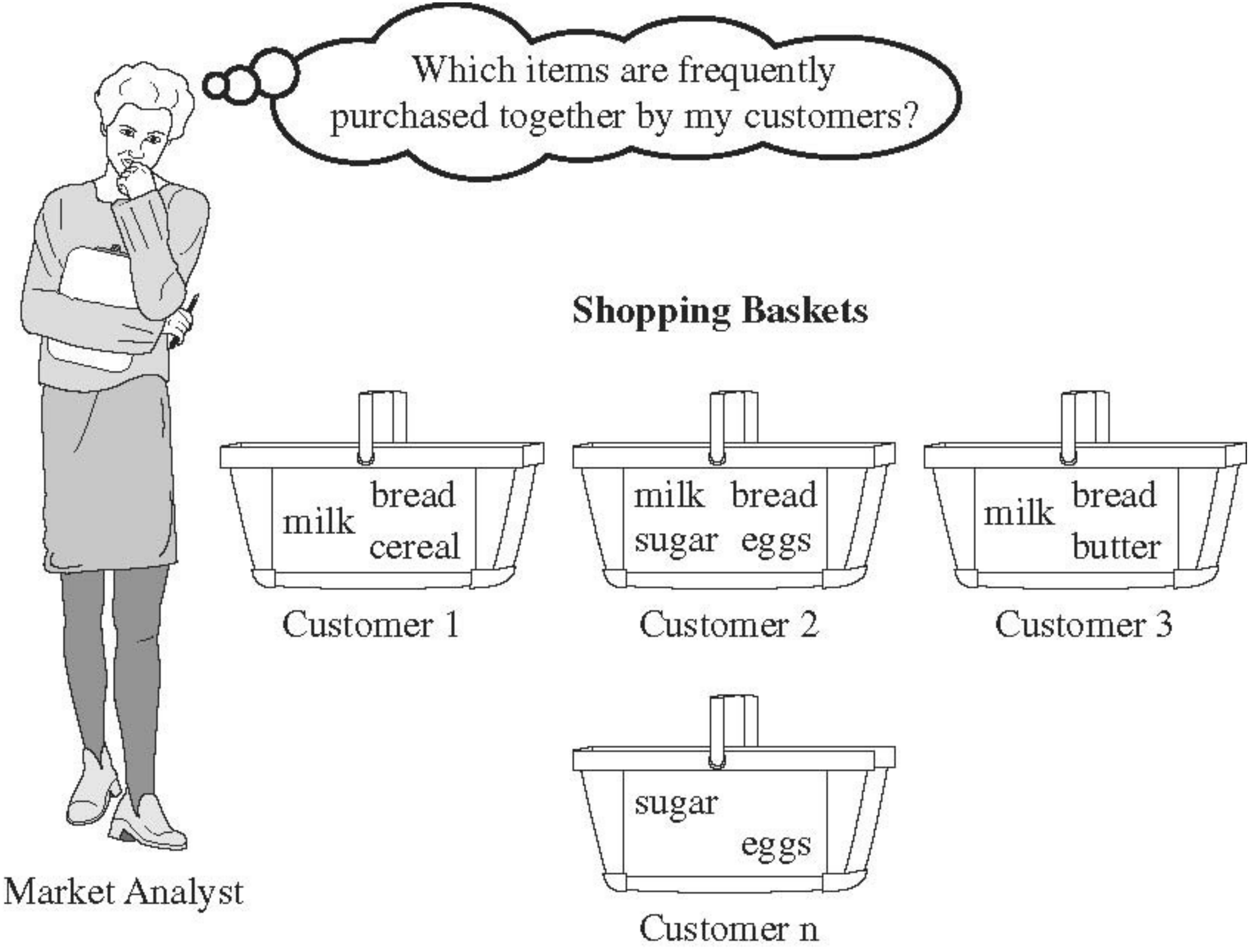
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Transaction Database

Transaction ID	Items bought
T01	A, B, D
T02	A, C, D
T03	B, C, D, E
T04	A, B, D
T05	A, B, C, E
T06	A, C
T07	B, C, D
T08	B, D
T09	A, C, E
T10	B, D

Association Analysis

Market Basket Analysis

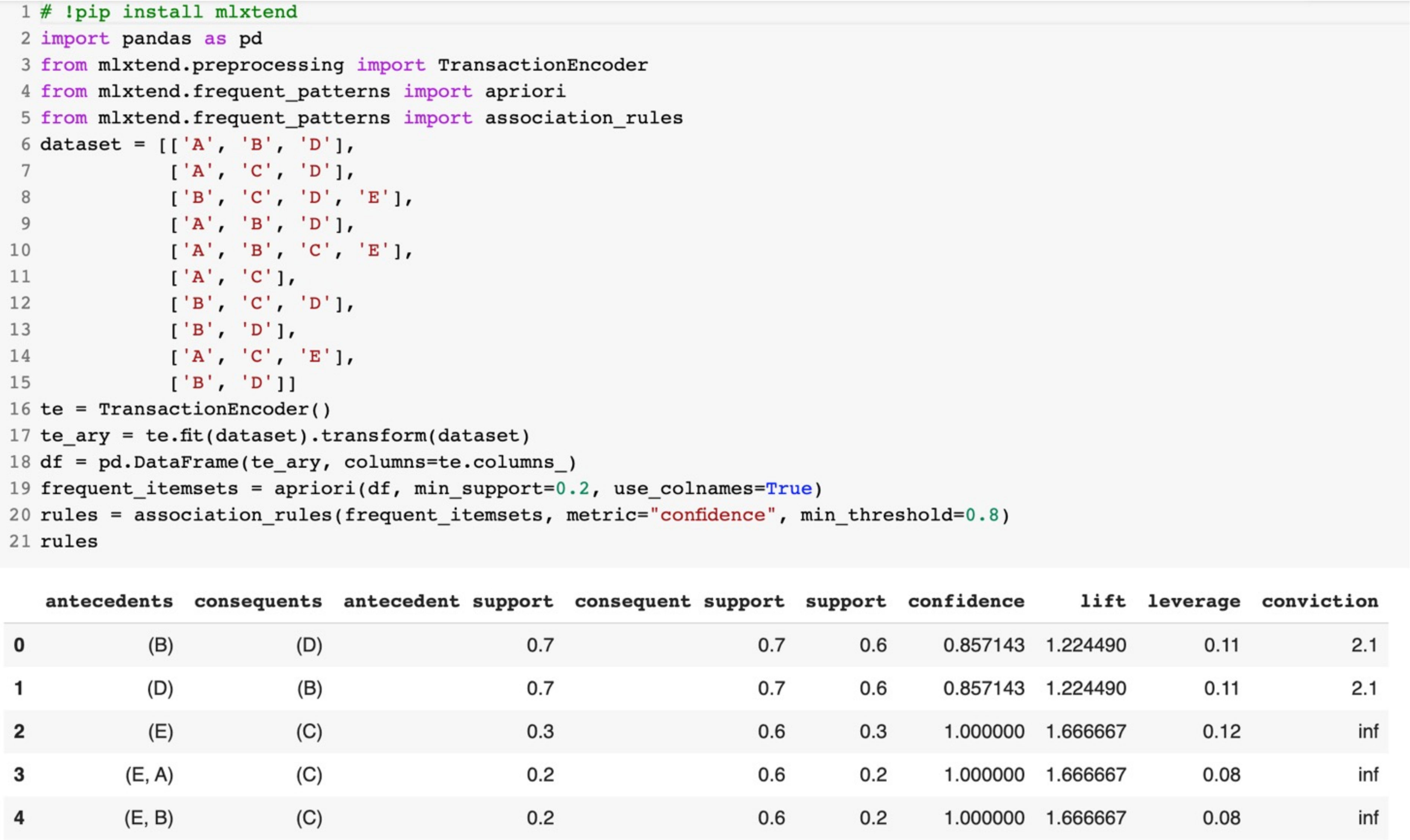


Source: Han & Kamber (2006)

Python mlxtend Association Rules

```
# !pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

dataset = [['A', 'B', 'D’],
           ['A', 'C', 'D'],
           ['B', 'C', 'D', 'E'],
           ['A', 'B', 'D’],
           ['A', 'B', 'C', 'E’],
           ['A', 'C’],
           ['B', 'C', 'D’],
           ['B', 'D’],
           ['A', 'C', 'E'],
           ['B', 'D']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
```

Python mlxtend Association Rules



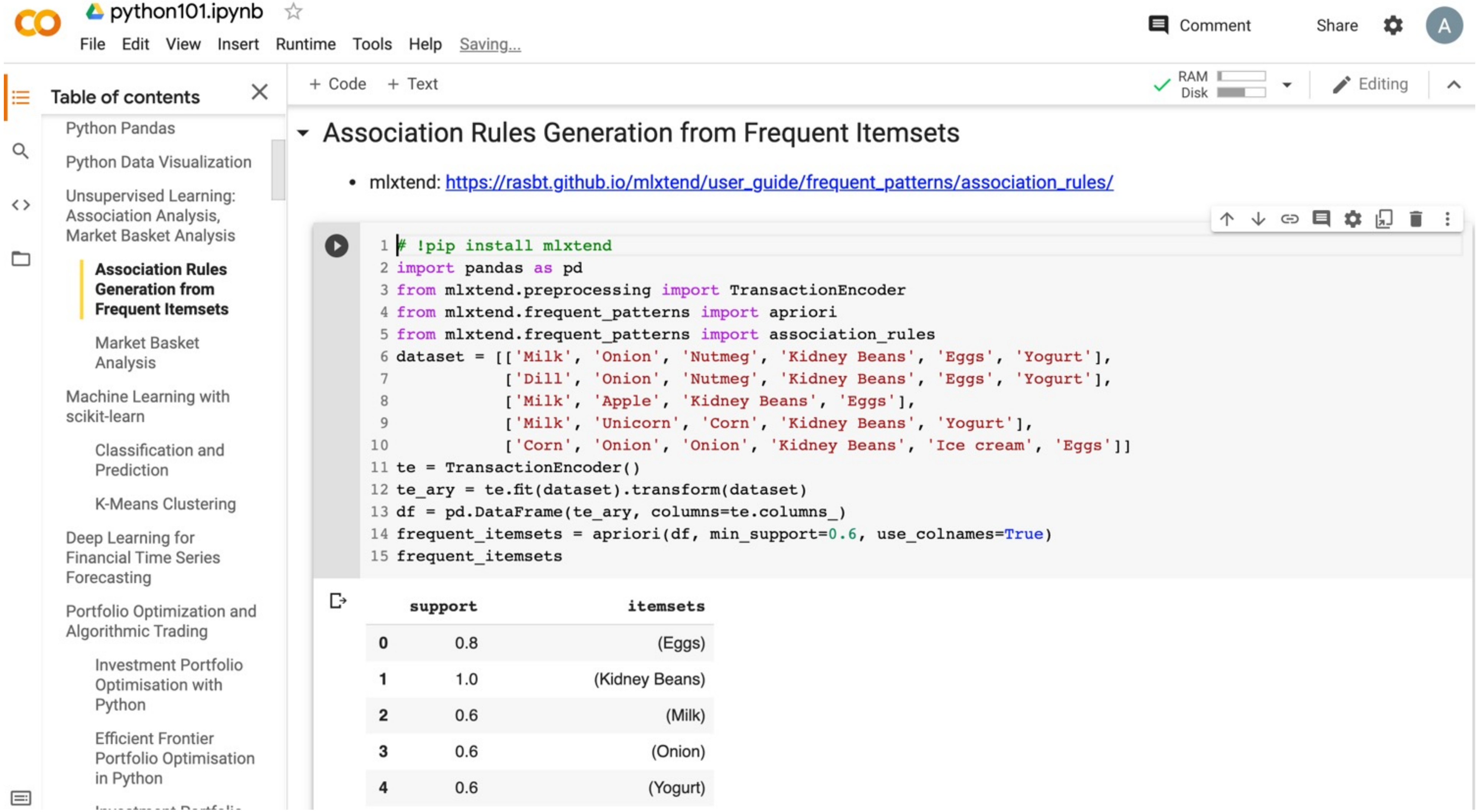
```python
# !pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
dataset = [['A', 'B', 'D'],
           ['A', 'C', 'D'],
           ['B', 'C', 'D', 'E'],
           ['A', 'B', 'D'],
           ['A', 'B', 'C', 'E'],
           ['A', 'C'],
           ['B', 'C', 'D'],
           ['B', 'D'],
           ['A', 'C', 'E'],
           ['B', 'D']]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(B)	(D)	0.7	0.7	0.6	0.857143	1.224490	0.11	2.1
1	(D)	(B)	0.7	0.7	0.6	0.857143	1.224490	0.11	2.1
2	(E)	(C)	0.3	0.6	0.3	1.000000	1.666667	0.12	inf
3	(E, A)	(C)	0.2	0.6	0.2	1.000000	1.666667	0.08	inf
4	(E, B)	(C)	0.2	0.6	0.2	1.000000	1.666667	0.08	inf

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



Association Rules Generation from Frequent Itemsets

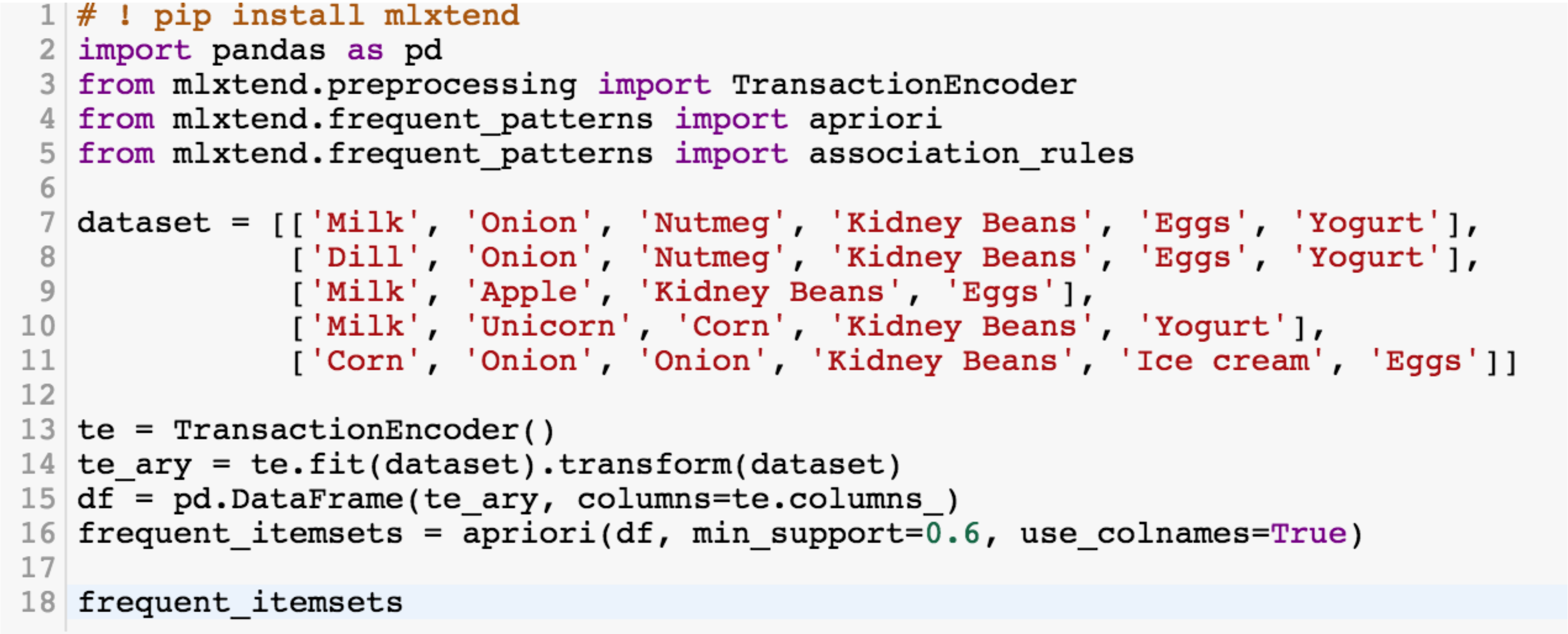
- mlxtend: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

```
# !pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
frequent_itemsets
```

	support	itemsets
0	0.8	(Eggs)
1	1.0	(Kidney Beans)
2	0.6	(Milk)
3	0.6	(Onion)
4	0.6	(Yogurt)

https://tinyurl.com/aintpupython101

```
# ! pip install mlxtend
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

frequent_itemsets = apriori(df, min_support=0.6,
use_colnames=True)
```

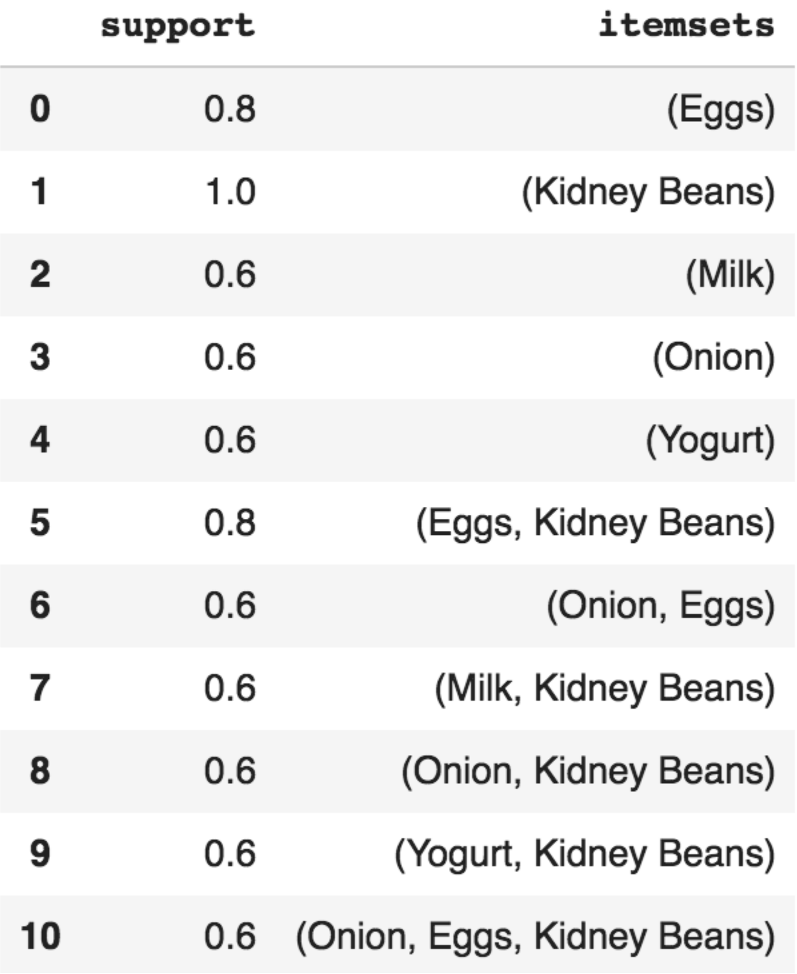


```
# ! pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

frequent_itemsets
```

https://tinyurl.com/aintpupython101

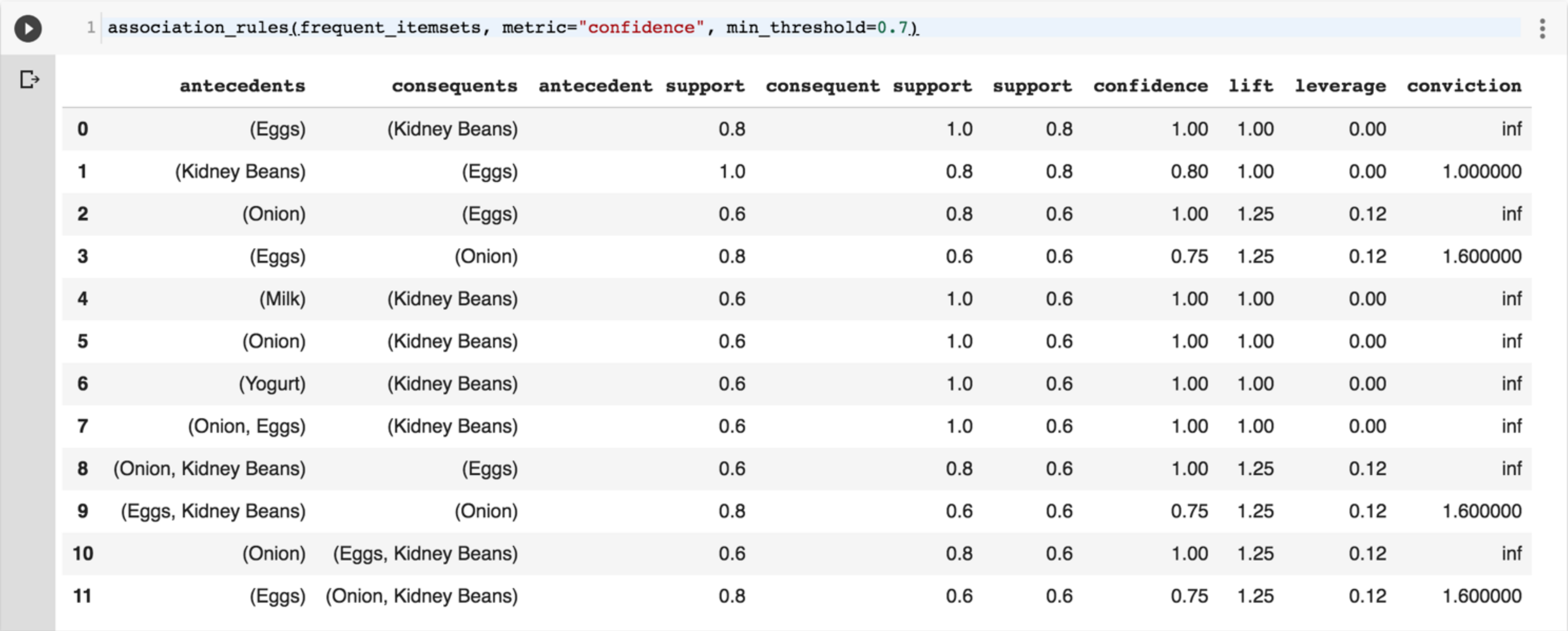
```
# ! pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.6,
use_colnames=True)

frequent_itemsets
```

```
frequent_itemsets = apriori(df,
min_support=0.6,
use_colnames=True)
```



	support	itemsets
0	0.8	(Eggs)
1	1.0	(Kidney Beans)
2	0.6	(Milk)
3	0.6	(Onion)
4	0.6	(Yogurt)
5	0.8	(Eggs, Kidney Beans)
6	0.6	(Onion, Eggs)
7	0.6	(Milk, Kidney Beans)
8	0.6	(Onion, Kidney Beans)
9	0.6	(Yogurt, Kidney Beans)
10	0.6	(Onion, Eggs, Kidney Beans)

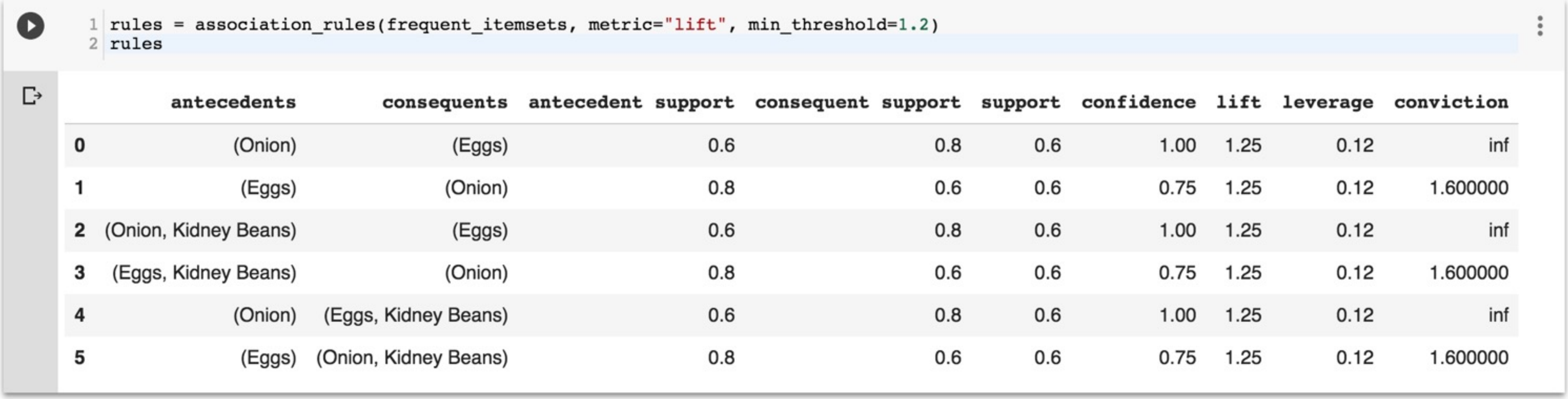
```
association_rules(frequent_itemsets,
metric="confidence", min_threshold=0.7)
```



```
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.00	1.00	0.00	inf
1	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.80	1.00	0.00	1.000000
2	(Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf
3	(Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000
4	(Milk)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
5	(Onion)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
6	(Yogurt)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
7	(Onion, Eggs)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	inf
8	(Onion, Kidney Beans)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf
9	(Eggs, Kidney Beans)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000
10	(Onion)	(Eggs, Kidney Beans)	0.6	0.8	0.6	1.00	1.25	0.12	inf
11	(Eggs)	(Onion, Kidney Beans)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000

```
rules =
association_rules(frequent_itemsets,
metric="lift", min_threshold=1.2)
rules
```

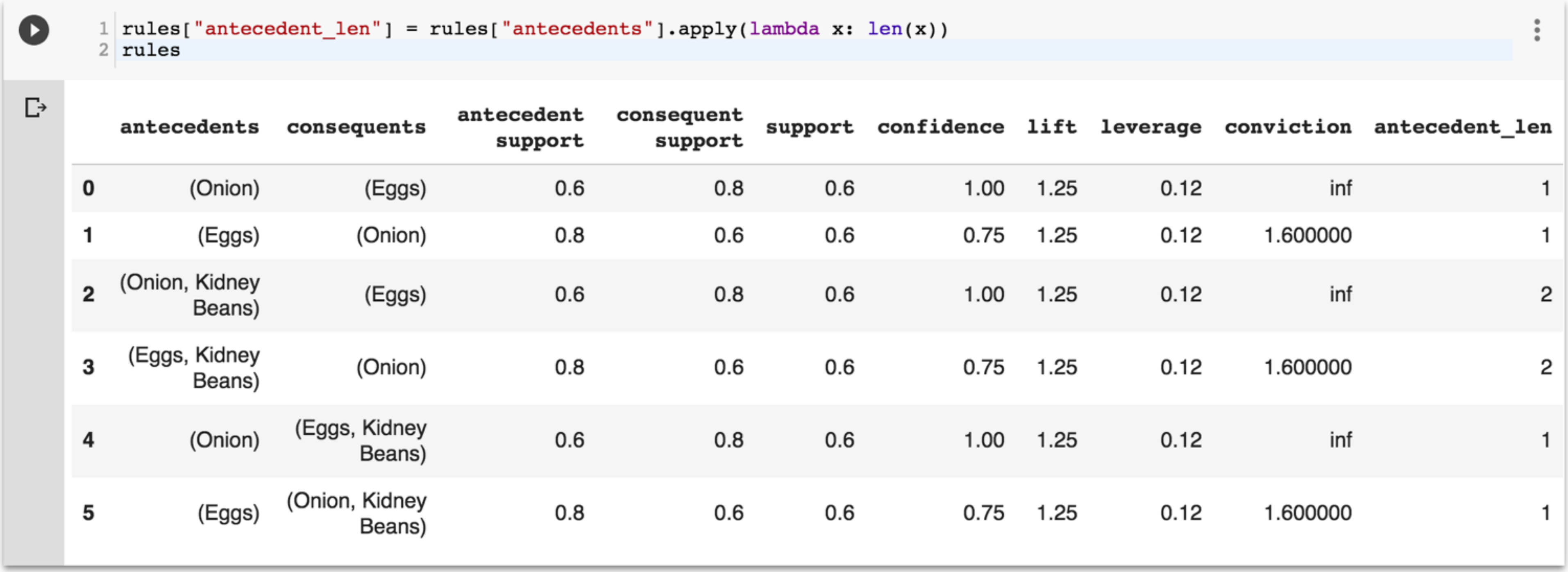


```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
rules
```

```
           antecedents          consequents  antecedent support  consequent support  support  confidence  lift  leverage  conviction
0              (Onion)               (Eggs)                 0.6                 0.8      0.6        1.00  1.25      0.12         inf
1               (Eggs)              (Onion)                 0.8                 0.6      0.6        0.75  1.25      0.12    1.600000
2  (Onion, Kidney Beans)               (Eggs)                 0.6                 0.8      0.6        1.00  1.25      0.12         inf
3   (Eggs, Kidney Beans)              (Onion)                 0.8                 0.6      0.6        0.75  1.25      0.12    1.600000
4              (Onion)  (Eggs, Kidney Beans)                 0.6                 0.8      0.6        1.00  1.25      0.12         inf
5               (Eggs)  (Onion, Kidney Beans)                 0.8                 0.6      0.6        0.75  1.25      0.12    1.600000
```

https://tinyurl.com/aintpupython101

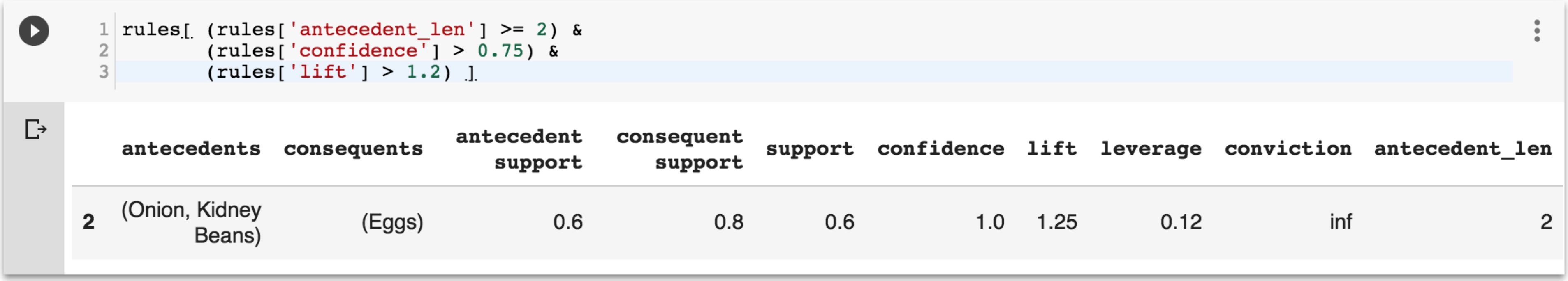
```
rules["antecedent_len"] =
rules["antecedents"].apply(lambda x: len(x))
rules
```



```
rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
0	(Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf	1
1	(Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000	1
2	(Onion, Kidney Beans)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	inf	2
3	(Eggs, Kidney Beans)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000	2
4	(Onion)	(Eggs, Kidney Beans)	0.6	0.8	0.6	1.00	1.25	0.12	inf	1
5	(Eggs)	(Onion, Kidney Beans)	0.8	0.6	0.6	0.75	1.25	0.12	1.600000	1

```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

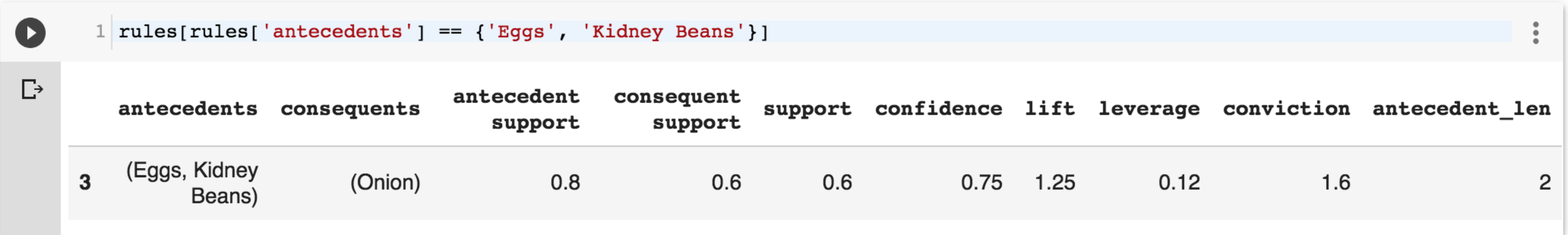


```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
2	(Onion, Kidney Beans)	(Eggs)	0.6	0.8	0.6	1.0	1.25	0.12	inf	2

https://tinyurl.com/aintpupython101

```
rules[rules['antecedents'] ==
{'Eggs', 'Kidney Beans’}]
```



```
rules[rules['antecedents'] == {'Eggs', 'Kidney Beans'}]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
3	(Eggs, Kidney Beans)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	1.6	2

The
Theory of
Learning

The Theory of Learning

- Computational Learning Theory
  - Probably Approximately Correct (PAC) Learning
  - Vapnik-Chervonenkis (VC) Dimension
  - Bias-Variance Trade-off
- Overfitting and Underfitting
  - Avoid overfitting: Regularization, Cross-Validation

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

What is Learning in Machine Learning?

- **Learning** involves **finding patterns** from **data** to make **predictions**.
- Performance is measured through **generalization** to unseen data.
- Three paradigms of learning:
  - **Supervised Learning** (e.g., classification, regression)
  - **Unsupervised Learning** (e.g., clustering, dimensionality reduction)
  - **Reinforcement Learning** (e.g., learning through **rewards**)

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

The Theory of Learning

- How can we be sure that our learned hypothesis will predict well for previously unseen inputs?
  - How do we know that the hypothesis h is close to the target function f if we don't know what is?
- How many examples do we need to get a good h?
- What hypothesis space should we use?
- If the hypothesis space is very complex, can we even find the best h or do we have to settle for a local maximum?
- How complex should h be?
- How do we avoid overfitting?

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Computational Learning Theory

- Intersection of AI, statistics, and theoretical computer science.
- Any hypothesis that is seriously wrong will almost certainly be "found out" with high probability after a small number of examples.

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Probably Approximately Correct (PAC) Learning

- **Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong.**
- **PAC learning algorithm:**
  - **Any learning algorithm that returns hypotheses that are probably approximately correct.**

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Probably Approximately Correct (PAC) Learning

- **PAC Learning** provides a way to understand **generalization**
- Key Components:
  - Error threshold (ε)
  - Confidence level (1 - δ)
- Example: Linear models are PAC-learnable with enough data

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Linear function

$$y = f(x)$$

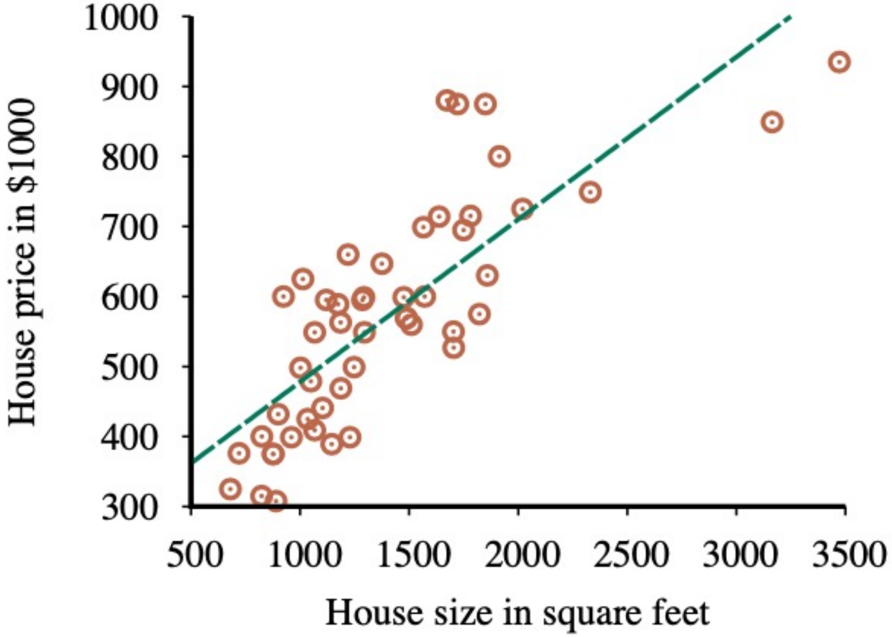
$$y = w_1 x + w_0$$

$$h_w(x) = w_1 x + w_0$$

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

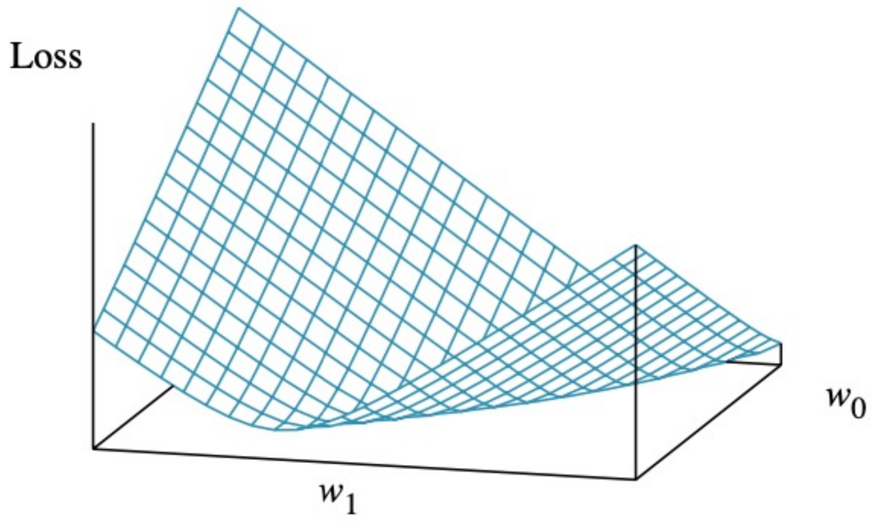
Linear Regression Weight Space

$$h_w(x) = w_1 x + w_0$$



$y = 0.232\, x + 246$

$w^* = \text{argmin}_w \, Loss(h_w)$



Loss function for Weights (w_1, w_0)

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Vapnik-Chervonenkis (VC) Dimension

- VC Dimension measures a model's capacity to fit various patterns
- Higher VC Dimension
  Higher model complexity
- Balancing model complexity helps improve generalization

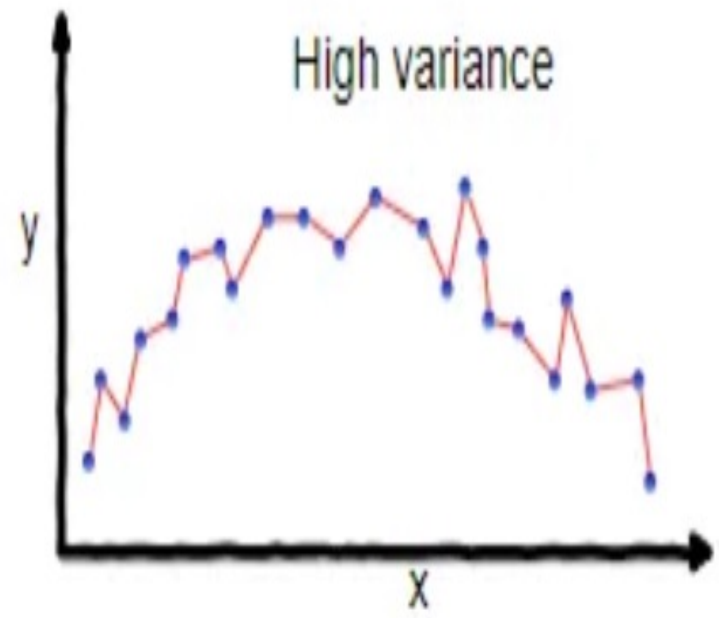
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Bias-Variance Trade-off

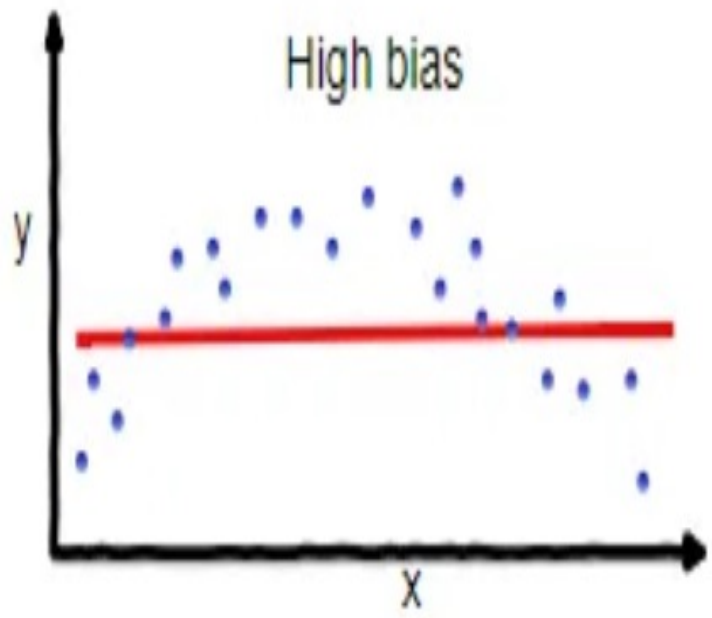
- Bias: Error due to simple models (e.g., linear models)
- Variance: Error from models too sensitive to noise (e.g., deep trees)
- Bias-Variance Trade-off:
  - Low bias-high variance: Risk of overfitting
  - High bias-low variance: Risk of underfitting
- Selecting the right model balances bias and variance

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

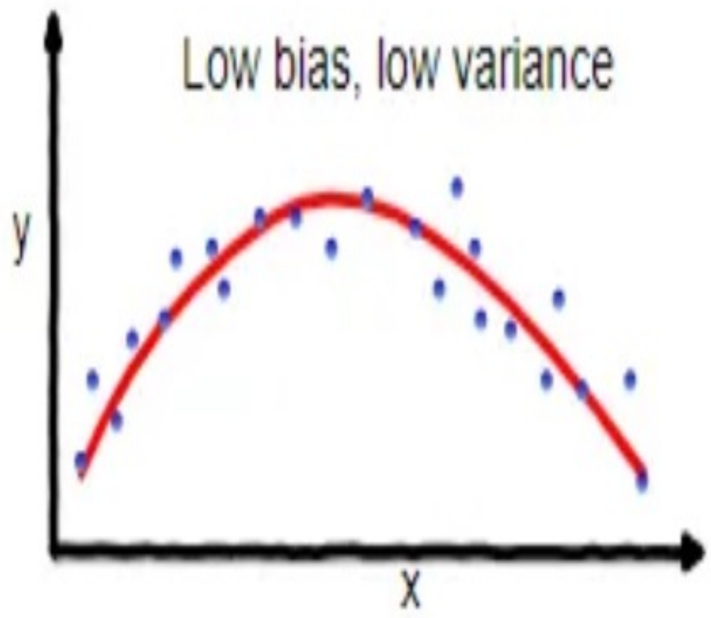
Overfitting vs. Underfitting



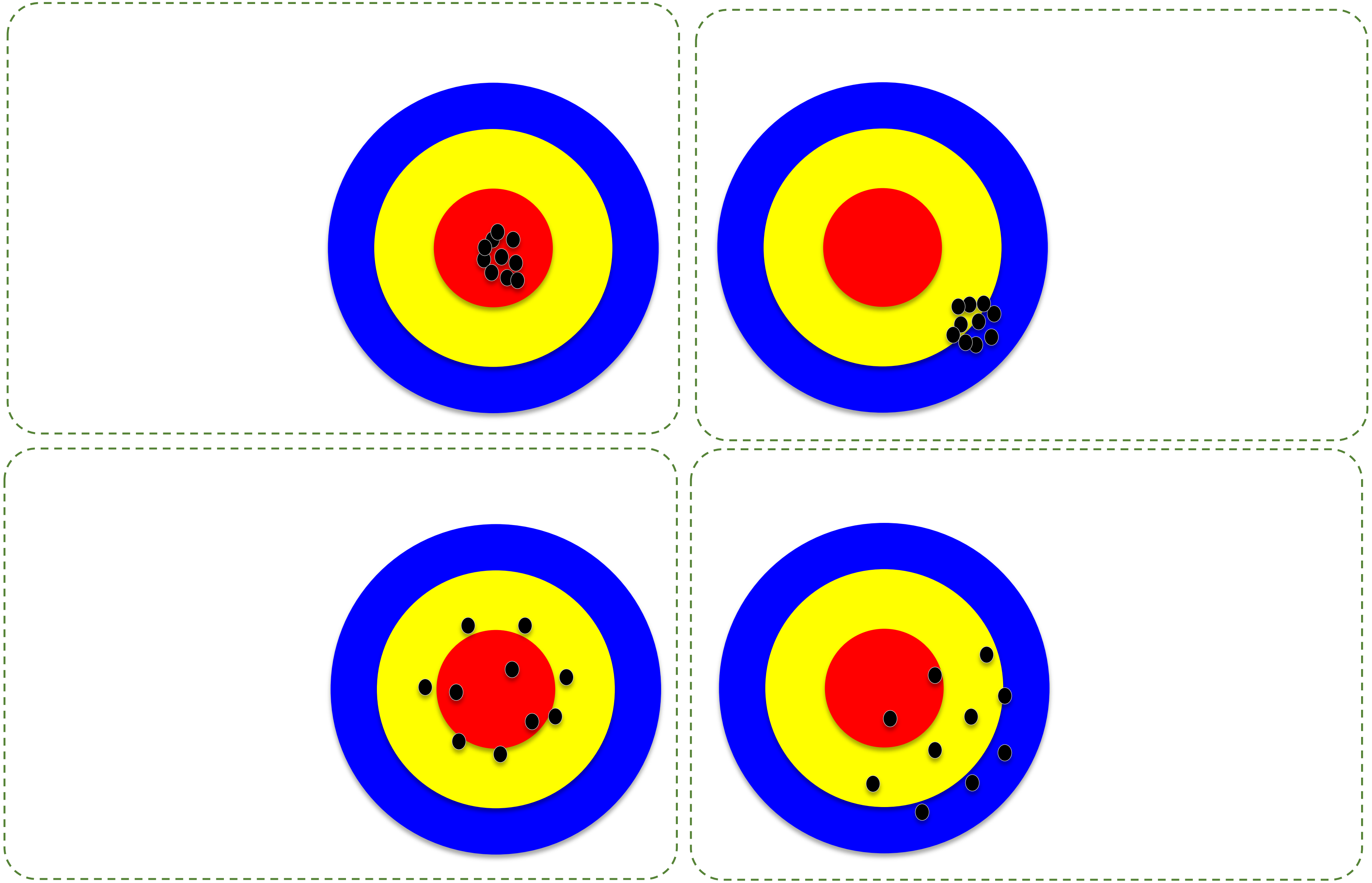
Overfitting



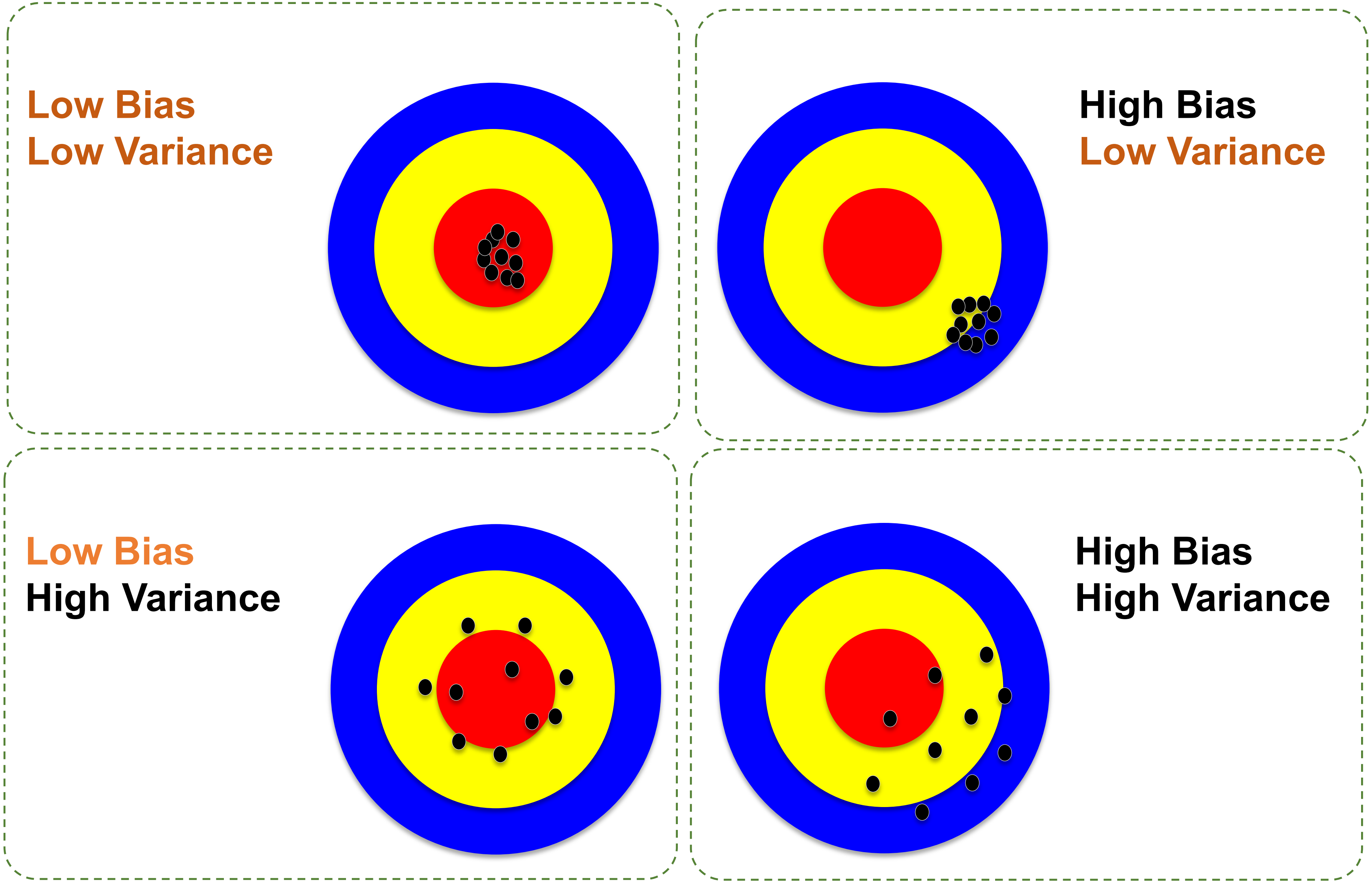
Underfitting



Good Balance



Bias vs. Variance



Low Variance

Low Bias
Low Variance

High Bias
Low Variance

High Variance

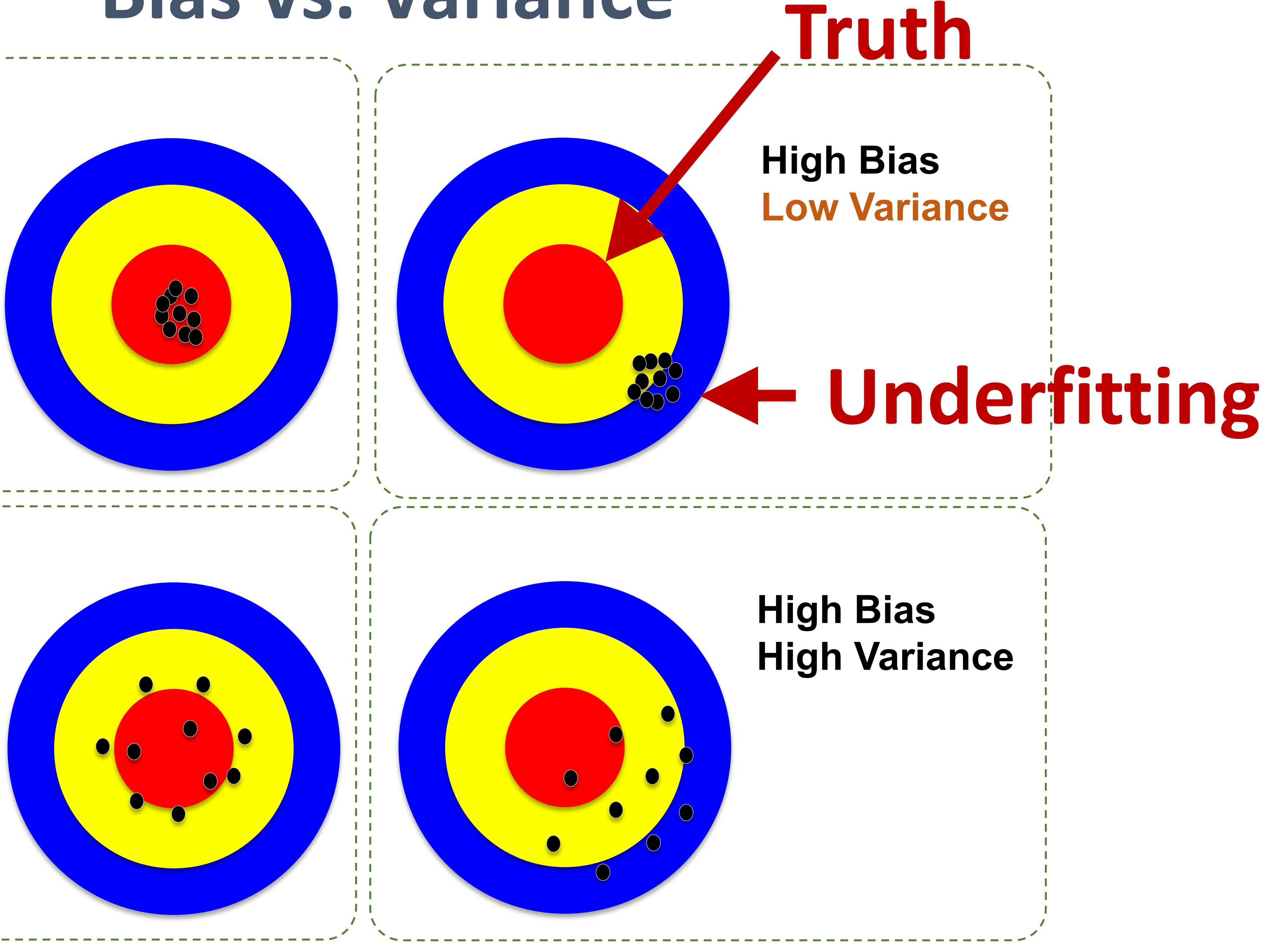
Low Bias
High Variance

High Bias
High Variance

Low Bias

High Bias

Bias vs. Variance



Truth

Low Variance

Low Bias
Low Variance

High Bias
Low Variance

Underfitting

High Variance

Low Bias
High Variance

High Bias
High Variance

Overfitting

Low Bias

High Bias

Bias vs. Variance

Low Variance

A

Low Bias
Low Variance

High Accuracy
High Precision

B

High Bias
Low Variance

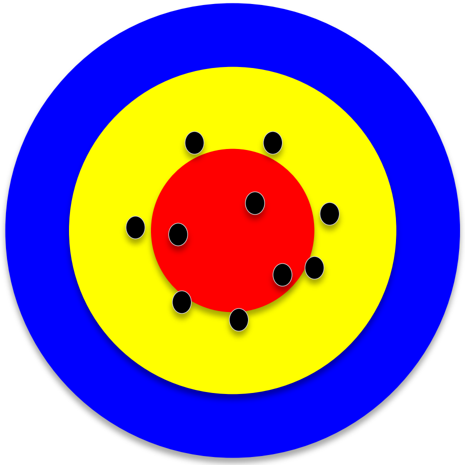
Low Accuracy
High Precision

High Variance

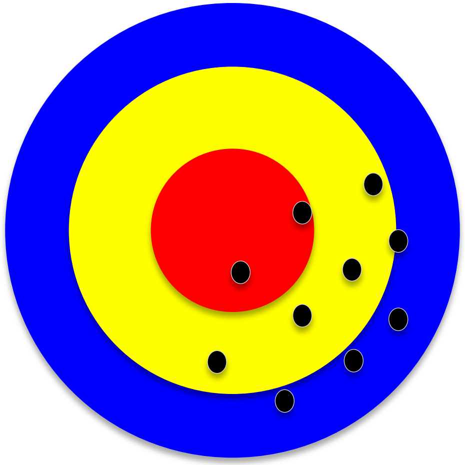
C

Low Bias
High Variance

High Accuracy
Low Precision



D



High Bias
High Variance

Low Accuracy
Low Precision

Low Bias

High Bias

Accuracy vs. Precision

A

High Accuracy
High Precision

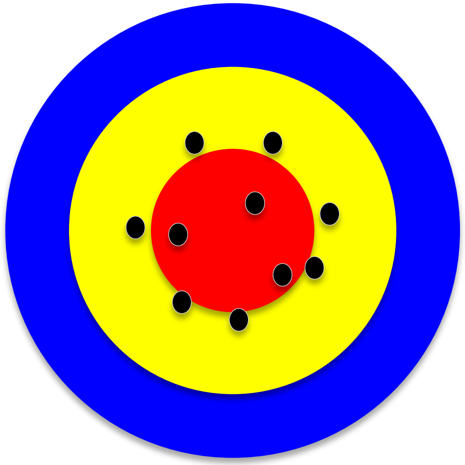
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

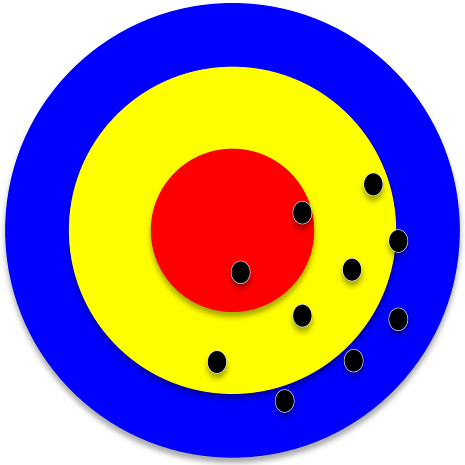
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Accuracy (Validity) vs. Precision (Reliability)

A

High Accuracy
High Precision

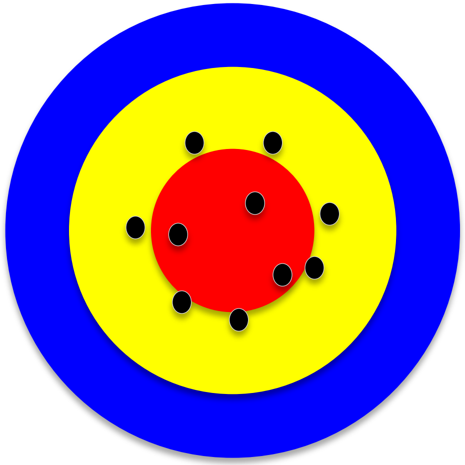
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

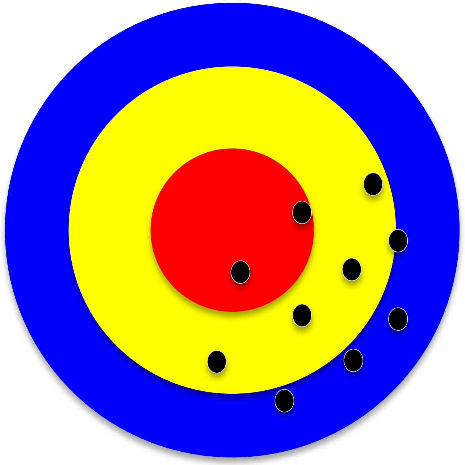
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Learning Theory

- **PAC Learning**:
  - Focuses on **generalization**
- **VC Dimension**:
  - Measures model capacity and impacts **generalization**
- **Bias-Variance Trade-off**:
  - Helps in **model selection** and **tuning**

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning

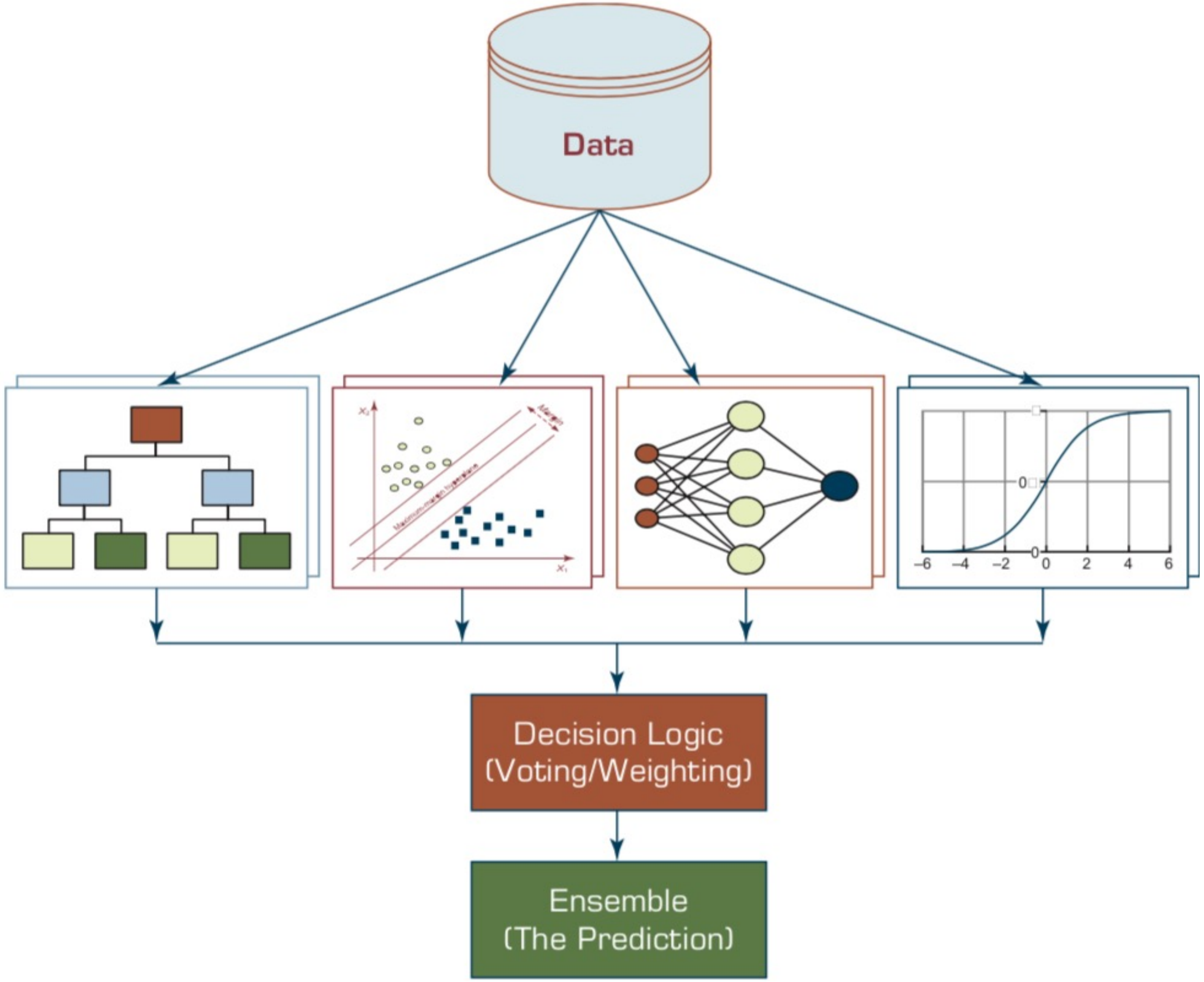
Ensemble Learning

- **Select a collection**, or **ensemble**, of hypotheses, $h_1, h_2, \ldots, h_n$, and **combine** their predictions by **averaging, voting**, or by another level of machine learning.

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Models

Heterogeneous Ensemble

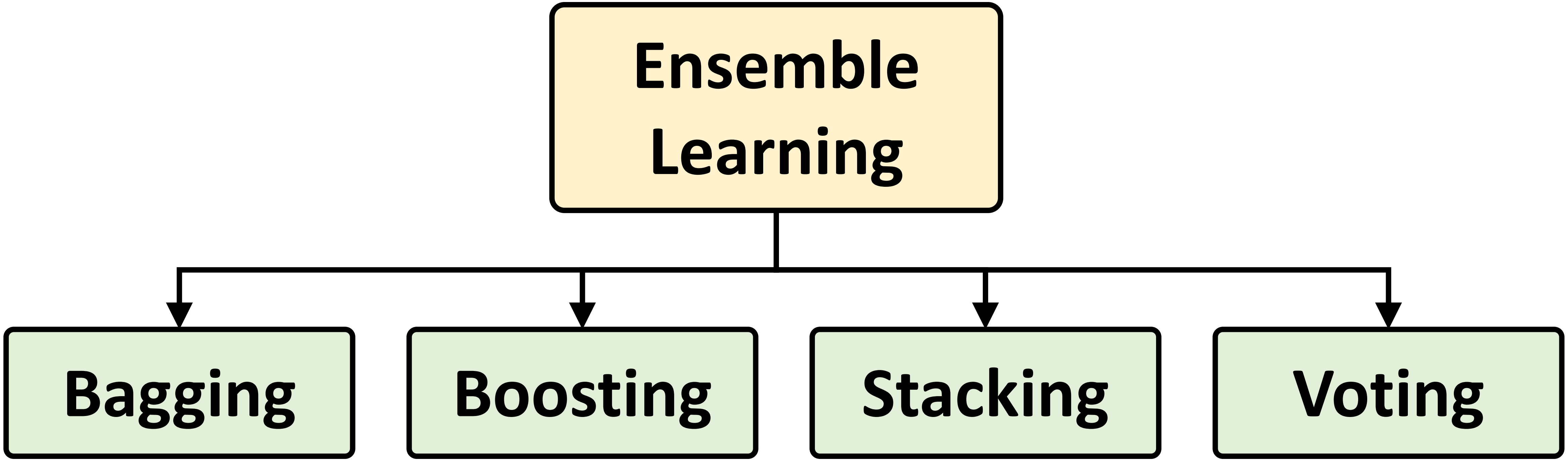


Decision Logic (Voting/Weighting)

Ensemble (The Prediction)

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Ensemble Learning:
Bagging, Boosting, Stacking, Voting



Ensemble Learning

Bagging

Boosting

Stacking

Voting

Ensemble Learning

- Base model
  - individual hypotheses
  - $h_1, h_2, \ldots, h_n$
- Ensemble model
  - hypotheses combination

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Why Ensemble Learning

- Reduce bias
- Reduce variance

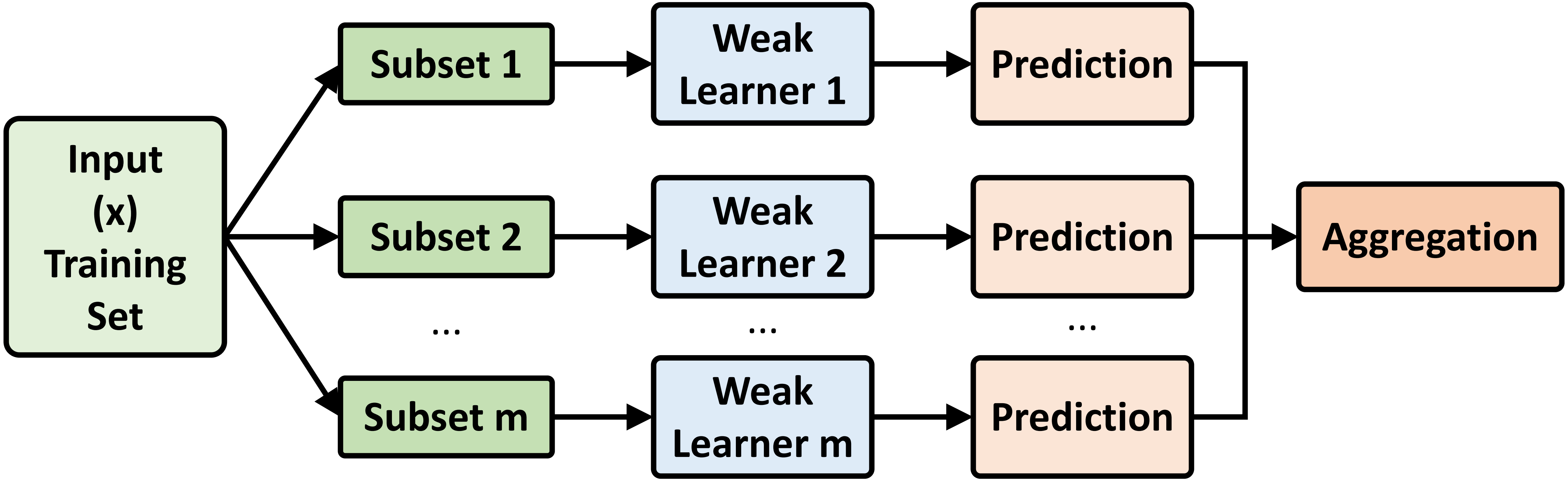
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning

- Bagging
  - Random Forests (RF)
- Boosting
  - Gradient Boosting, XGBoost, LightGBM, CatBoost
- Stacking
- Online learning

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

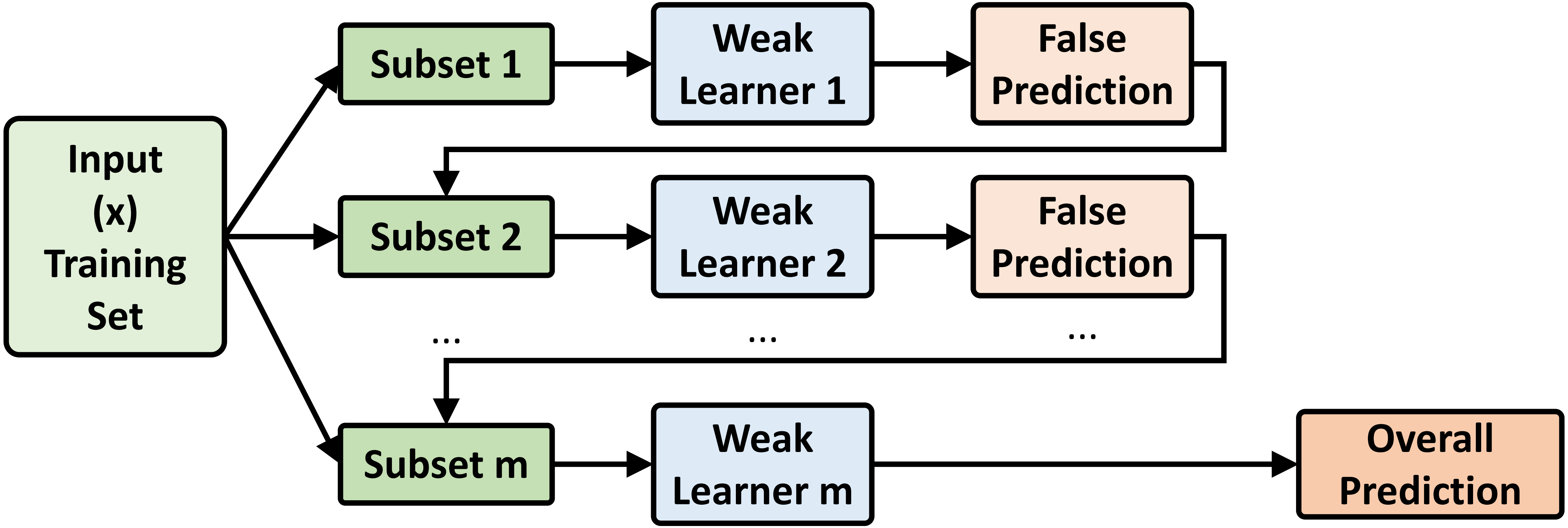
Ensemble Learning:
Bagging (Bootstrap Aggregation)



Bootstrap
Samples

Parallel

Ensemble Learning: Boosting



Input (x) Training Set

Subset 1

Weak Learner 1

False Prediction

Subset 2

Weak Learner 2

False Prediction

...

...

...

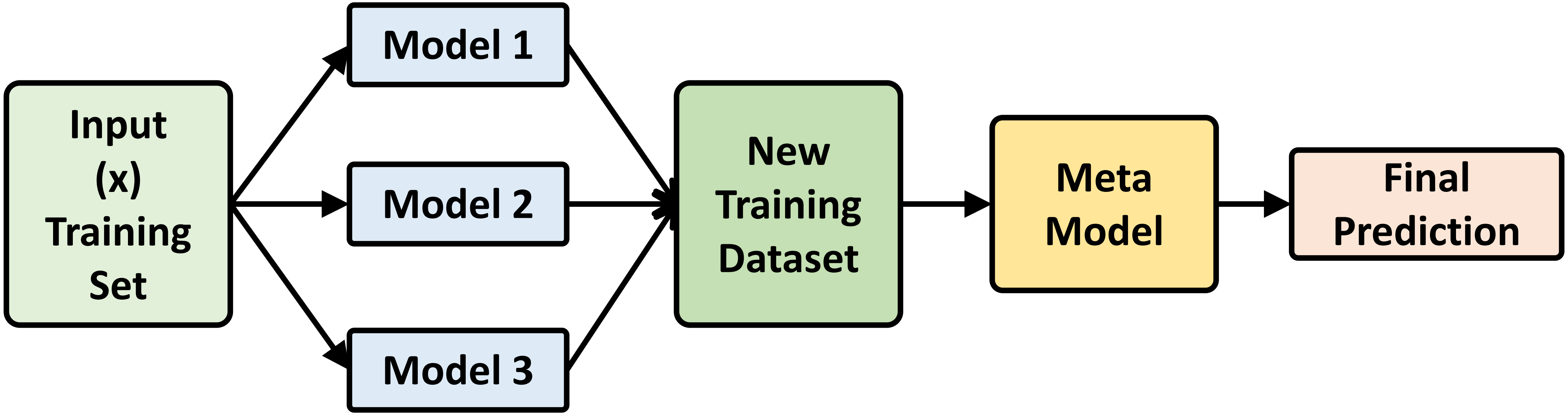
Subset m

Weak Learner m

Overall Prediction

Sequential

Ensemble Learning: Stacking



Input (x) Training Set

Model 1

Model 2

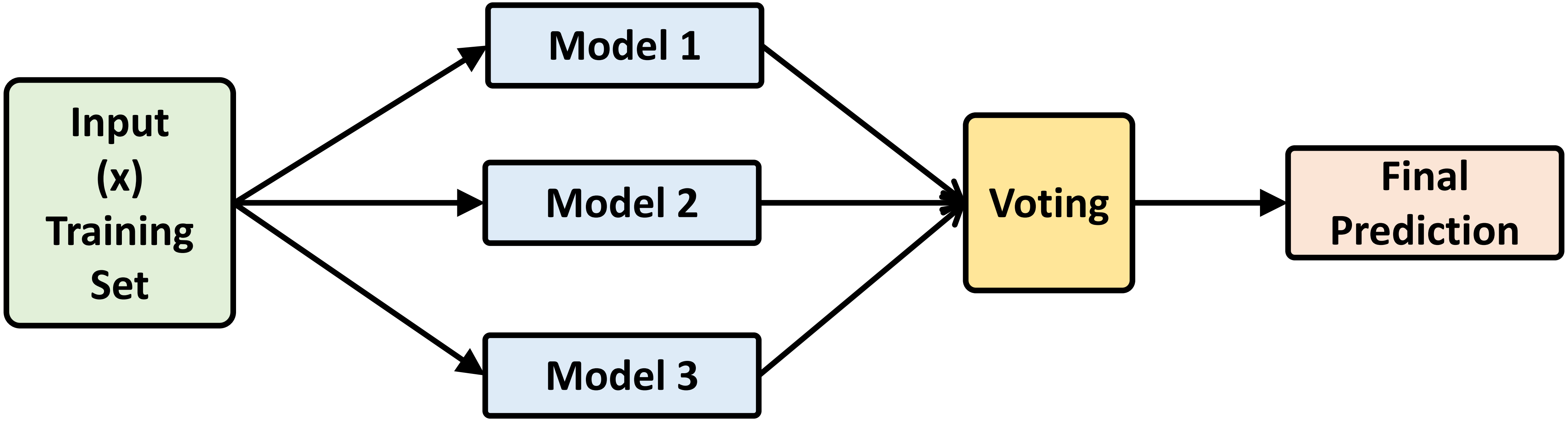
Model 3

New Training Dataset

Meta Model

Final Prediction

Ensemble Learning: Voting



Input (x) Training Set

Model 1

Model 2

Model 3

Voting

Final Prediction

Ensemble Learning: Bagging

- **Bagging**
    - Generate distinct training sets by sampling with replacement from the original training set.
- Classification:
    - Plurality Vote (Majority Vote)
- Regression:
    - Average

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning: Random forests

- **Random forest** model is a form of **decision tree bagging** in which we take extra steps to make the ensemble of trees more diverse, to reduce variance.
- The key idea is to randomly vary the **attribute** choices (rather than the training **examples**)

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning: Random forests

- **Extremely randomized trees (ExtraTrees)**
  - **Use randomness in selecting the split point value**
  - **for each selected attribute, we randomly sample several candidate values from a uniform distribution over the attribute’s range**

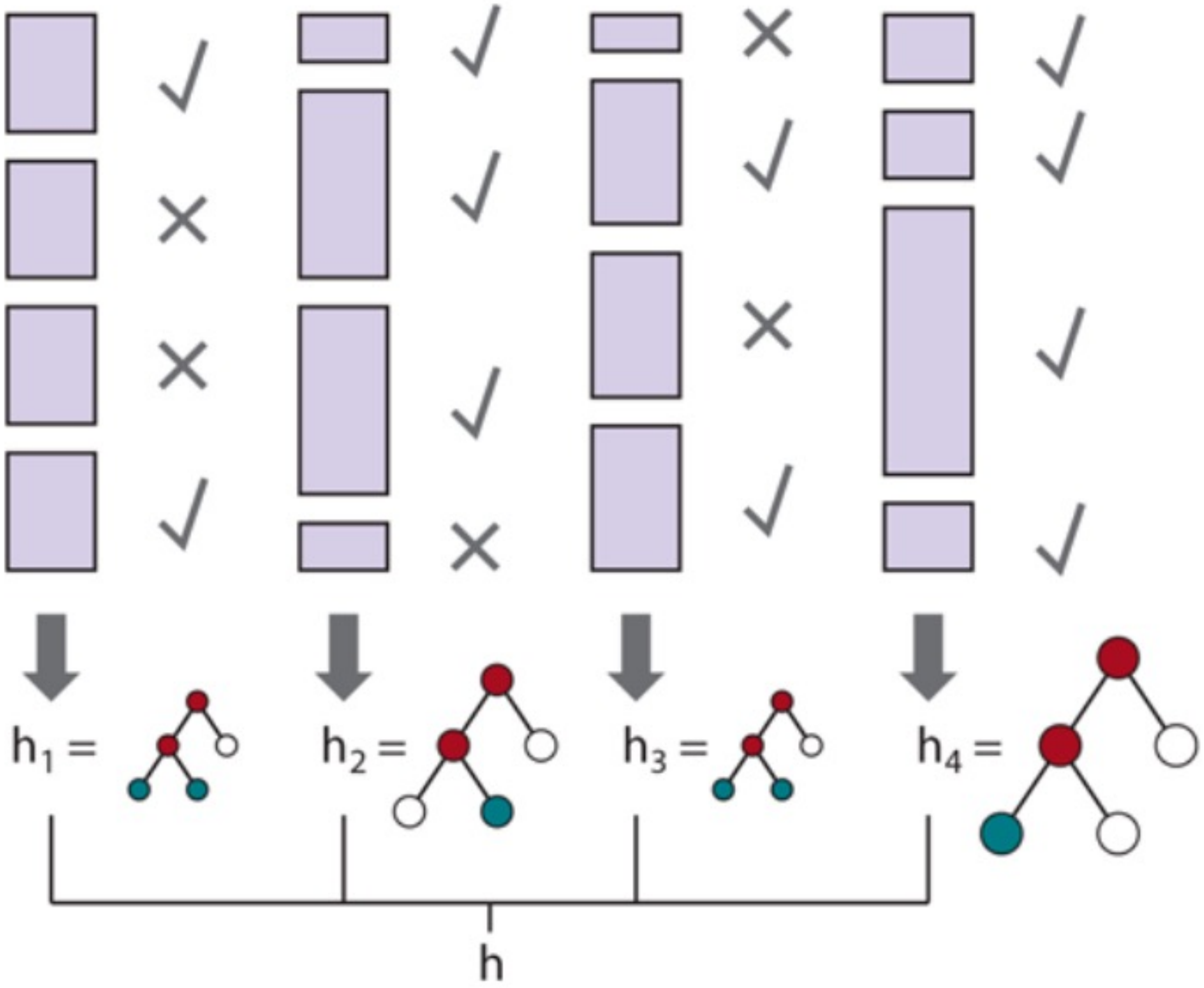
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning: Boosting

- **Boosting**
  - The most popular ensemble method
- **Weighted training set**

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning: Boosting



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning:
Gradient boosting

- **Gradient boosting**
  - Gradient boosting is a form of boosting using gradient descent
- **Gradient boosting machines (GBM)**
- **Gradient boosted regression trees (GBRT)**
- Popular method for regression and classification of factored tabular data

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning: Stacking

- Staking
  - Stacked generalization combines multiple base models from different model classes trained on the same data.
- Bagging
  - Combines multiple base models of the same model class trained on different data.

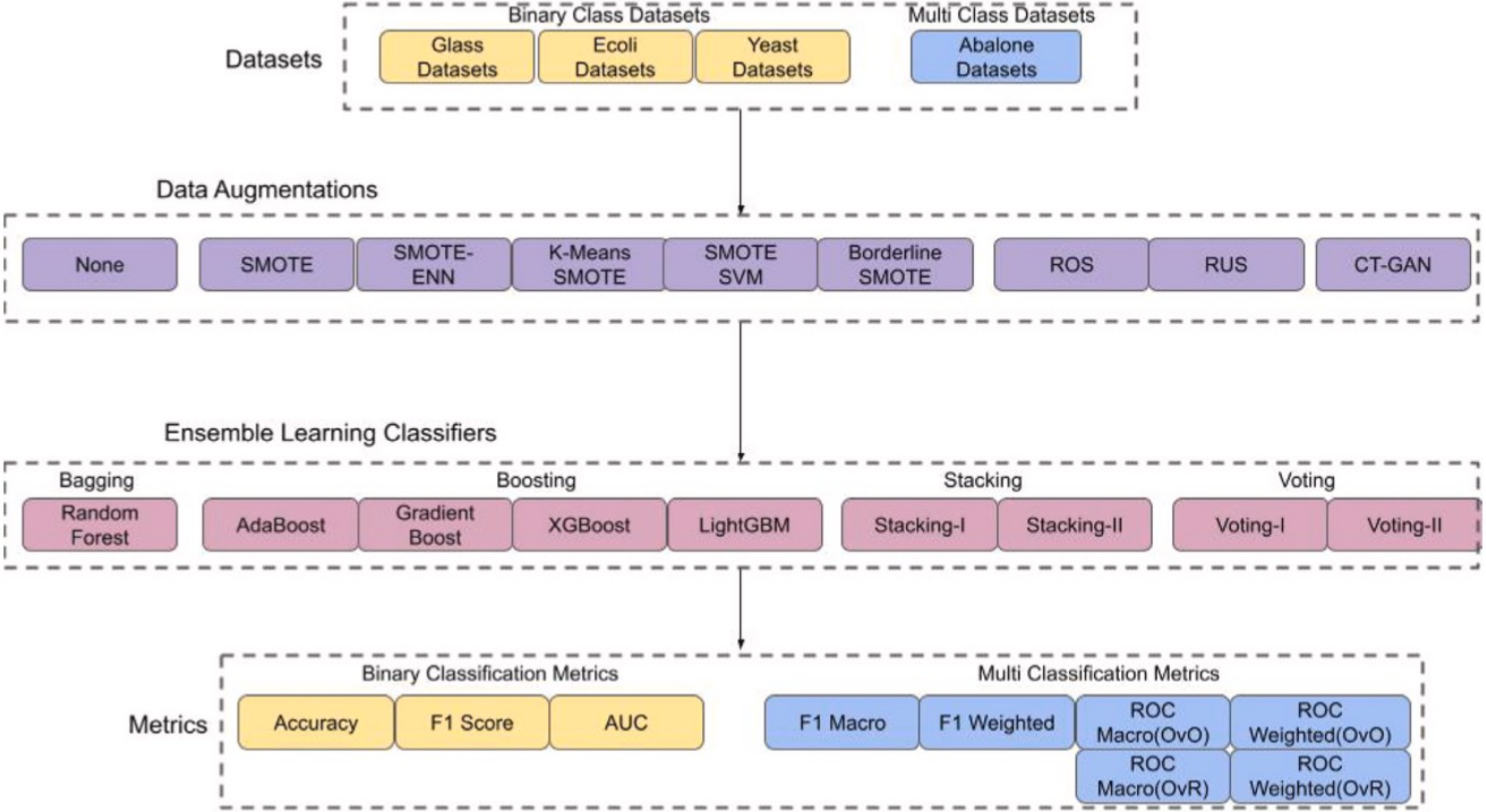
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning:
Online learning

- **Online learning**
  - **Data are not i.i.d.**
    (independent and identically distributed)
  - **An agent receives an input x_i from nature, predicts the corresponding y_i and then is told the correct answer.**

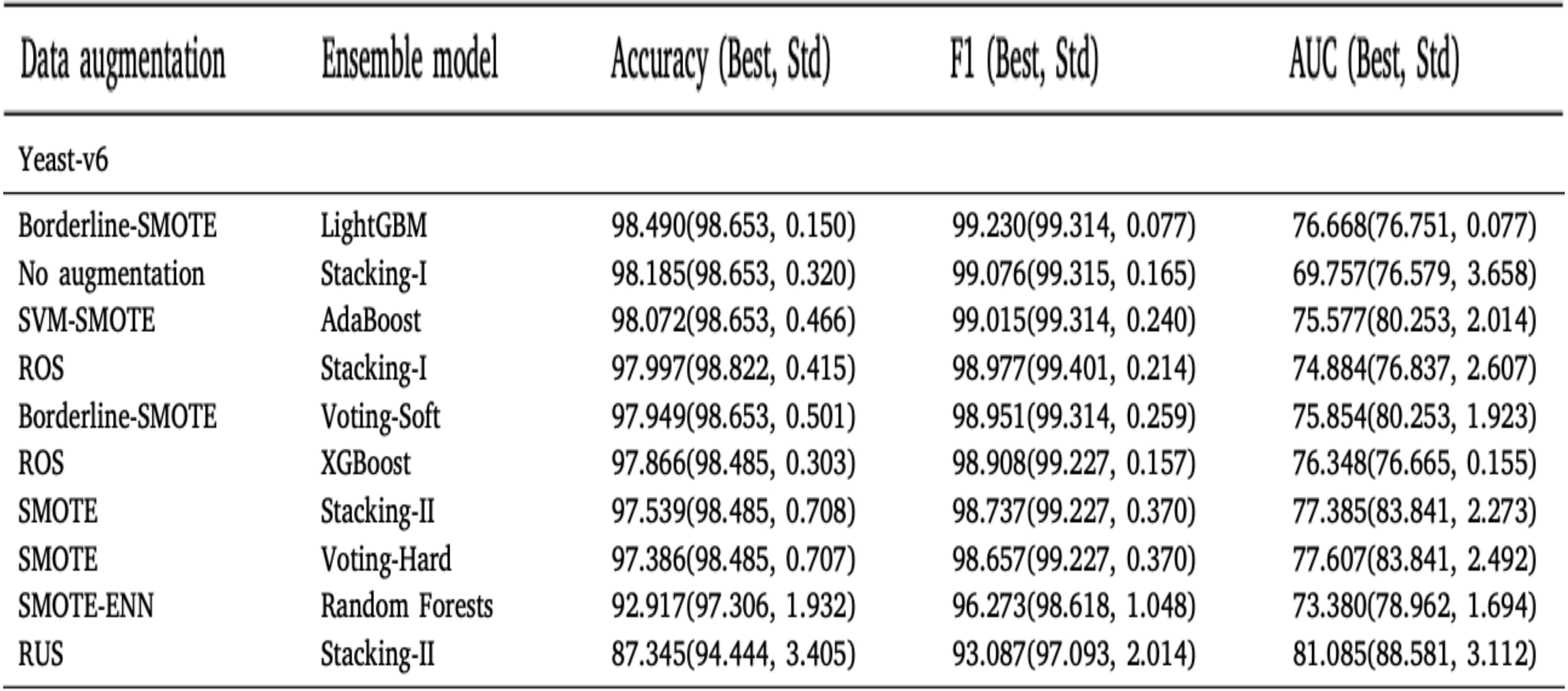
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Ensemble Learning and Data Augmentations (DA)



Source: Khan, A. A., Chaudhari, O., & Chandra, R. (2023). A review of ensemble learning and data augmentation models for class imbalanced problems: combination, implementation and evaluation. Expert Systems with Applications, 122778.

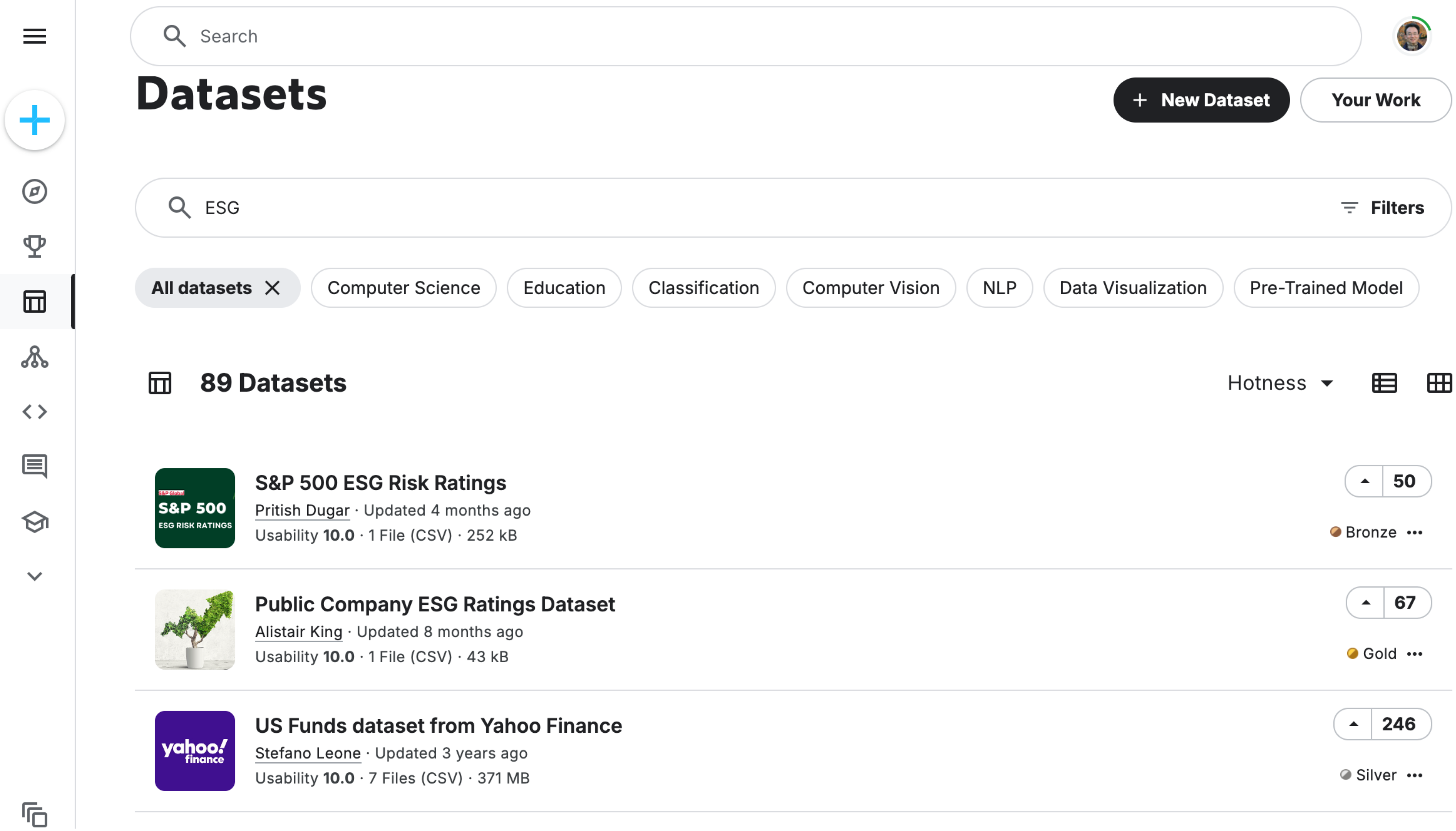
ML Evaluation of Imbalanced Dataset: Ensemble Learning and Data Augmentations (DA)



Data augmentation	Ensemble model	Accuracy (Best, Std)	F1 (Best, Std)	AUC (Best, Std)
Yeast-v6				
Borderline-SMOTE	LightGBM	98.490(98.653, 0.150)	99.230(99.314, 0.077)	76.668(76.751, 0.077)
No augmentation	Stacking-I	98.185(98.653, 0.320)	99.076(99.315, 0.165)	69.757(76.579, 3.658)
SVM-SMOTE	AdaBoost	98.072(98.653, 0.466)	99.015(99.314, 0.240)	75.577(80.253, 2.014)
ROS	Stacking-I	97.997(98.822, 0.415)	98.977(99.401, 0.214)	74.884(76.837, 2.607)
Borderline-SMOTE	Voting-Soft	97.949(98.653, 0.501)	98.951(99.314, 0.259)	75.854(80.253, 1.923)
ROS	XGBoost	97.866(98.485, 0.303)	98.908(99.227, 0.157)	76.348(76.665, 0.155)
SMOTE	Stacking-II	97.539(98.485, 0.708)	98.737(99.227, 0.370)	77.385(83.841, 2.273)
SMOTE	Voting-Hard	97.386(98.485, 0.707)	98.657(99.227, 0.370)	77.607(83.841, 2.492)
SMOTE-ENN	Random Forests	92.917(97.306, 1.932)	96.273(98.618, 1.048)	73.380(78.962, 1.694)
RUS	Stacking-II	87.345(94.444, 3.405)	93.087(97.093, 2.014)	81.085(88.581, 3.112)

Source: Khan, A. A., Chaudhari, O., & Chandra, R. (2023). A review of ensemble learning and data augmentation models for class imbalanced problems: combination, implementation and evaluation. Expert Systems with Applications, 122778.

Kaggle Datasets for Machine Learning



Datasets

89 Datasets

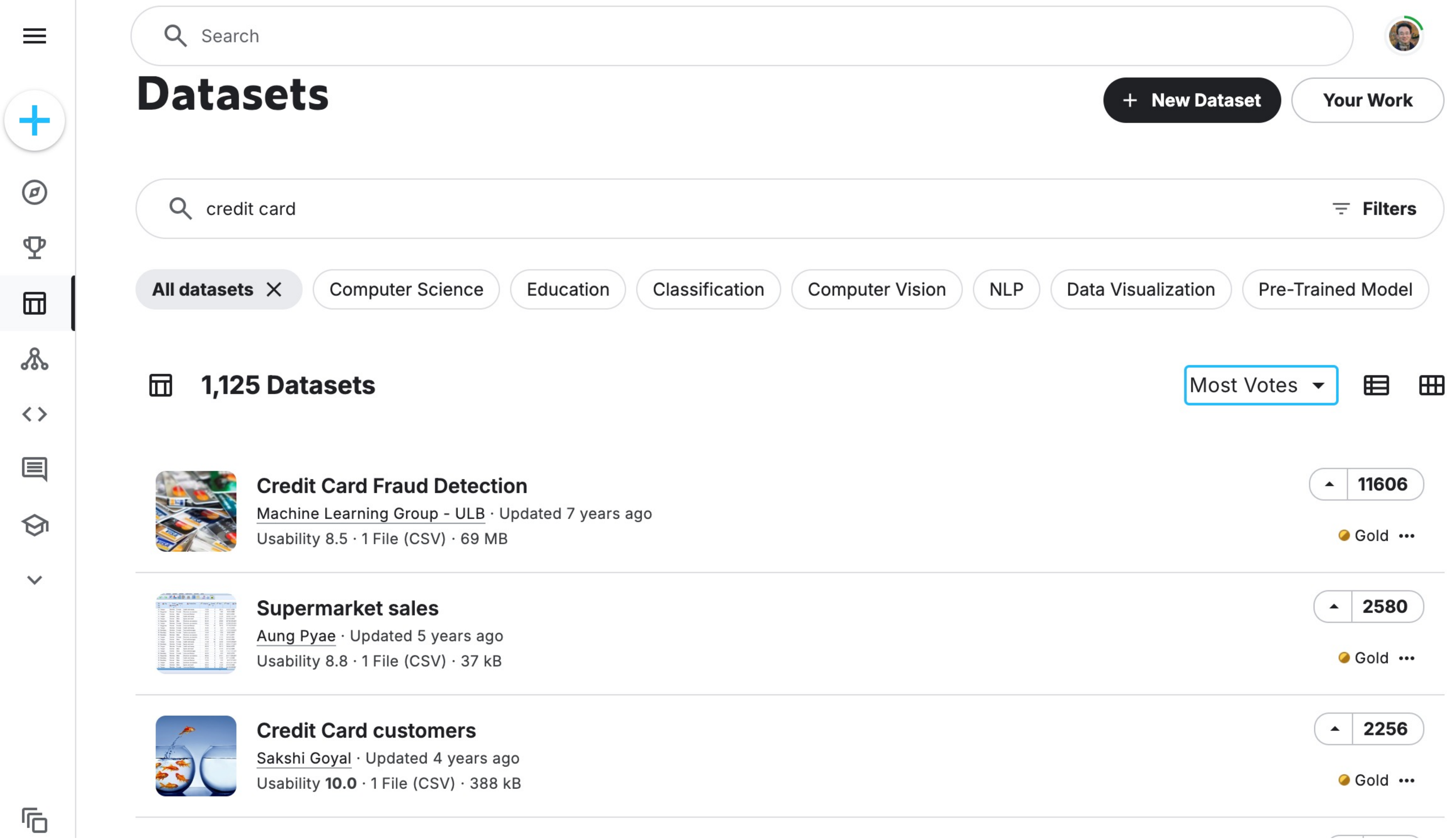
S&P 500 ESG Risk Ratings
Pritish Dugar · Updated 4 months ago
Usability **10.0** · 1 File (CSV) · 252 kB
50
Bronze

Public Company ESG Ratings Dataset
Alistair King · Updated 8 months ago
Usability **10.0** · 1 File (CSV) · 43 kB
67
Gold

US Funds dataset from Yahoo Finance
Stefano Leone · Updated 3 years ago
Usability **10.0** · 7 Files (CSV) · 371 MB
246
Silver

Source: https://www.kaggle.com/datasets?search=ESG

Kaggle Datasets for Machine Learning



Search

Datasets

credit card

1,125 Datasets

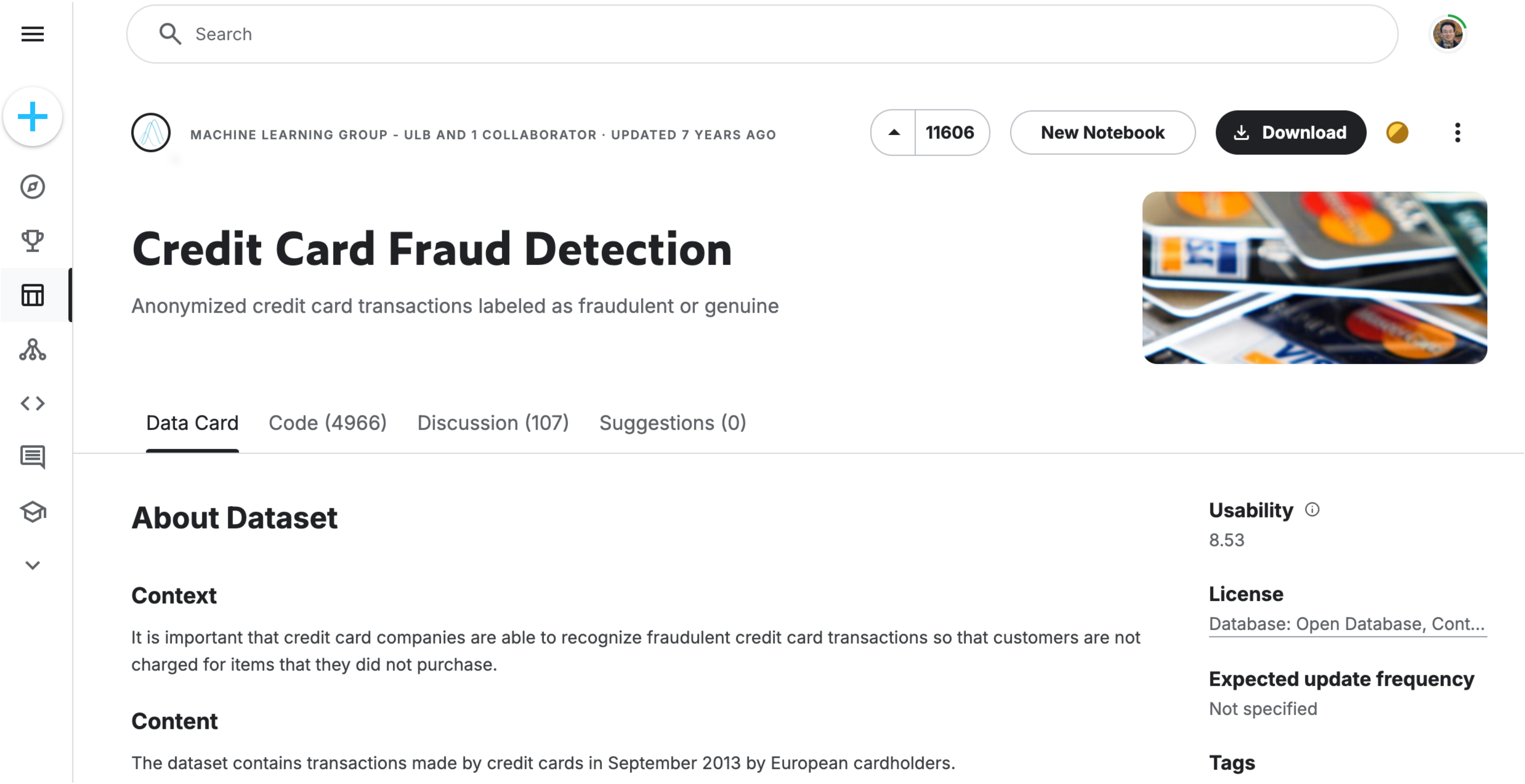
Credit Card Fraud Detection
Machine Learning Group - ULB · Updated 7 years ago
Usability 8.5 · 1 File (CSV) · 69 MB
11606
Gold

Supermarket sales
Aung Pyae · Updated 5 years ago
Usability 8.8 · 1 File (CSV) · 37 kB
2580
Gold

Credit Card customers
Sakshi Goyal · Updated 4 years ago
Usability **10.0** · 1 File (CSV) · 388 kB
2256
Gold

Source: https://www.kaggle.com/datasets?search=credit+card&sort=votes

Kaggle Datasets: Credit Card Fraud Detection



MACHINE LEARNING GROUP - ULB AND 1 COLLABORATOR · UPDATED 7 YEARS AGO

Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

About Dataset

Context

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Content

The dataset contains transactions made by credit cards in September 2013 by European cardholders.

Usability
8.53

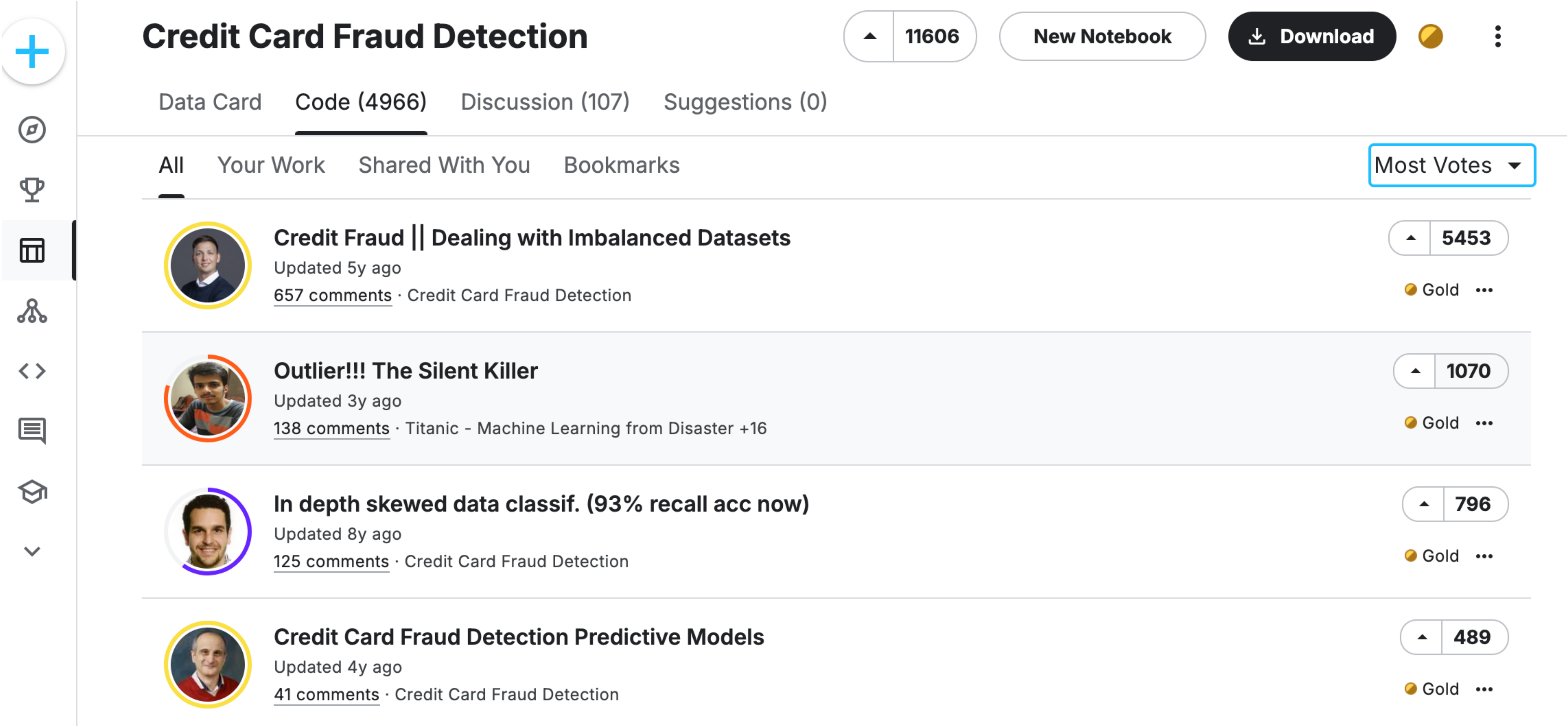
License
Database: Open Database, Cont...

Expected update frequency
Not specified

Tags

Source: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Kaggle Code: Credit Card Fraud Detection



Credit Card Fraud Detection

Credit Fraud || Dealing with Imbalanced Datasets
Updated 5y ago
657 comments · Credit Card Fraud Detection
5453 · Gold

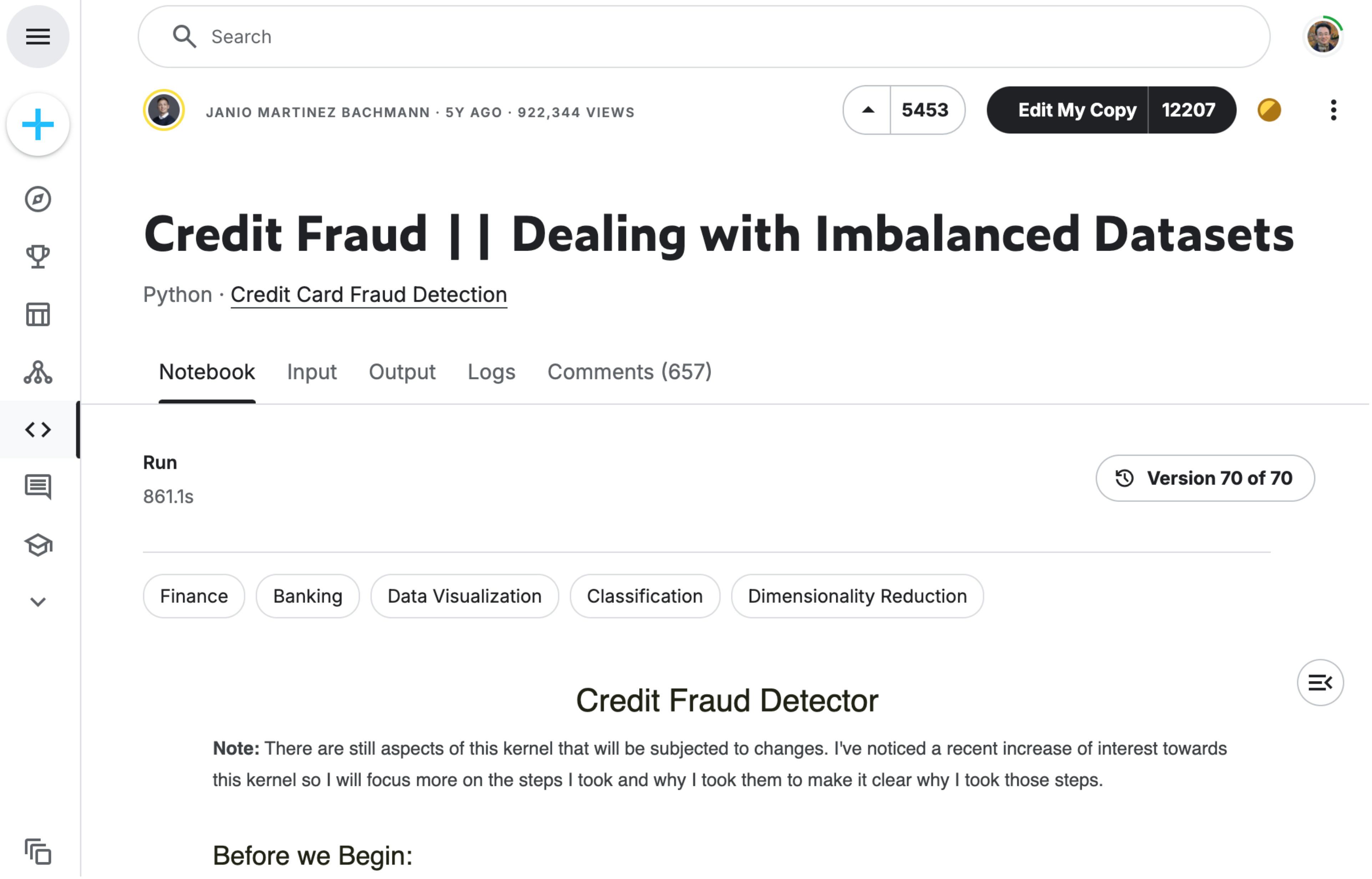
Outlier!!! The Silent Killer
Updated 3y ago
138 comments · Titanic - Machine Learning from Disaster +16
1070 · Gold

In depth skewed data classif. (93% recall acc now)
Updated 8y ago
125 comments · Credit Card Fraud Detection
796 · Gold

Credit Card Fraud Detection Predictive Models
Updated 4y ago
41 comments · Credit Card Fraud Detection
489 · Gold

Source: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/code?datasetId=310&sortBy=voteCount

Kaggle Code: Credit Card Fraud Detection



Credit Fraud || Dealing with Imbalanced Datasets

Python · Credit Card Fraud Detection

Run
861.1s

Credit Fraud Detector

Note: There are still aspects of this kernel that will be subjected to changes. I've noticed a recent increase of interest towards this kernel so I will focus more on the steps I took and why I took them to make it clear why I took those steps.

Before we Begin:

Source: https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets

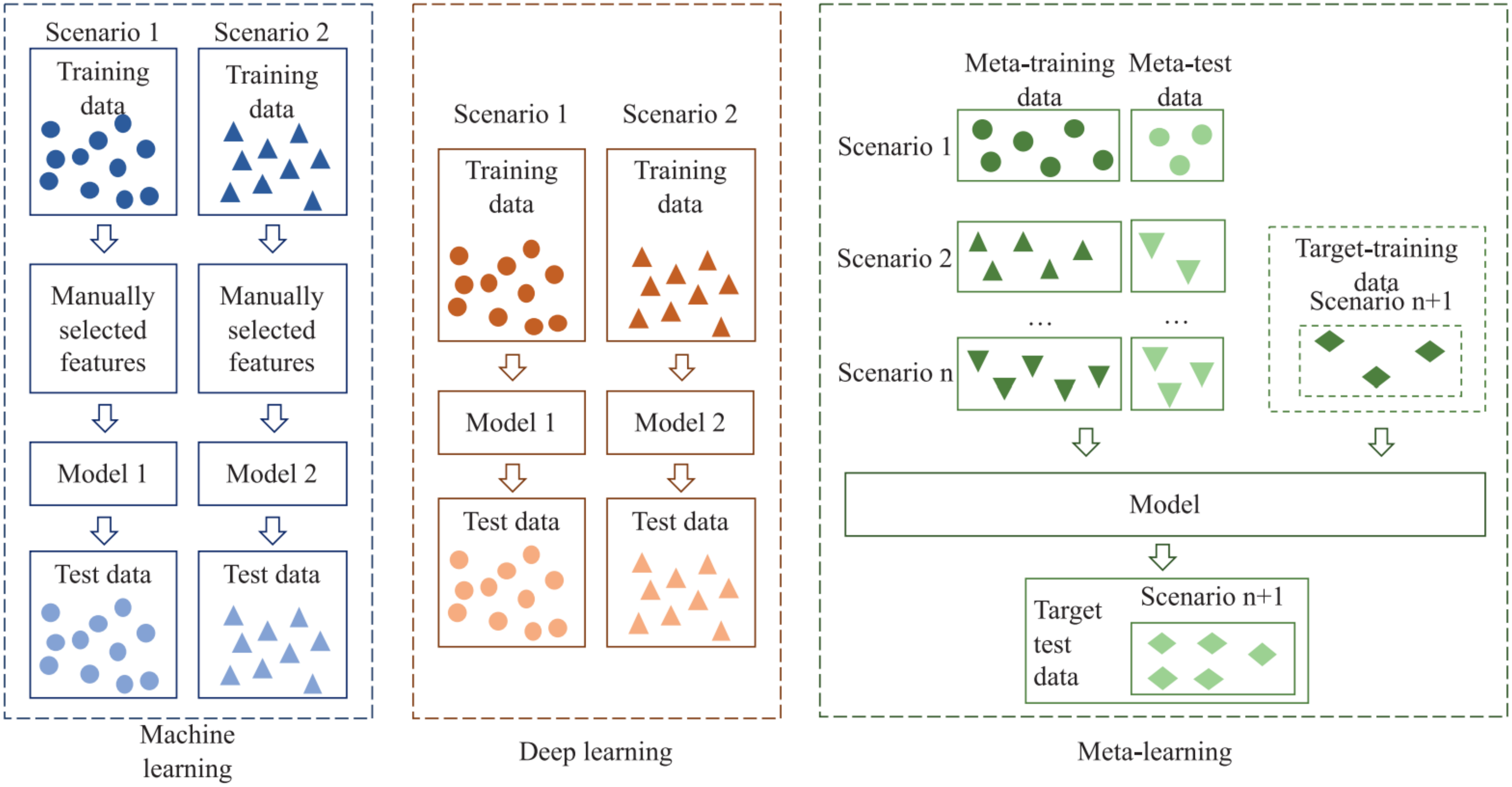
Meta Learning:
Learning to Learn

Deep Learning
Transfer Learning
Few-Shot Learning
Meta Learning

Deep Learning, Transfer Learning, Few-Shot Learning, Meta Learning

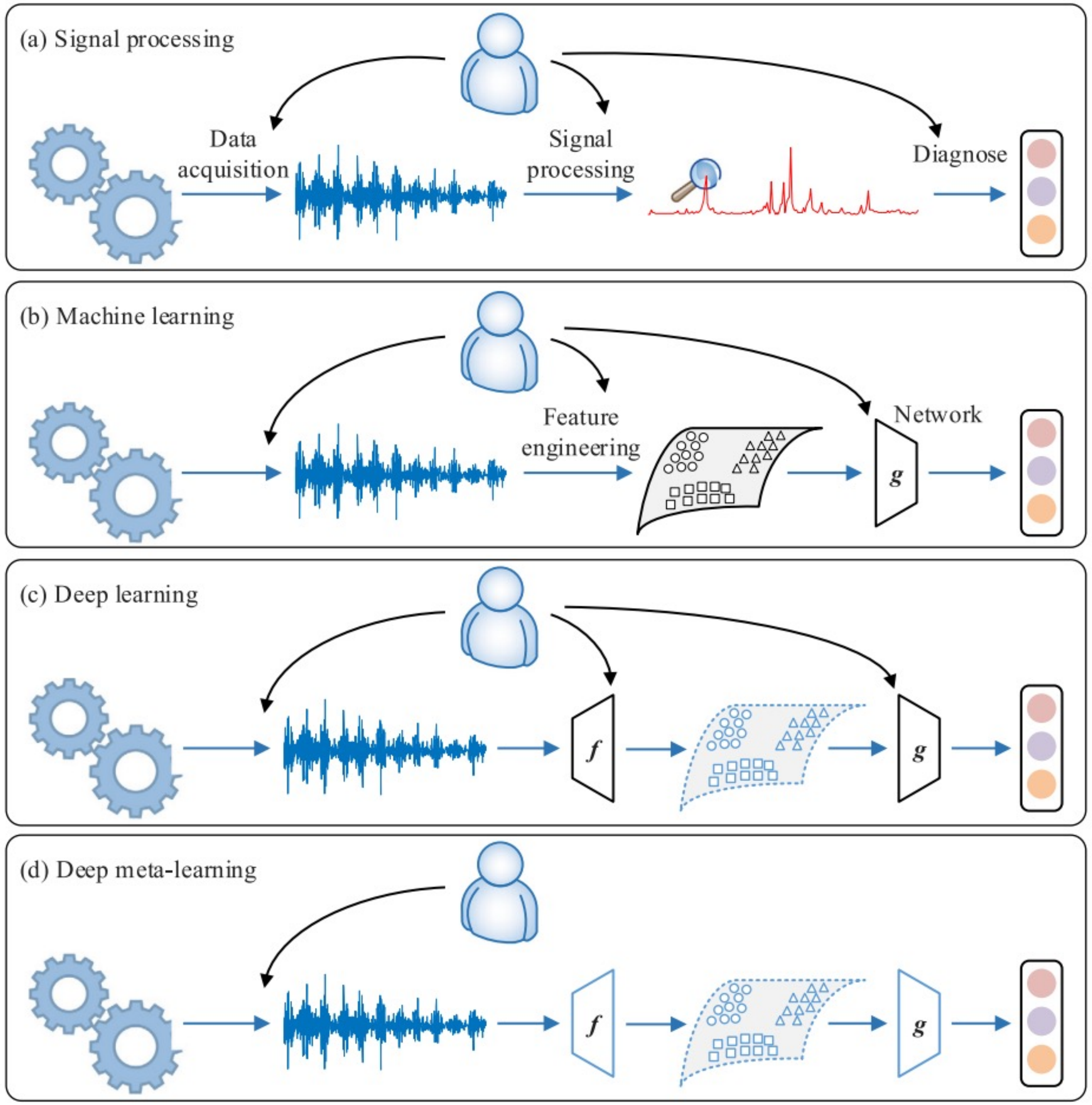
- Deep Learning
  - Transfer Learning
    - Pre-training, Fine-Tuning (FT)
- Meta Learning: Learning to Learn
- Few-Shot Learning (FSL)
- One-Shot Learning (1SL)
- Zero-Shot Learning (0SL)(ZSL)

Machine Learning, Deep Learning, Meta Learning



Source: Luo, Shuai, Yujie Li, Pengxiang Gao, Yichuan Wang, and Seiichi Serikawa. "Meta-seg: A survey of meta-learning for image segmentation." Pattern Recognition (2022): 108586.

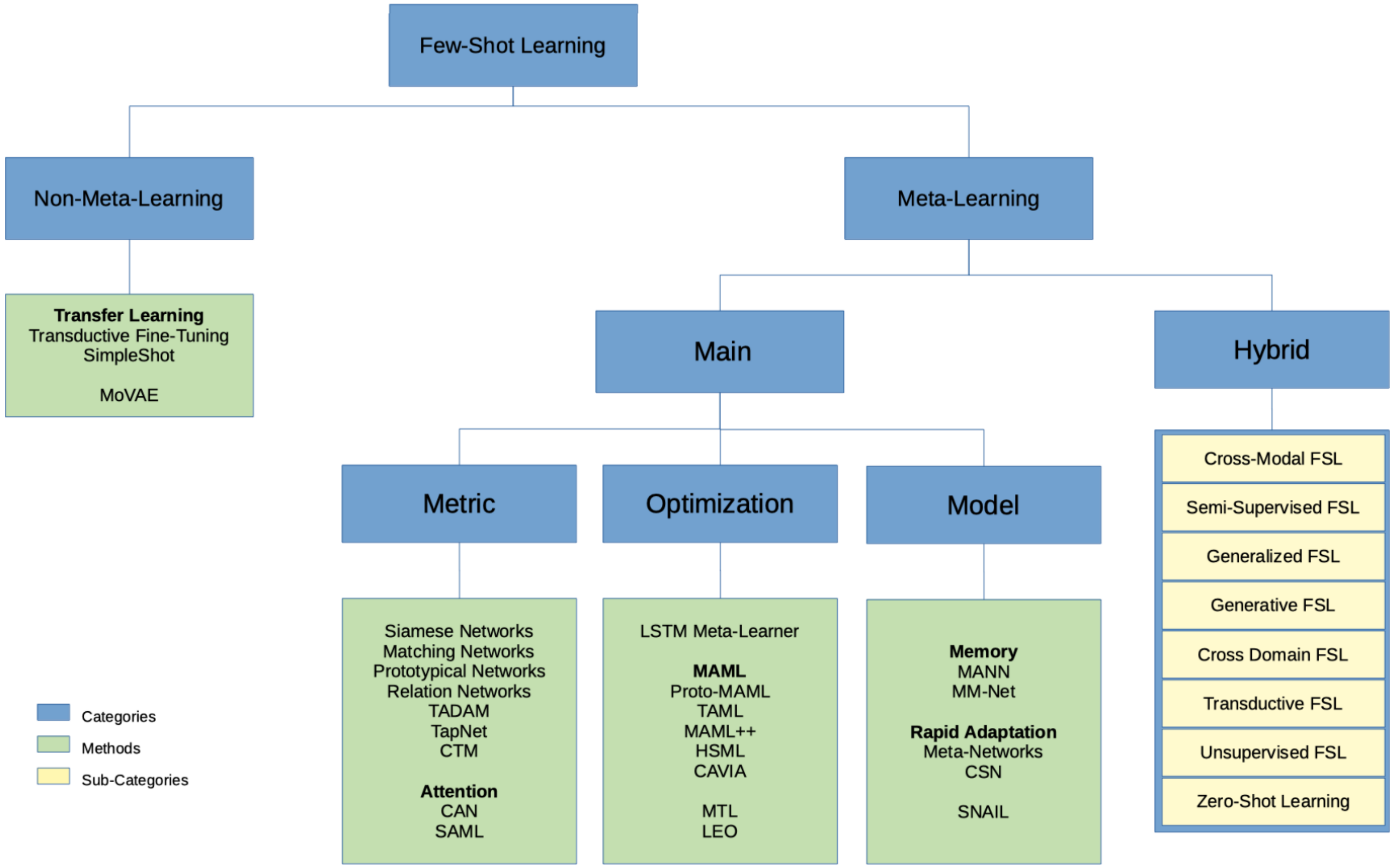
Machine Learning, Deep Learning, Meta Learning



Source: Feng, Yong, Jinglong Chen, Jingsong Xie, Tianci Zhang, Haixin Lv, and Tongyang Pan. "Meta-learning as a promising approach for few-shot cross-domain fault diagnosis: Algorithms, applications, and prospects." Knowledge-Based Systems 235 (2022): 107646.

Few-Shot Learning (FSL) and Meta Learning

Machine learning from few training examples



Few-Shot Learning

Non-Meta-Learning

Transfer Learning
Transductive Fine-Tuning
SimpleShot

MoVAE

Meta-Learning

Main

Hybrid

Metric

Siamese Networks
Matching Networks
Prototypical Networks
Relation Networks
TADAM
TapNet
CTM

Attention
CAN
SAML

Optimization

LSTM Meta-Learner

MAML
Proto-MAML
TAML
MAML++
HSML
CAVIA

MTL
LEO

Model

Memory
MANN
MM-Net

Rapid Adaptation
Meta-Networks
CSN

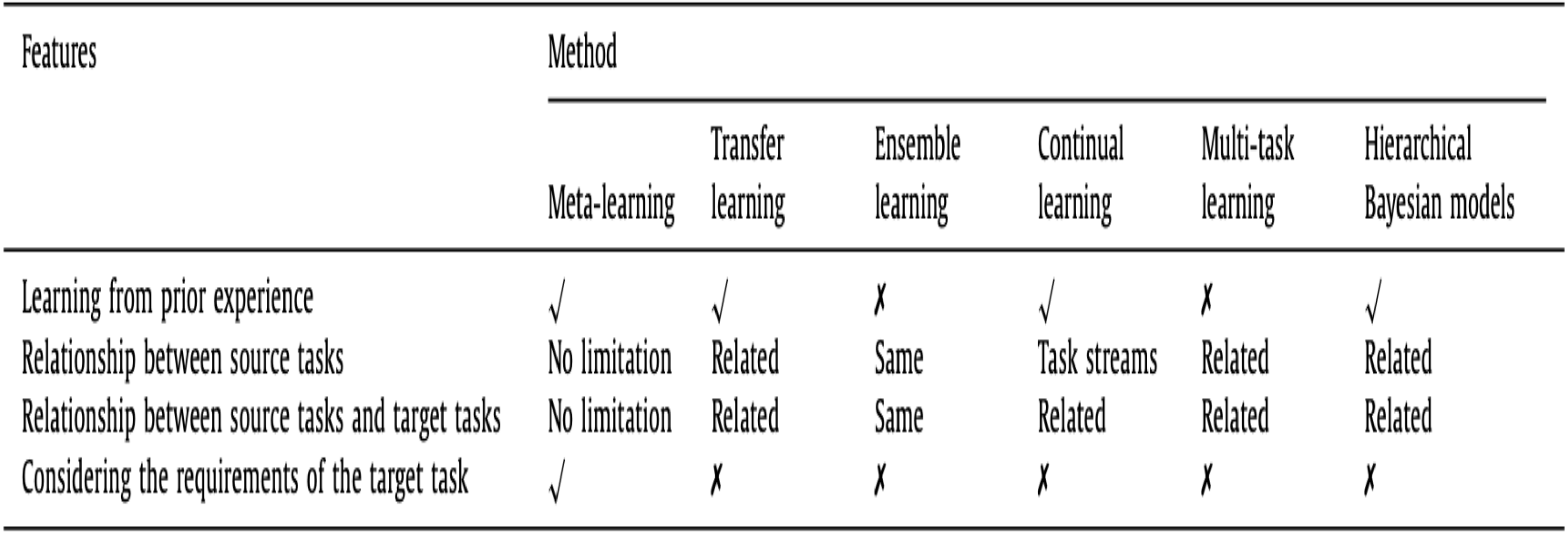
SNAIL

- Cross-Modal FSL
- Semi-Supervised FSL
- Generalized FSL
- Generative FSL
- Cross Domain FSL
- Transductive FSL
- Unsupervised FSL
- Zero-Shot Learning

🟦 Categories
🟩 Methods
🟨 Sub-Categories

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Meta Learning, Transfer Learning, Ensemble Learning, Continual Learning, Multi-Task Learning

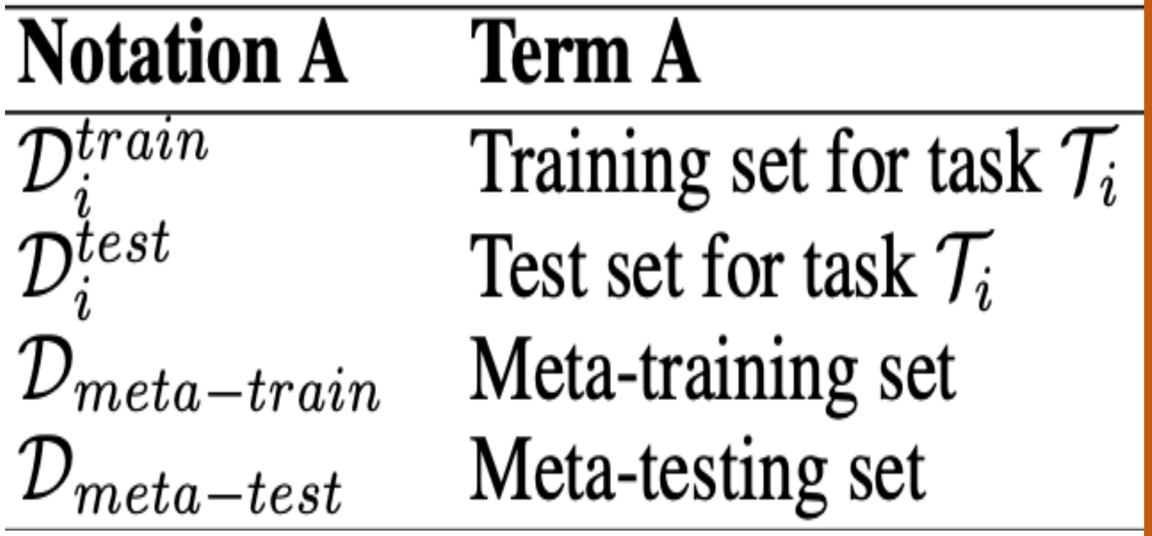


Features	Method					
	Meta-learning	Transfer learning	Ensemble learning	Continual learning	Multi-task learning	Hierarchical Bayesian models
Learning from prior experience	√	√	✗	√	✗	√
Relationship between source tasks	No limitation	Related	Same	Task streams	Related	Related
Relationship between source tasks and target tasks	No limitation	Related	Same	Related	Related	Related
Considering the requirements of the target task	√	✗	✗	✗	✗	✗

Source: Luo, Shuai, Yujie Li, Pengxiang Gao, Yichuan Wang, and Seiichi Serikawa. "Meta-seg: A survey of meta-learning for image segmentation." Pattern Recognition (2022): 108586.

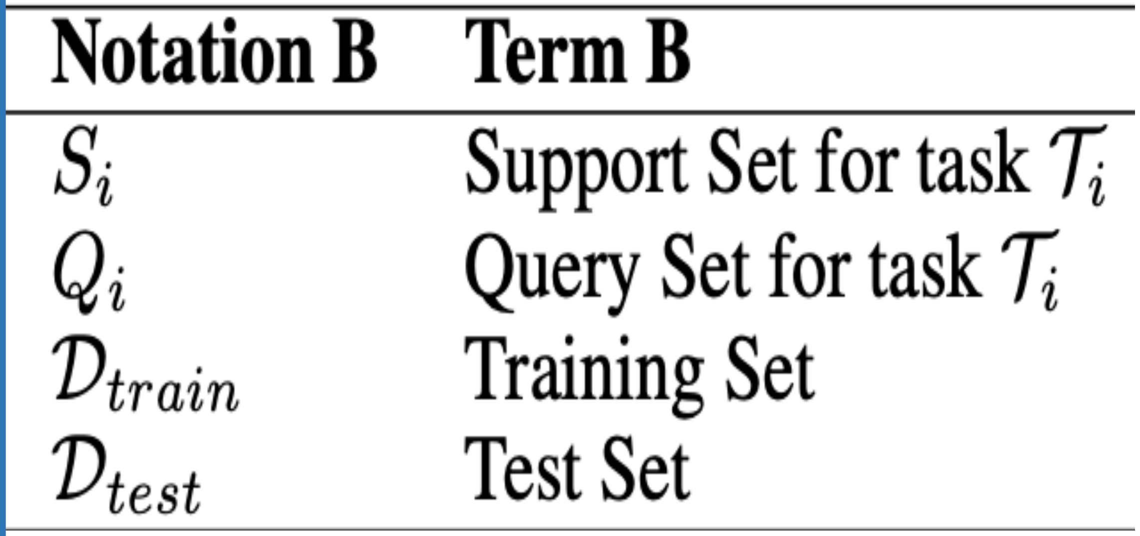
Meta-Learning and Few-shot Learning
Notations and Terms

Optimization-based Meta-learning



Notation A	Term A
$\mathcal{D}_i^{train}$	Training set for task $\mathcal{T}_i$
$\mathcal{D}_i^{test}$	Test set for task $\mathcal{T}_i$
$\mathcal{D}_{meta-train}$	Meta-training set
$\mathcal{D}_{meta-test}$	Meta-testing set

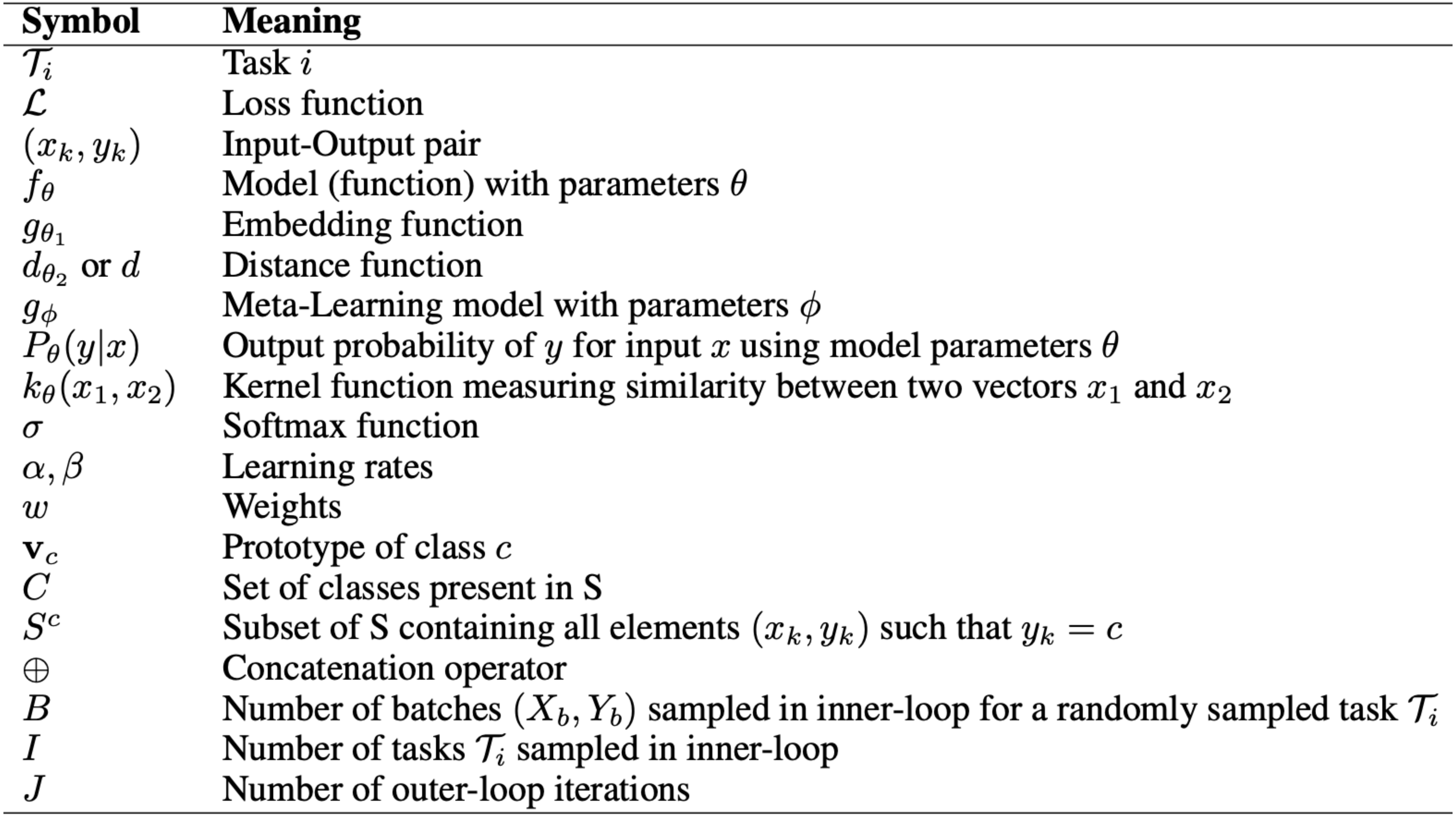
Metric-based Meta-learning



Notation B	Term B
S_i	Support Set for task $\mathcal{T}_i$
Q_i	Query Set for task $\mathcal{T}_i$
$\mathcal{D}_{train}$	Training Set
$\mathcal{D}_{test}$	Test Set

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

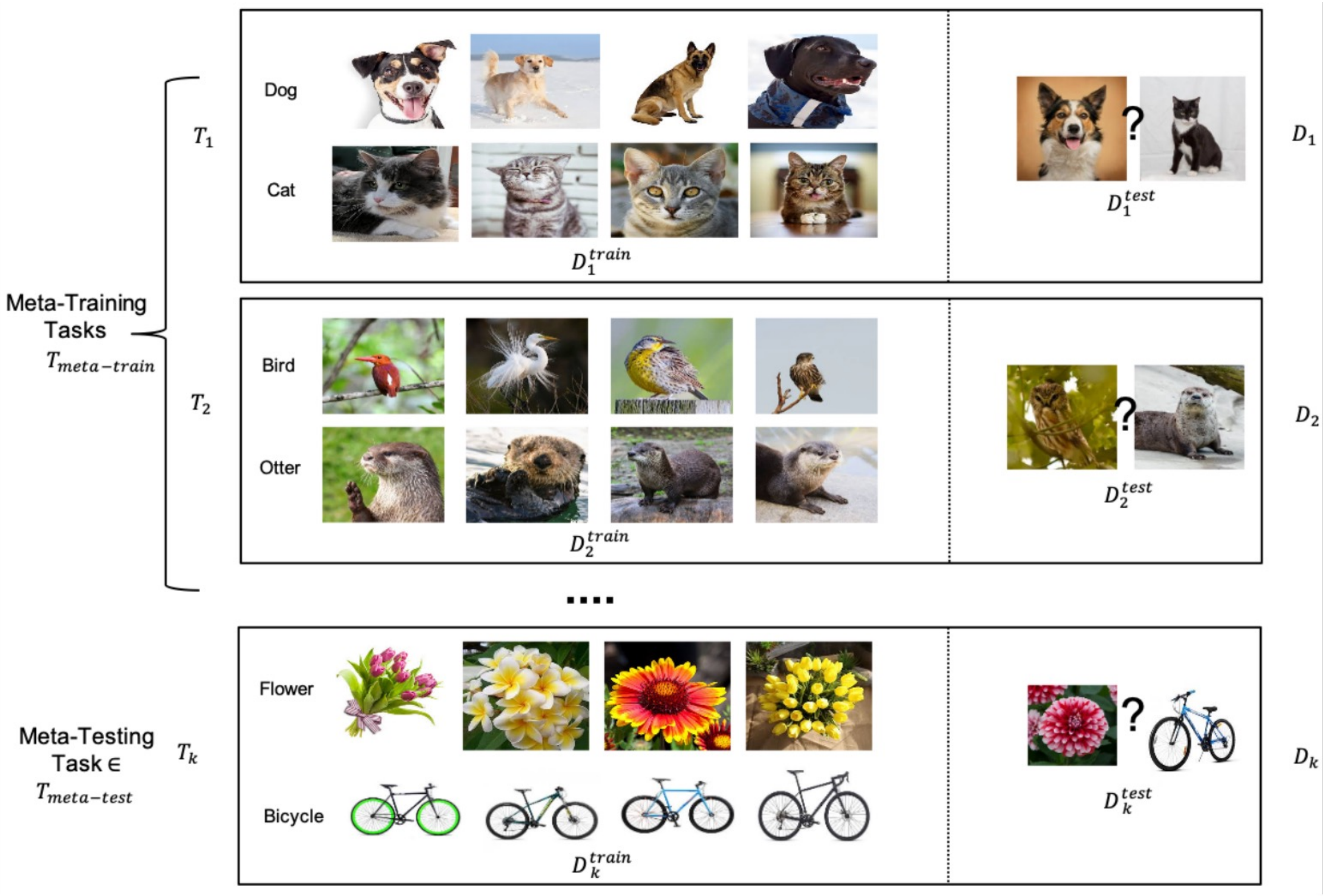
Meta-Learning Symbols



Symbol	Meaning
$\mathcal{T}_i$	Task i
$\mathcal{L}$	Loss function
(x_k, y_k)	Input-Output pair
f_θ	Model (function) with parameters θ
g_{θ_1}	Embedding function
d_{θ_2} or d	Distance function
g_ϕ	Meta-Learning model with parameters ϕ
$P_\theta(y\|x)$	Output probability of y for input x using model parameters θ
$k_\theta(x_1, x_2)$	Kernel function measuring similarity between two vectors x_1 and x_2
σ	Softmax function
α, β	Learning rates
w	Weights
$\mathbf{v}_c$	Prototype of class c
C	Set of classes present in S
S^c	Subset of S containing all elements (x_k, y_k) such that $y_k = c$
$\bigoplus$	Concatenation operator
B	Number of batches (X_b, Y_b) sampled in inner-loop for a randomly sampled task $\mathcal{T}_i$
I	Number of tasks $\mathcal{T}_i$ sampled in inner-loop
J	Number of outer-loop iterations

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

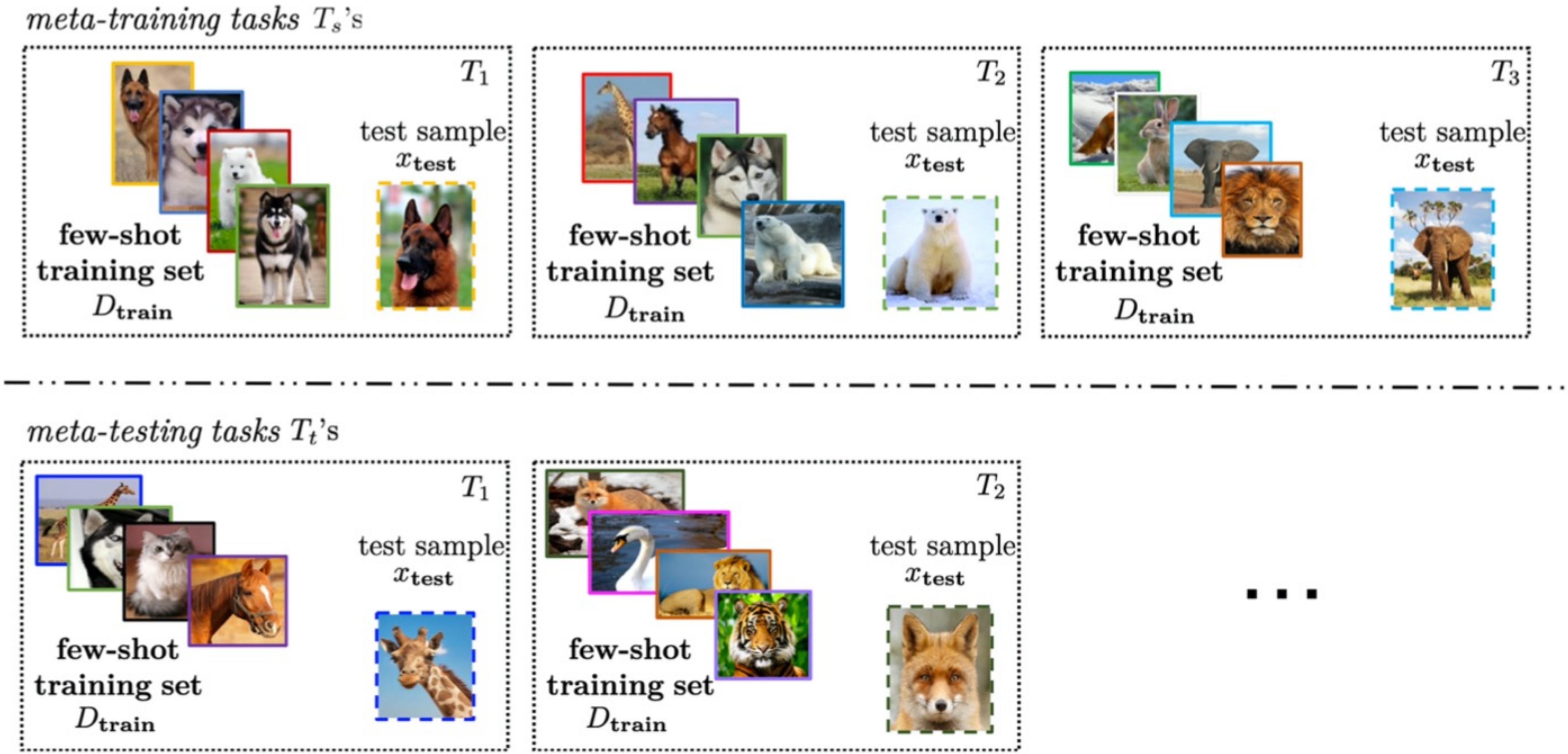
Meta-Learning Example Setup



Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Few-Shot Learning (FSL)

Solving the FSL problem by meta-learning

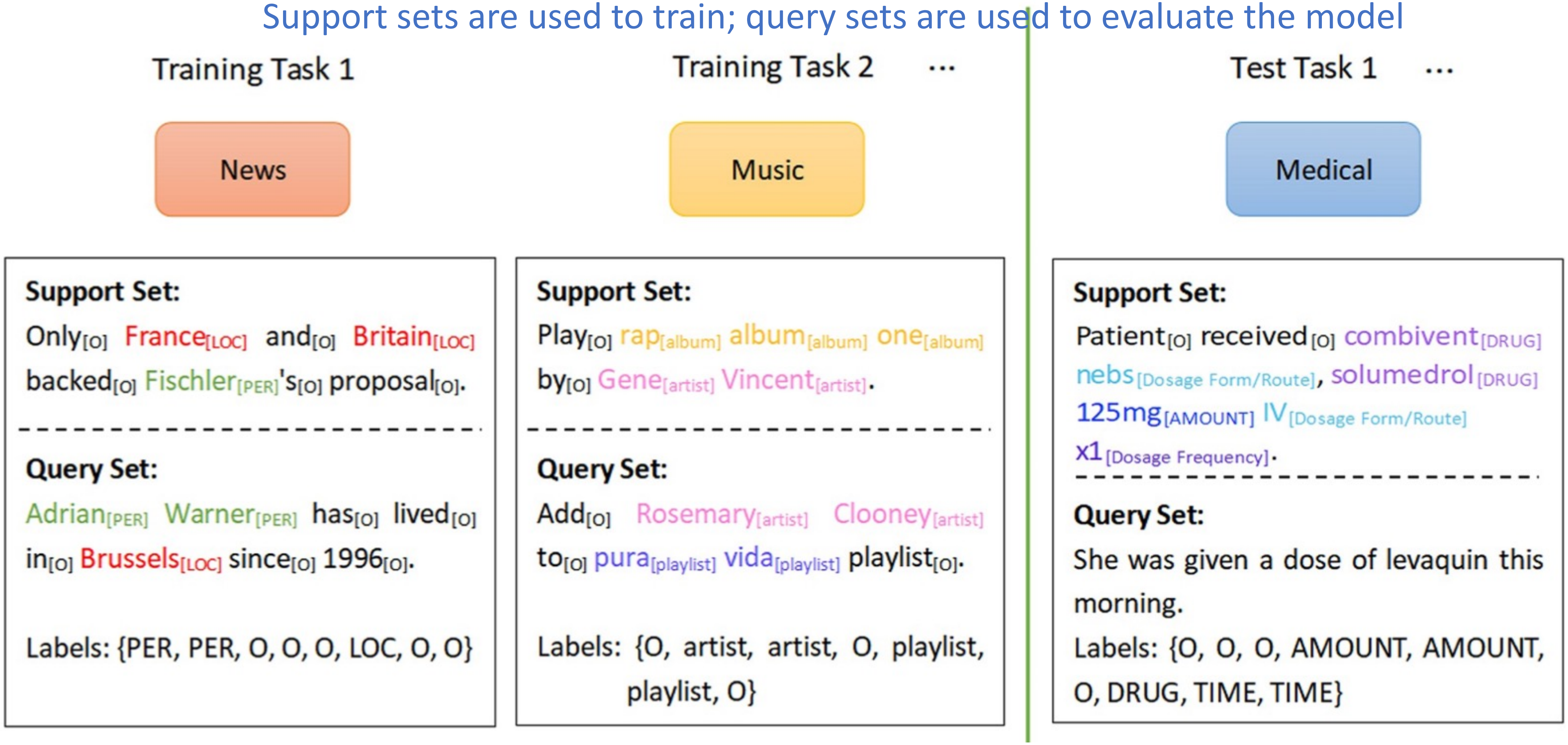


Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Meta-learning

Each task mimics the few-shot scenario, and can be completely non-overlapping.
Support sets are used to train; query sets are used to evaluate the model



Support Set:
Only[O] France[LOC] and[O] Britain[LOC] backed[O] Fischler[PER]'s[O] proposal[O].

Query Set:
Adrian[PER] Warner[PER] has[O] lived[O] in[O] Brussels[LOC] since[O] 1996[O].

Labels: {PER, PER, O, O, O, LOC, O, O}

Support Set:
Play[O] rap[album] album[album] one[album] by[O] Gene[artist] Vincent[artist].

Query Set:
Add[O] Rosemary[artist] Clooney[artist] to[O] pura[playlist] vida[playlist] playlist[O].

Labels: {O, artist, artist, O, playlist, playlist, O}

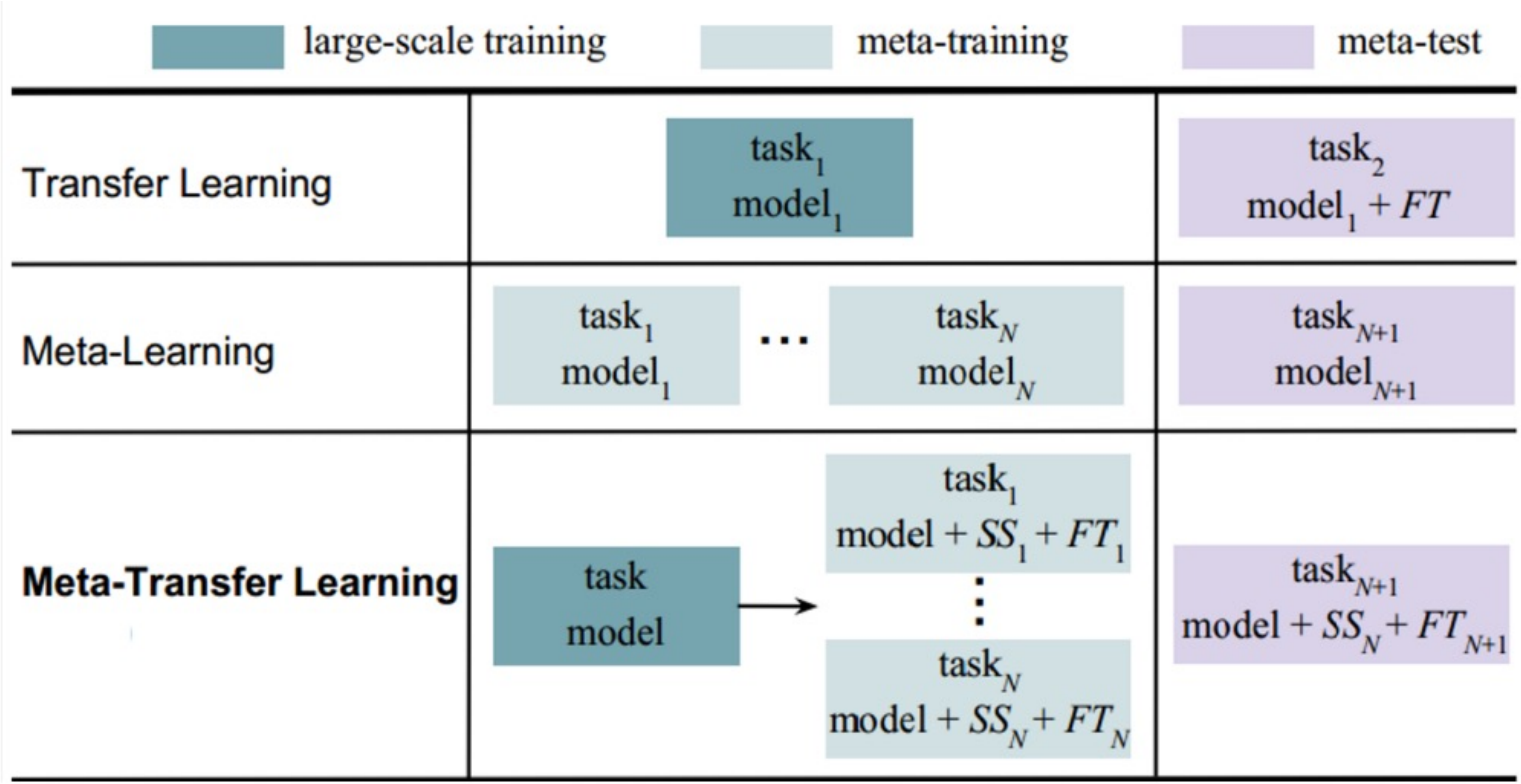
Support Set:
Patient[O] received[O] combivent[DRUG] nebs[Dosage Form/Route], solumedrol[DRUG] 125mg[AMOUNT] IV[Dosage Form/Route] x1[Dosage Frequency].

Query Set:
She was given a dose of levaquin this morning.
Labels: {O, O, O, AMOUNT, AMOUNT, O, DRUG, TIME, TIME}

Source: Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).

Meta-Task Learning (MTL)

Transfer Learning Strategy for Meta-Learning



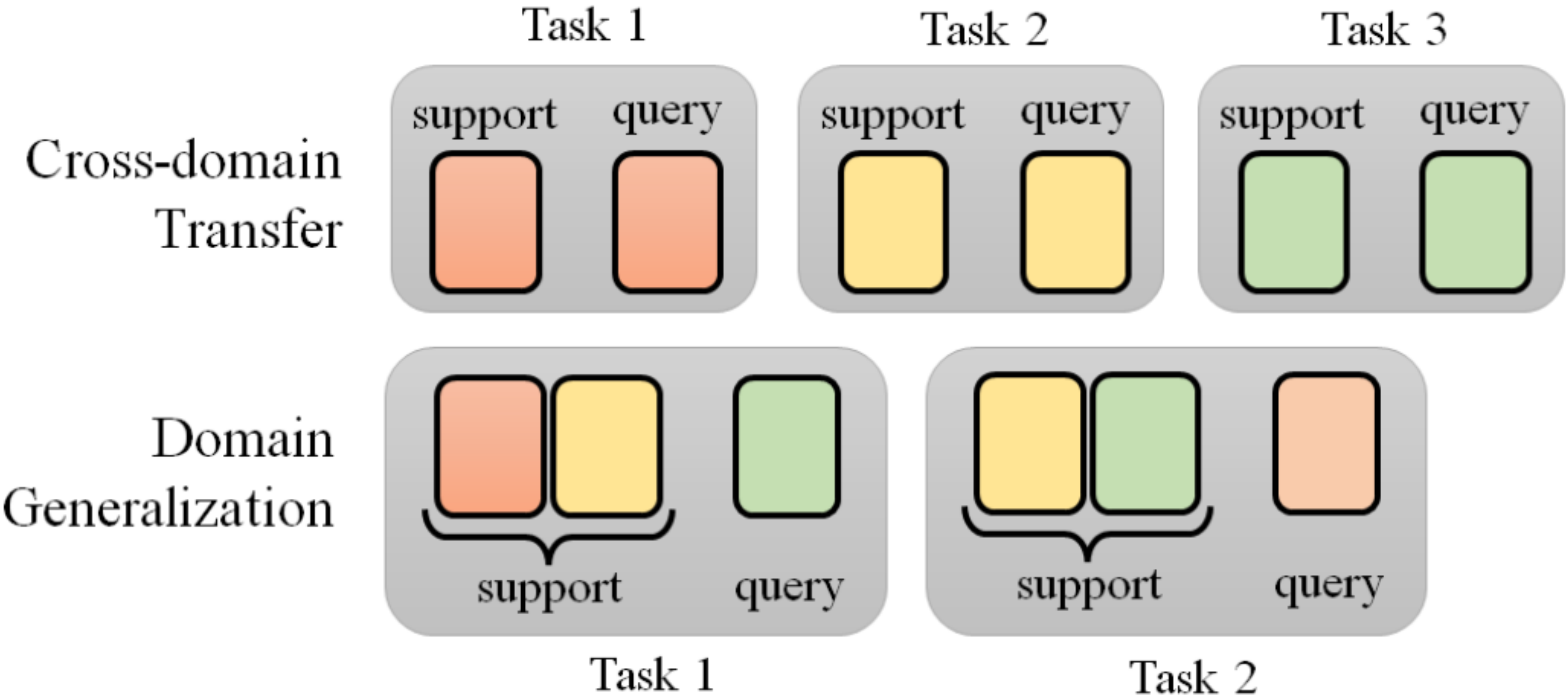
■ large-scale training □ meta-training 🟪 meta-test

Transfer Learning	task$_1$ model$_1$	task$_2$ model$_1$ + FT
Meta-Learning	task$_1$ model$_1$ … task$_N$ model$_N$	task$_{N+1}$ model$_{N+1}$
Meta-Transfer Learning	task model → task$_1$ model + SS_1 + FT_1 ⋮ task$_N$ model + SS_N + FT_N	task$_{N+1}$ model + SS_N + FT_{N+1}

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

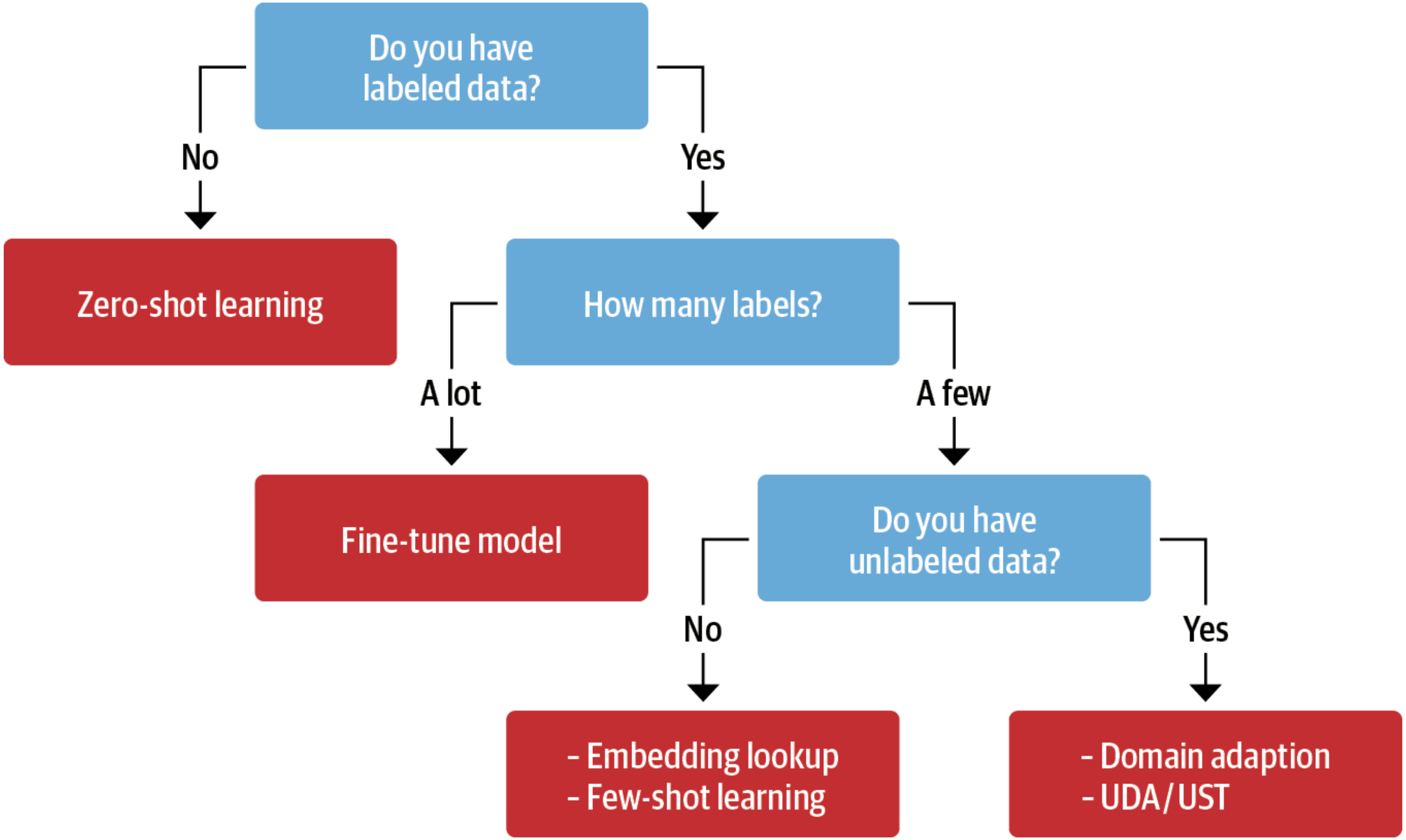
Meta Learning

The task construction of cross-domain transfer and domain generalization



Source: Lee, Hung-yi, Shang-Wen Li, and Ngoc Thang Vu. "Meta Learning for Natural Language Processing: A Survey." arXiv preprint arXiv:2205.01500 (2022).

Transfer Learning, Fine-tuning, Few-shot learning



Do you have labeled data?

No

Yes

Zero-shot learning

How many labels?

A lot

A few

Fine-tune model

Do you have unlabeled data?

No

Yes

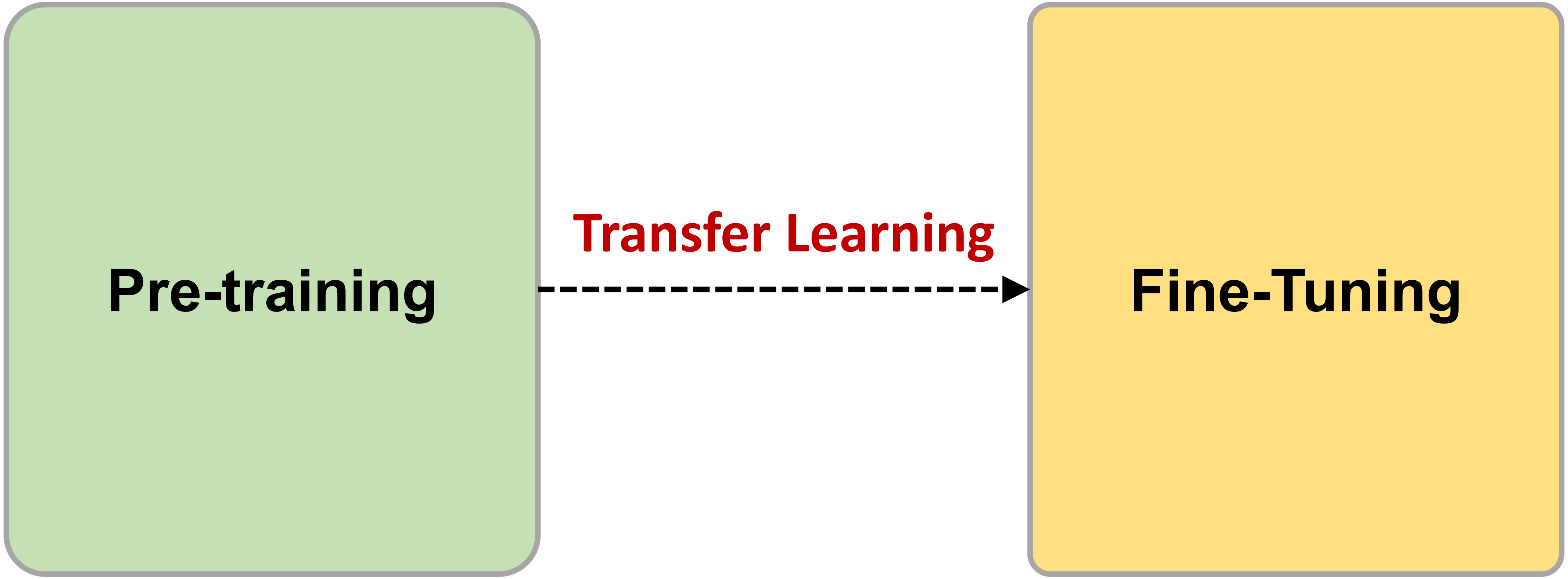
- Embedding lookup
- Few-shot learning

- Domain adaption
- UDA/UST

Unsupervised Data Augmentation (UDA)
Uncertainty-aware Self-Training (UST)

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Transfer Learning



Pre-training

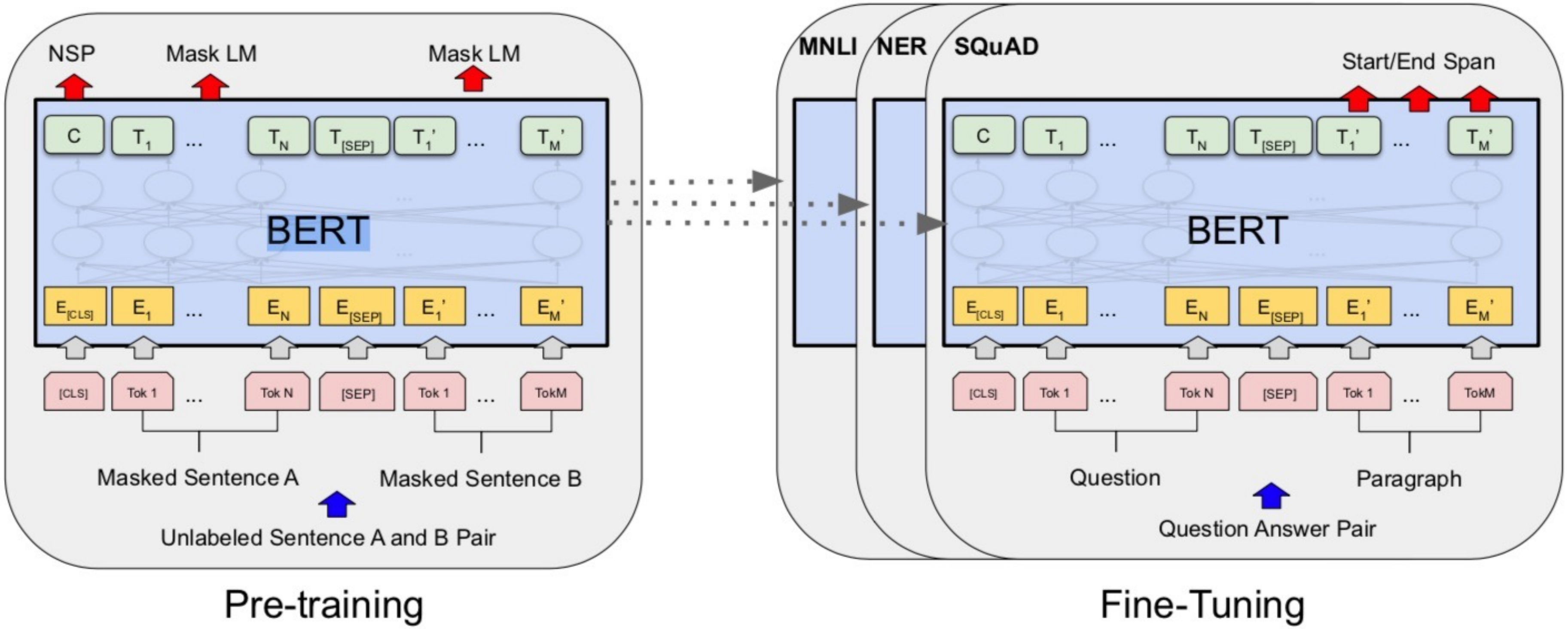
Transfer Learning

Fine-Tuning

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

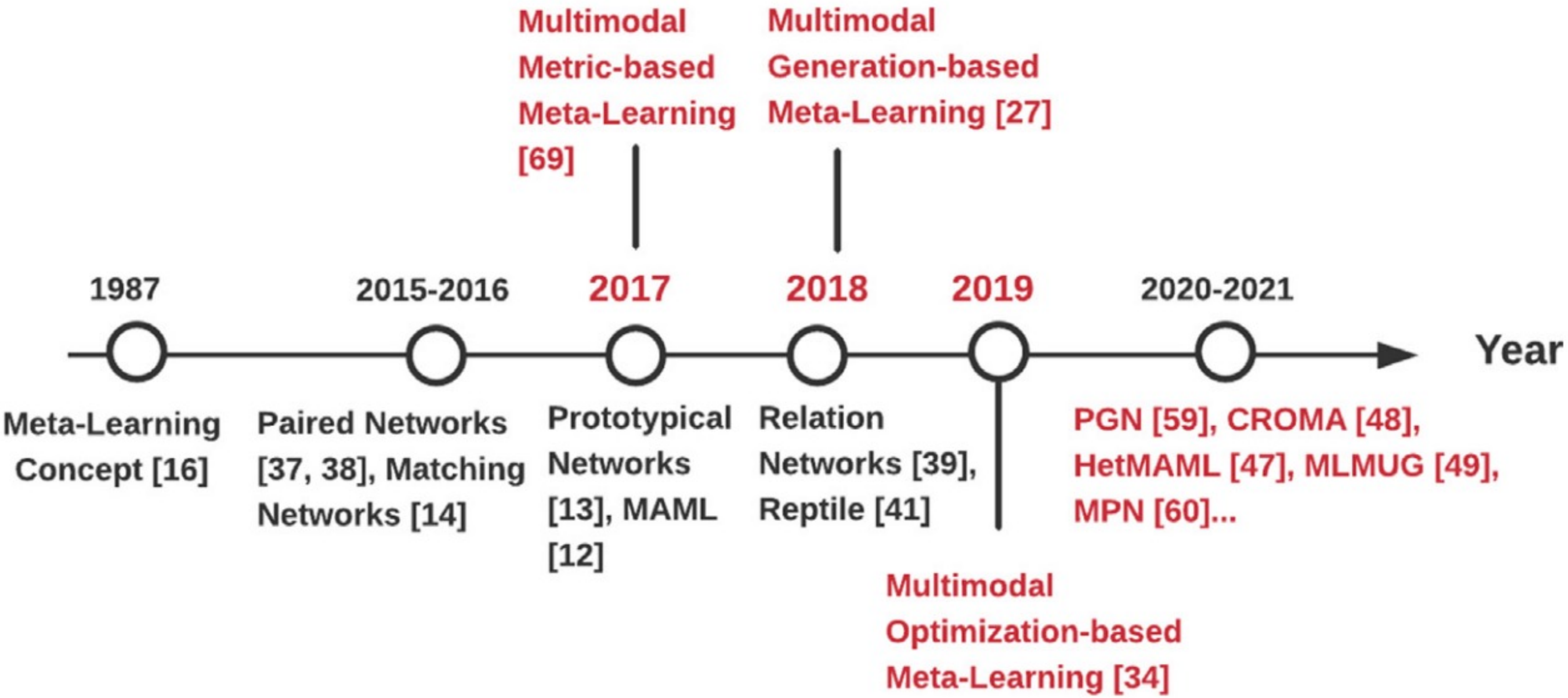
BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT



Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Meta Learning



Multimodal Metric-based Meta-Learning [69]

Multimodal Generation-based Meta-Learning [27]

1987

Meta-Learning Concept [16]

2015-2016

Paired Networks [37, 38], Matching Networks [14]

2017

Prototypical Networks [13], MAML [12]

2018

Relation Networks [39], Reptile [41]

2019

Multimodal Optimization-based Meta-Learning [34]

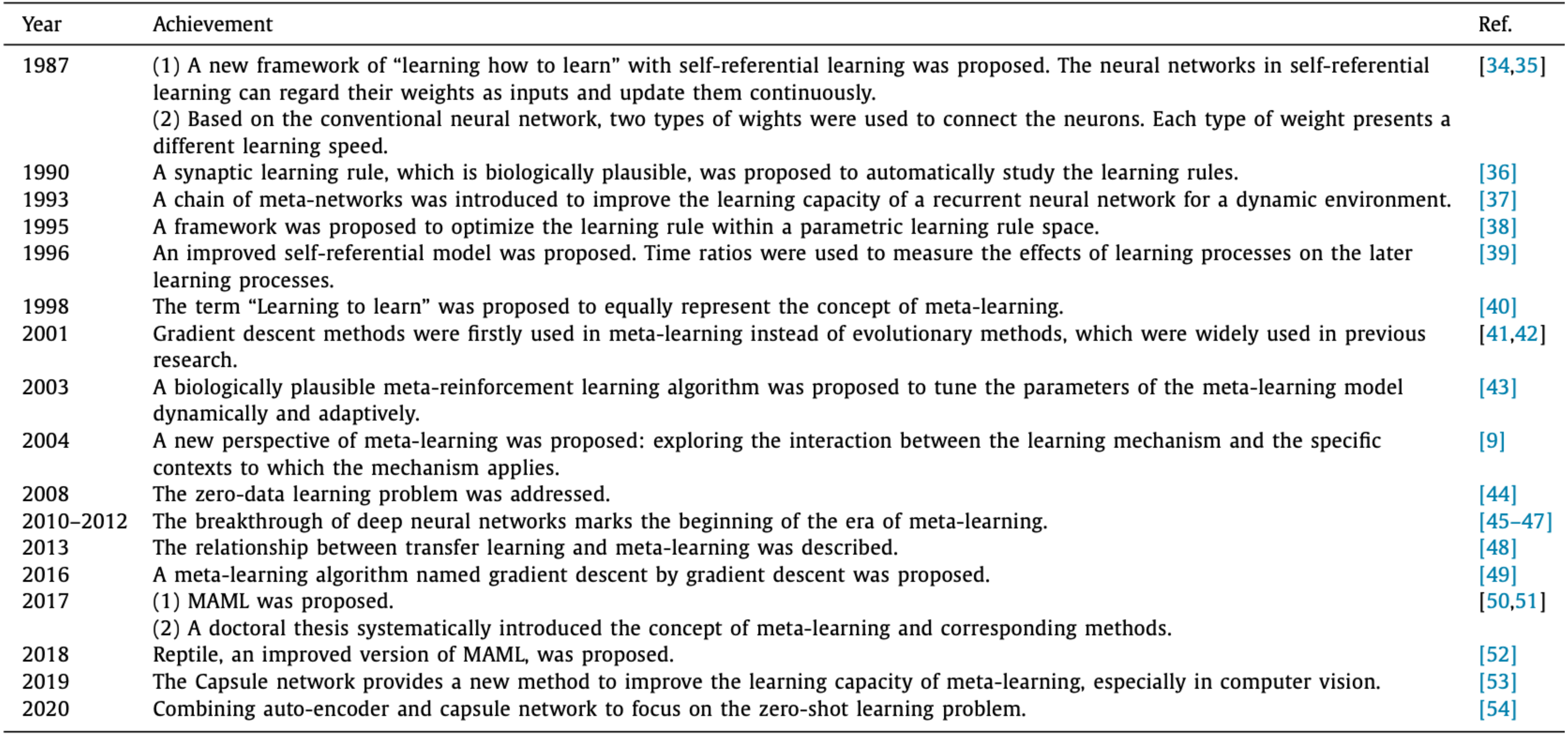
2020-2021

PGN [59], CROMA [48], HetMAML [47], MLMUG [49], MPN [60]...

Year

Source: TMa, Yao, Shilin Zhao, Weixiao Wang, Yaoman Li, and Irwin King. "Multimodality in meta-learning: A comprehensive survey." Knowledge-Based Systems (2022): 108976.

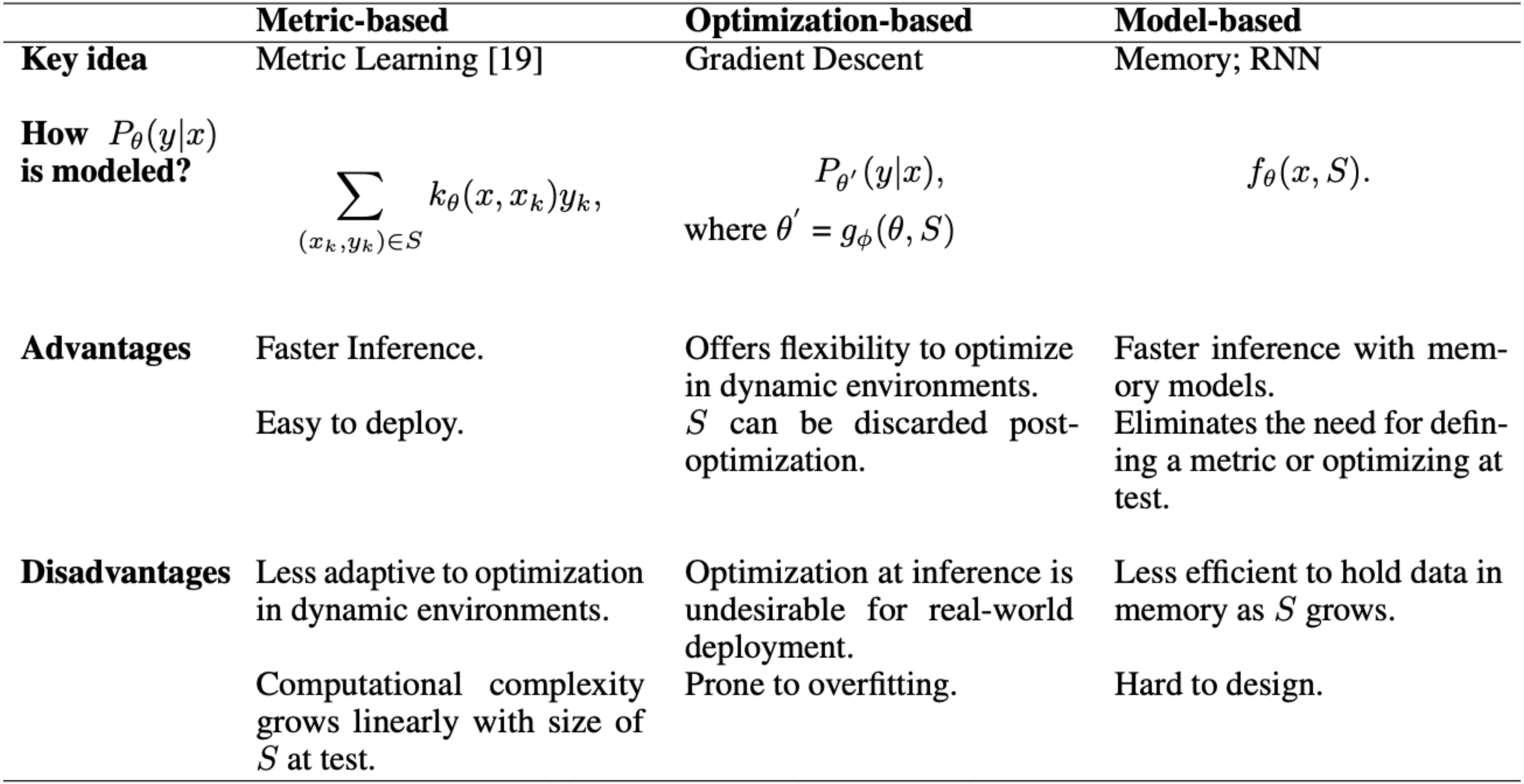
Meta Learning



Year	Achievement	Ref.
1987	(1) A new framework of "learning how to learn" with self-referential learning was proposed. The neural networks in self-referential learning can regard their weights as inputs and update them continuously. (2) Based on the conventional neural network, two types of wights were used to connect the neurons. Each type of weight presents a different learning speed.	[34,35]
1990	A synaptic learning rule, which is biologically plausible, was proposed to automatically study the learning rules.	[36]
1993	A chain of meta-networks was introduced to improve the learning capacity of a recurrent neural network for a dynamic environment.	[37]
1995	A framework was proposed to optimize the learning rule within a parametric learning rule space.	[38]
1996	An improved self-referential model was proposed. Time ratios were used to measure the effects of learning processes on the later learning processes.	[39]
1998	The term "Learning to learn" was proposed to equally represent the concept of meta-learning.	[40]
2001	Gradient descent methods were firstly used in meta-learning instead of evolutionary methods, which were widely used in previous research.	[41,42]
2003	A biologically plausible meta-reinforcement learning algorithm was proposed to tune the parameters of the meta-learning model dynamically and adaptively.	[43]
2004	A new perspective of meta-learning was proposed: exploring the interaction between the learning mechanism and the specific contexts to which the mechanism applies.	[9]
2008	The zero-data learning problem was addressed.	[44]
2010–2012	The breakthrough of deep neural networks marks the beginning of the era of meta-learning.	[45–47]
2013	The relationship between transfer learning and meta-learning was described.	[48]
2016	A meta-learning algorithm named gradient descent by gradient descent was proposed.	[49]
2017	(1) MAML was proposed. (2) A doctoral thesis systematically introduced the concept of meta-learning and corresponding methods.	[50,51]
2018	Reptile, an improved version of MAML, was proposed.	[52]
2019	The Capsule network provides a new method to improve the learning capacity of meta-learning, especially in computer vision.	[53]
2020	Combining auto-encoder and capsule network to focus on the zero-shot learning problem.	[54]

Source: TMa, Yao, Shilin Zhao, Weixiao Wang, Yaoman Li, and Irwin King. "Multimodality in meta-learning: A comprehensive survey." Knowledge-Based Systems (2022): 108976.

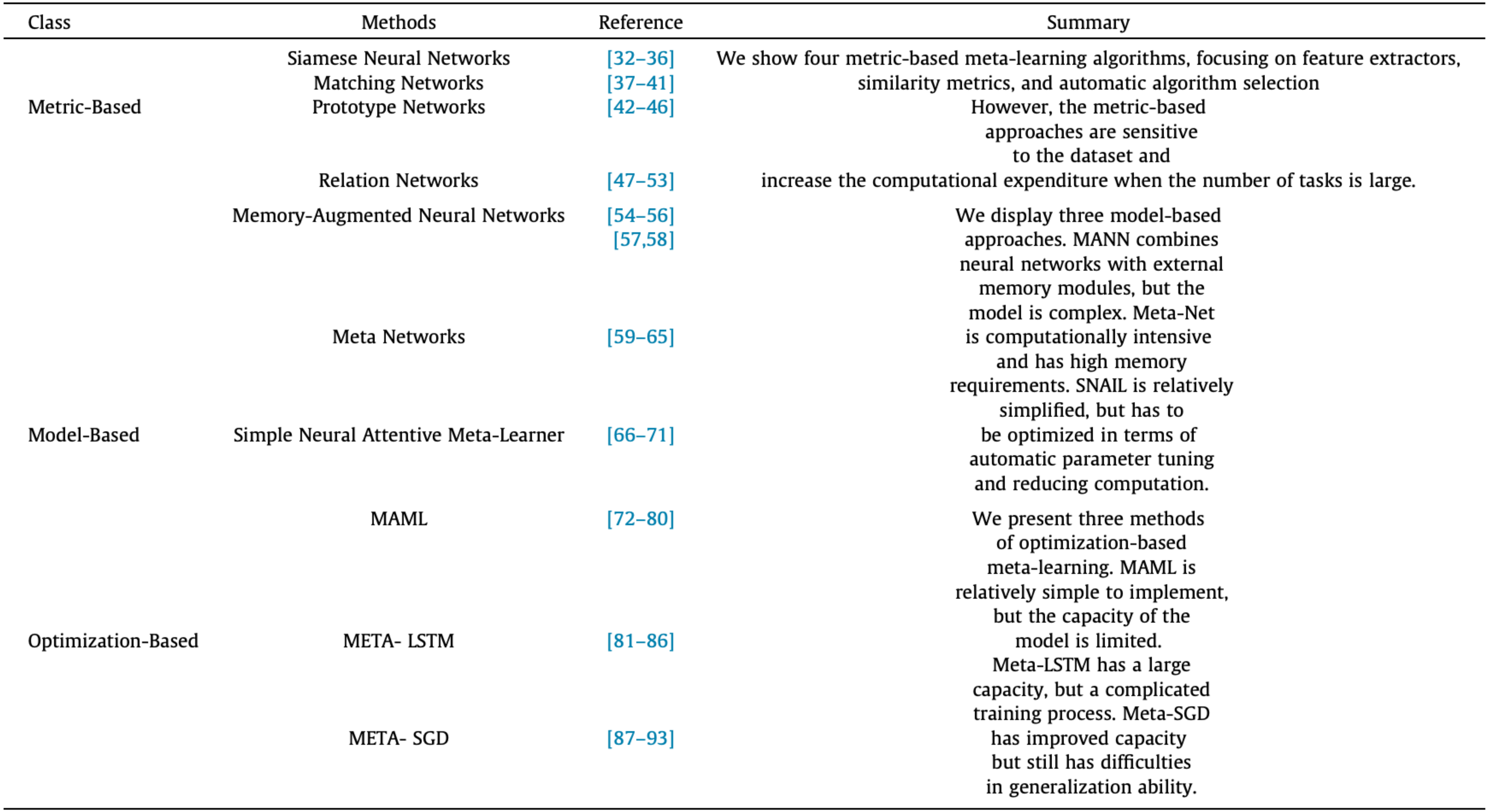
Meta-learning Approaches



	Metric-based	**Optimization-based**	**Model-based**
Key idea	Metric Learning [19]	Gradient Descent	Memory; RNN
How $P_\theta(y\|x)$ **is modeled?**	$\sum_{(x_k,y_k)\in S} k_\theta(x, x_k)y_k,$	$P_{\theta'}(y\|x),$ where $\theta' = g_\phi(\theta, S)$	$f_\theta(x, S).$
Advantages	Faster Inference. Easy to deploy.	Offers flexibility to optimize in dynamic environments. S can be discarded post-optimization.	Faster inference with memory models. Eliminates the need for defining a metric or optimizing at test.
Disadvantages	Less adaptive to optimization in dynamic environments. Computational complexity grows linearly with size of S at test.	Optimization at inference is undesirable for real-world deployment. Prone to overfitting.	Less efficient to hold data in memory as S grows. Hard to design.

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Meta Learning: Learning to Learn

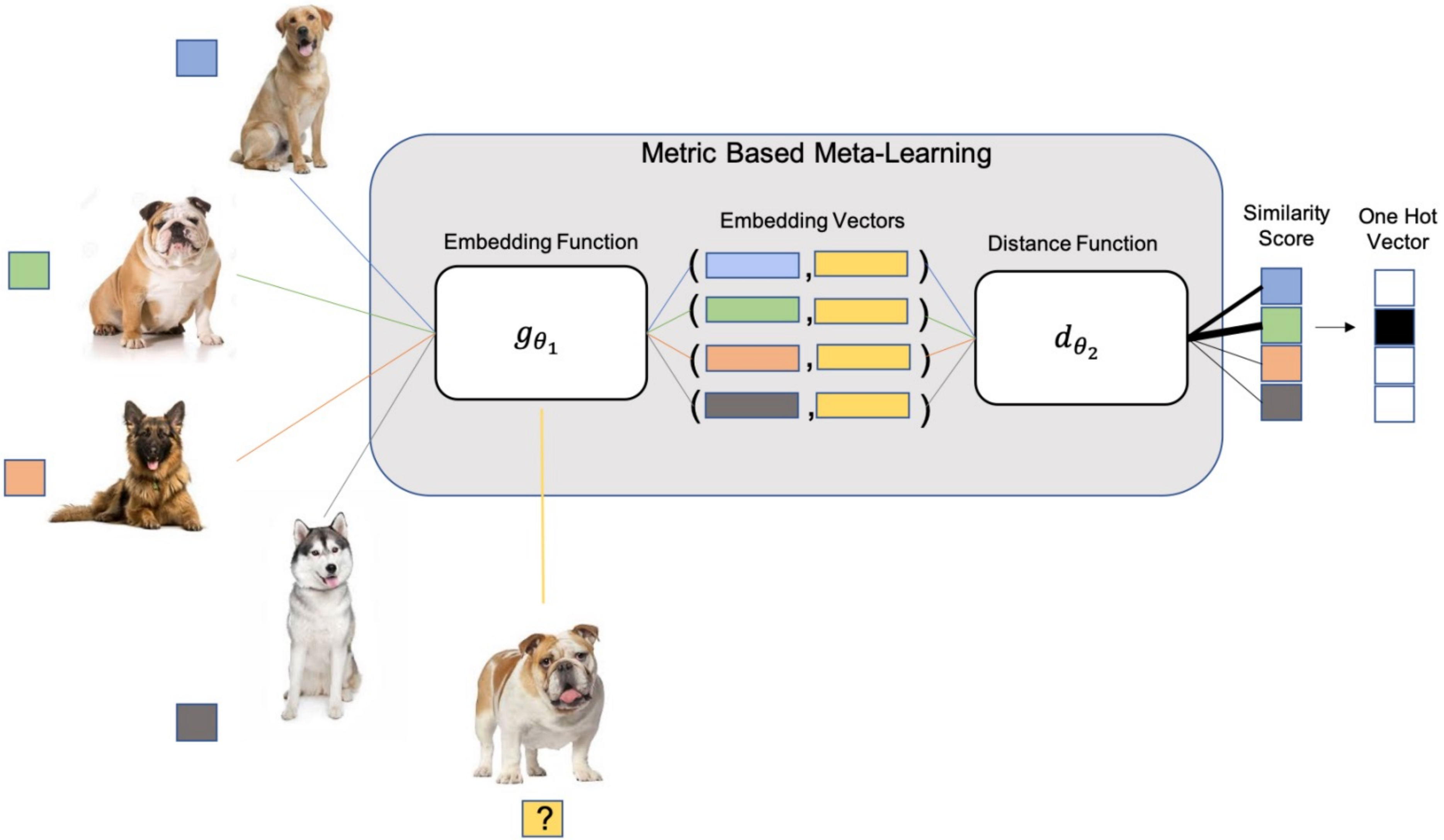


Class	Methods	Reference	Summary
Metric-Based	Siamese Neural Networks	[32–36]	We show four metric-based meta-learning algorithms, focusing on feature extractors, similarity metrics, and automatic algorithm selection However, the metric-based approaches are sensitive to the dataset and increase the computational expenditure when the number of tasks is large.
	Matching Networks	[37–41]	
	Prototype Networks	[42–46]	
	Relation Networks	[47–53]	
Model-Based	Memory-Augmented Neural Networks	[54–56] [57,58]	We display three model-based approaches. MANN combines neural networks with external memory modules, but the model is complex. Meta-Net is computationally intensive and has high memory requirements. SNAIL is relatively simplified, but has to be optimized in terms of automatic parameter tuning and reducing computation.
	Meta Networks	[59–65]	
	Simple Neural Attentive Meta-Learner	[66–71]	
Optimization-Based	MAML	[72–80]	We present three methods of optimization-based meta-learning. MAML is relatively simple to implement, but the capacity of the model is limited. Meta-LSTM has a large capacity, but a complicated training process. Meta-SGD has improved capacity but still has difficulties in generalization ability.
	META- LSTM	[81–86]	
	META- SGD	[87–93]	

Source: Tian, Yingjie, Xiaoxi Zhao, and Wei Huang. "Meta-learning approaches for learning-to-learn in deep learning: A survey." Neurocomputing 494 (2022): 203-223.

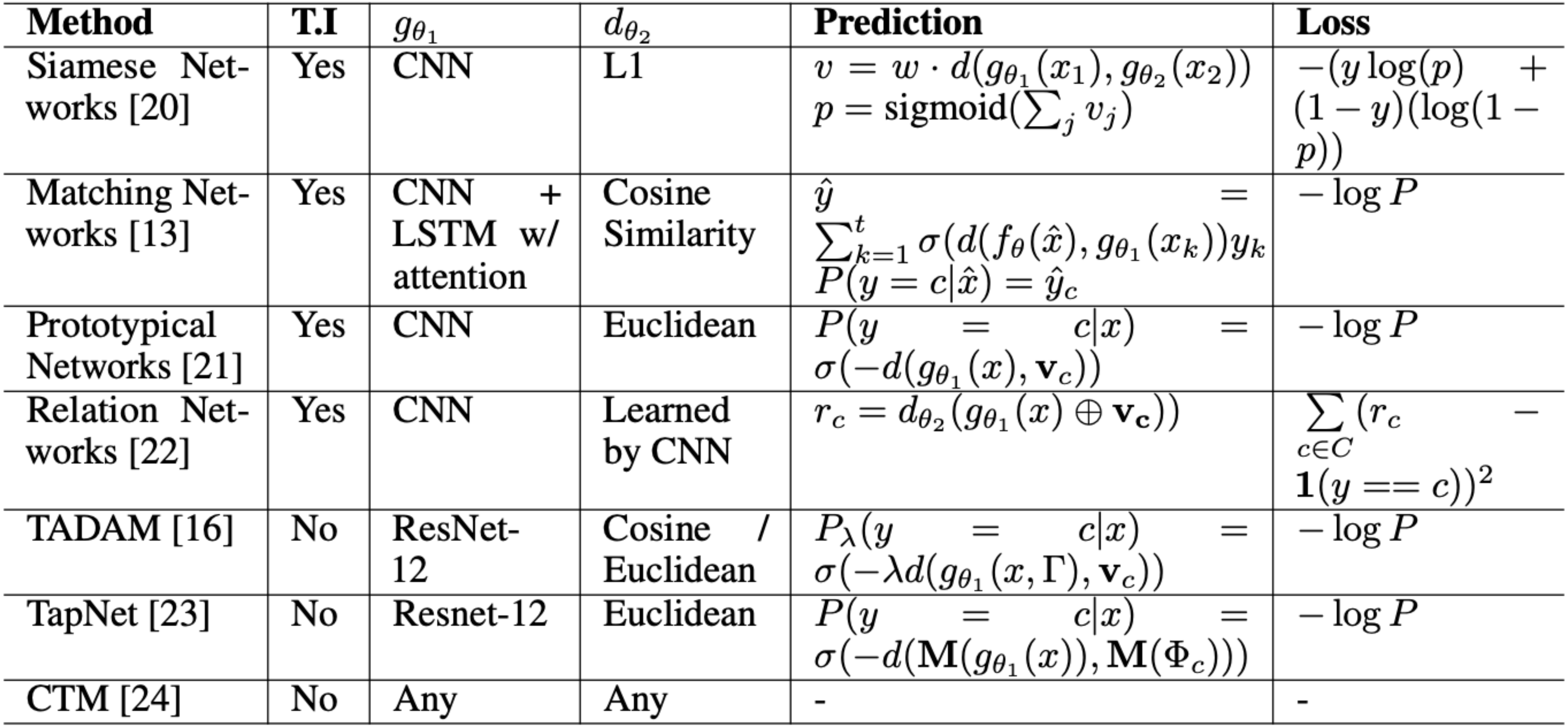
Metric-based Meta-learning

M-Way K-Shot Task (4-way-1-shot classification task)



Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

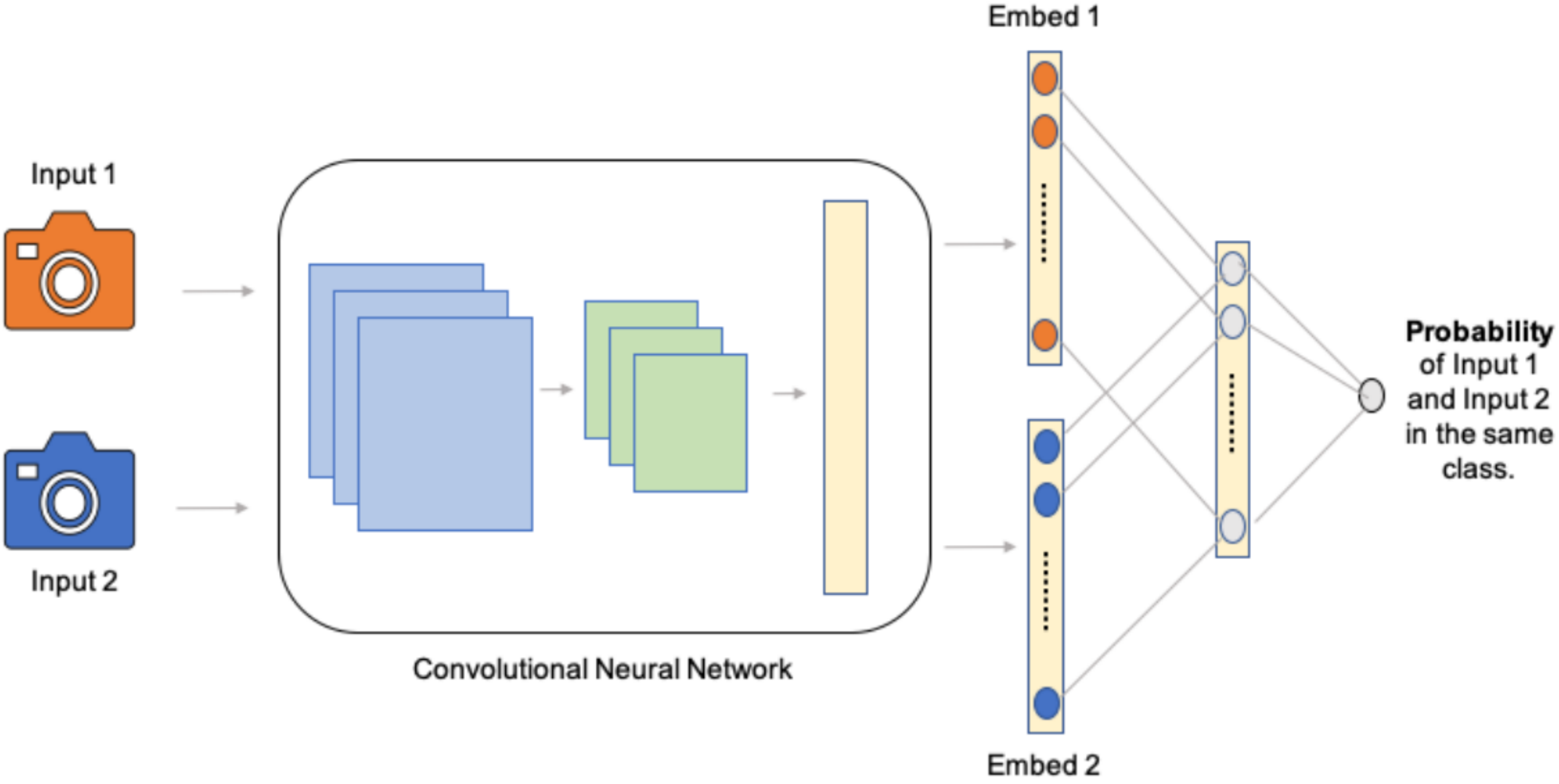
Metric-based Meta-Learning Methods



Method	**T.I**	g_{θ_1}	d_{θ_2}	**Prediction**	**Loss**
Siamese Networks [20]	Yes	CNN	L1	$v = w \cdot d(g_{\theta_1}(x_1), g_{\theta_2}(x_2))$ $p = \text{sigmoid}(\sum_j v_j)$	$-(y \log(p) + (1-y)(\log(1-p))$
Matching Networks [13]	Yes	CNN + LSTM w/ attention	Cosine Similarity	$\hat{y} = \sum_{k=1}^{t} \sigma(d(f_\theta(\hat{x}), g_{\theta_1}(x_k))y_k$ $P(y = c\|\hat{x}) = \hat{y}_c$	$-\log P$
Prototypical Networks [21]	Yes	CNN	Euclidean	$P(y = c\|x) = \sigma(-d(g_{\theta_1}(x), \mathbf{v}_c))$	$-\log P$
Relation Networks [22]	Yes	CNN	Learned by CNN	$r_c = d_{\theta_2}(g_{\theta_1}(x) \oplus \mathbf{v_c}))$	$\sum_{c \in C}(r_c - \mathbf{1}(y == c))^2$
TADAM [16]	No	ResNet-12	Cosine / Euclidean	$P_\lambda(y = c\|x) = \sigma(-\lambda d(g_{\theta_1}(x, \Gamma), \mathbf{v}_c))$	$-\log P$
TapNet [23]	No	Resnet-12	Euclidean	$P(y = c\|x) = \sigma(-d(\mathbf{M}(g_{\theta_1}(x)), \mathbf{M}(\Phi_c)))$	$-\log P$
CTM [24]	No	Any	Any	-	-

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

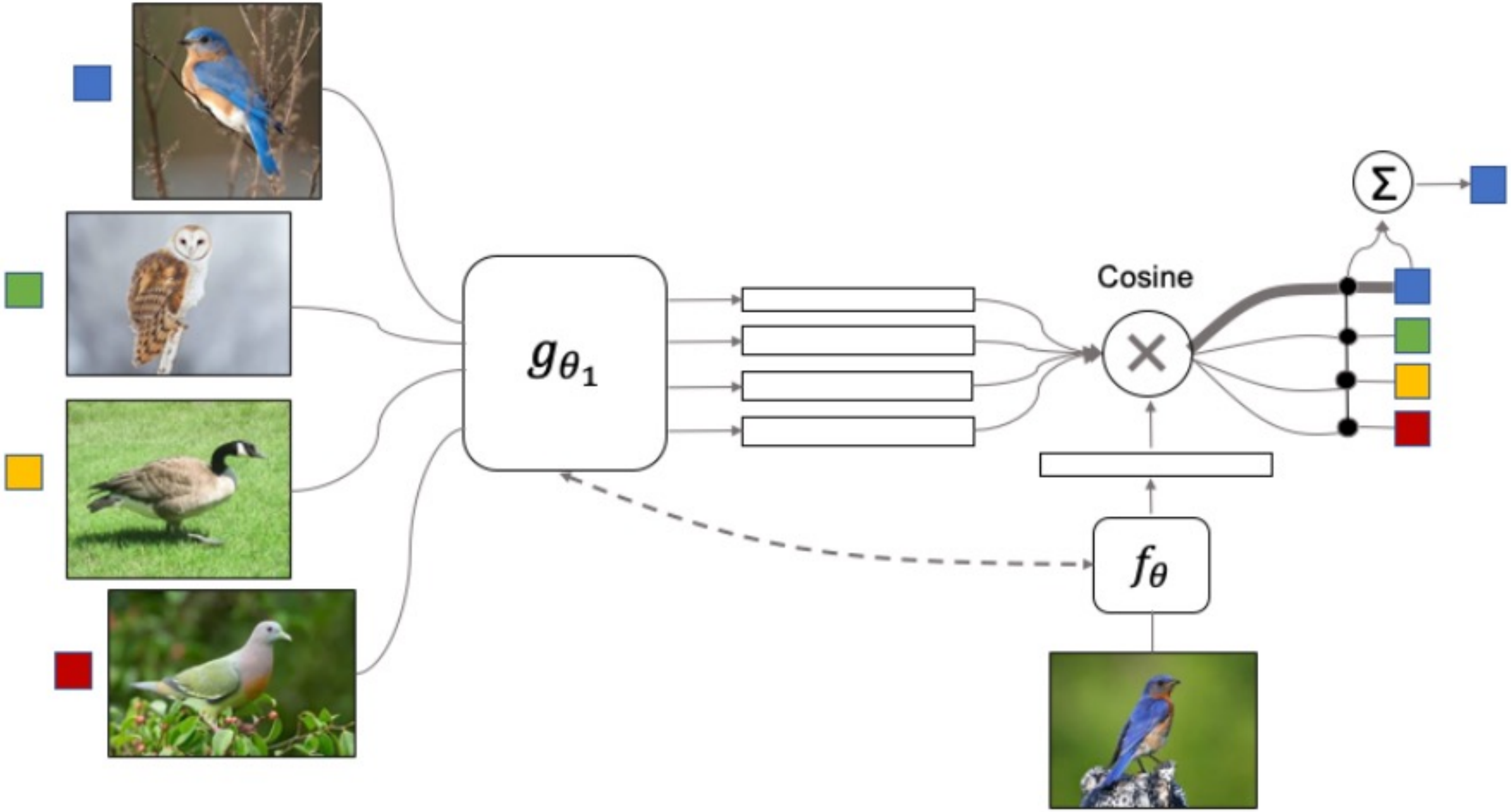
Convolutional Siamese Network



Probability of Input 1 and Input 2 in the same class.

Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

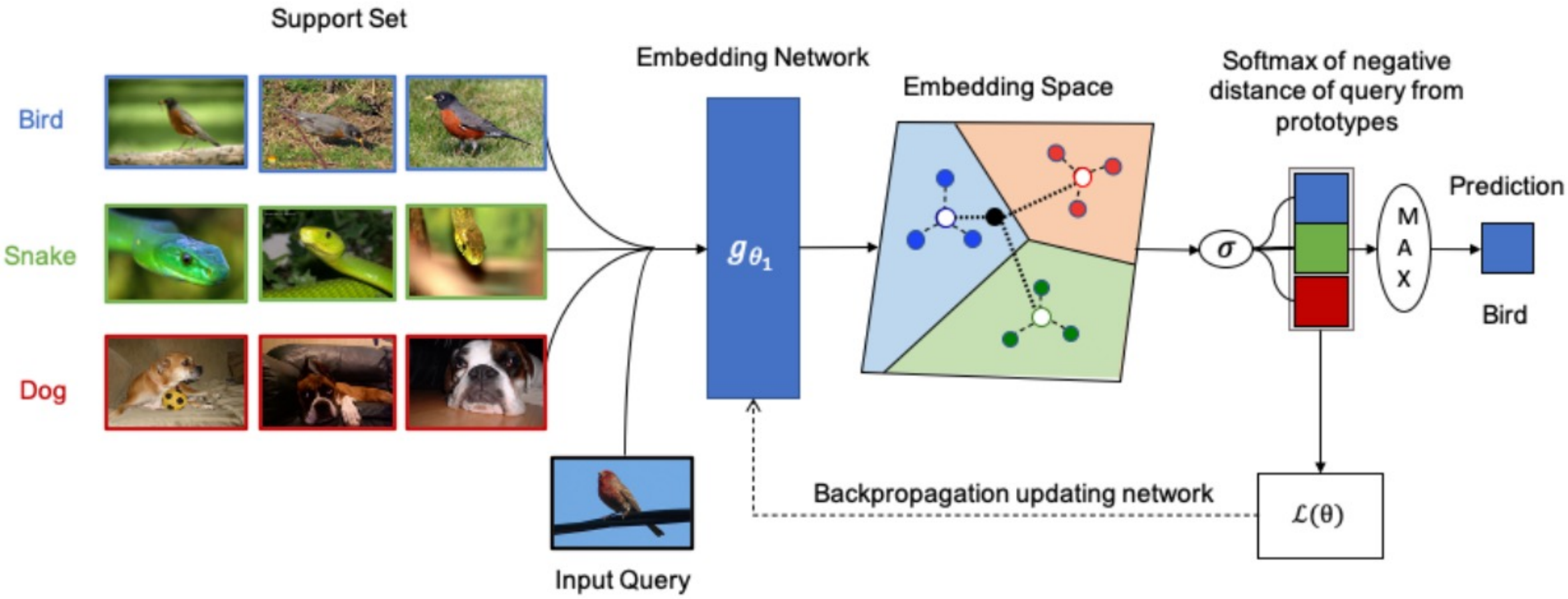
Meta Learning: Matching Networks



Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

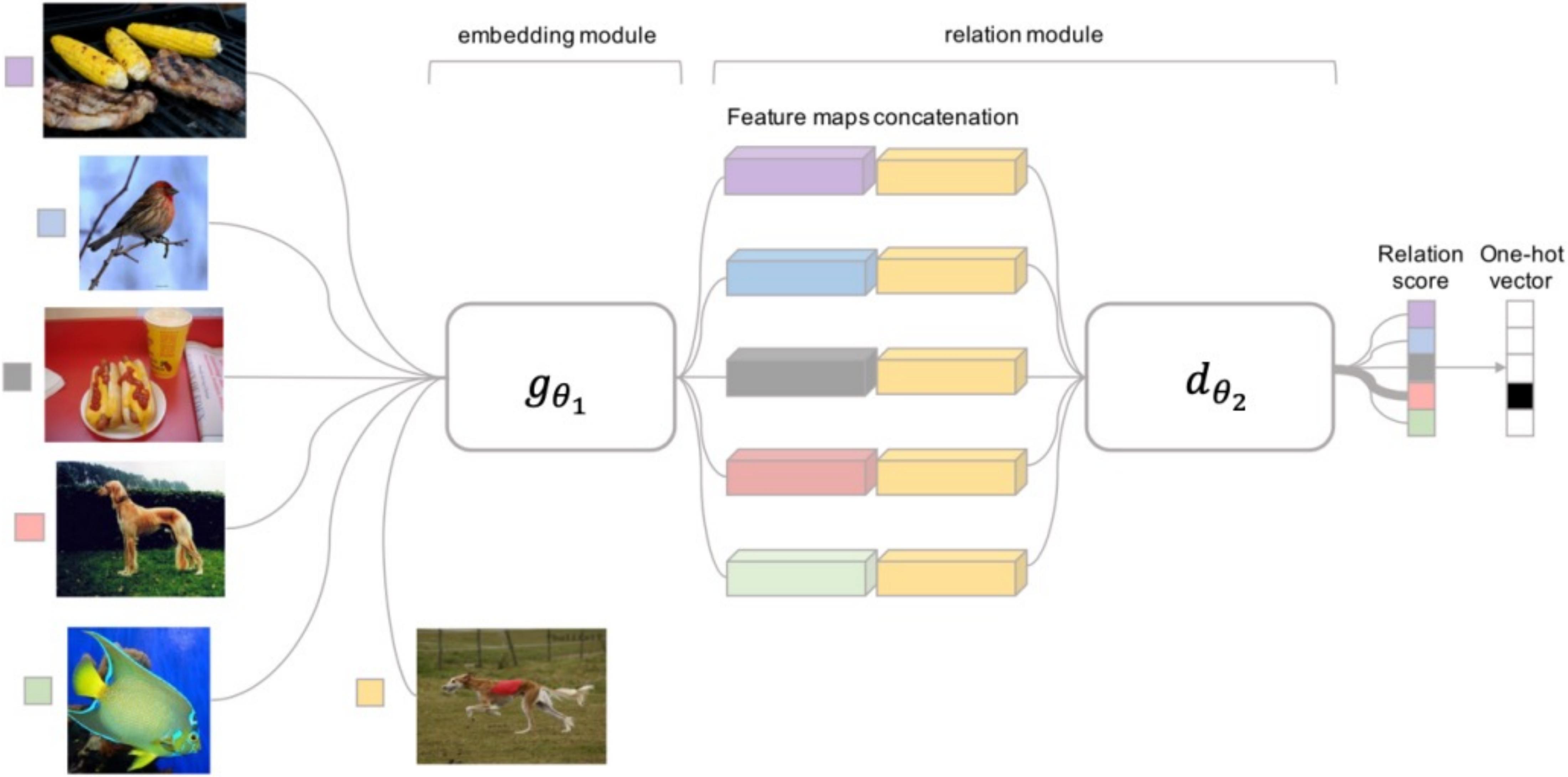
Few-shot Prototypes

v_c are computed as the mean of embedded support examples for each class



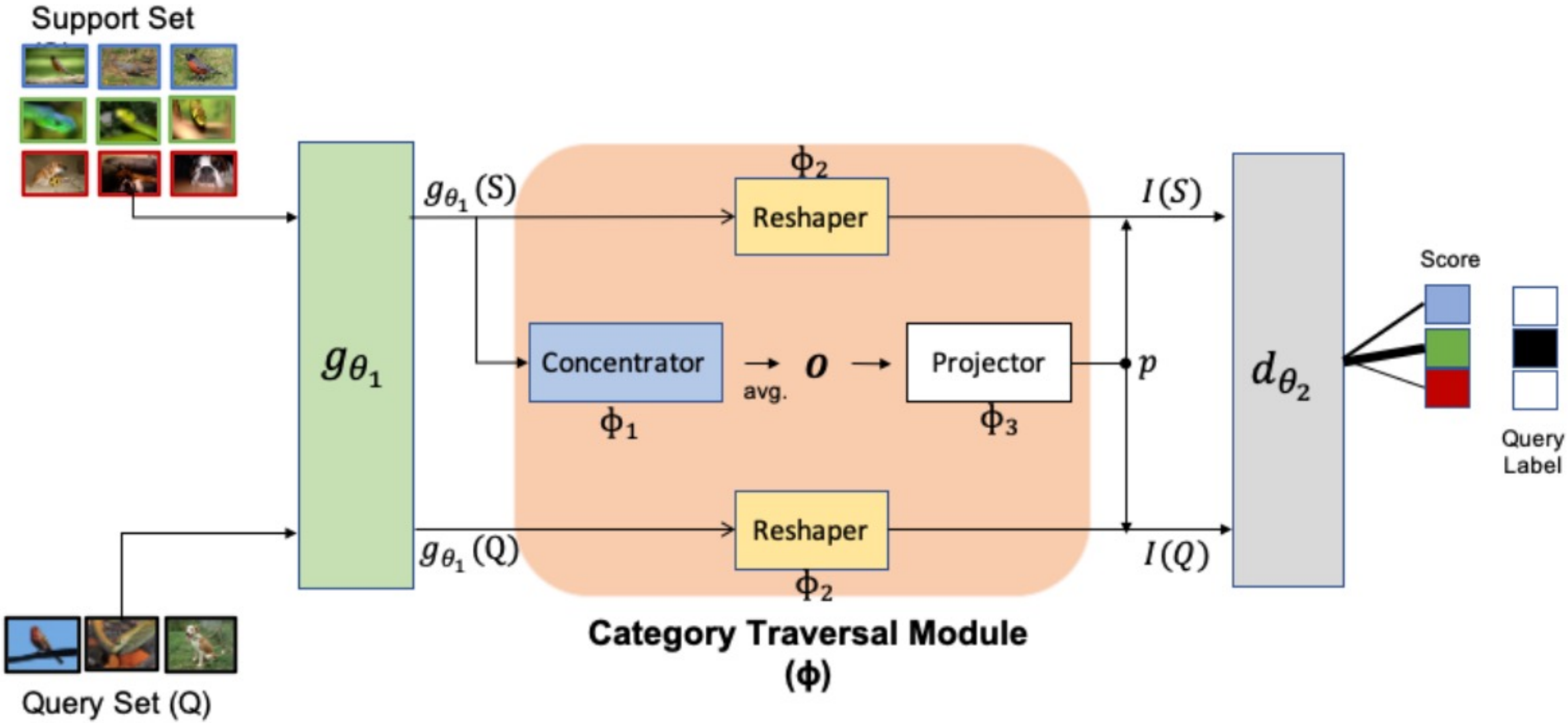
Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Meta Learning: Relation Network



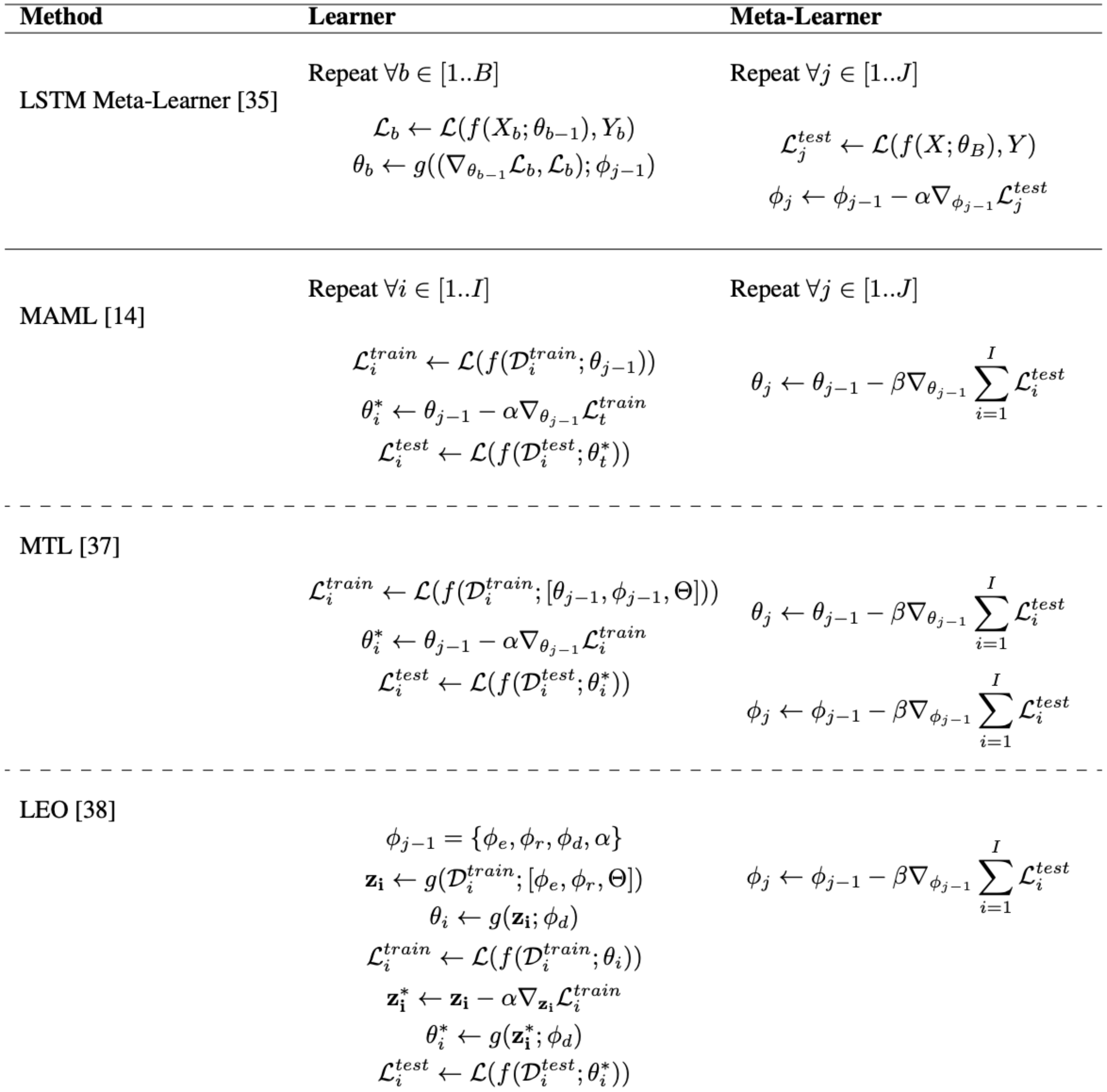
Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Meta Learning: Category Traversal Module (CTM)



Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

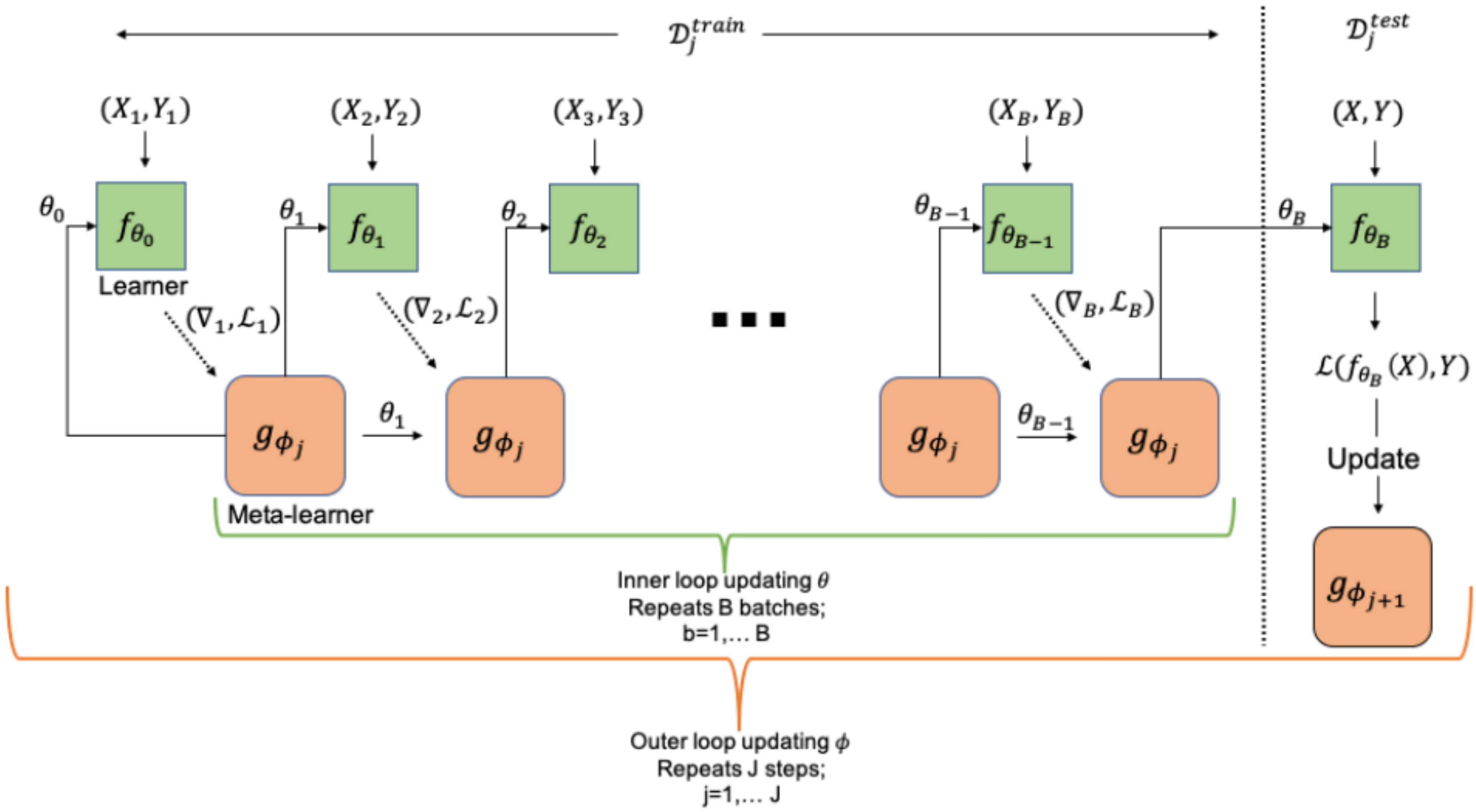
Optimization-based Meta-Learning Methods



Method	Learner	Meta-Learner
LSTM Meta-Learner [35]	Repeat $\forall b \in [1..B]$ $\mathcal{L}_b \leftarrow \mathcal{L}(f(X_b; \theta_{b-1}), Y_b)$ $\theta_b \leftarrow g((\nabla_{\theta_{b-1}} \mathcal{L}_b, \mathcal{L}_b); \phi_{j-1})$	Repeat $\forall j \in [1..J]$ $\mathcal{L}_j^{test} \leftarrow \mathcal{L}(f(X; \theta_B), Y)$ $\phi_j \leftarrow \phi_{j-1} - \alpha \nabla_{\phi_{j-1}} \mathcal{L}_j^{test}$
MAML [14]	Repeat $\forall i \in [1..I]$ $\mathcal{L}_i^{train} \leftarrow \mathcal{L}(f(\mathcal{D}_i^{train}; \theta_{j-1}))$ $\theta_i^* \leftarrow \theta_{j-1} - \alpha \nabla_{\theta_{j-1}} \mathcal{L}_t^{train}$ $\mathcal{L}_i^{test} \leftarrow \mathcal{L}(f(\mathcal{D}_i^{test}; \theta_t^*))$	Repeat $\forall j \in [1..J]$ $\theta_j \leftarrow \theta_{j-1} - \beta \nabla_{\theta_{j-1}} \sum_{i=1}^{I} \mathcal{L}_i^{test}$
MTL [37]	$\mathcal{L}_i^{train} \leftarrow \mathcal{L}(f(\mathcal{D}_i^{train}; [\theta_{j-1}, \phi_{j-1}, \Theta]))$ $\theta_i^* \leftarrow \theta_{j-1} - \alpha \nabla_{\theta_{j-1}} \mathcal{L}_i^{train}$ $\mathcal{L}_i^{test} \leftarrow \mathcal{L}(f(\mathcal{D}_i^{test}; \theta_i^*))$	$\theta_j \leftarrow \theta_{j-1} - \beta \nabla_{\theta_{j-1}} \sum_{i=1}^{I} \mathcal{L}_i^{test}$ $\phi_j \leftarrow \phi_{j-1} - \beta \nabla_{\phi_{j-1}} \sum_{i=1}^{I} \mathcal{L}_i^{test}$
LEO [38]	$\phi_{j-1} = \{\phi_e, \phi_r, \phi_d, \alpha\}$ $\mathbf{z_i} \leftarrow g(\mathcal{D}_i^{train}; [\phi_e, \phi_r, \Theta])$ $\theta_i \leftarrow g(\mathbf{z_i}; \phi_d)$ $\mathcal{L}_i^{train} \leftarrow \mathcal{L}(f(\mathcal{D}_i^{train}; \theta_i))$ $\mathbf{z_i^*} \leftarrow \mathbf{z_i} - \alpha \nabla_{\mathbf{z_i}} \mathcal{L}_i^{train}$ $\theta_i^* \leftarrow g(\mathbf{z_i^*}; \phi_d)$ $\mathcal{L}_i^{test} \leftarrow \mathcal{L}(f(\mathcal{D}_i^{test}; \theta_i^*))$	$\phi_j \leftarrow \phi_{j-1} - \beta \nabla_{\phi_{j-1}} \sum_{i=1}^{I} \mathcal{L}_i^{test}$

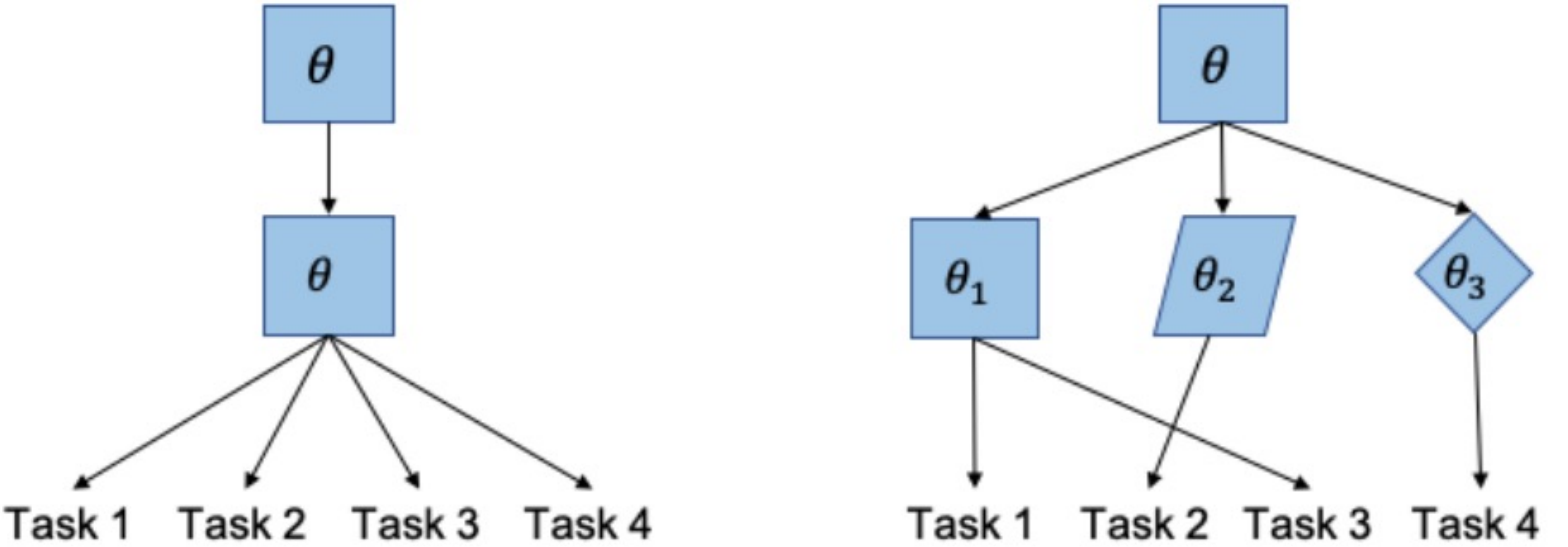
Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Computational Graph for the Forward Pass of the Meta-learner



Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Model-Agnostic Meta-Learning (MAML)
Hierarchically Structured Meta-Learning (HSML)

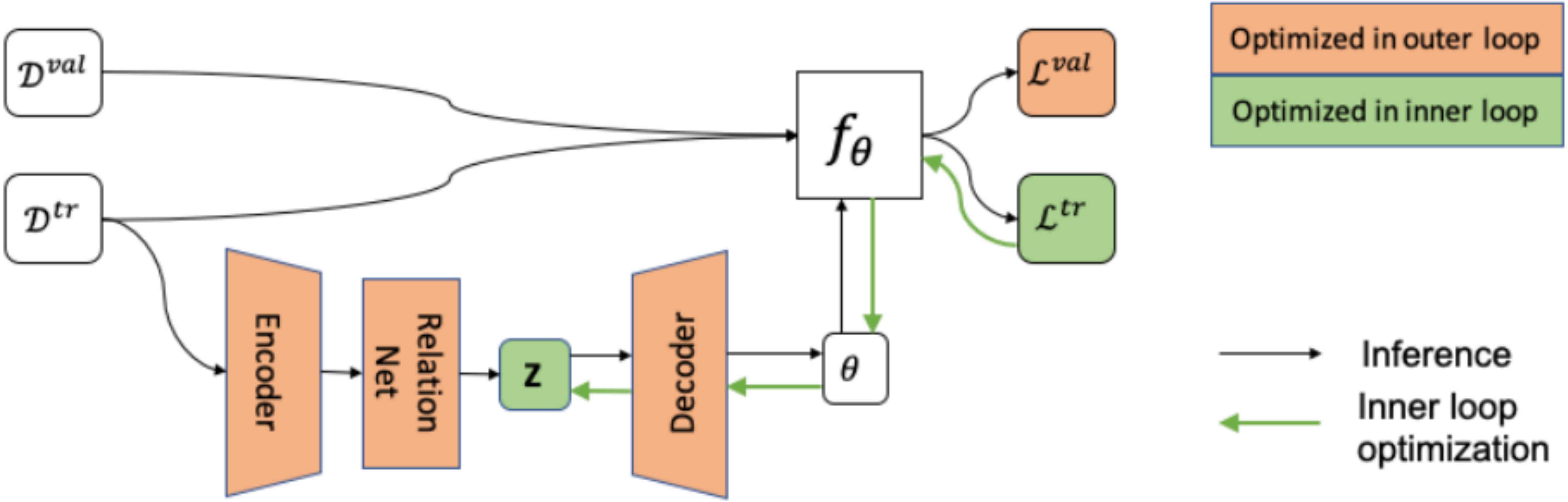


Globally Shared Initialization
(MAML)

Hierarchically Clustered Initialization
(HSML)

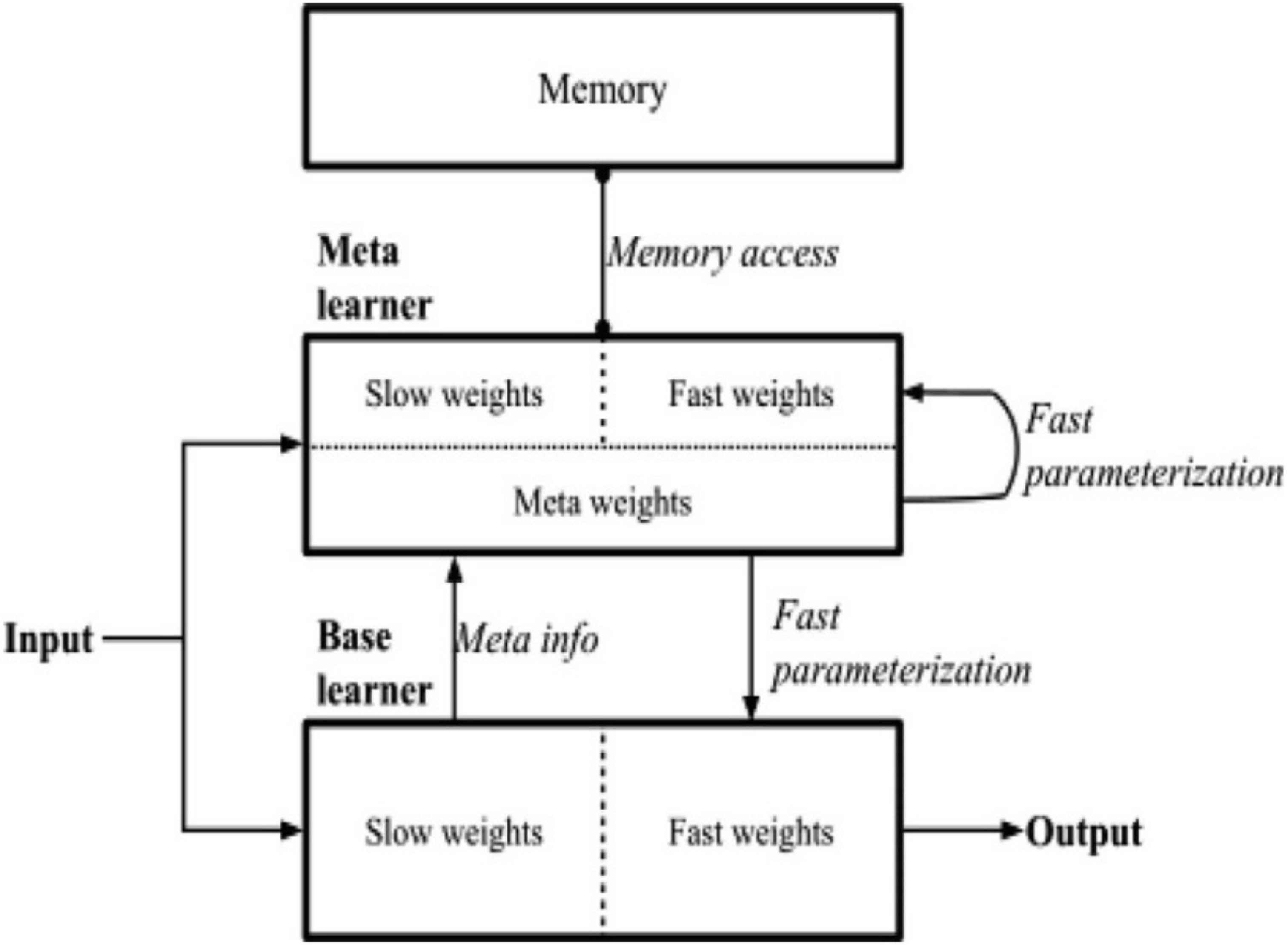
Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Latent Embedding Optimization (LEO)



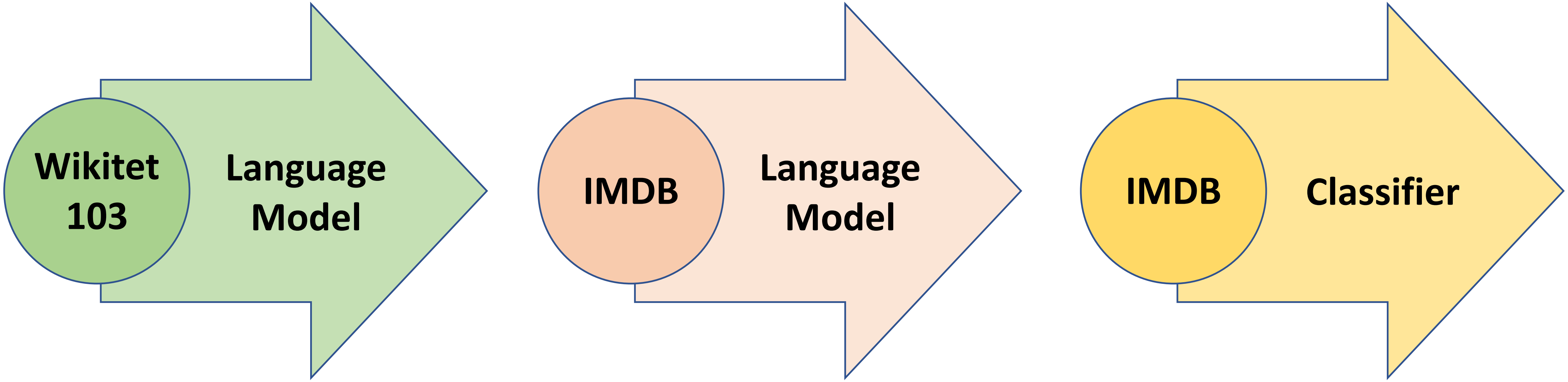
Source: Parnami, Archit, and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning." arXiv preprint arXiv:2203.04291 (2022).

Overall Architecture of Meta Networks



Source: Tian, Yingjie, Xiaoxi Zhao, and Wei Huang. "Meta-learning approaches for learning-to-learn in deep learning: A survey." Neurocomputing 494 (2022): 203-223.

ULMFiT: 3 Steps
Transfer Learning in NLP



1. Pretraining

2. Domain adaptation

3. Fine-tuning

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

A typical pipeline for training transformer models

with the Datasets, Tokenizers, and Transformers libraries

Datasets	Tokenizers	Transformers	Datasets
Load and process datasets	Tokenize input texts	Load models, train and infer	Load metrics evaluate models

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Few-Shot Learning (FSL)
Typical Scenarios

- **Acting as a test bed for learning like human**
- **Learning for rare cases**
- **Reducing data gathering effort and computational cost**

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

- **Few-Shot Learning (FSL) is a sub-area in machine learning.**
- **Machine Learning Definition**
  - **A computer program is said to learn from experience E with respect to some classes of task T and performance measure P if its performance can improve with E on T measured by P.**
    - Example: Image classification task (T), a machine learning program can improve its classification accuracy (P) through E obtained by training on a large number of labeled images (e.g., the ImageNet data set).

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Machine Learning

task T	experience E	performance P
image classification [73]	large-scale labeled images for each class	classification accuracy
the ancient game of Go [120]	a database containing around 30 million recorded moves of human experts and self-play records	winning rate

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

- **Few-shot Learning (FSL) is a type of machine learning problems (specified by E, T, and P),**
  where E contains only a limited number of examples with supervised information for the target T.
  - **Existing FSL problems are mainly supervised learning problems.**
  - **Few-shot classification learns classifiers given only a few labeled examples of each class.**
    - image classification
    - sentiment classification from short text
    - object recognition

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

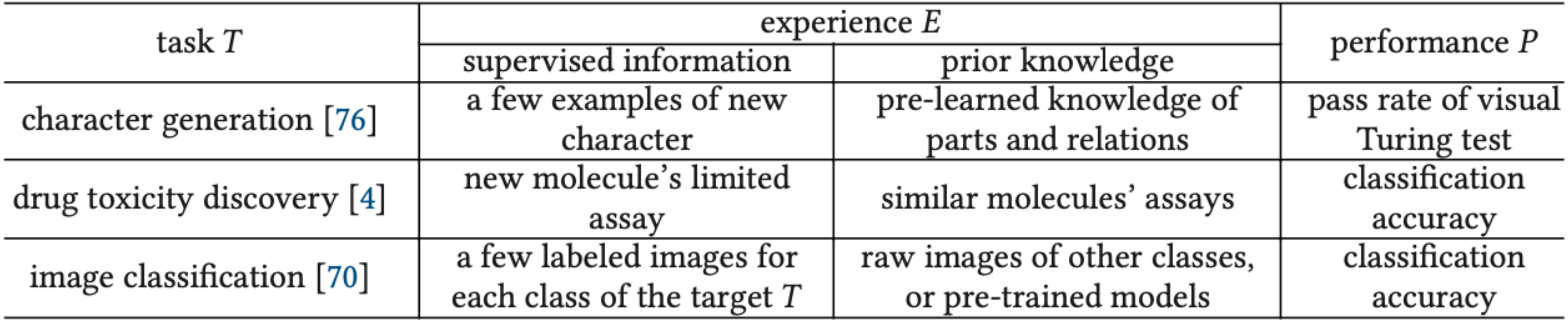
- **Few-shot classification learns a classifier h, which predicts label y_i for each input x_i.**
- **Usually, one considers the *N-way-K-shot* classification, in which D_{train} contains $I = KN$ examples from N classes each with K examples**

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

- **Few-Shot Learning (FSL)**
  - $K = 10 \sim 100$ **examples**
- **One-Shot Learning (1SL)**
  - $K = 1$ **example**
- **Zero-Shot Learning (0SL)(ZSL)**
  - $K = 0$

Few-Shot Learning (FSL)

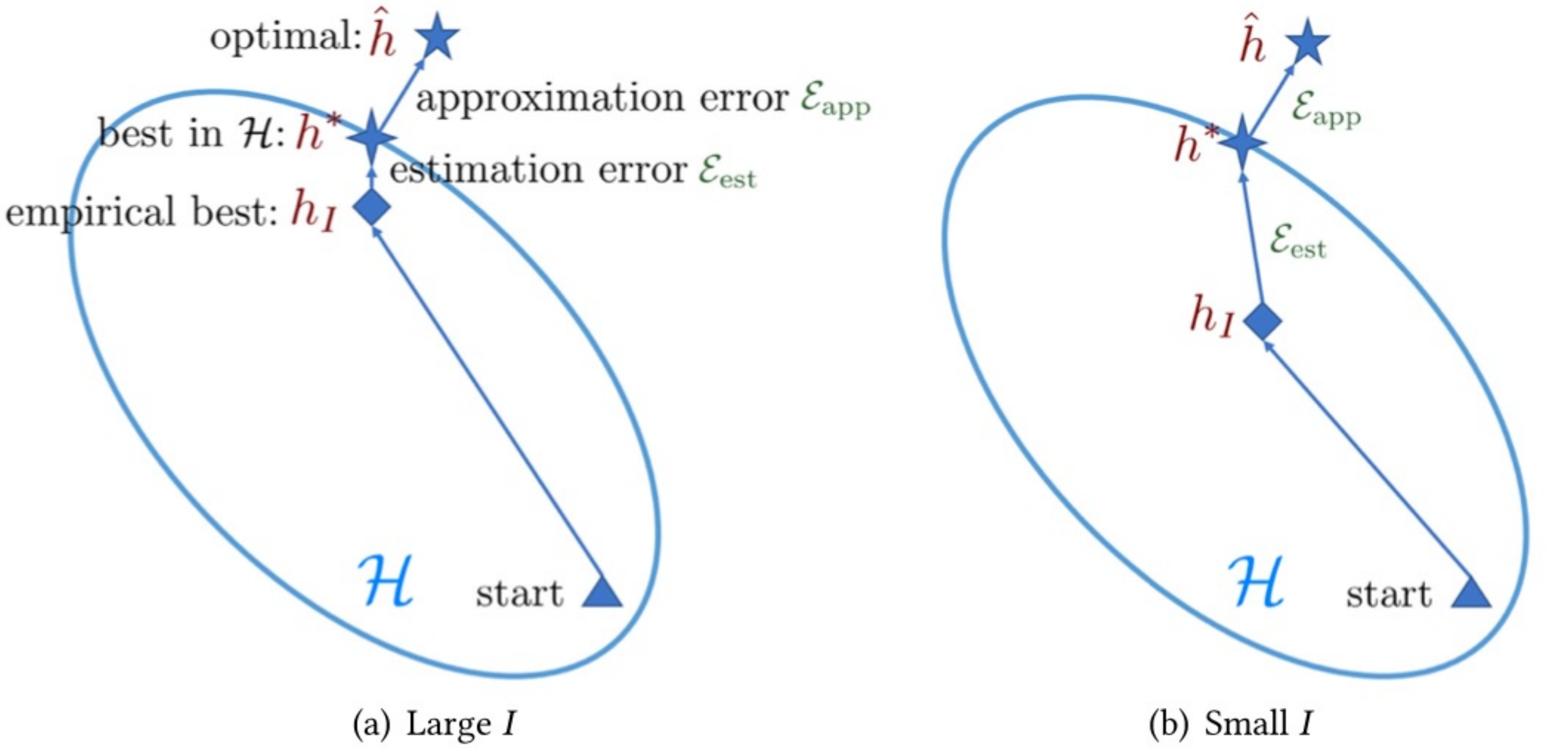


task T	experience E		performance P
	supervised information	prior knowledge	
character generation [76]	a few examples of new character	pre-learned knowledge of parts and relations	pass rate of visual Turing test
drug toxicity discovery [4]	new molecule's limited assay	similar molecules' assays	classification accuracy
image classification [70]	a few labeled images for each class of the target T	raw images of other classes, or pre-trained models	classification accuracy

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Comparison of learning with sufficient and few training samples



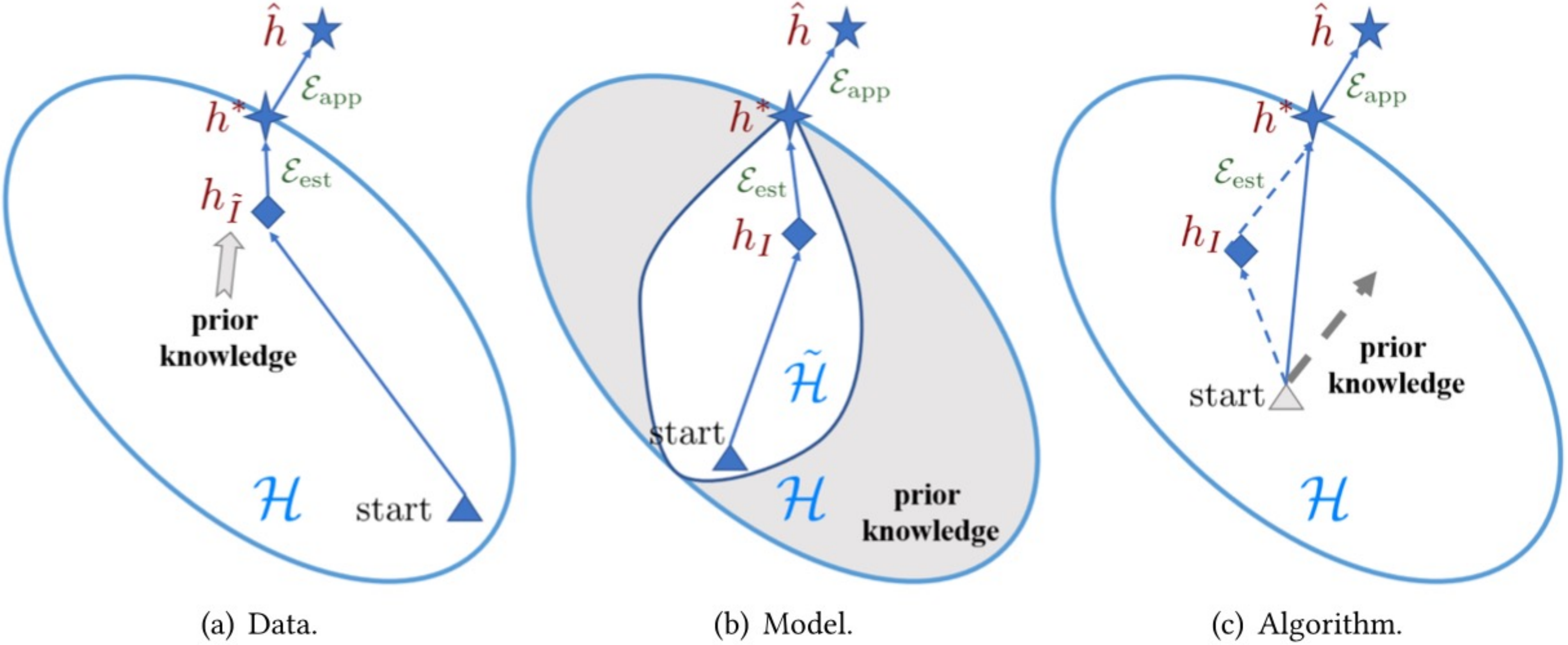
(a) Large I

(b) Small I

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

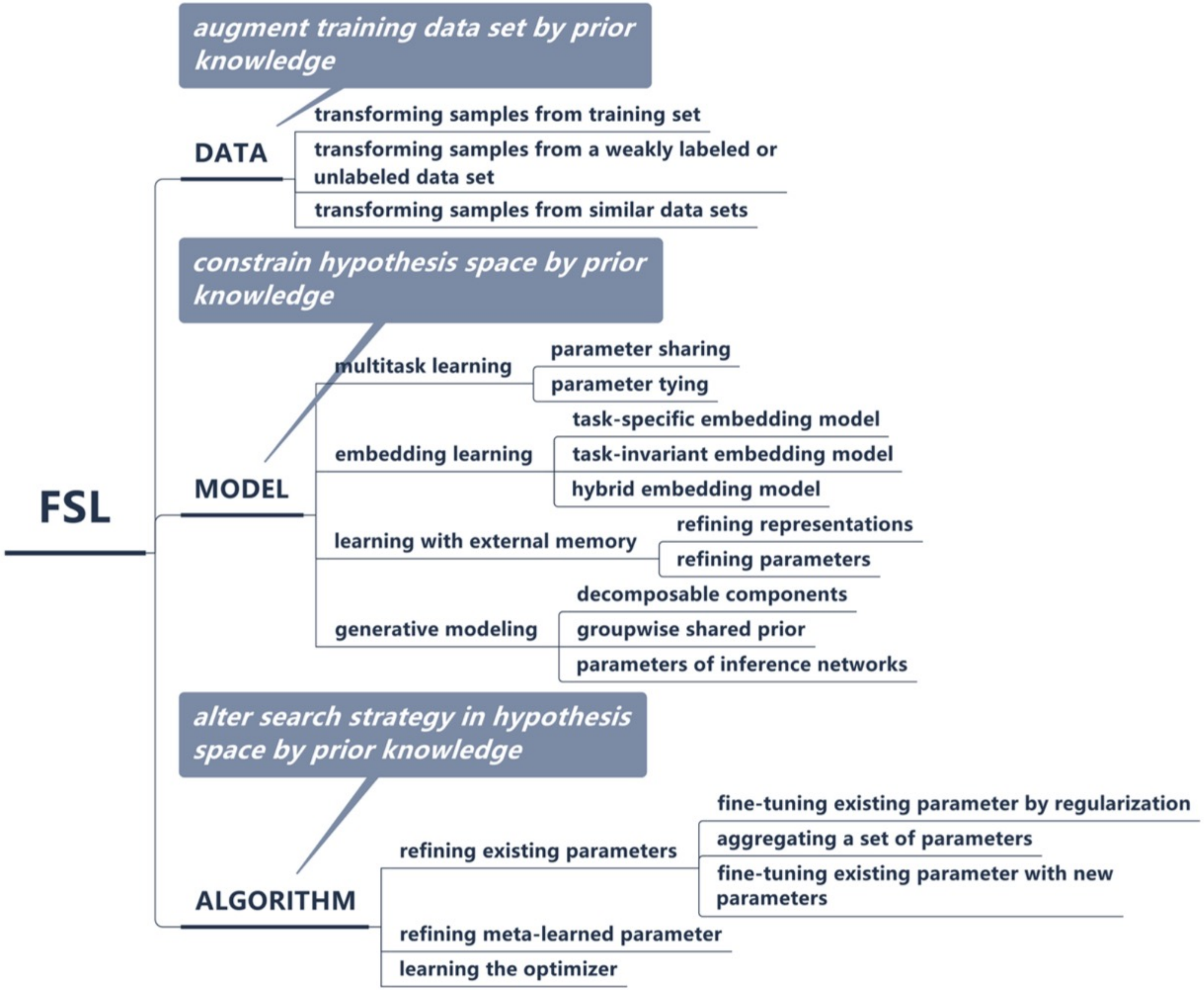
Different perspectives on how FSL methods solve the few-shot problem



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

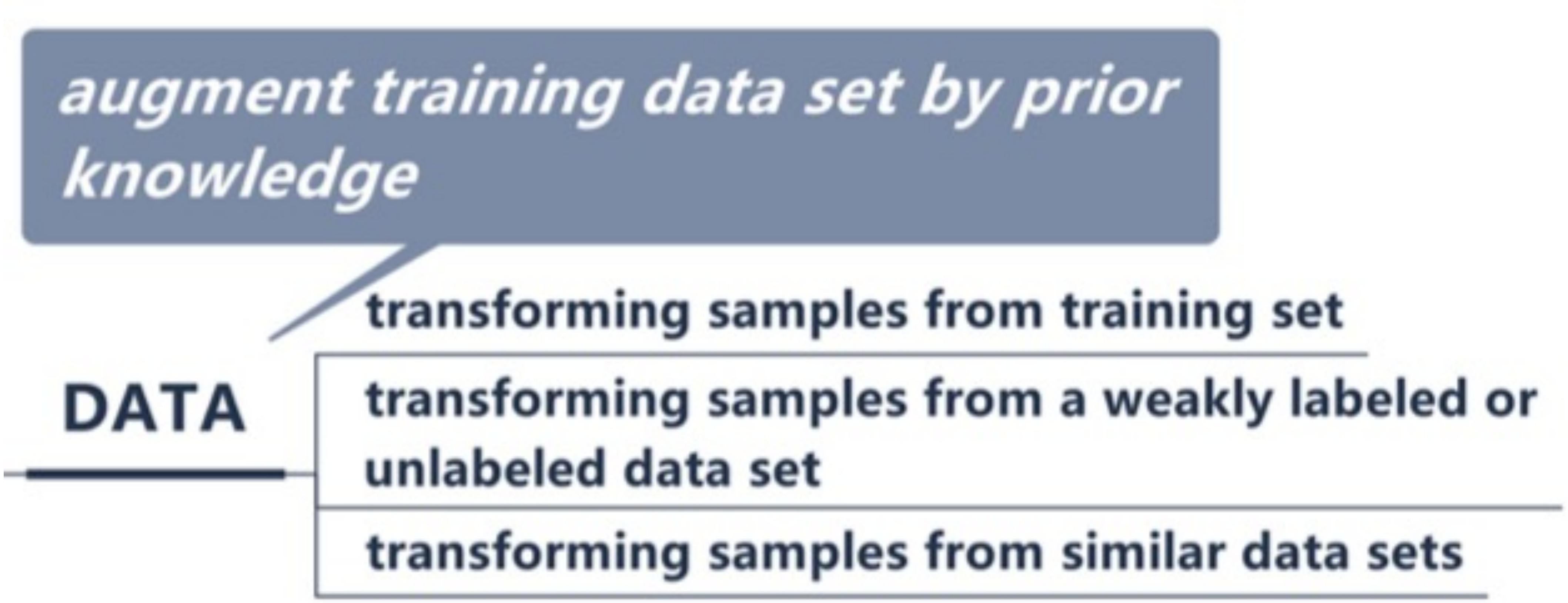
A taxonomy of FSL methods



- **FSL**
  - **DATA** — *augment training data set by prior knowledge*
    - transforming samples from training set
    - transforming samples from a weakly labeled or unlabeled data set
    - transforming samples from similar data sets
  - **MODEL** — *constrain hypothesis space by prior knowledge*
    - multitask learning
      - parameter sharing
      - parameter tying
    - embedding learning
      - task-specific embedding model
      - task-invariant embedding model
      - hybrid embedding model
    - learning with external memory
      - refining representations
      - refining parameters
    - generative modeling
      - decomposable components
      - groupwise shared prior
      - parameters of inference networks
  - **ALGORITHM** — *alter search strategy in hypothesis space by prior knowledge*
    - refining existing parameters
      - fine-tuning existing parameter by regularization
      - aggregating a set of parameters
      - fine-tuning existing parameter with new parameters
    - refining meta-learned parameter
    - learning the optimizer

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)



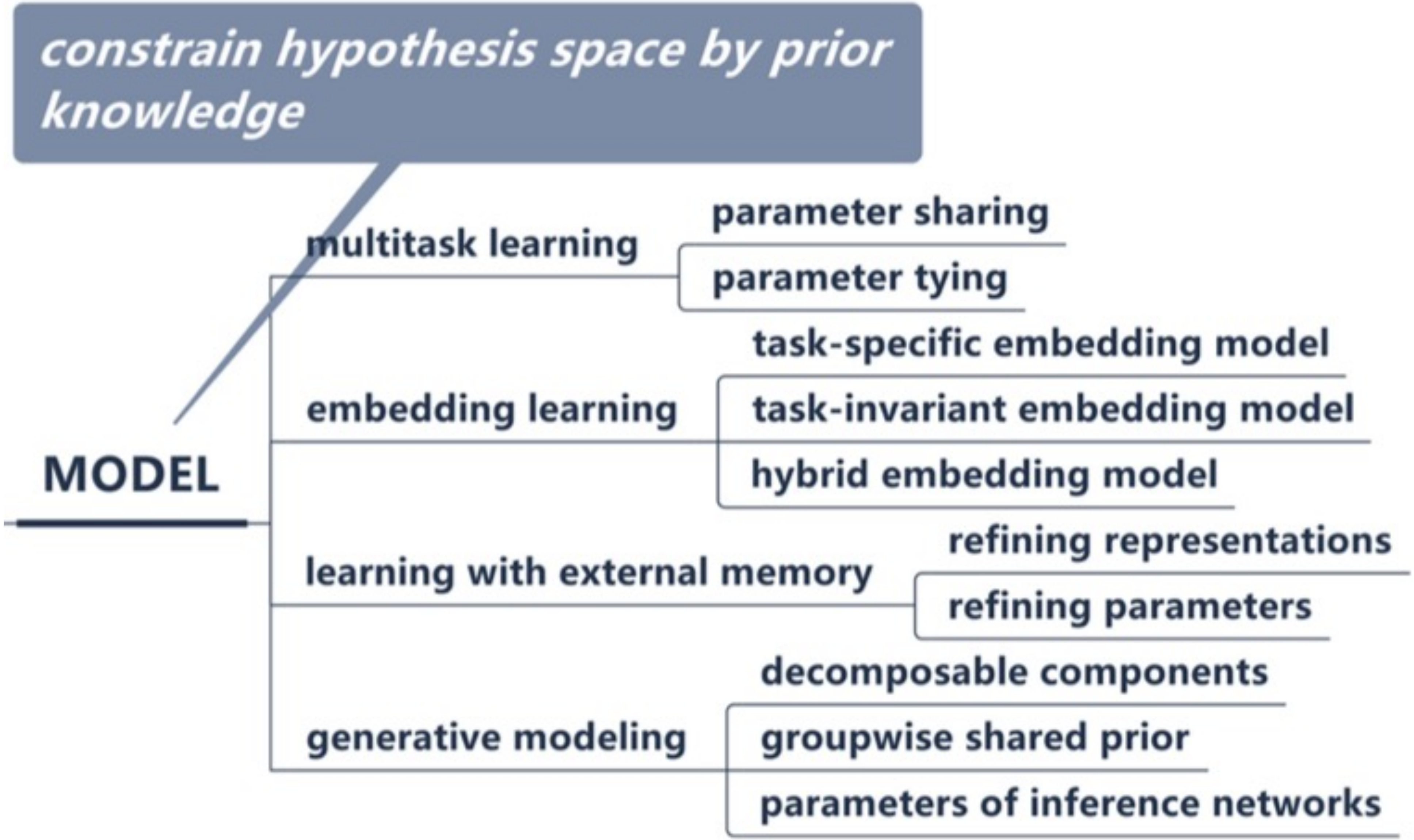
augment training data set by prior knowledge

DATA

- transforming samples from training set
- transforming samples from a weakly labeled or unlabeled data set
- transforming samples from similar data sets

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)



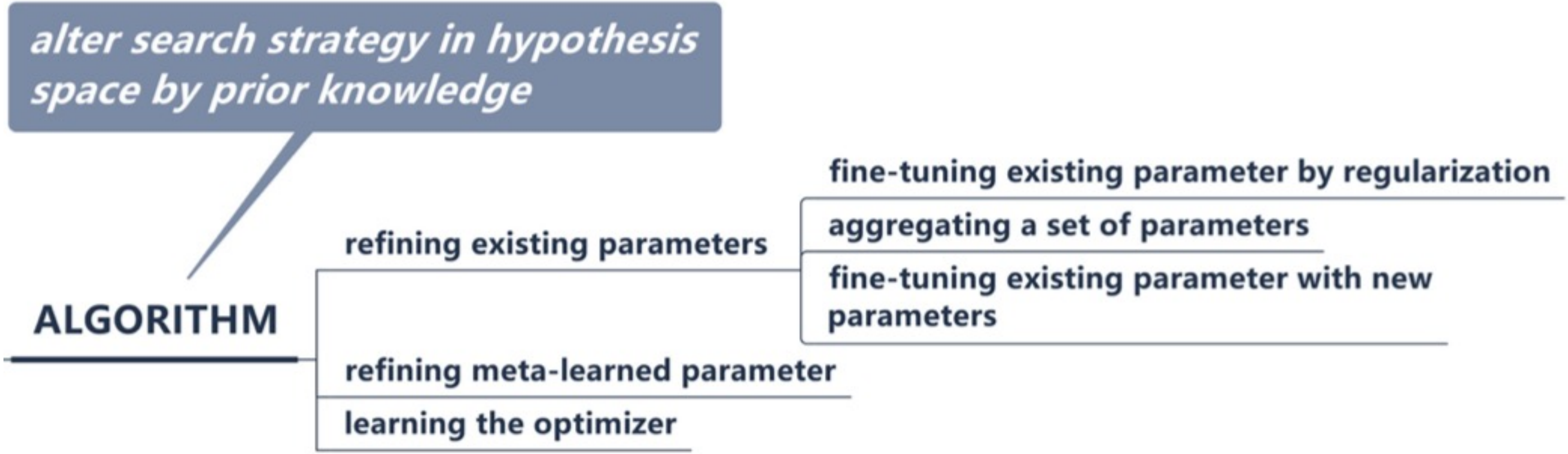
constrain hypothesis space by prior knowledge

MODEL

- multitask learning
  - parameter sharing
  - parameter tying
- embedding learning
  - task-specific embedding model
  - task-invariant embedding model
  - hybrid embedding model
- learning with external memory
  - refining representations
  - refining parameters
- generative modeling
  - decomposable components
  - groupwise shared prior
  - parameters of inference networks

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)



alter search strategy in hypothesis space by prior knowledge

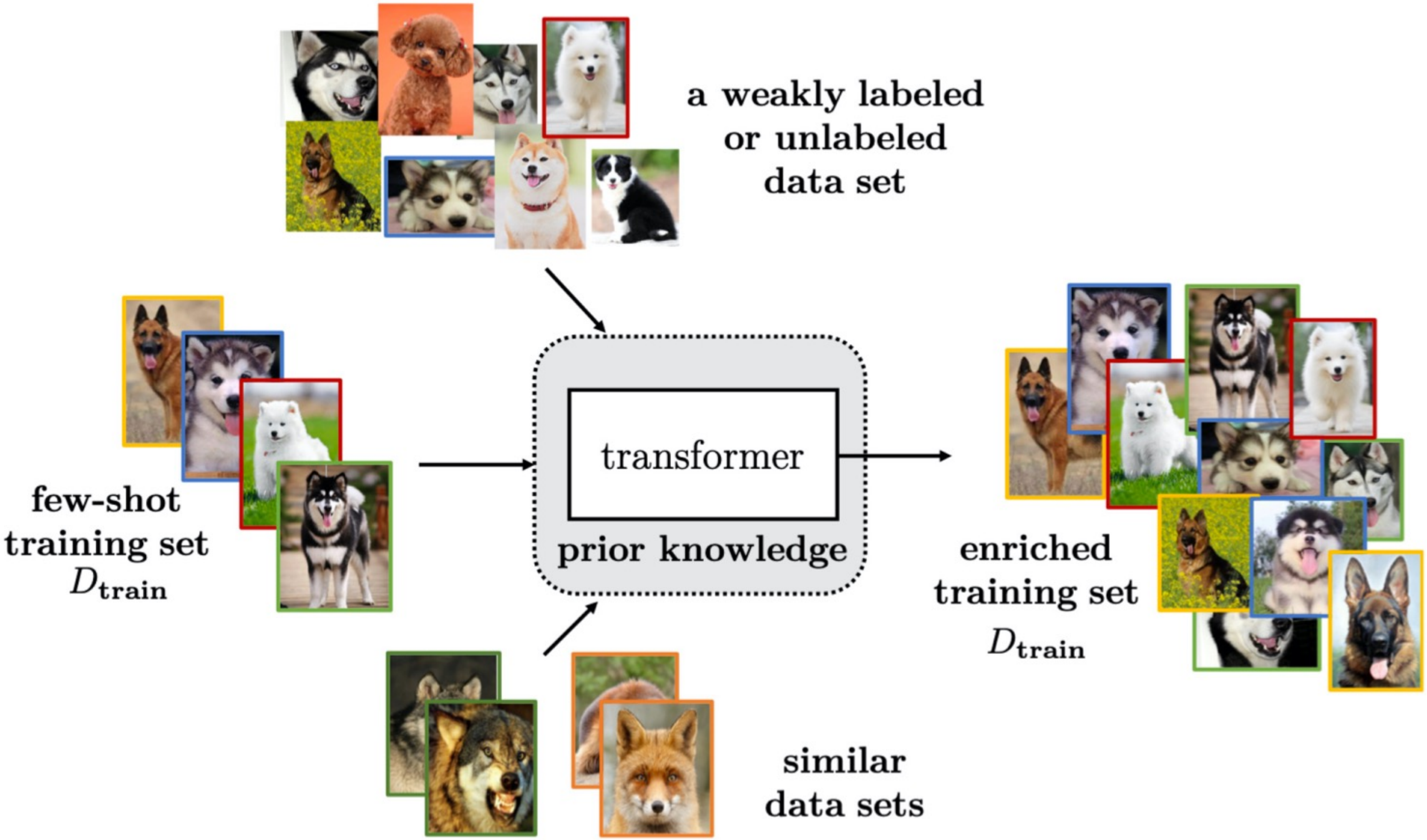
ALGORITHM

- refining existing parameters
  - fine-tuning existing parameter by regularization
  - aggregating a set of parameters
  - fine-tuning existing parameter with new parameters
- refining meta-learned parameter
- learning the optimizer

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

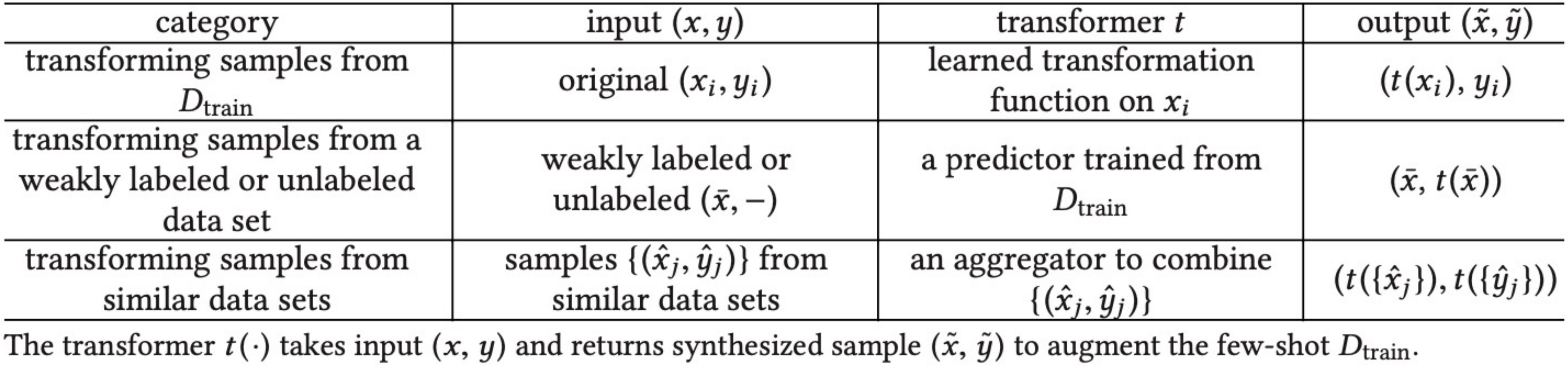
Solving the FSL problem by data augmentation



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Characteristics for FSL Methods Focusing on the Data Perspective



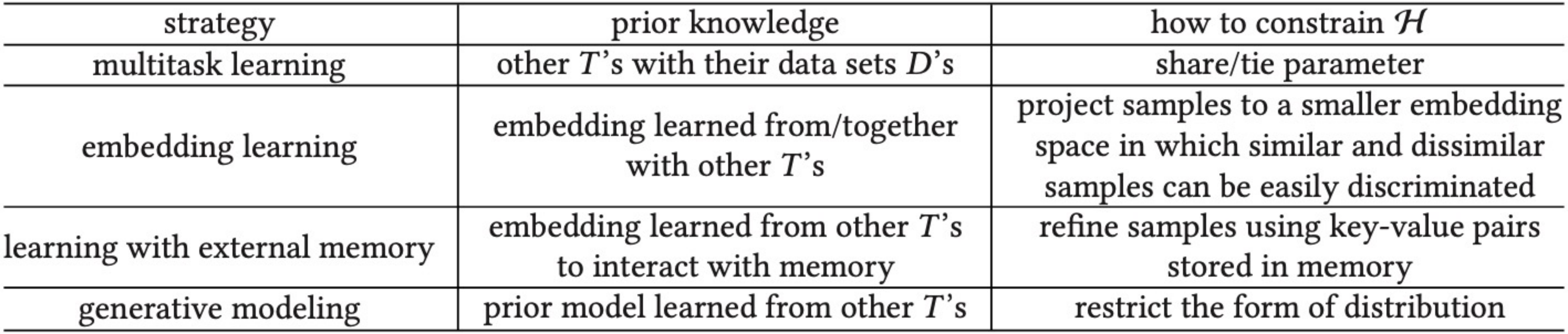
category	input (x, y)	transformer t	output $(\tilde{x}, \tilde{y})$
transforming samples from D_{train}	original (x_i, y_i)	learned transformation function on x_i	$(t(x_i), y_i)$
transforming samples from a weakly labeled or unlabeled data set	weakly labeled or unlabeled $(\bar{x}, -)$	a predictor trained from D_{train}	$(\bar{x}, t(\bar{x}))$
transforming samples from similar data sets	samples $\{(\hat{x}_j, \hat{y}_j)\}$ from similar data sets	an aggregator to combine $\{(\hat{x}_j, \hat{y}_j)\}$	$(t(\{\hat{x}_j\}), t(\{\hat{y}_j\}))$

The transformer $t(\cdot)$ takes input (x, y) and returns synthesized sample $(\tilde{x}, \tilde{y})$ to augment the few-shot D_{train}.

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Characteristics for FSL Methods Focusing on the Model Perspective

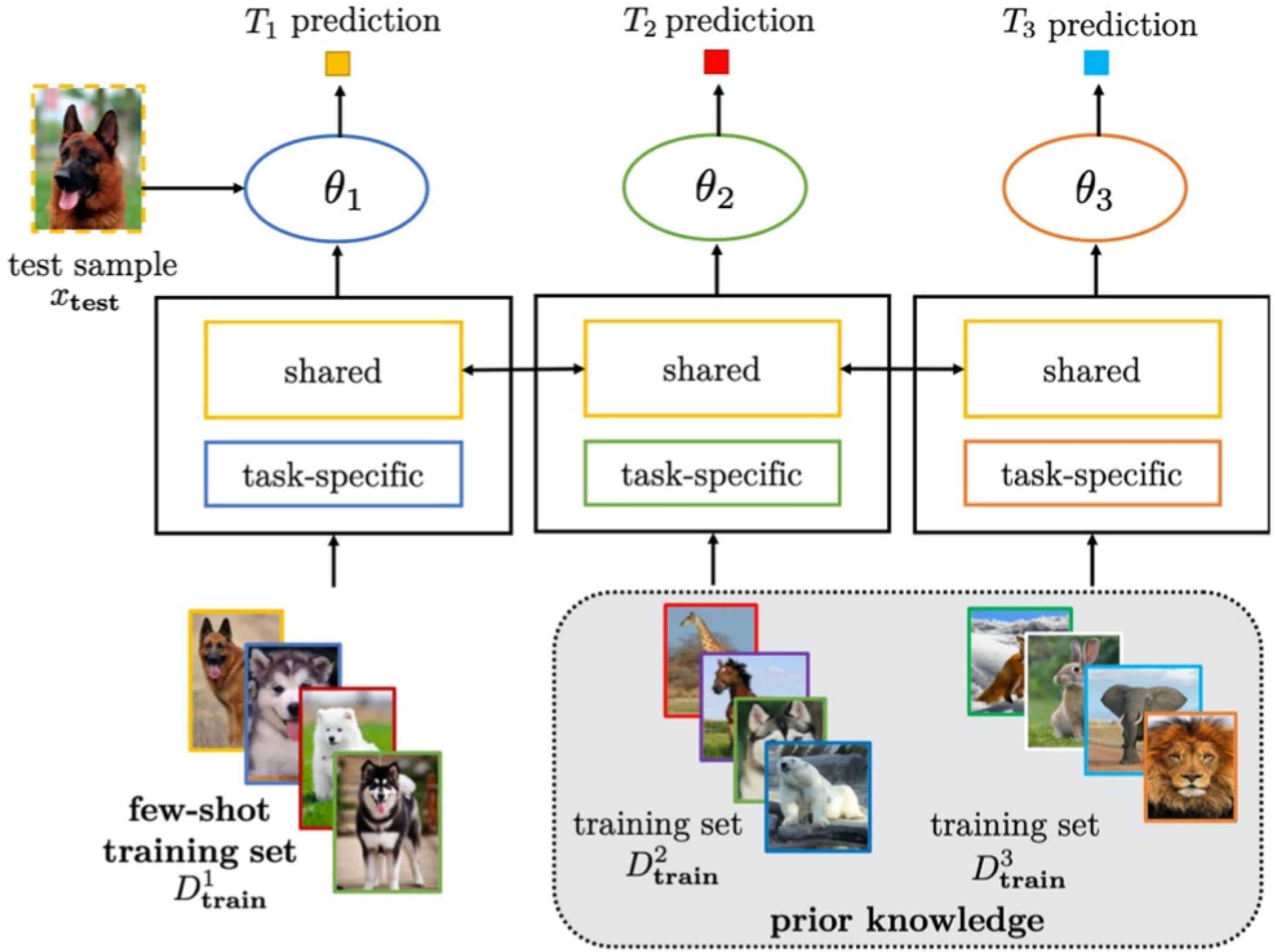


strategy	prior knowledge	how to constrain $\mathcal{H}$
multitask learning	other T's with their data sets D's	share/tie parameter
embedding learning	embedding learned from/together with other T's	project samples to a smaller embedding space in which similar and dissimilar samples can be easily discriminated
learning with external memory	embedding learned from other T's to interact with memory	refine samples using key-value pairs stored in memory
generative modeling	prior model learned from other T's	restrict the form of distribution

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

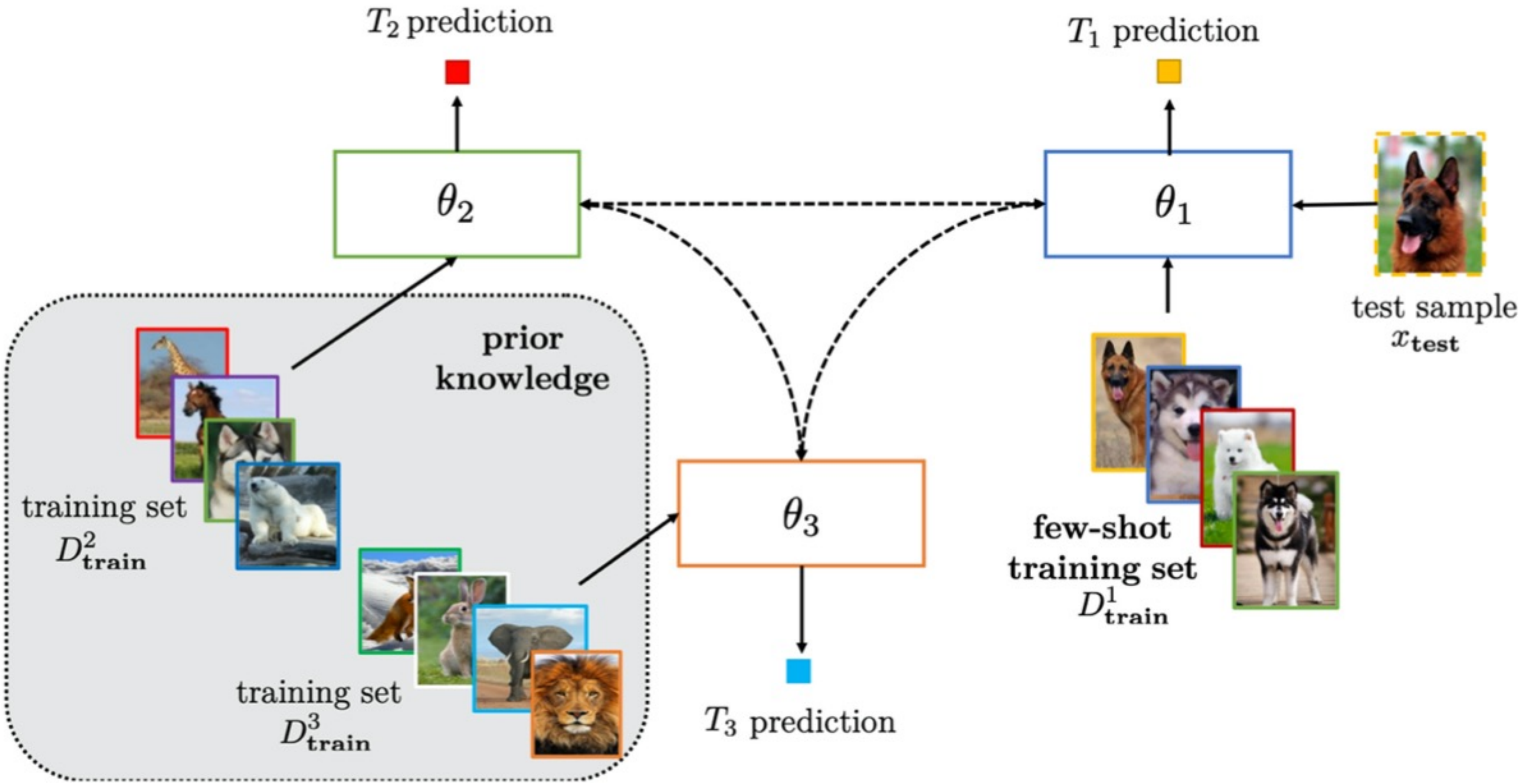
Solving the FSL problem by multitask learning with parameter sharing



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

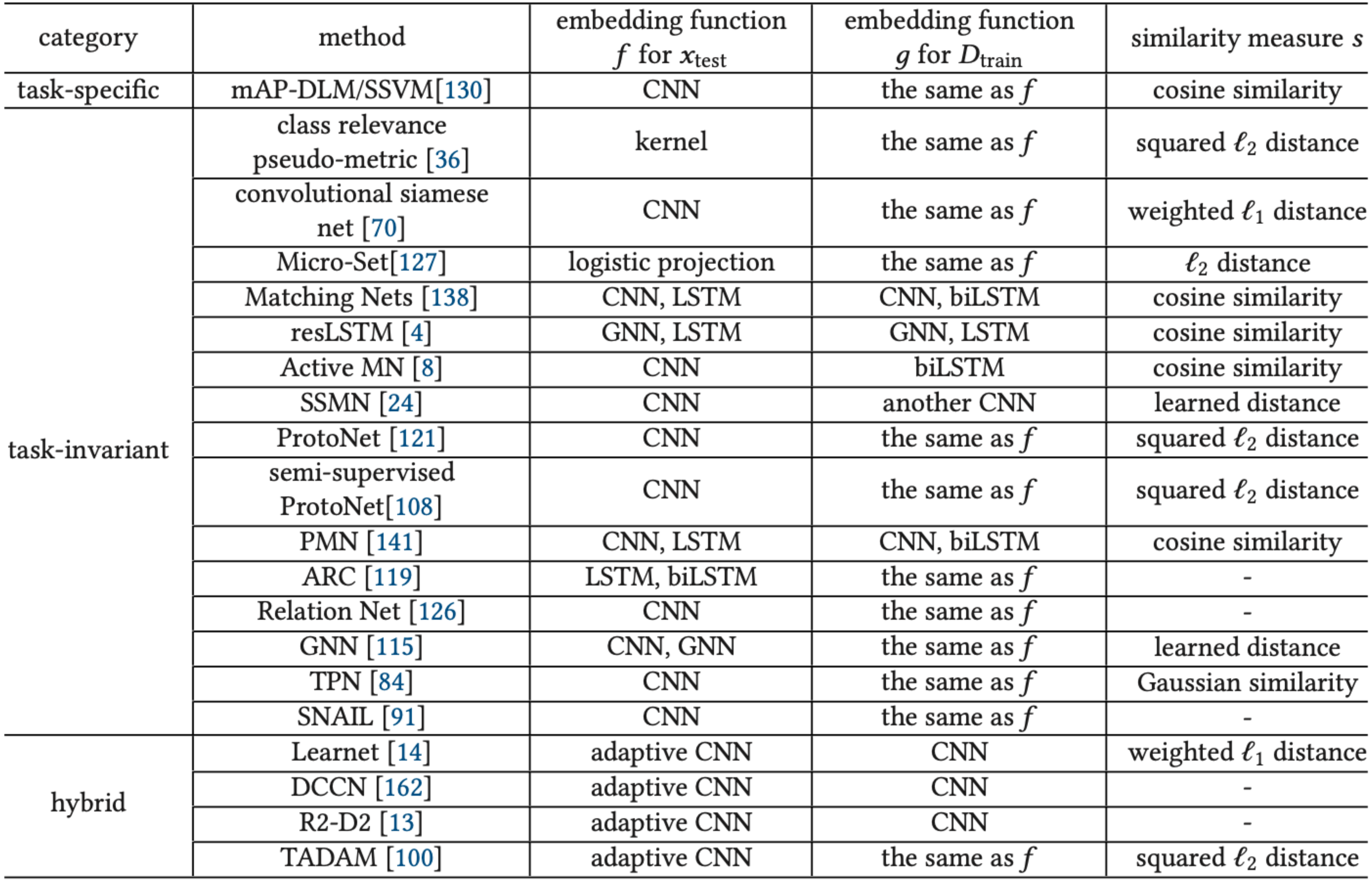
Solving the FSL problem by multitask learning with parameter tying



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Characteristics of Embedding Learning Methods

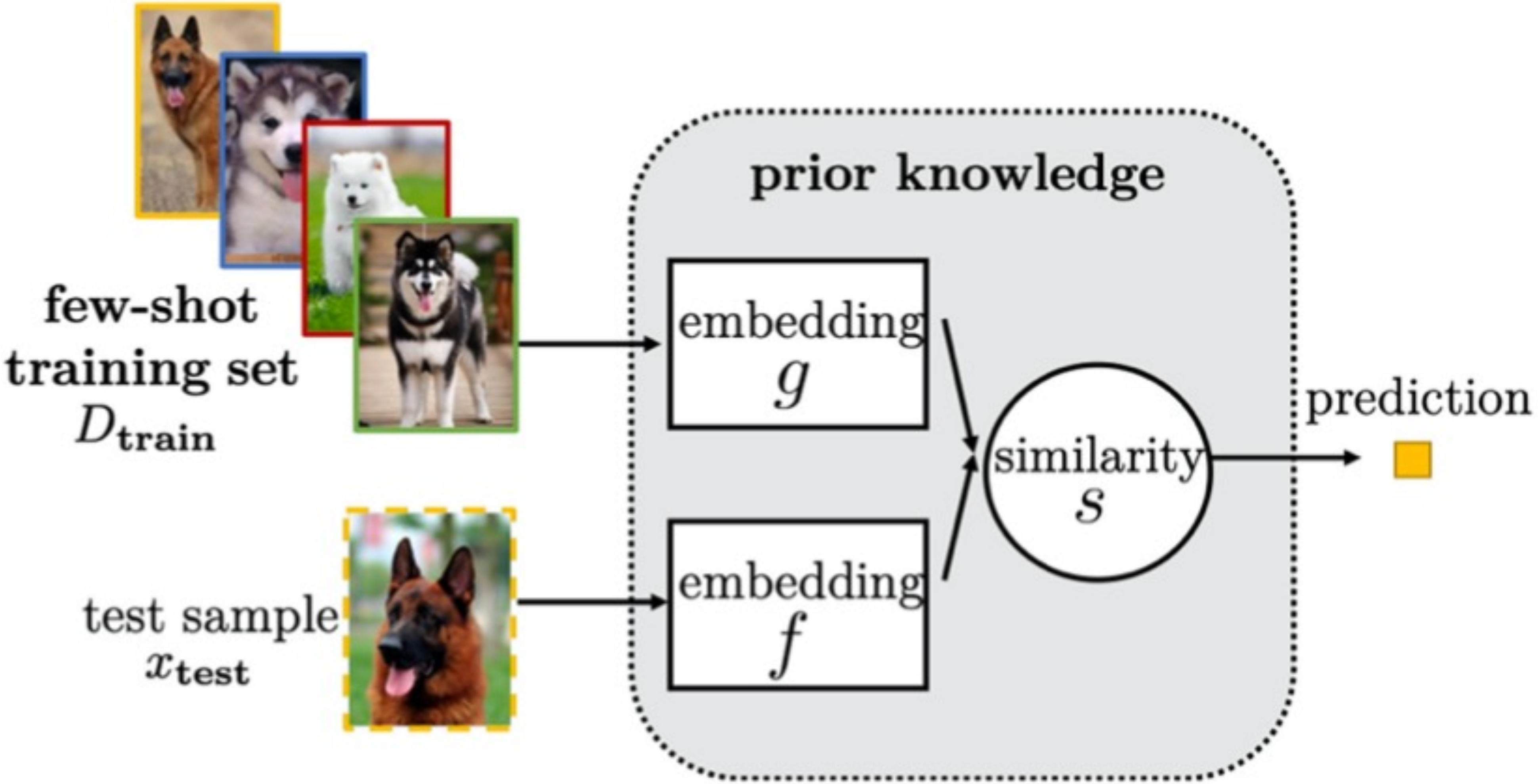


category	method	embedding function f for x_{test}	embedding function g for D_{train}	similarity measure s
task-specific	mAP-DLM/SSVM[130]	CNN	the same as f	cosine similarity
task-invariant	class relevance pseudo-metric [36]	kernel	the same as f	squared ℓ_2 distance
	convolutional siamese net [70]	CNN	the same as f	weighted ℓ_1 distance
	Micro-Set[127]	logistic projection	the same as f	ℓ_2 distance
	Matching Nets [138]	CNN, LSTM	CNN, biLSTM	cosine similarity
	resLSTM [4]	GNN, LSTM	GNN, LSTM	cosine similarity
	Active MN [8]	CNN	biLSTM	cosine similarity
	SSMN [24]	CNN	another CNN	learned distance
	ProtoNet [121]	CNN	the same as f	squared ℓ_2 distance
	semi-supervised ProtoNet[108]	CNN	the same as f	squared ℓ_2 distance
	PMN [141]	CNN, LSTM	CNN, biLSTM	cosine similarity
	ARC [119]	LSTM, biLSTM	the same as f	-
	Relation Net [126]	CNN	the same as f	-
	GNN [115]	CNN, GNN	the same as f	learned distance
	TPN [84]	CNN	the same as f	Gaussian similarity
	SNAIL [91]	CNN	the same as f	-
hybrid	Learnet [14]	adaptive CNN	CNN	weighted ℓ_1 distance
	DCCN [162]	adaptive CNN	CNN	-
	R2-D2 [13]	adaptive CNN	CNN	-
	TADAM [100]	adaptive CNN	the same as f	squared ℓ_2 distance

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

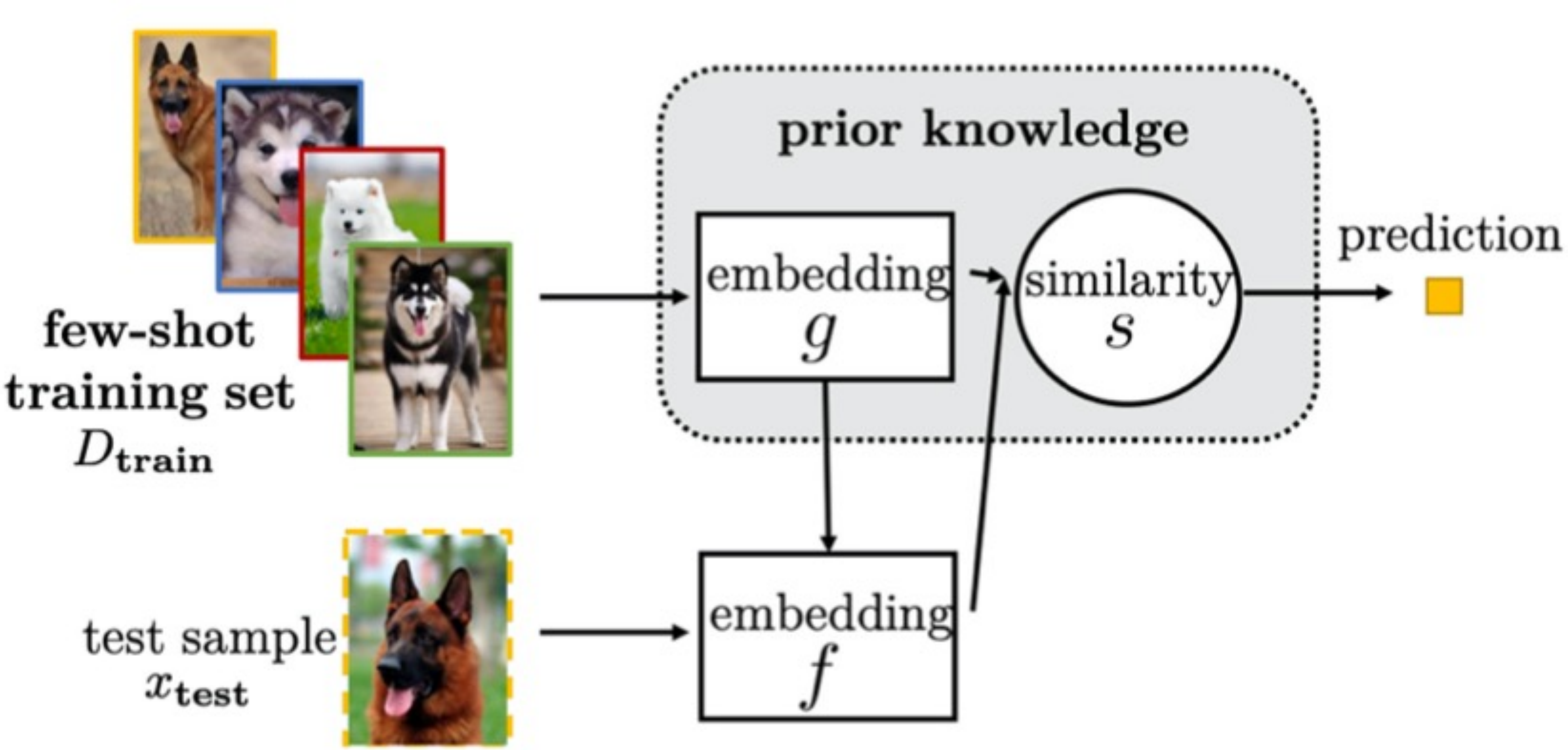
Solving the FSL problem by task-invariant embedding model



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

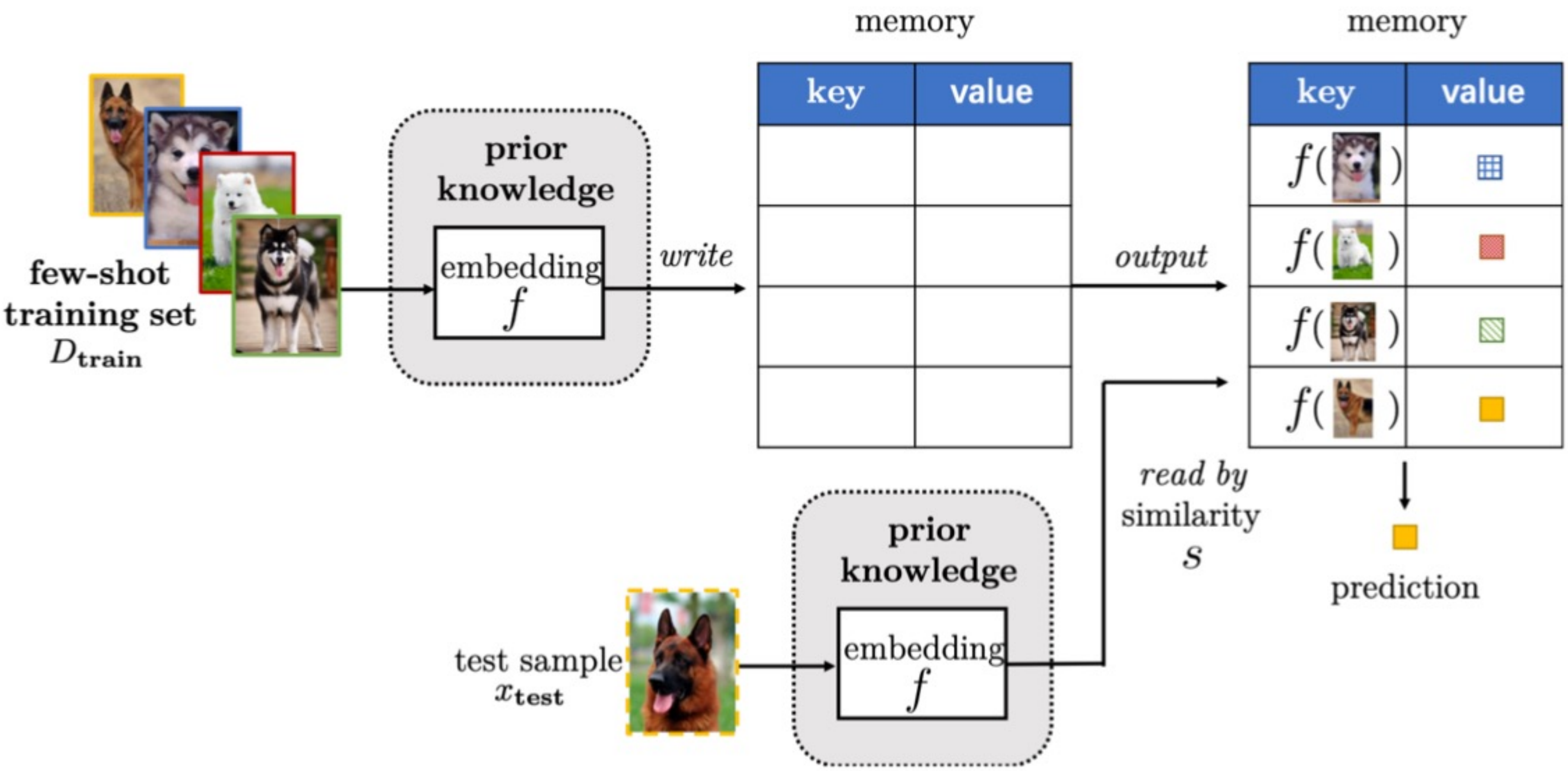
Solving the FSL problem by hybrid embedding model



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

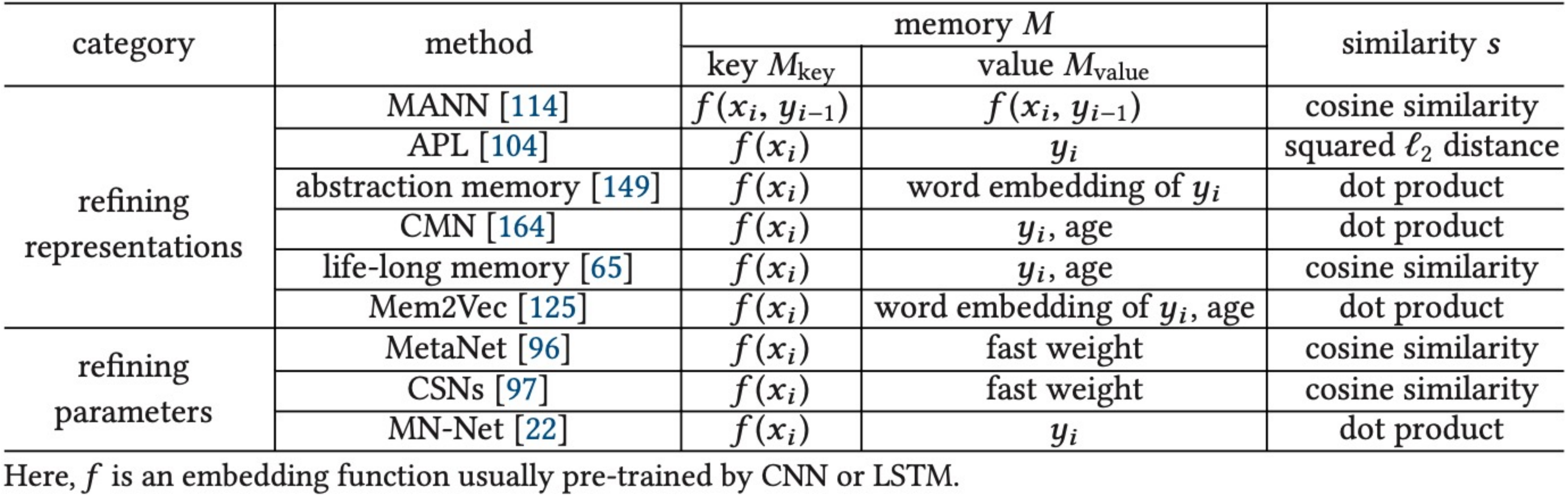
Solving the FSL problem by learning with external memory



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Characteristics of FSL Methods Based on Learning with External Memory



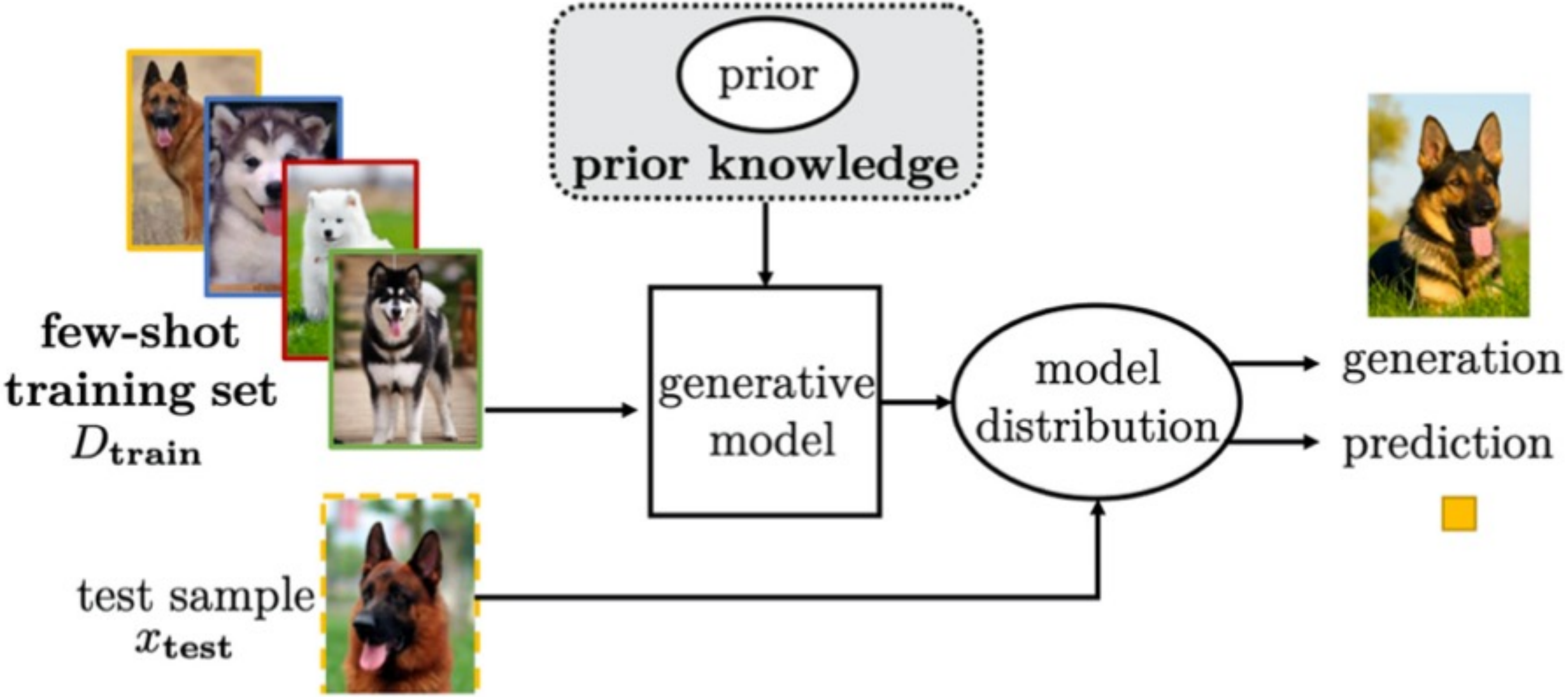
category	method	memory M		similarity s
		key M_{key}	value M_{value}	
refining representations	MANN [114]	$f(x_i, y_{i-1})$	$f(x_i, y_{i-1})$	cosine similarity
	APL [104]	$f(x_i)$	y_i	squared ℓ_2 distance
	abstraction memory [149]	$f(x_i)$	word embedding of y_i	dot product
	CMN [164]	$f(x_i)$	y_i, age	dot product
	life-long memory [65]	$f(x_i)$	y_i, age	cosine similarity
	Mem2Vec [125]	$f(x_i)$	word embedding of y_i, age	dot product
refining parameters	MetaNet [96]	$f(x_i)$	fast weight	cosine similarity
	CSNs [97]	$f(x_i)$	fast weight	cosine similarity
	MN-Net [22]	$f(x_i)$	y_i	dot product

Here, f is an embedding function usually pre-trained by CNN or LSTM.

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

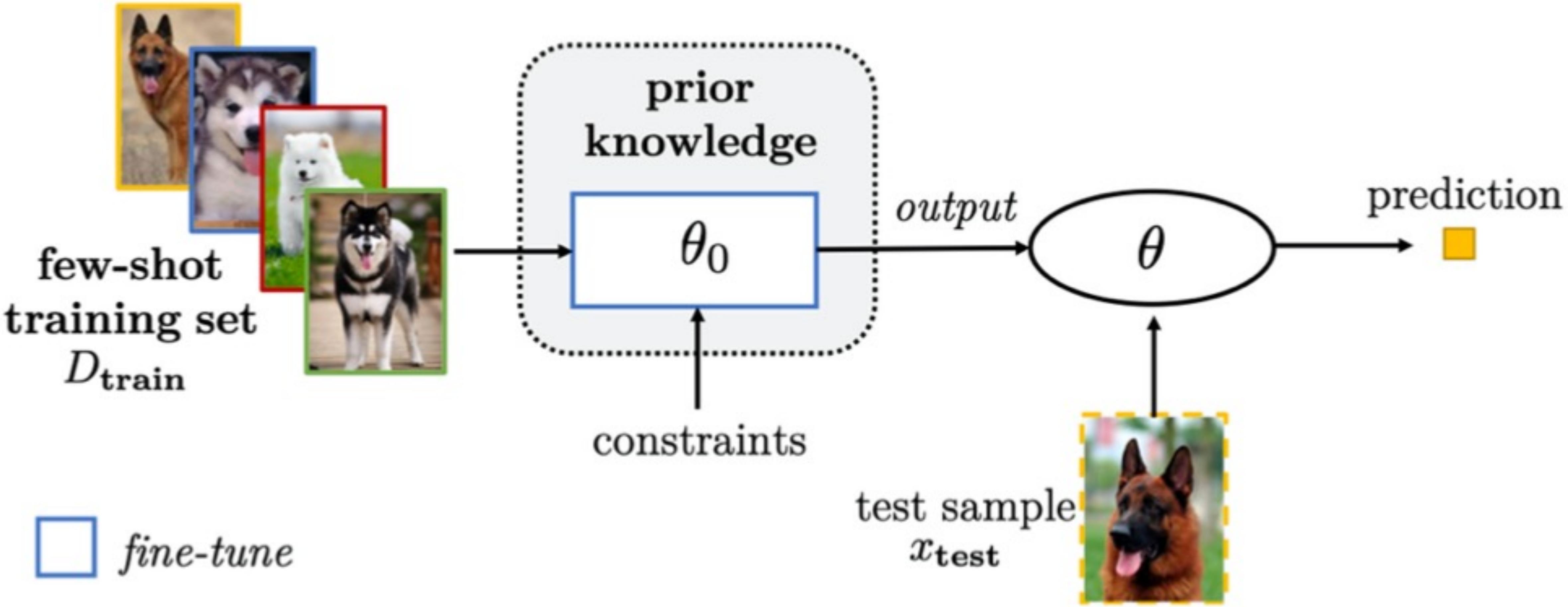
Solving the FSL problem by generative modeling



Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Solving the FSL problem by fine-tuning existing parameter θ_0 by regularization

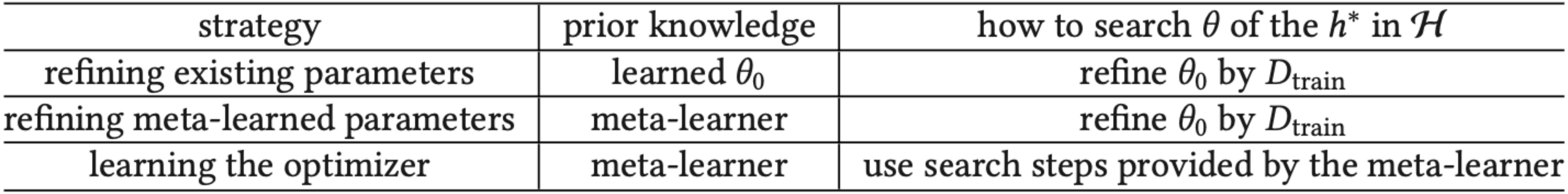


Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Characteristics for FSL Methods

Focusing on the Algorithm Perspective

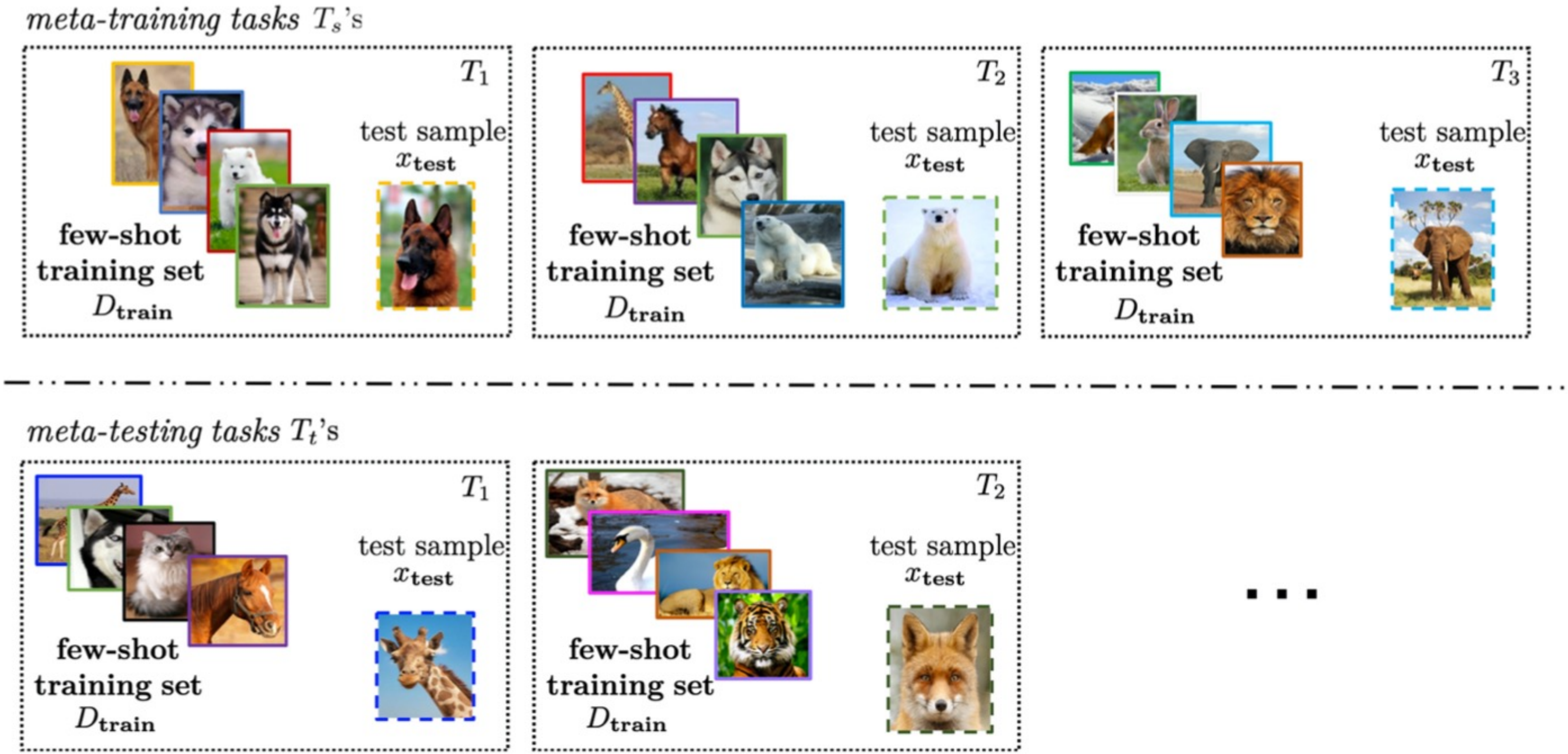


strategy	prior knowledge	how to search θ of the h^* in $\mathcal{H}$
refining existing parameters	learned θ_0	refine θ_0 by D_{train}
refining meta-learned parameters	meta-learner	refine θ_0 by D_{train}
learning the optimizer	meta-learner	use search steps provided by the meta-learner

Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Solving the FSL problem by meta-learning

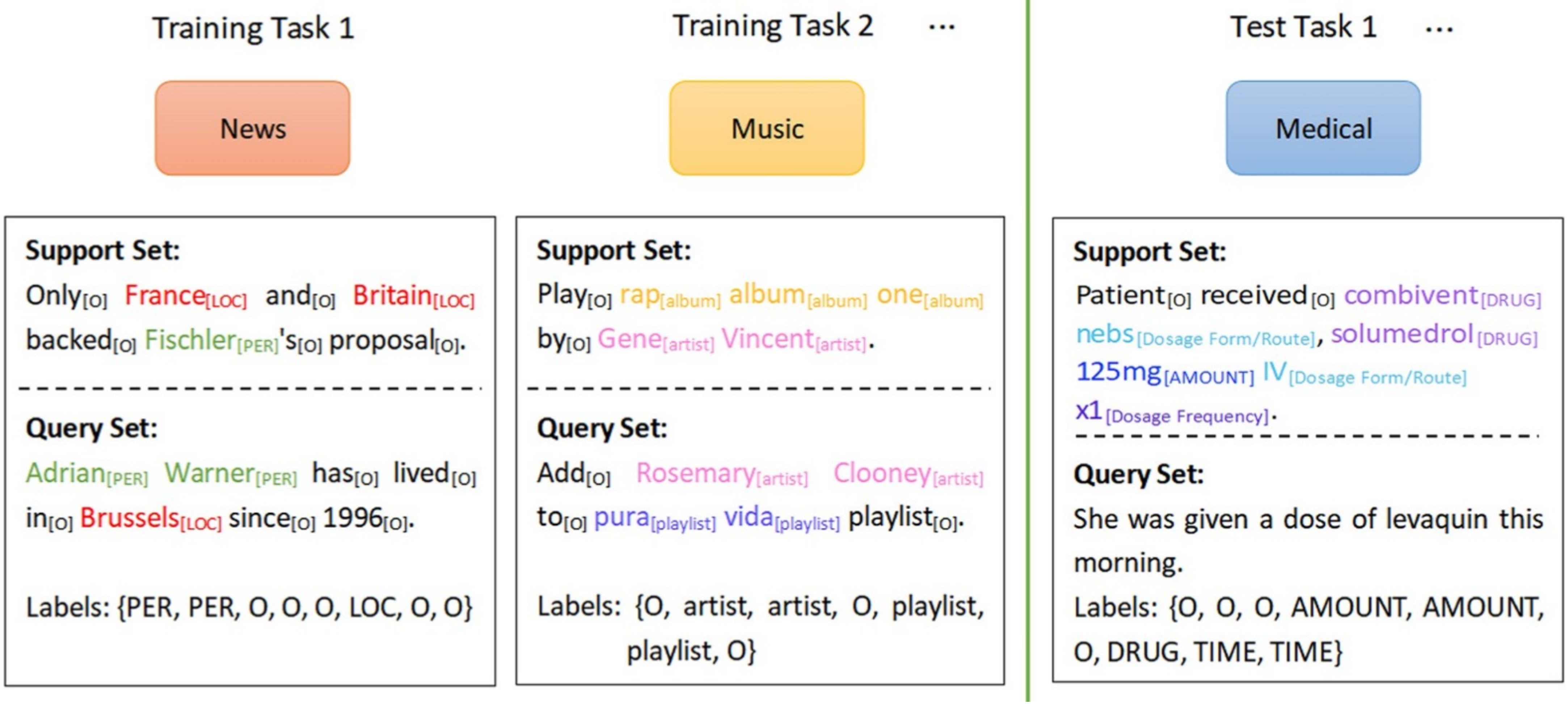


Source: Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.

Few-Shot Learning (FSL)

Meta-learning

Each task mimics the few-shot scenario, and can be completely non-overlapping.
Support sets are used to train; query sets are used to evaluate the model



Training Task 1 (News)

Support Set:

Only$_{[O]}$ France$_{[LOC]}$ and$_{[O]}$ Britain$_{[LOC]}$ backed$_{[O]}$ Fischler$_{[PER]}$'s$_{[O]}$ proposal$_{[O]}$.

Query Set:

Adrian$_{[PER]}$ Warner$_{[PER]}$ has$_{[O]}$ lived$_{[O]}$ in$_{[O]}$ Brussels$_{[LOC]}$ since$_{[O]}$ 1996$_{[O]}$.

Labels: {PER, PER, O, O, O, LOC, O, O}

Training Task 2 (Music)

Support Set:

Play$_{[O]}$ rap$_{[album]}$ album$_{[album]}$ one$_{[album]}$ by$_{[O]}$ Gene$_{[artist]}$ Vincent$_{[artist]}$.

Query Set:

Add$_{[O]}$ Rosemary$_{[artist]}$ Clooney$_{[artist]}$ to$_{[O]}$ pura$_{[playlist]}$ vida$_{[playlist]}$ playlist$_{[O]}$.

Labels: {O, artist, artist, O, playlist, playlist, O}

Test Task 1 (Medical)

Support Set:

Patient$_{[O]}$ received$_{[O]}$ combivent$_{[DRUG]}$ nebs$_{[Dosage Form/Route]}$, solumedrol$_{[DRUG]}$ 125mg$_{[AMOUNT]}$ IV$_{[Dosage Form/Route]}$ x1$_{[Dosage Frequency]}$.

Query Set:

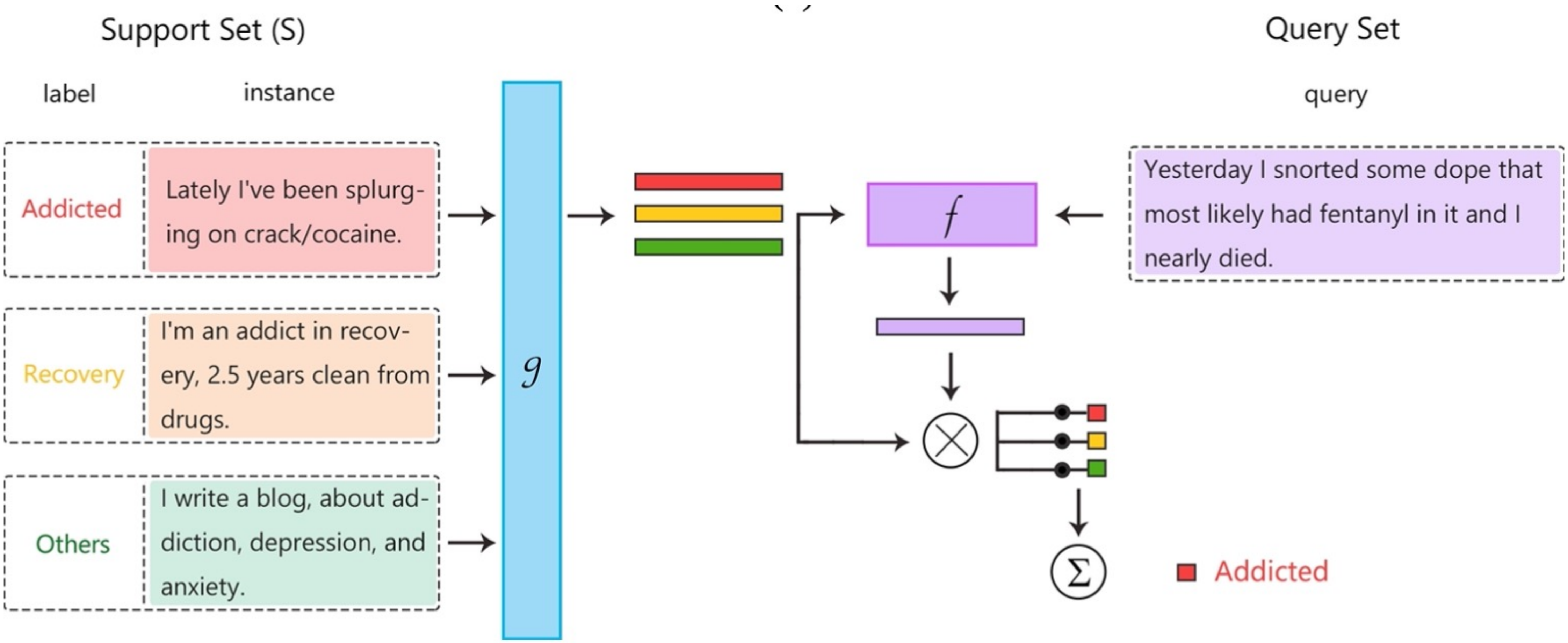
She was given a dose of levaquin this morning.

Labels: {O, O, O, AMOUNT, AMOUNT, O, DRUG, TIME, TIME}

Source: Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).

Few-Shot Learning (FSL)

Matching networks



Support Set (S)

label | instance

Addicted | Lately I've been splurging on crack/cocaine.

Recovery | I'm an addict in recovery, 2.5 years clean from drugs.

Others | I write a blog, about addiction, depression, and anxiety.

Query Set

query

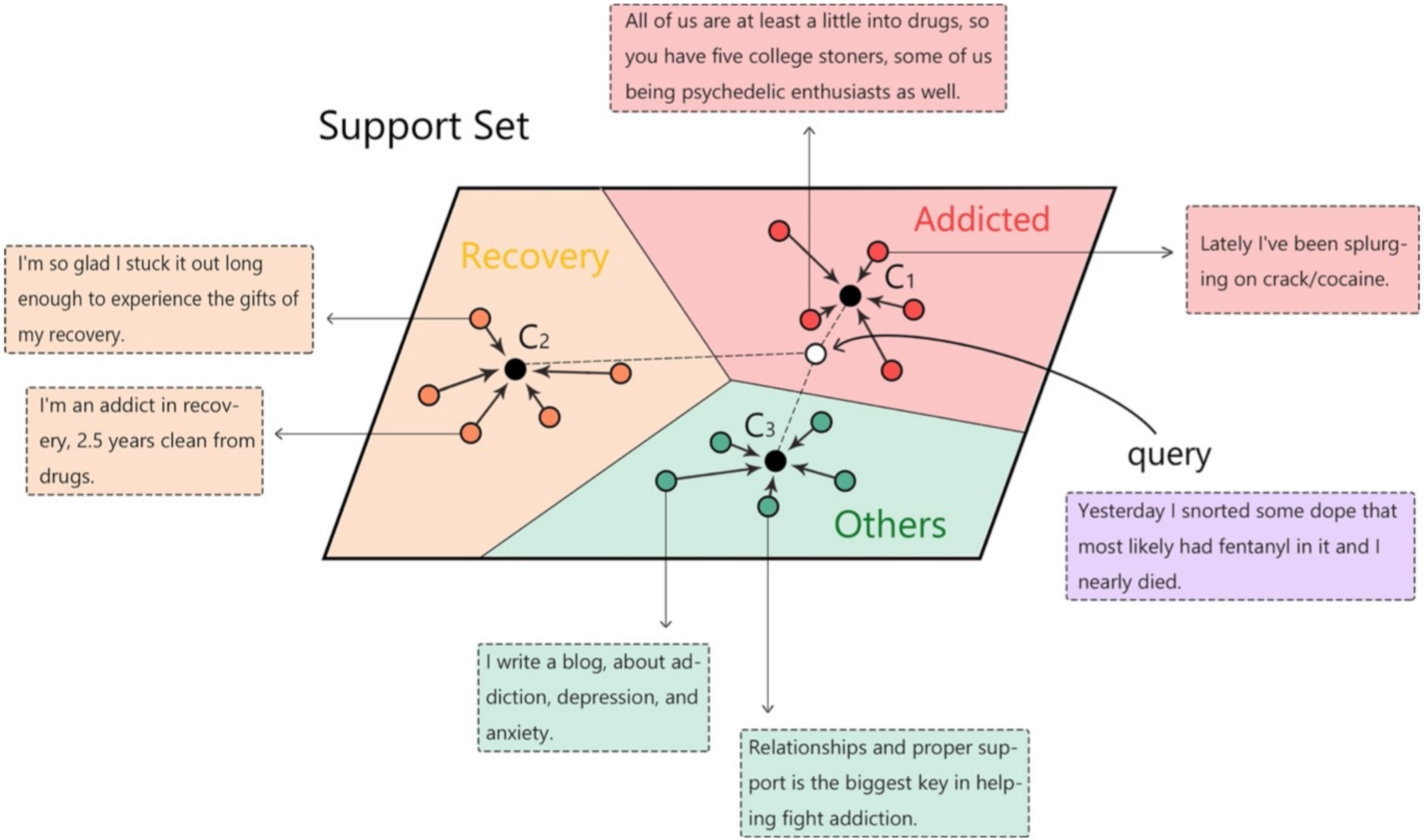
Yesterday I snorted some dope that most likely had fentanyl in it and I nearly died.

🟥 Addicted

Source: Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).

Few-Shot Learning (FSL)

Prototypical network



All of us are at least a little into drugs, so you have five college stoners, some of us being psychedelic enthusiasts as well.

I'm so glad I stuck it out long enough to experience the gifts of my recovery.

Lately I've been splurging on crack/cocaine.

I'm an addict in recovery, 2.5 years clean from drugs.

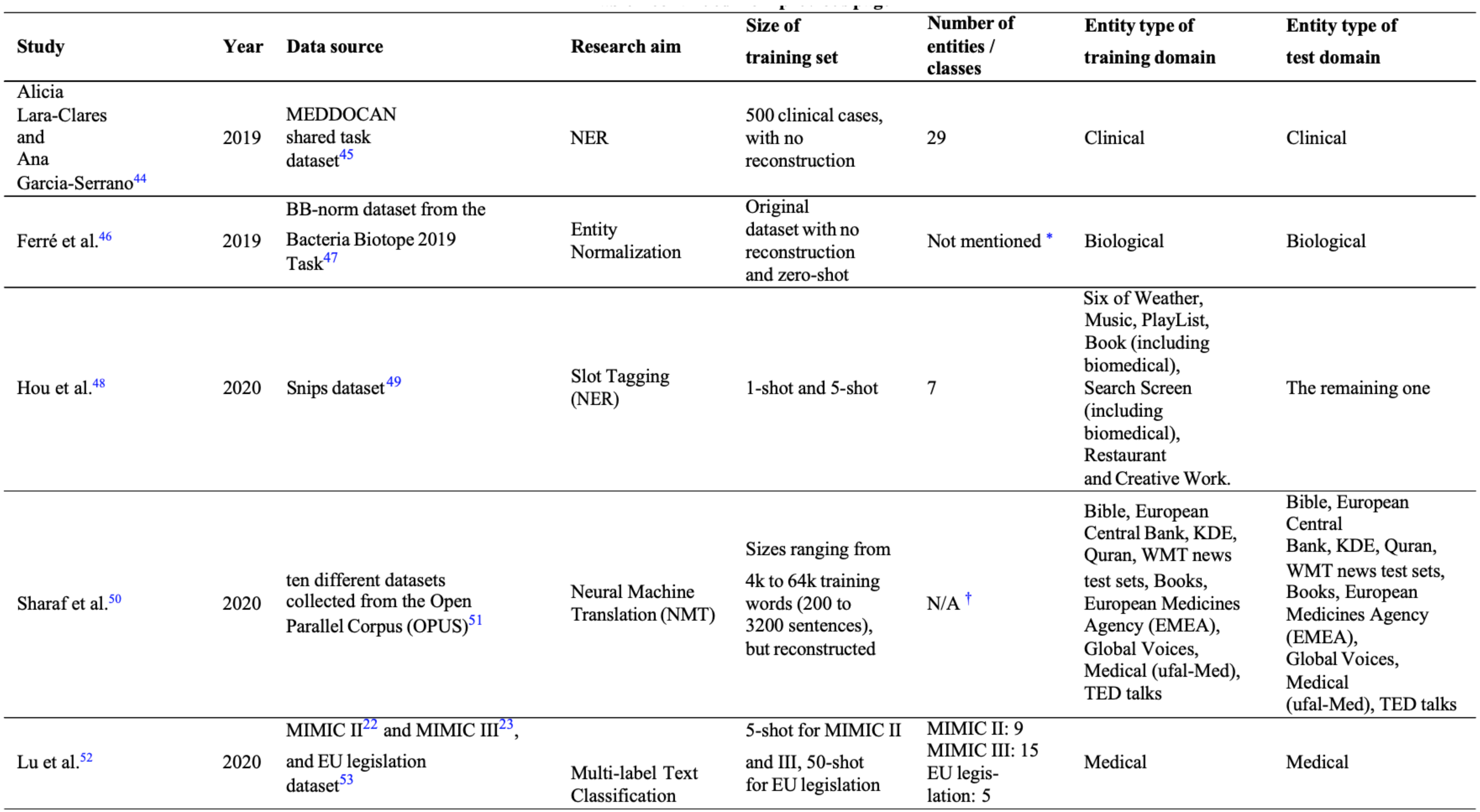
Yesterday I snorted some dope that most likely had fentanyl in it and I nearly died.

I write a blog, about addiction, depression, and anxiety.

Relationships and proper support is the biggest key in helping fight addiction.

Source: Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).

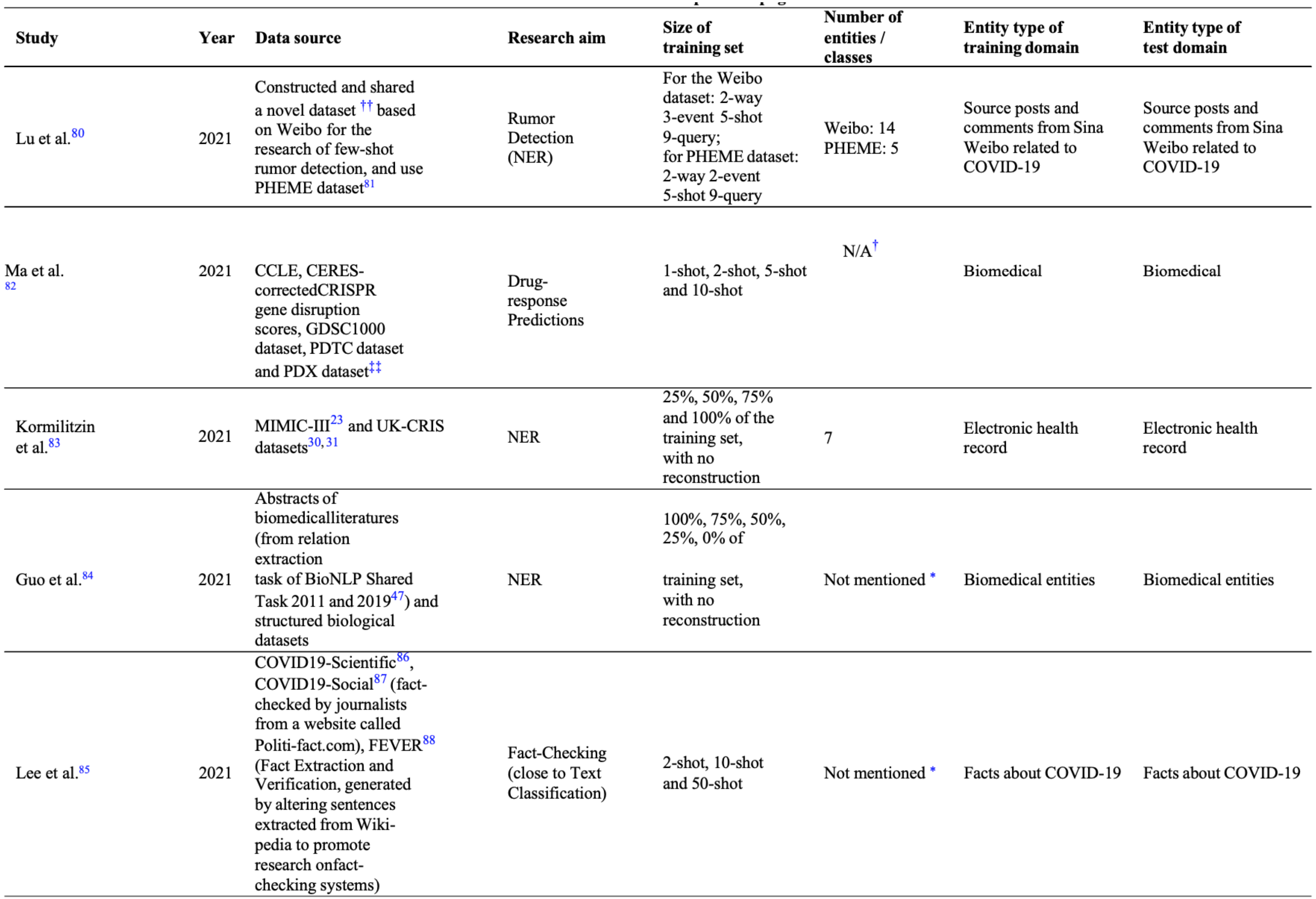
Few-Shot Learning (FSL) for medical text



Study	Year	Data source	Research aim	Size of training set	Number of entities / classes	Entity type of training domain	Entity type of test domain
Alicia Lara-Clares and Ana Garcia-Serrano[44]	2019	MEDDOCAN shared task dataset[45]	NER	500 clinical cases, with no reconstruction	29	Clinical	Clinical
Ferré et al.[46]	2019	BB-norm dataset from the Bacteria Biotope 2019 Task[47]	Entity Normalization	Original dataset with no reconstruction and zero-shot	Not mentioned *	Biological	Biological
Hou et al.[48]	2020	Snips dataset[49]	Slot Tagging (NER)	1-shot and 5-shot	7	Six of Weather, Music, PlayList, Book (including biomedical), Search Screen (including biomedical), Restaurant and Creative Work.	The remaining one
Sharaf et al.[50]	2020	ten different datasets collected from the Open Parallel Corpus (OPUS)[51]	Neural Machine Translation (NMT)	Sizes ranging from 4k to 64k training words (200 to 3200 sentences), but reconstructed	N/A †	Bible, European Central Bank, KDE, Quran, WMT news test sets, Books, European Medicines Agency (EMEA), Global Voices, Medical (ufal-Med), TED talks	Bible, European Central Bank, KDE, Quran, WMT news test sets, Books, European Medicines Agency (EMEA), Global Voices, Medical (ufal-Med), TED talks
Lu et al.[52]	2020	MIMIC II[22] and MIMIC III[23], and EU legislation dataset[53]	Multi-label Text Classification	5-shot for MIMIC II and III, 50-shot for EU legislation	MIMIC II: 9 MIMIC III: 15 EU legislation: 5	Medical	Medical

Source: Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).

Few-Shot Learning (FSL) for medical text



Study	Year	Data source	Research aim	Size of training set	Number of entities / classes	Entity type of training domain	Entity type of test domain
Lu et al.[80]	2021	Constructed and shared a novel dataset [††] based on Weibo for the research of few-shot rumor detection, and use PHEME dataset[81]	Rumor Detection (NER)	For the Weibo dataset: 2-way 3-event 5-shot 9-query; for PHEME dataset: 2-way 2-event 5-shot 9-query	Weibo: 14 PHEME: 5	Source posts and comments from Sina Weibo related to COVID-19	Source posts and comments from Sina Weibo related to COVID-19
Ma et al.[82]	2021	CCLE, CERES-correctedCRISPR gene disruption scores, GDSC1000 dataset, PDTC dataset and PDX dataset[‡‡]	Drug-response Predictions	1-shot, 2-shot, 5-shot and 10-shot	N/A[†]	Biomedical	Biomedical
Kormilitzin et al.[83]	2021	MIMIC-III[23] and UK-CRIS datasets[30, 31]	NER	25%, 50%, 75% and 100% of the training set, with no reconstruction	7	Electronic health record	Electronic health record
Guo et al.[84]	2021	Abstracts of biomedicalliteratures (from relation extraction task of BioNLP Shared Task 2011 and 2019[47]) and structured biological datasets	NER	100%, 75%, 50%, 25%, 0% of training set, with no reconstruction	Not mentioned [*]	Biomedical entities	Biomedical entities
Lee et al.[85]	2021	COVID19-Scientific[86], COVID19-Social[87] (fact-checked by journalists from a website called Politi-fact.com), FEVER[88] (Fact Extraction and Verification, generated by altering sentences extracted from Wiki-pedia to promote research onfact-checking systems)	Fact-Checking (close to Text Classification)	2-shot, 10-shot and 50-shot	Not mentioned [*]	Facts about COVID-19	Facts about COVID-19

Source: Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).

Multimodal Few-Shot Learning with Frozen Language Models

This person is like 😁.

This person is like 😞.

This person is like

Model Completion

😱. <EOS>

This was invented by Zacharias Janssen.

This was invented by Thomas Edison.

This was invented by

Model Completion

the Wright brothers. <EOS>

With one of these I can drive around a track, overtaking other cars and taking corners at speed

With one of these I can take off from a city and fly across the sky to somewhere on the other side of the world

With one of these I can

Model Completion

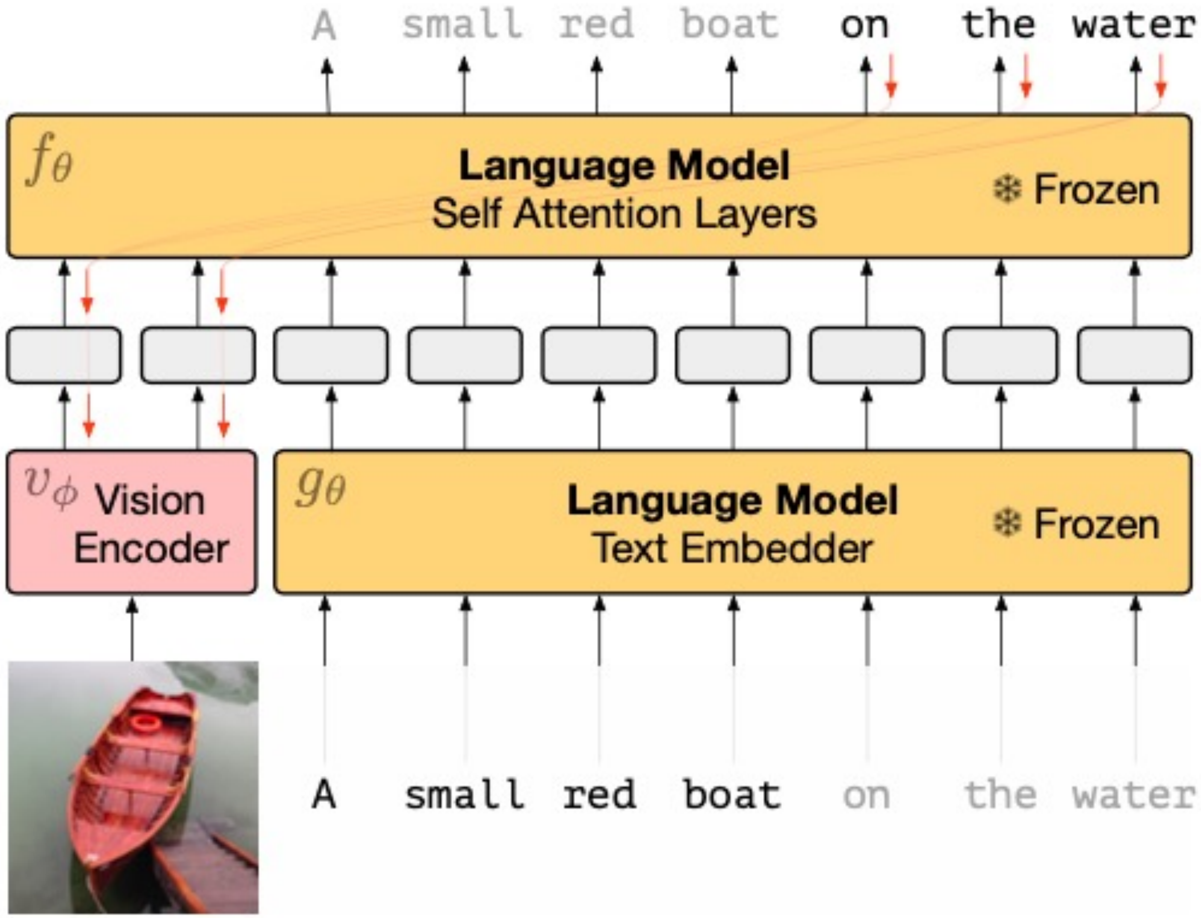
break into a secure building, unlock the door and walk right in <EOS>

Curated samples with about five seeds required to get past well-known language model failure modes of either repeating text for the prompt or emitting text that does not pertain to the image.

These samples demonstrate the ability to generate open-ended outputs that adapt to both images and text, and to make use of facts that it has learned during language-only pre-training.

Source: Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models." Advances in Neural Information Processing Systems 34 (2021): 200-212.

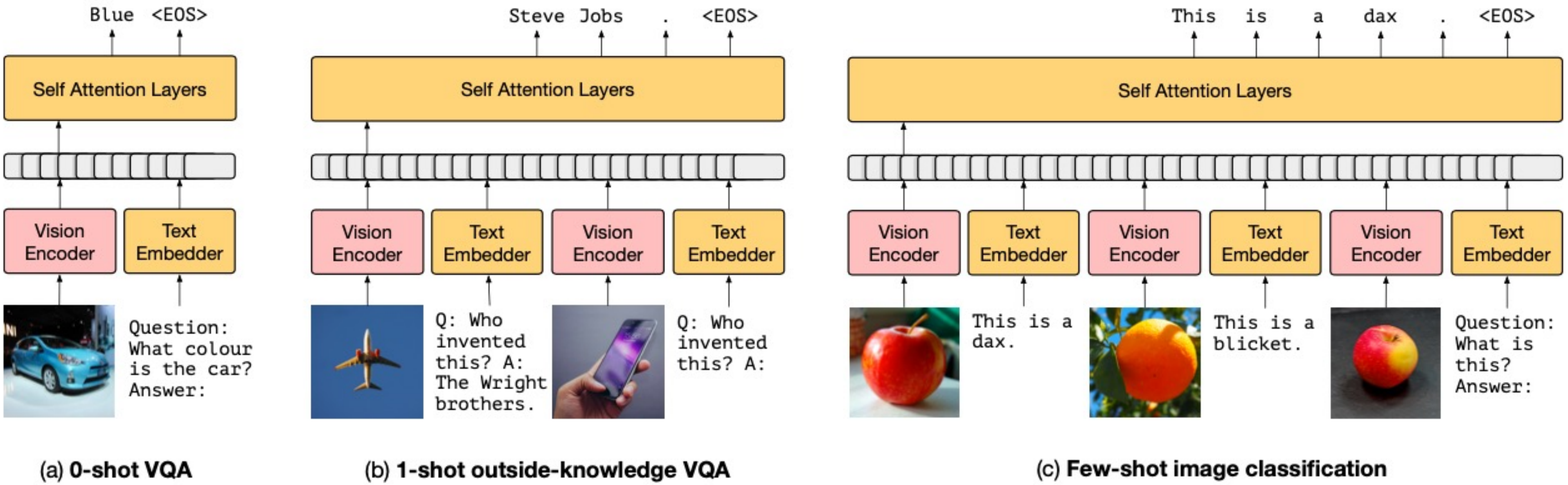
Multimodal Few-Shot Learning with Frozen Language Models



Gradients through a frozen language model's self attention layers are used to train the vision encoder.

Source: Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models." Advances in Neural Information Processing Systems 34 (2021): 200-212.

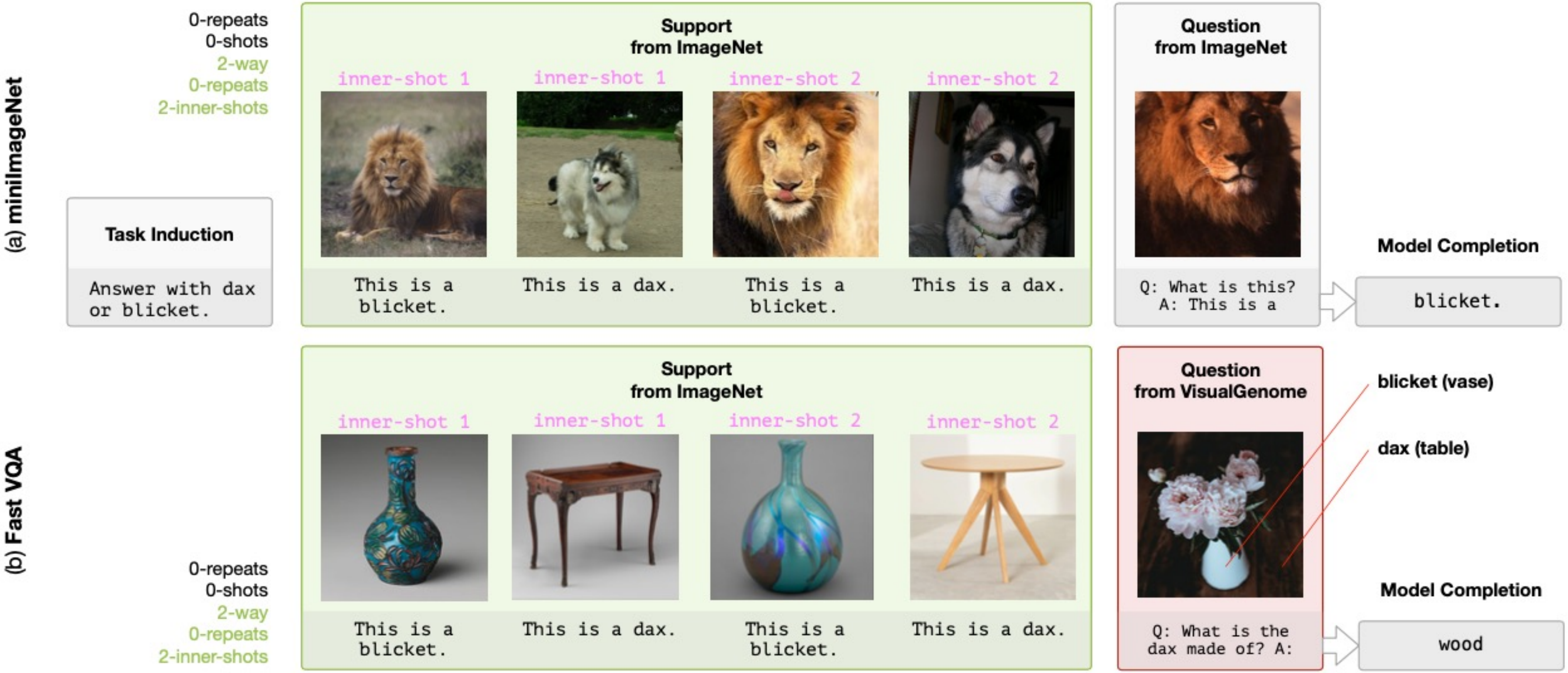
Multimodal Few-Shot Learning with Frozen Language Models



Inference-Time interface for *Frozen*. The figure demonstrates how we can support (a) visual question answering, (b) outside-knowledge question answering and (c) few-shot image classification via in-context learning.

Source: Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models." Advances in Neural Information Processing Systems 34 (2021): 200-212.

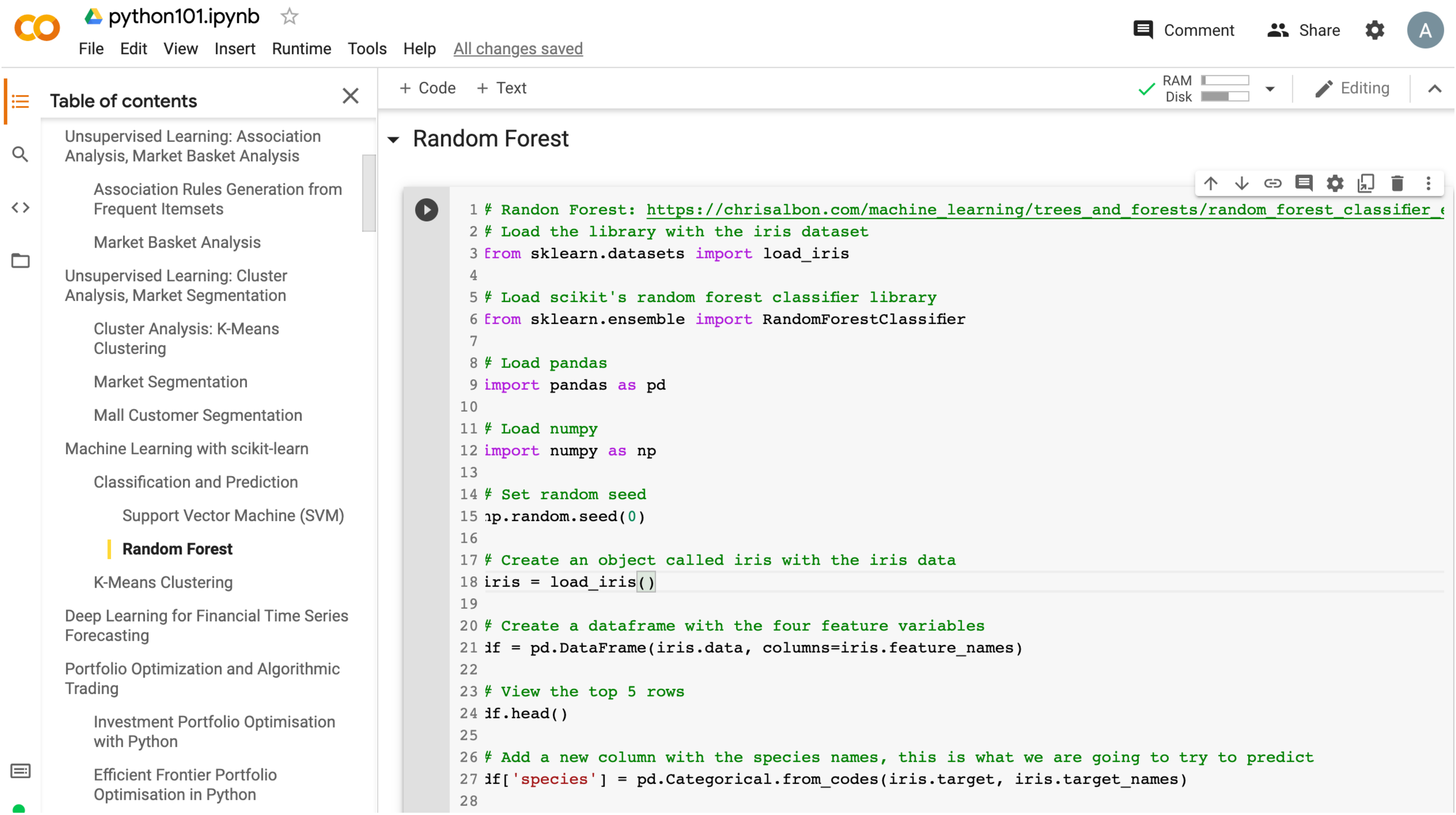
Multimodal Few-Shot Learning with Frozen Language Models



Examples of (a) the Open-Ended miniImageNet evaluation (b) the Fast VQA evaluation.

Source: Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models." Advances in Neural Information Processing Systems 34 (2021): 200-212.

Machine Learning: Ensemble Learning
Random Forest



```
# Randon Forest: https://chrisalbon.com/machine_learning/trees_and_forests/random_forest_classifier_
# Load the library with the iris dataset
from sklearn.datasets import load_iris

# Load scikit's random forest classifier library
from sklearn.ensemble import RandomForestClassifier

# Load pandas
import pandas as pd

# Load numpy
import numpy as np

# Set random seed
np.random.seed(0)

# Create an object called iris with the iris data
iris = load_iris()

# Create a dataframe with the four feature variables
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# View the top 5 rows
df.head()

# Add a new column with the species names, this is what we are going to try to predict
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

```

https://tinyurl.com/aintpupython101

Machine Learning: Supervised Learning Classification and Prediction

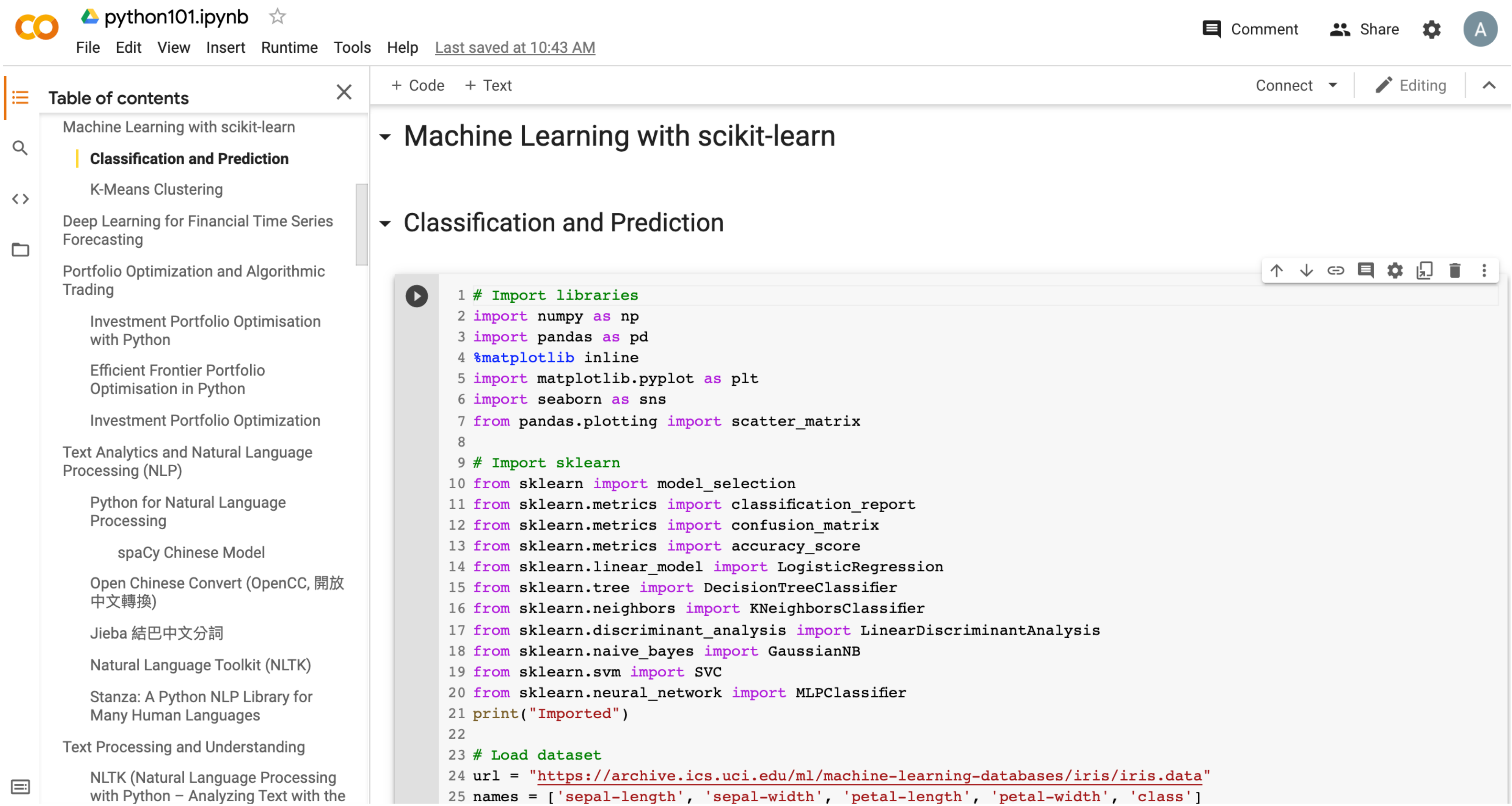


Table of contents

- Machine Learning with scikit-learn
  - **Classification and Prediction**
  - K-Means Clustering
- Deep Learning for Financial Time Series Forecasting
- Portfolio Optimization and Algorithmic Trading
  - Investment Portfolio Optimisation with Python
  - Efficient Frontier Portfolio Optimisation in Python
  - Investment Portfolio Optimization
- Text Analytics and Natural Language Processing (NLP)
  - Python for Natural Language Processing
    - spaCy Chinese Model
  - Open Chinese Convert (OpenCC, 開放中文轉換)
  - Jieba 結巴中文分詞
  - Natural Language Toolkit (NLTK)
  - Stanza: A Python NLP Library for Many Human Languages
- Text Processing and Understanding
  - NLTK (Natural Language Processing with Python – Analyzing Text with the

Machine Learning with scikit-learn

Classification and Prediction

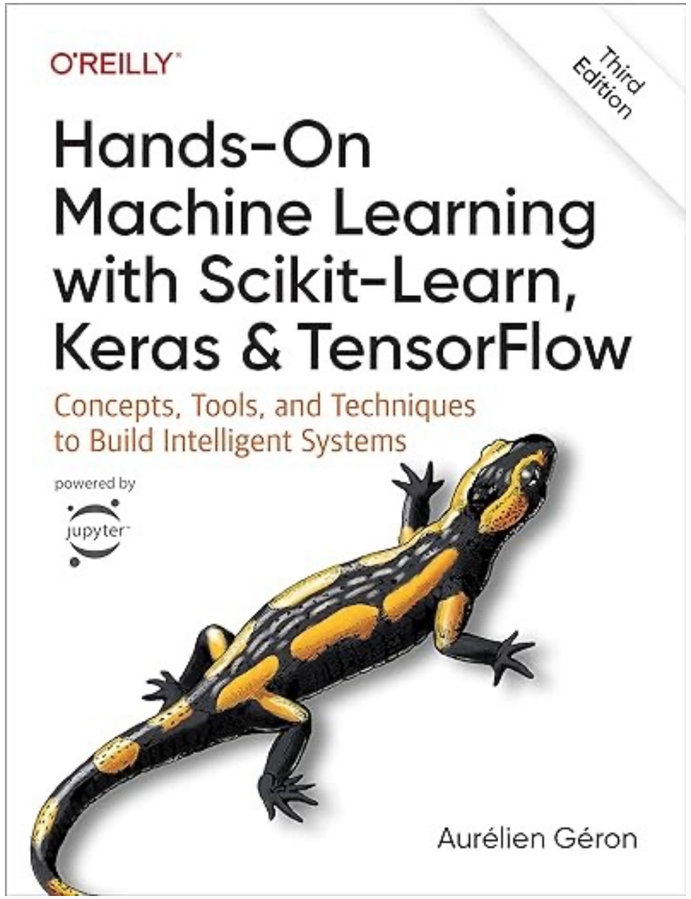
```python
# Import libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
```

https://tinyurl.com/aintpupython101

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks

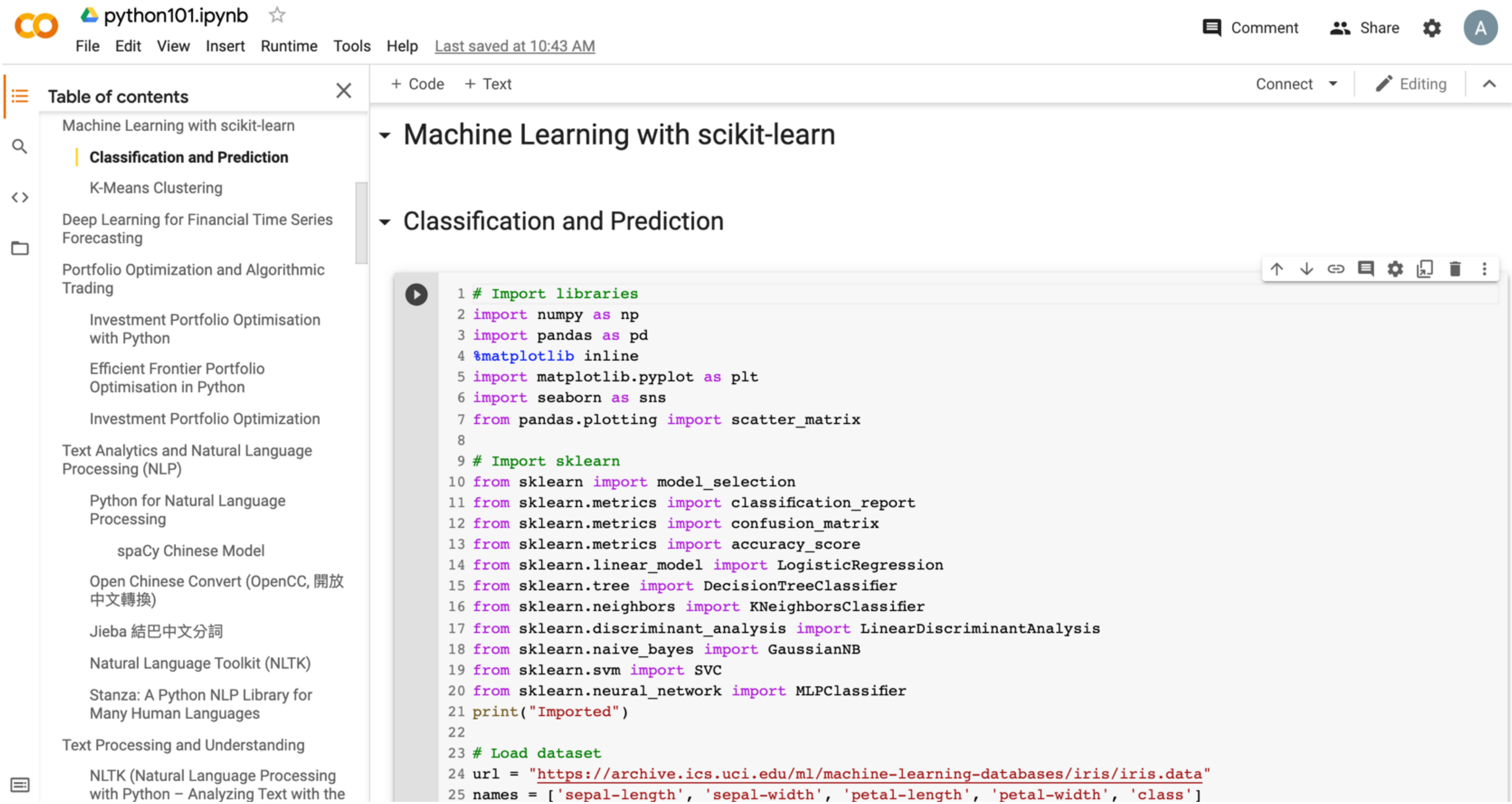
1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Autoencoders, GANs, and Diffusion Models
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml3

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



Machine Learning with scikit-learn

Classification and Prediction

```python
# Import libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
```

https://tinyurl.com/aintpupython101

Summary

- **The Theory of Learning**
  - **Computational Learning Theory**
  - **Probably Approximately Correct (PAC) Learning**
- **Ensemble Learning**
  - **Bagging: Random Forests (RF)**
  - **Boosting: Gradient Boosting, XGBoost, LightGBM, CatBoost**
  - **Stacking**
  - **Online learning**
- **Meta Learning: Learning to Learn**

References

- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Numa Dhamani and Maggie Engler (2024), Introduction to Generative AI, Manning
- Denis Rothman (2024), Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3, 3rd ed. Edition, Packt Publishing
- Thomas R. Caldwell (2025), The Agentic AI Bible: The Complete and Up-to-Date Guide to Design, Build, and Scale Goal-Driven, LLM-Powered Agents that Think, Execute and Evolve, Independently published
- Ben Auffarth (2023), Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT and other LLMs, Packt Publishing.
- Aurélien Géron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition, O’Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Nithin Buduma, Nikhil Buduma, Joe Papa (2022), Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 2nd Edition, O'Reilly Media.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. (2024) "Reasoning with Large Language Models, a Survey." arXiv preprint arXiv:2407.11511.
- Madaan, Aman, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon et al. (2024) "Self-refine: Iterative refinement with self-feedback." Advances in Neural Information Processing Systems 36.