Artificial Intelligence

Machine Learning: Supervised and Unsupervised Learning

1131AI04
MBA, IM, NTPU (M5276) (Fall 2024)
Tue 2, 3, 4 (9:10-12:00) (B3F17)

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

Min-Yuh Day, Ph.D, Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

2024-10-15

Syllabus

Week	Date	Subject/Topics
1	2024/09/10	Introduction to Artificial Intelligence
2	2024/09/17	Mid-Autumn Festival (Day off)
3	2024/09/24	Artificial Intelligence and Intelligent Agents; Problem Solving
4	2024/10/01	Knowledge, Reasoning and Knowledge Representation; Uncertain Knowledge and Reasoning
5	2024/10/08	Case Study on Artificial Intelligence I
6	2024/10/15	Machine Learning: Supervised and Unsupervised Learning

Syllabus

Week	Date	Subject/Topics
7	2024/10/22	The Theory of Learning and Ensemble Learning
8	2024/10/29	Midterm Project Report
9	2024/11/05	Self-Learning
10	2024/11/12	Deep Learning, Reinforcement Learning
11	2024/11/19	Case Study on Artificial Intelligence II
12	2024/11/26	Deep Learning for Natural Language Processing

Syllabus

Week	Date	Subject/Topics
13	2024/12/03	Computer Vision and Robotics
14	2024/12/10	Generative AI, Philosophy and Ethics of AI and the Future of AI
15	2024/12/17	Final Project Report I
16	2024/12/24	Final Project Report II

Machine Learning: Supervised and Unsupervised Learning

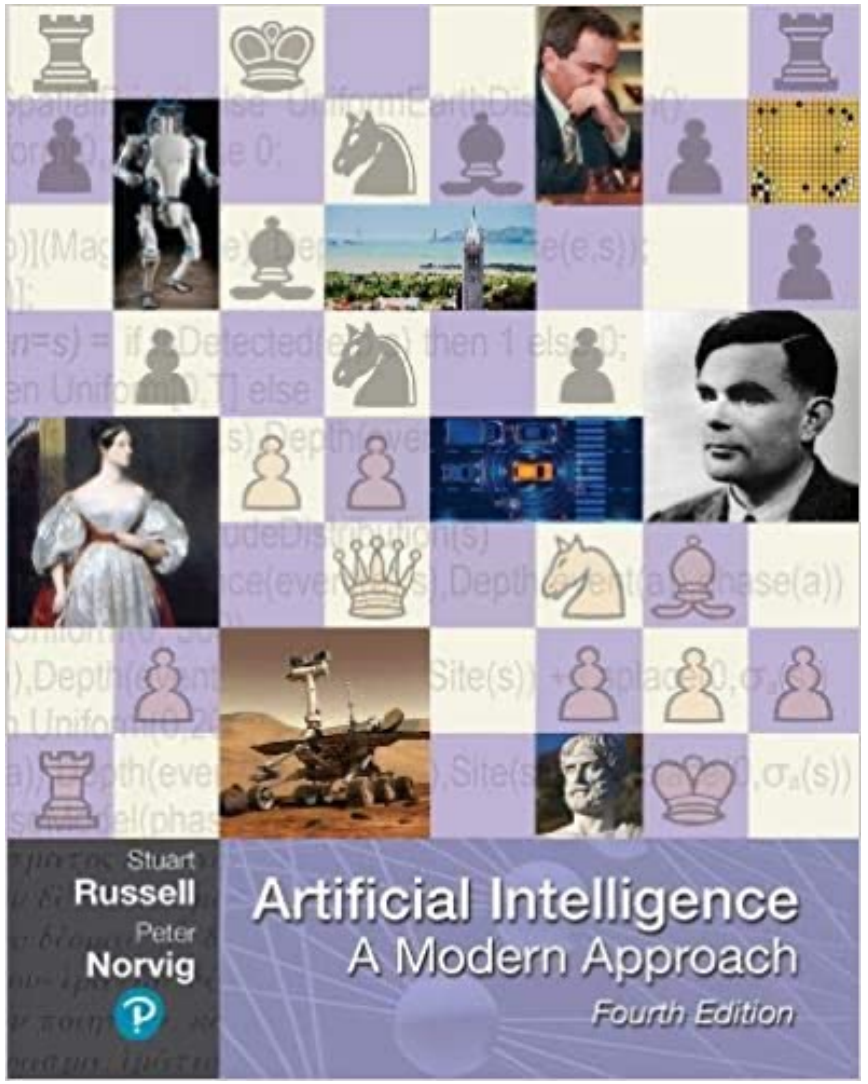
Outline

- **Machine Learning**
- **Supervised Learning**
- **Unsupervised Learning**

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

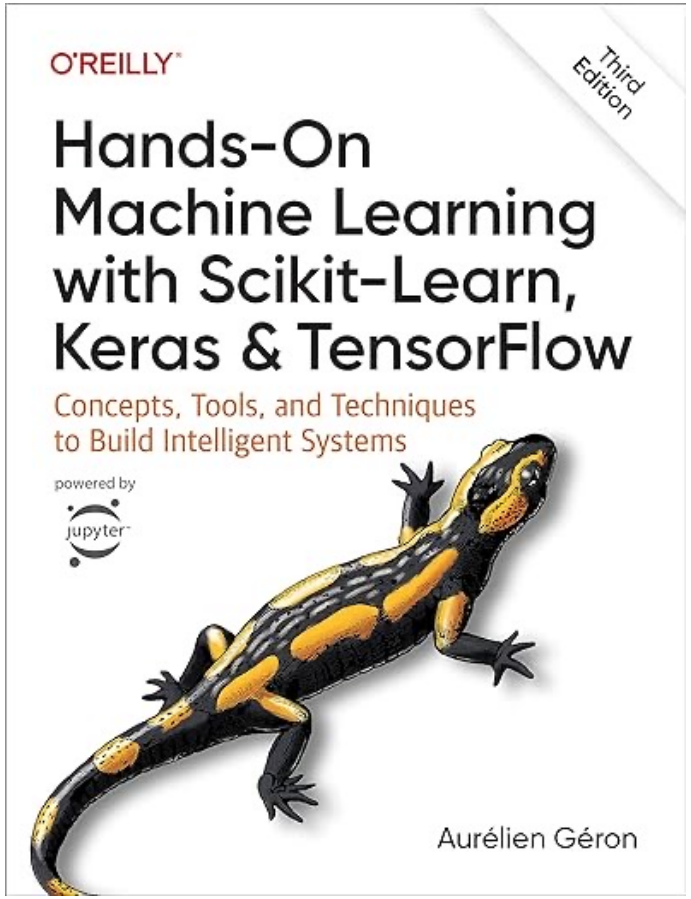
- **Learning from Examples**
- **Learning Probabilistic Models**
- **Deep Learning**
- **Reinforcement Learning**

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



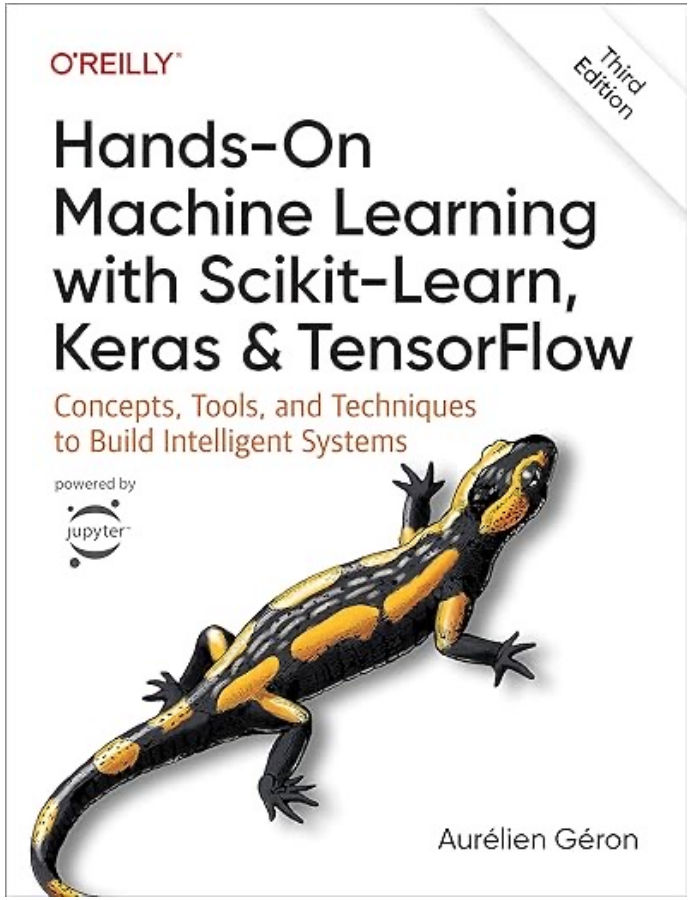
https://github.com/ageron/handson-ml3

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

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

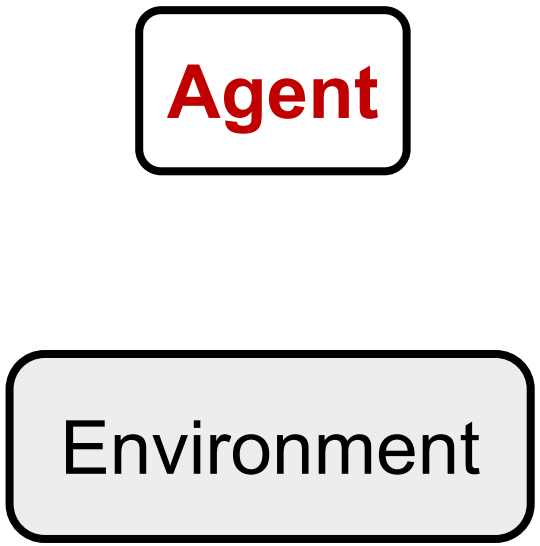
Notebooks

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



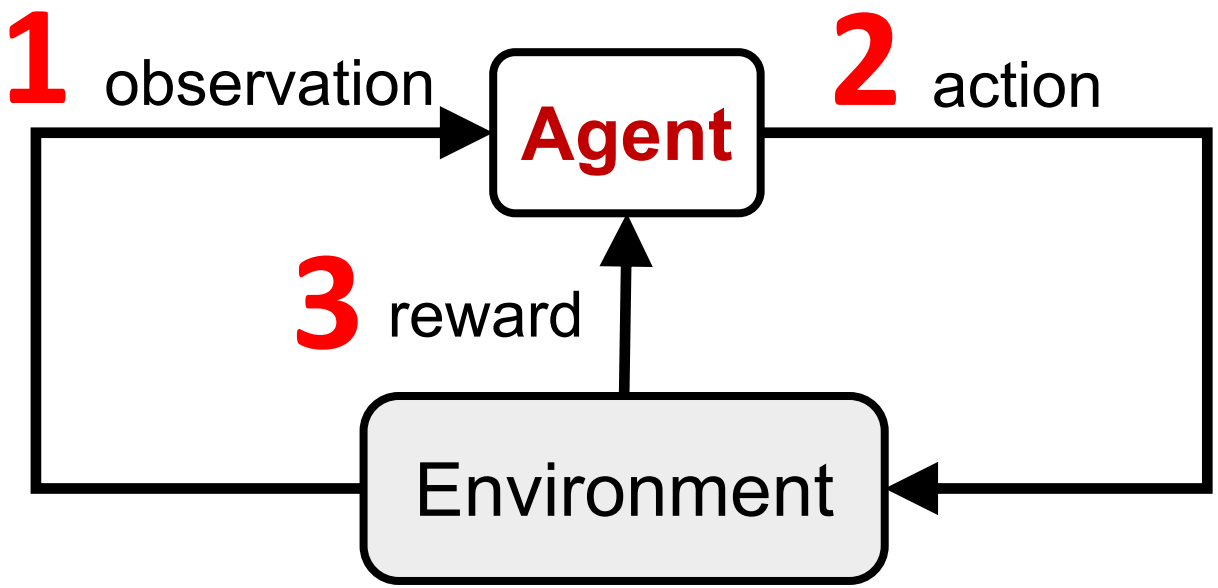
https://github.com/ageron/handson-ml3

Reinforcement Learning (DL)



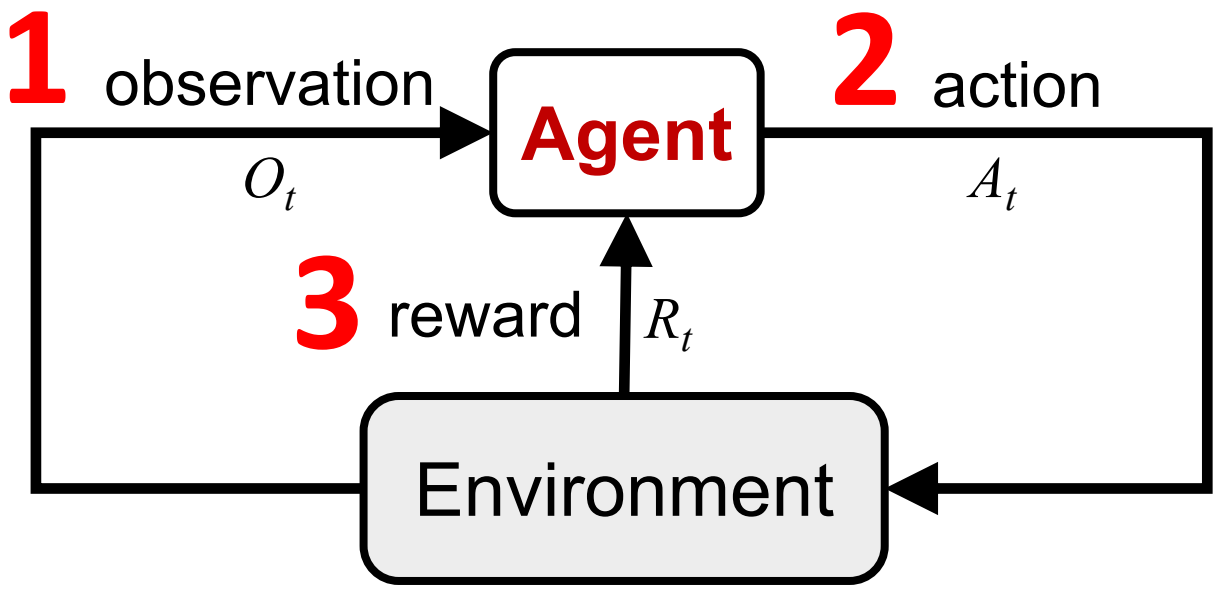
Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Reinforcement Learning (DL)



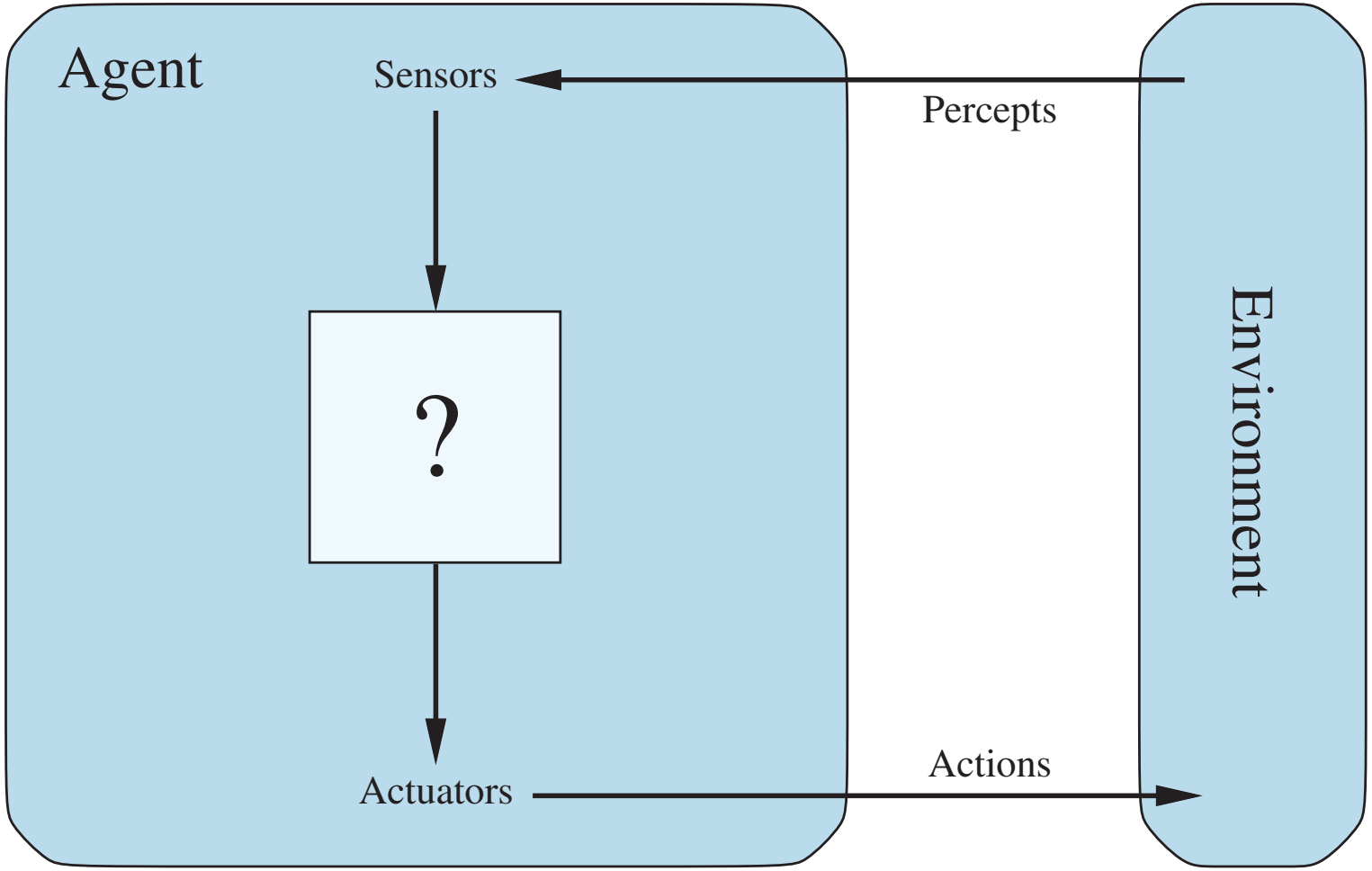
Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Reinforcement Learning (DL)



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

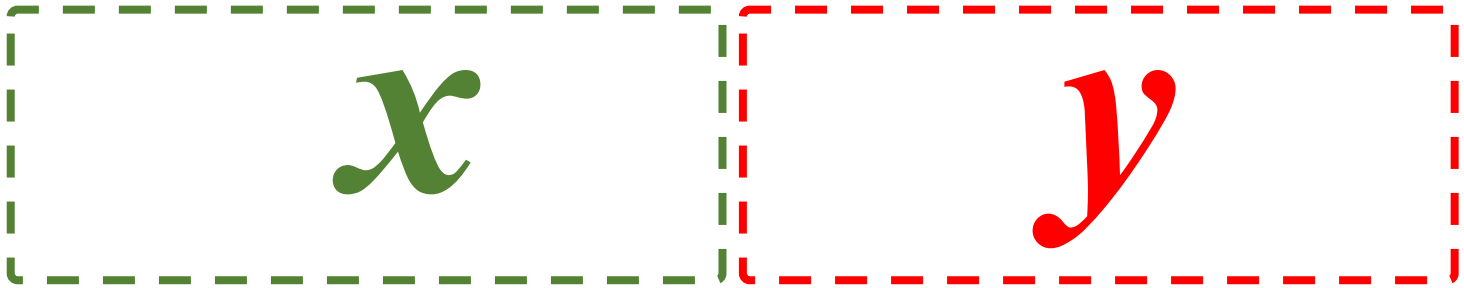
Agents interact with environments through sensors and actuators



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

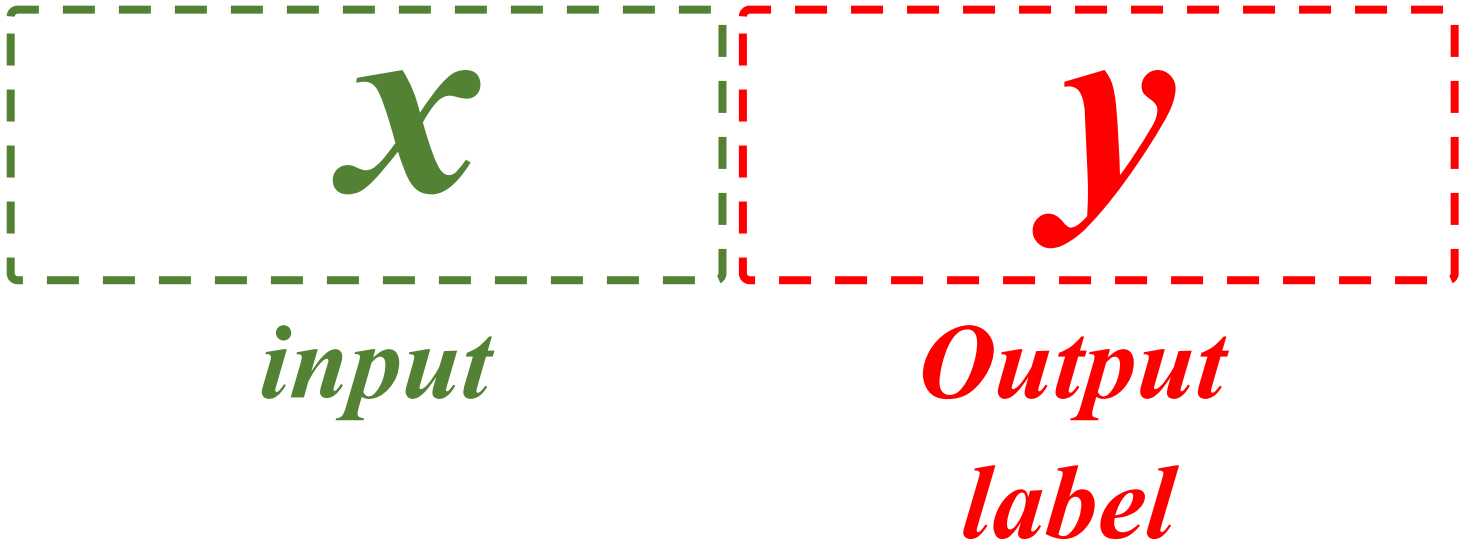
Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$



Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$



Iris flower data set

setosa

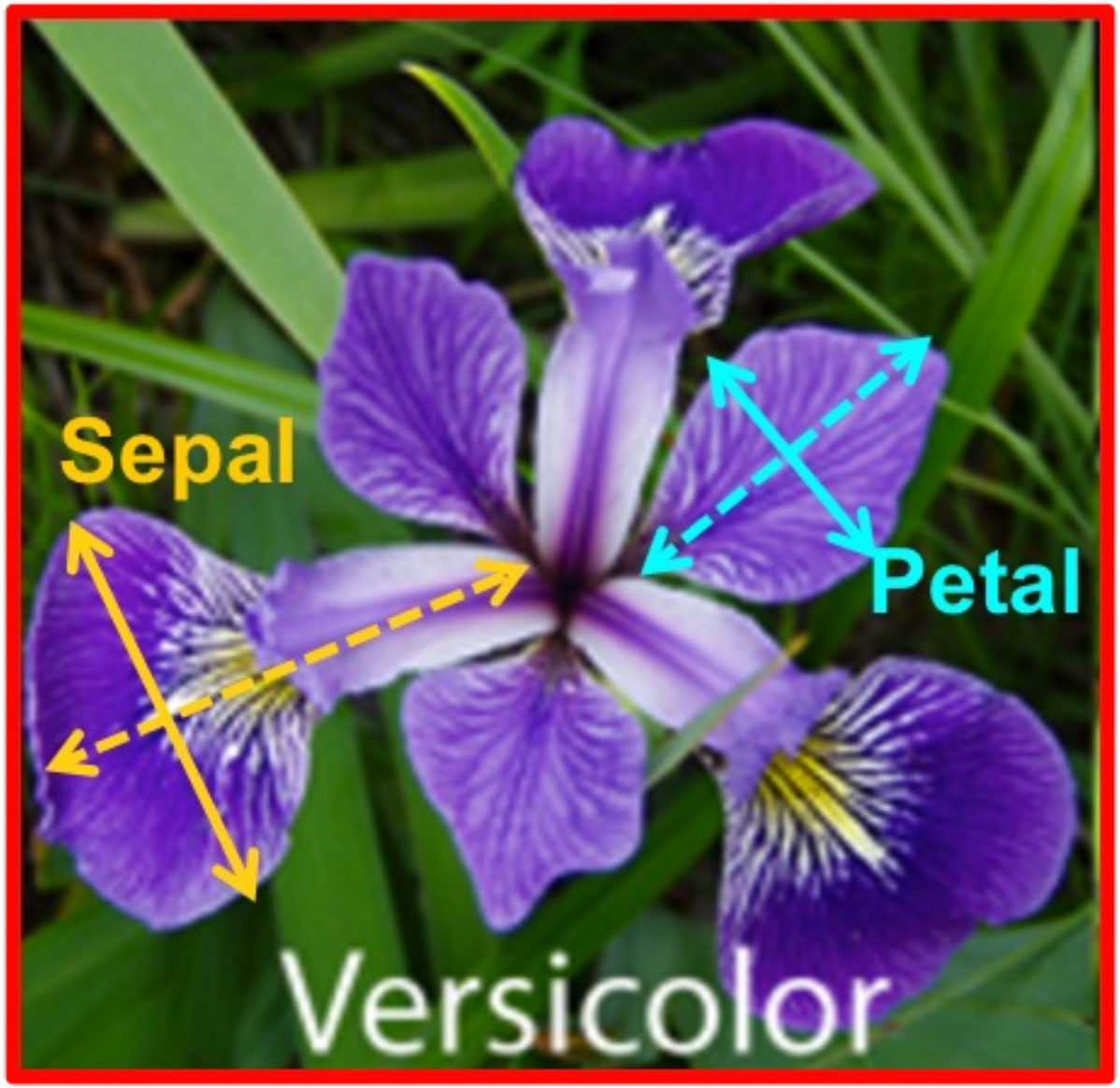
versicolor

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
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

$y = f(x)$

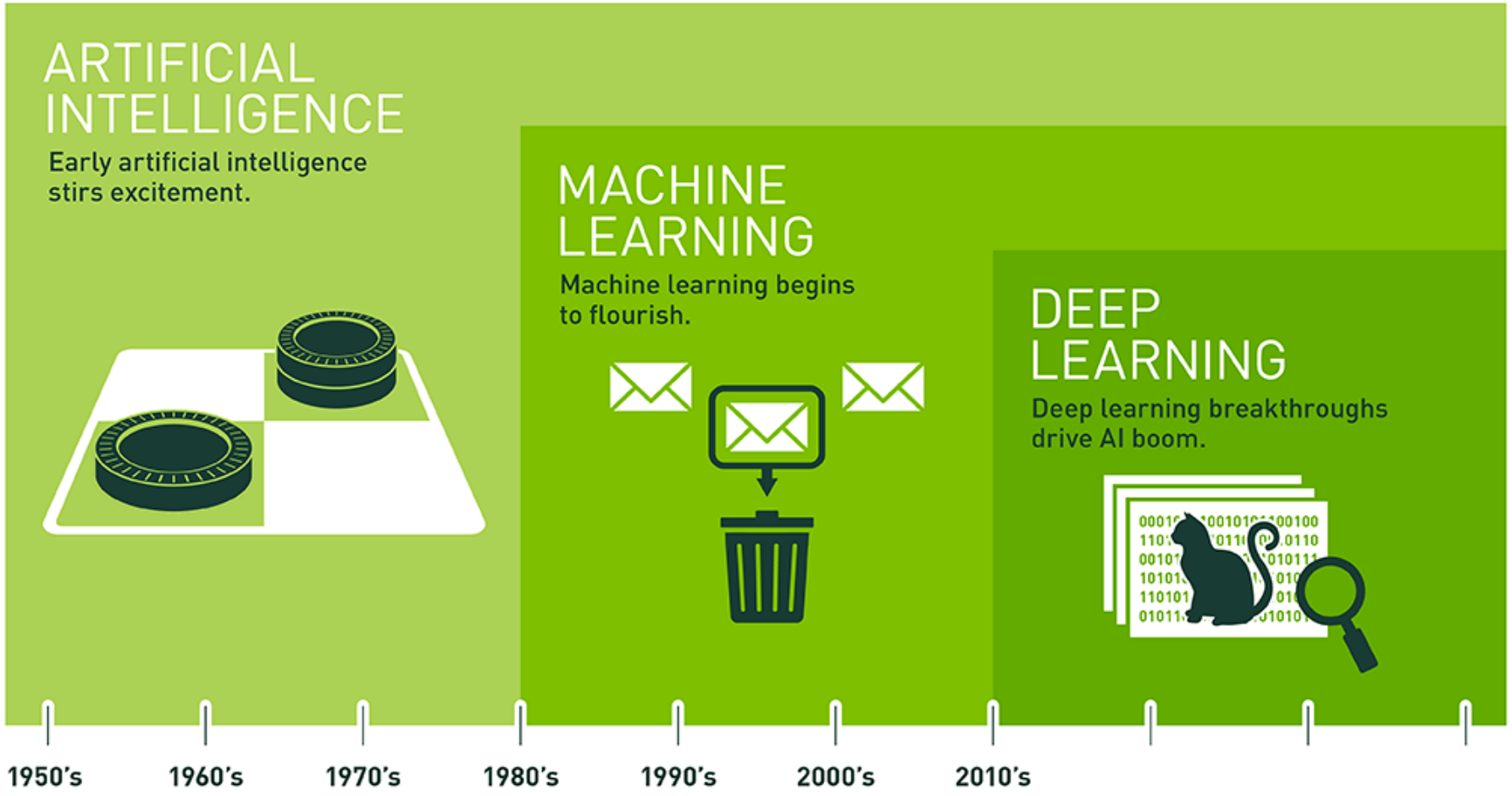
Example

```
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

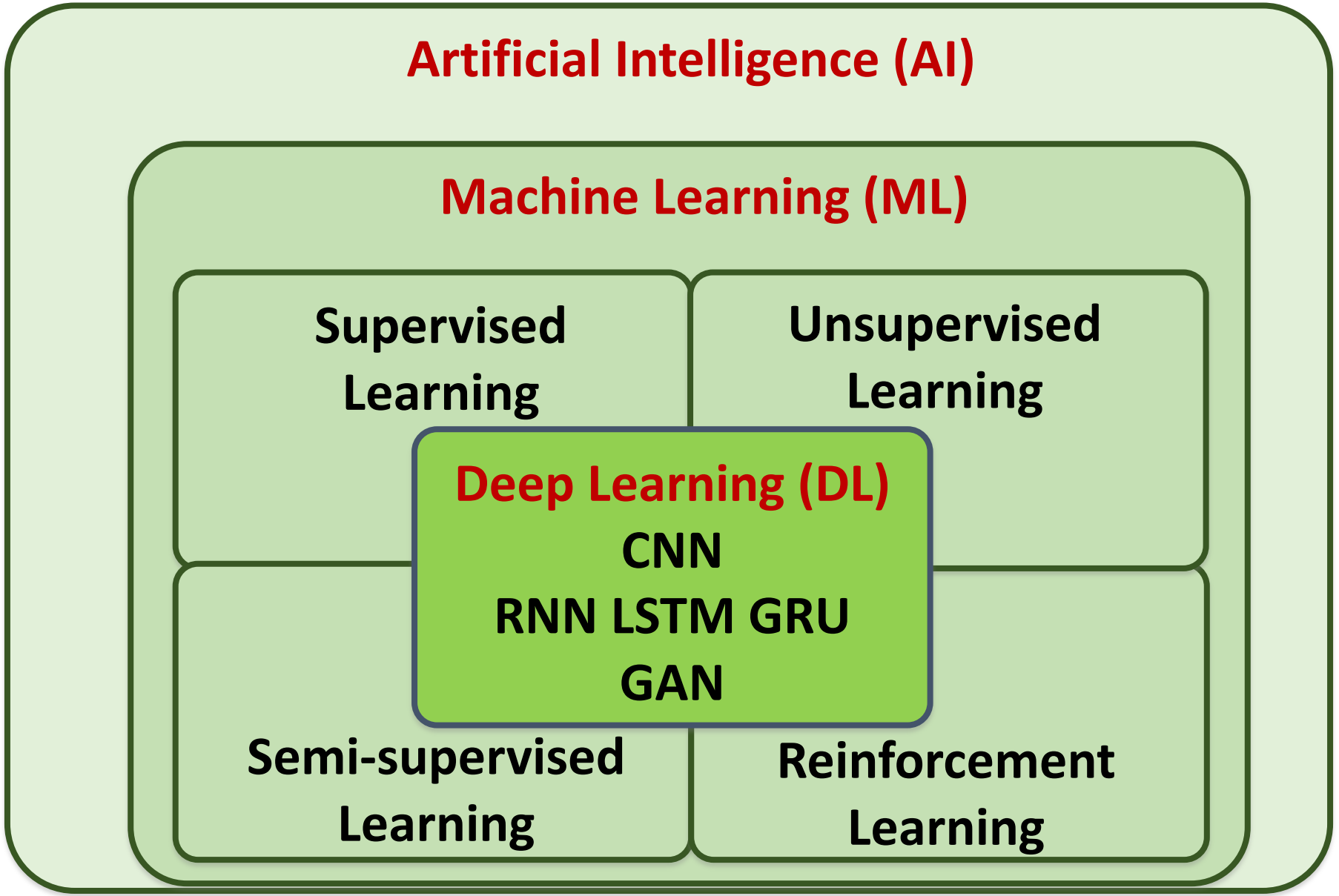
Artificial Intelligence
Machine Learning & Deep Learning



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.

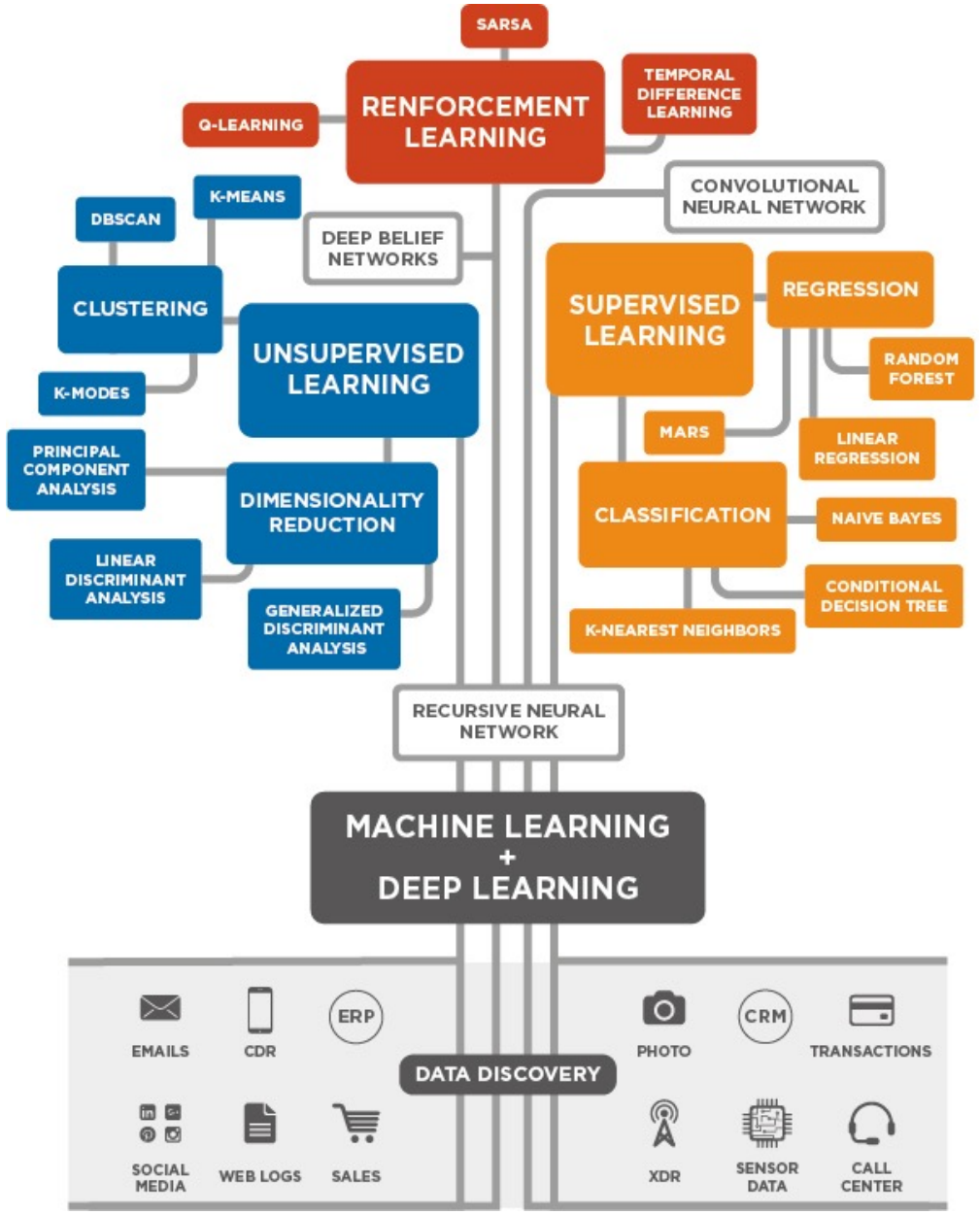
Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

AI, ML, DL



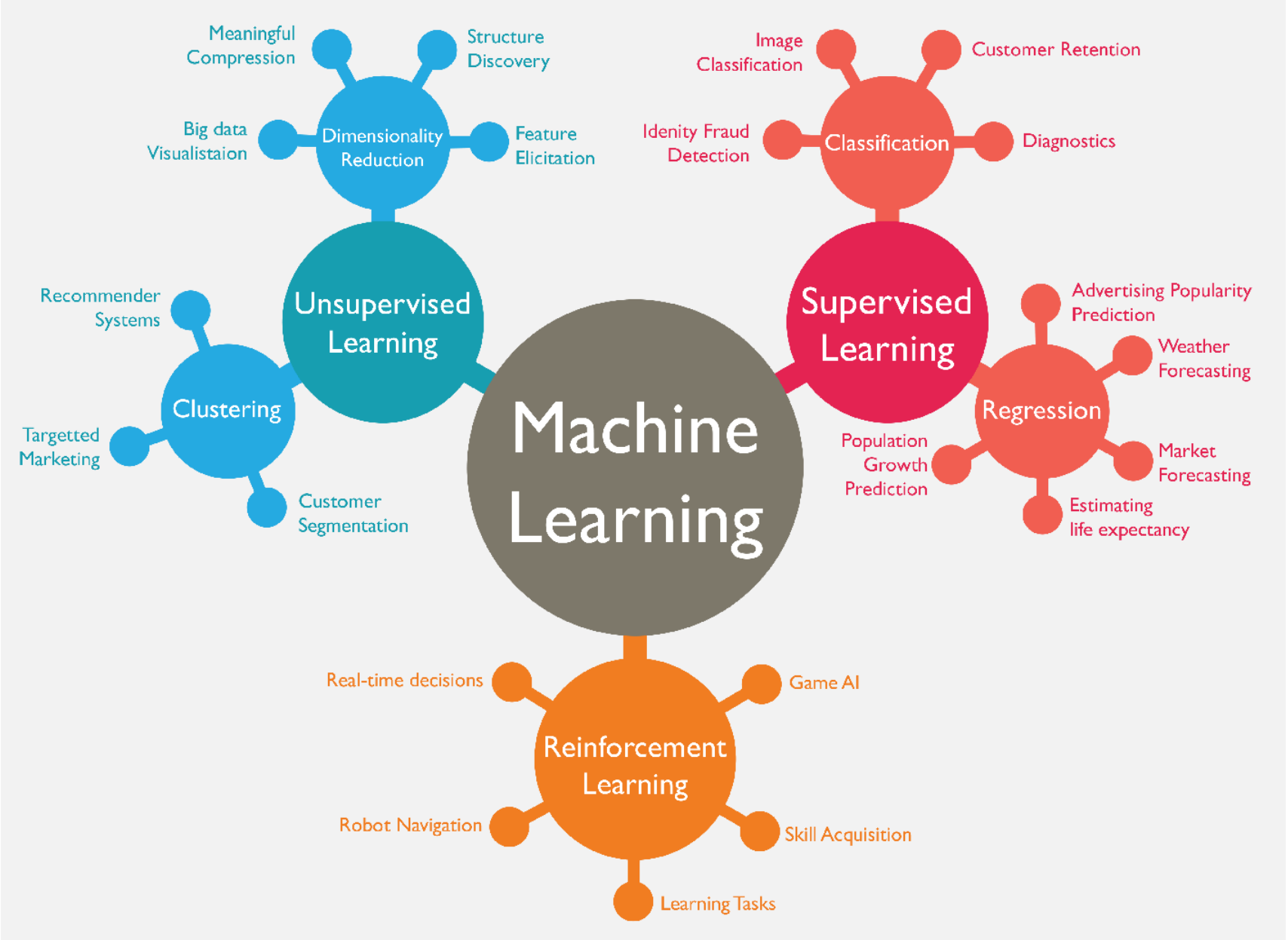
Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

3 Machine Learning Algorithms



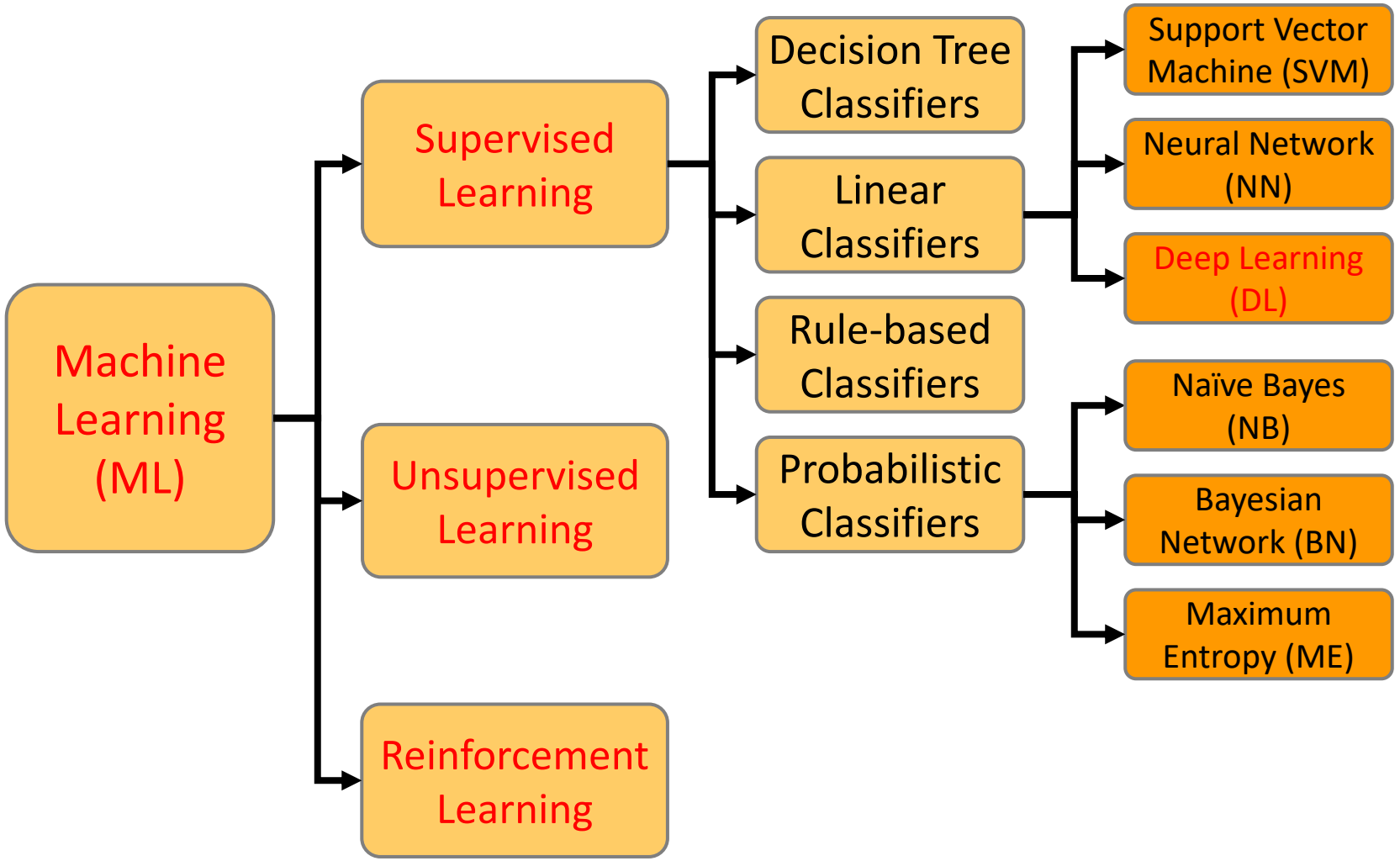
Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

Machine Learning (ML)



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

Machine Learning (ML) / Deep Learning (DL)



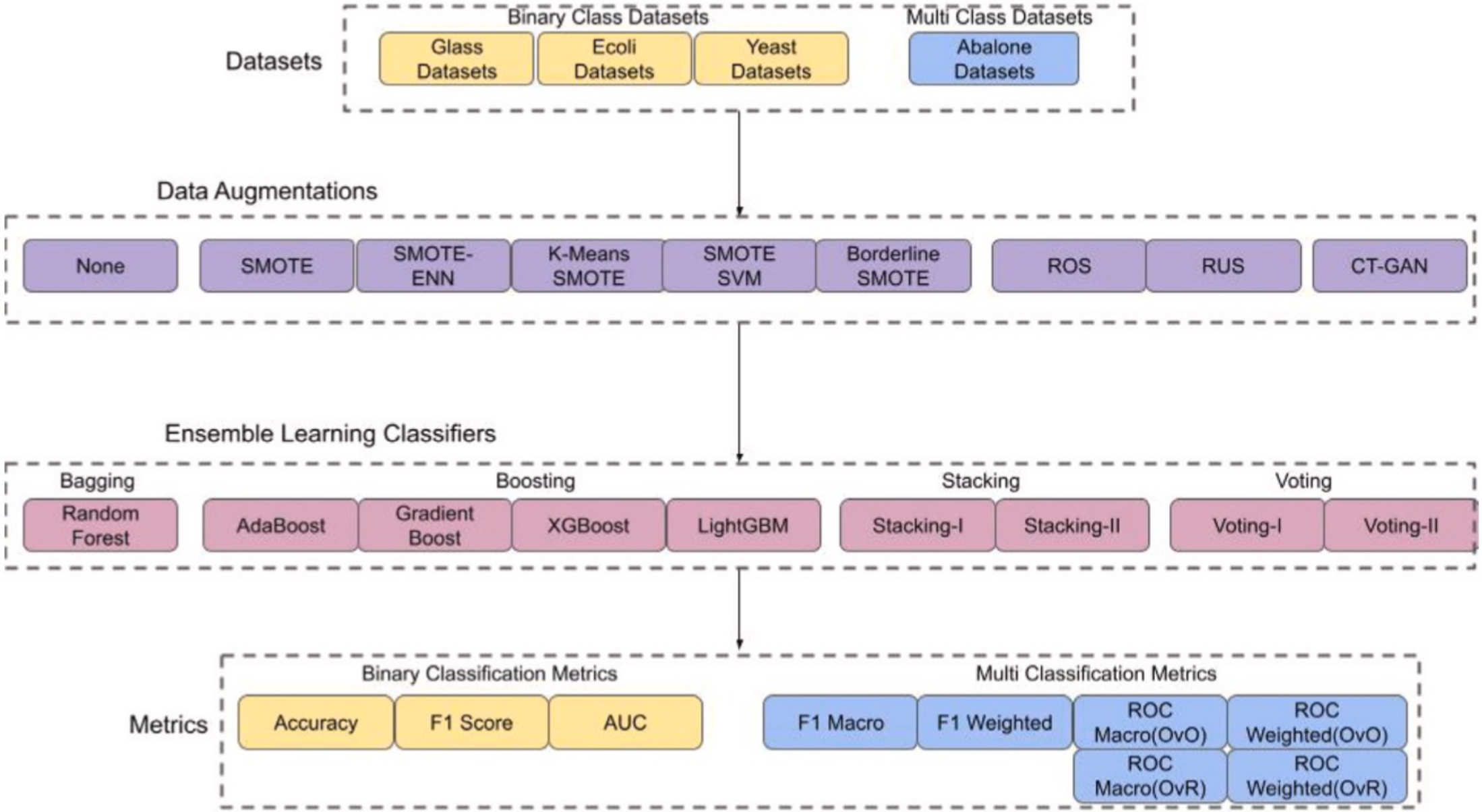
Source: Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero, and Enrique Herrera-Viedma (2015), "Sentiment analysis: A review and comparative analysis of web services," Information Sciences, 311, pp. 18-38.

Machine Learning Models

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

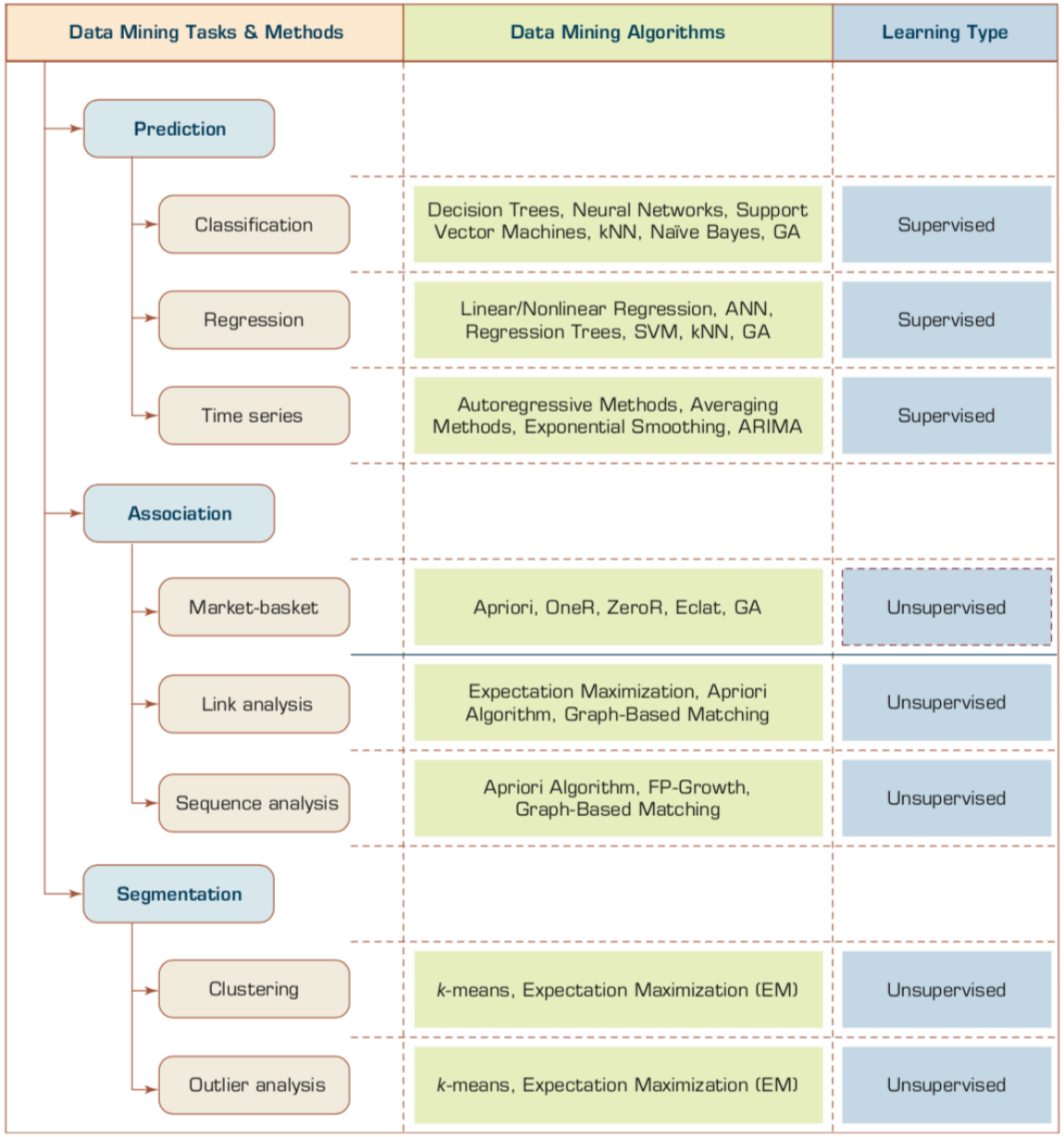
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

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



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**

Scikit-Learn

Machine Learning in Python

Scikit-Learn

scikit-learn

Machine Learning in Python

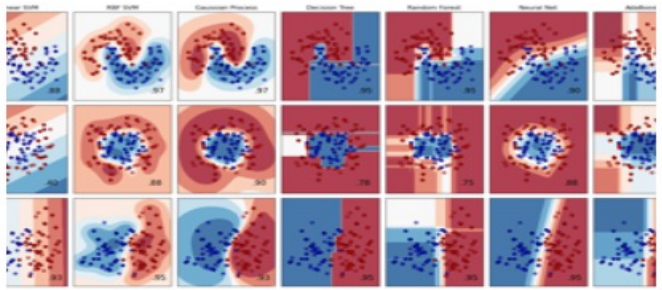
Getting Started | Release Highlights for 1.5

- 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...

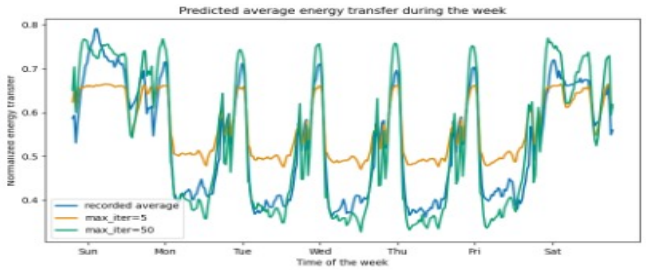


Examples

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...

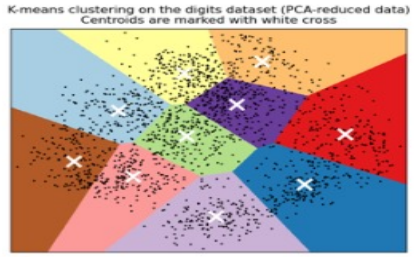


Examples

Clustering

Automatic grouping of similar objects into sets.

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



Examples

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...

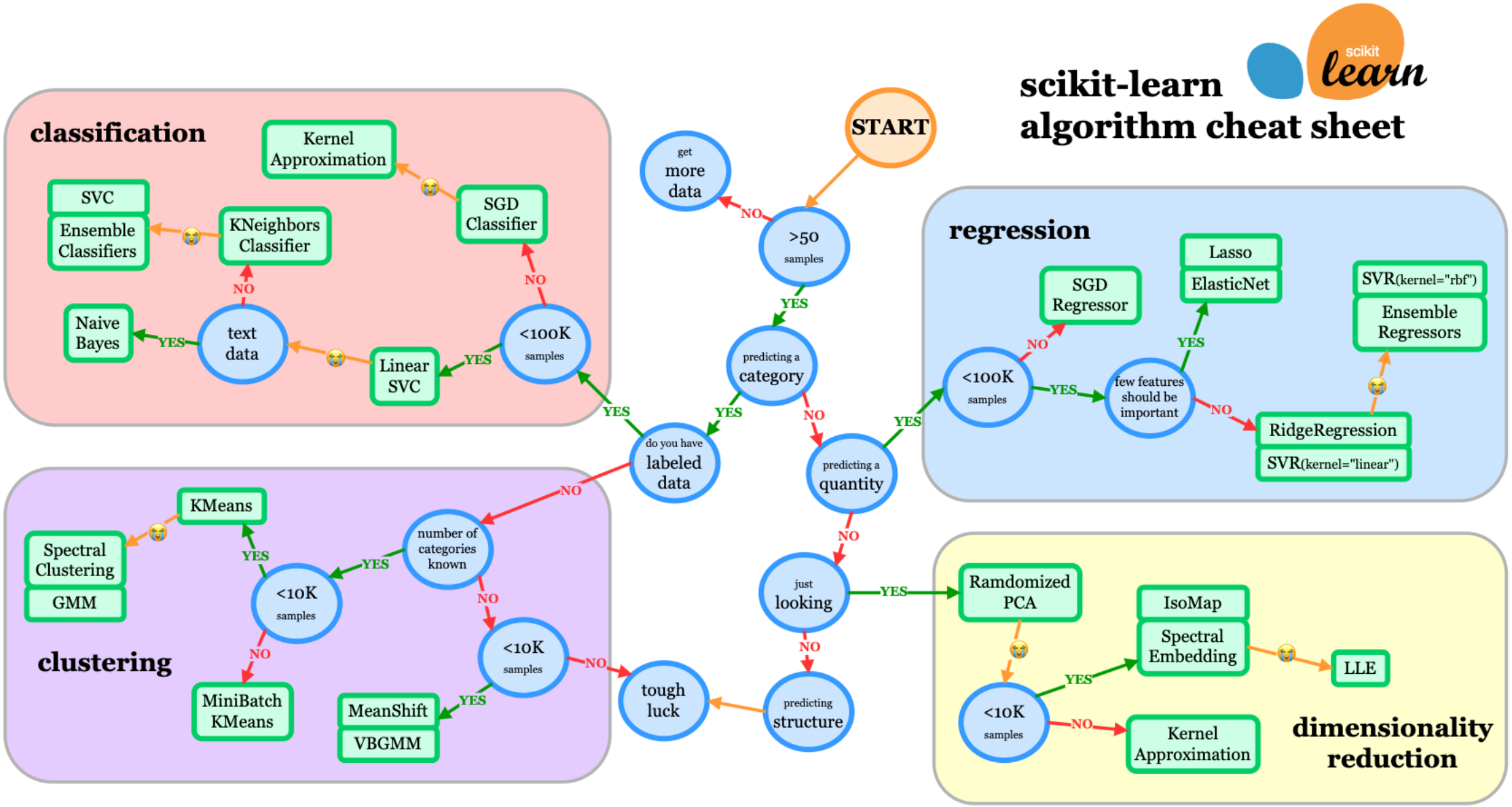
Preprocessing

Feature extraction and normalization.

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

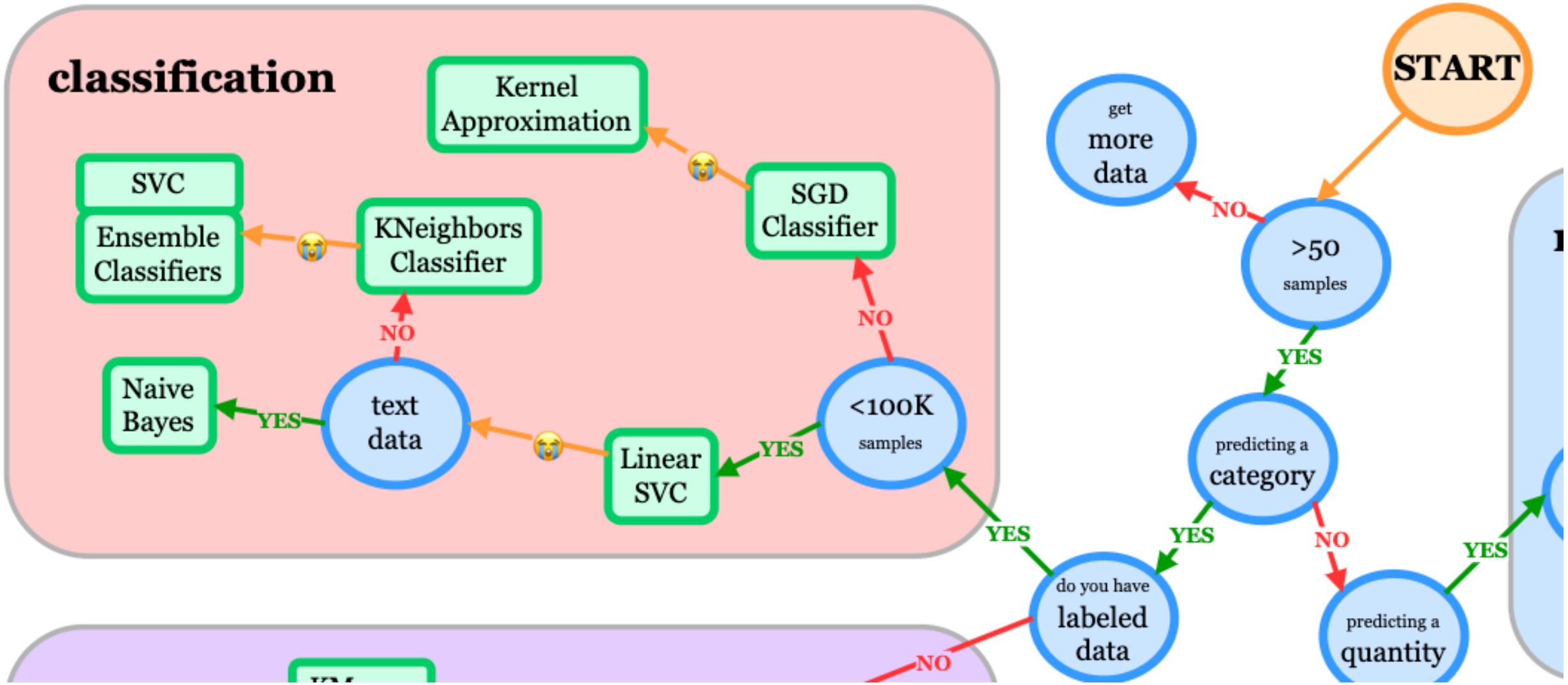
Source: http://scikit-learn.org/

Scikit-Learn Machine Learning Map



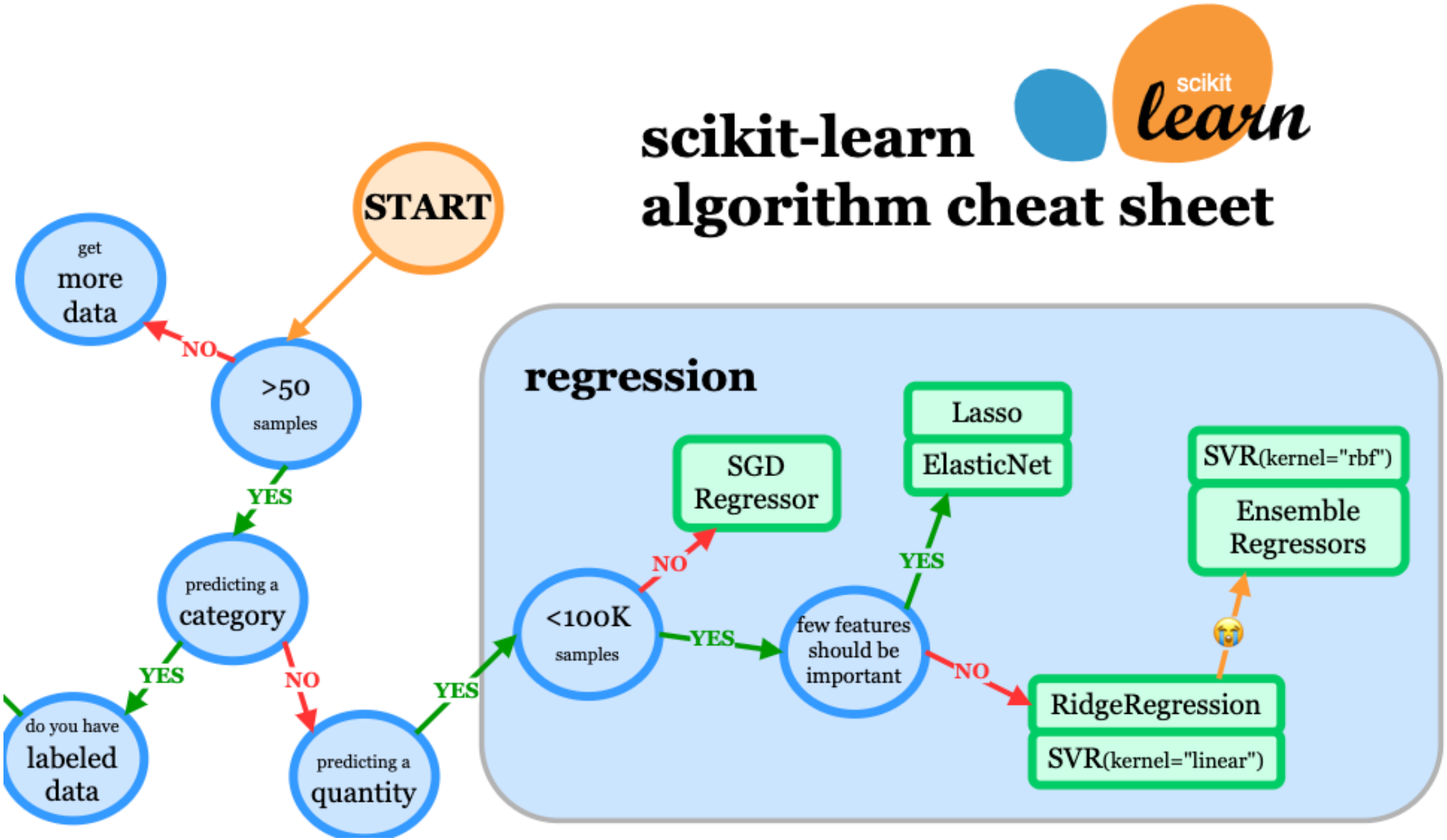
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



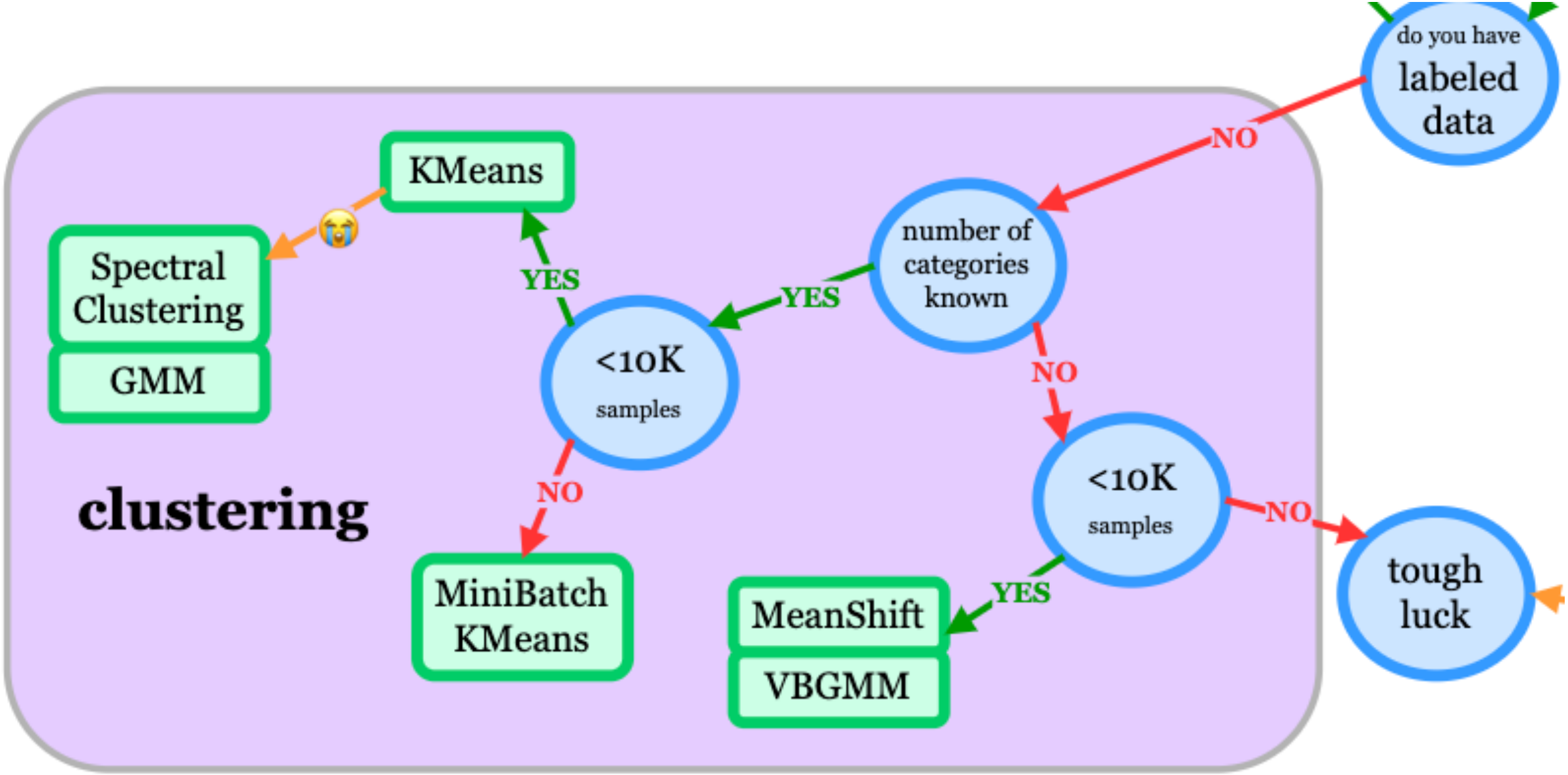
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



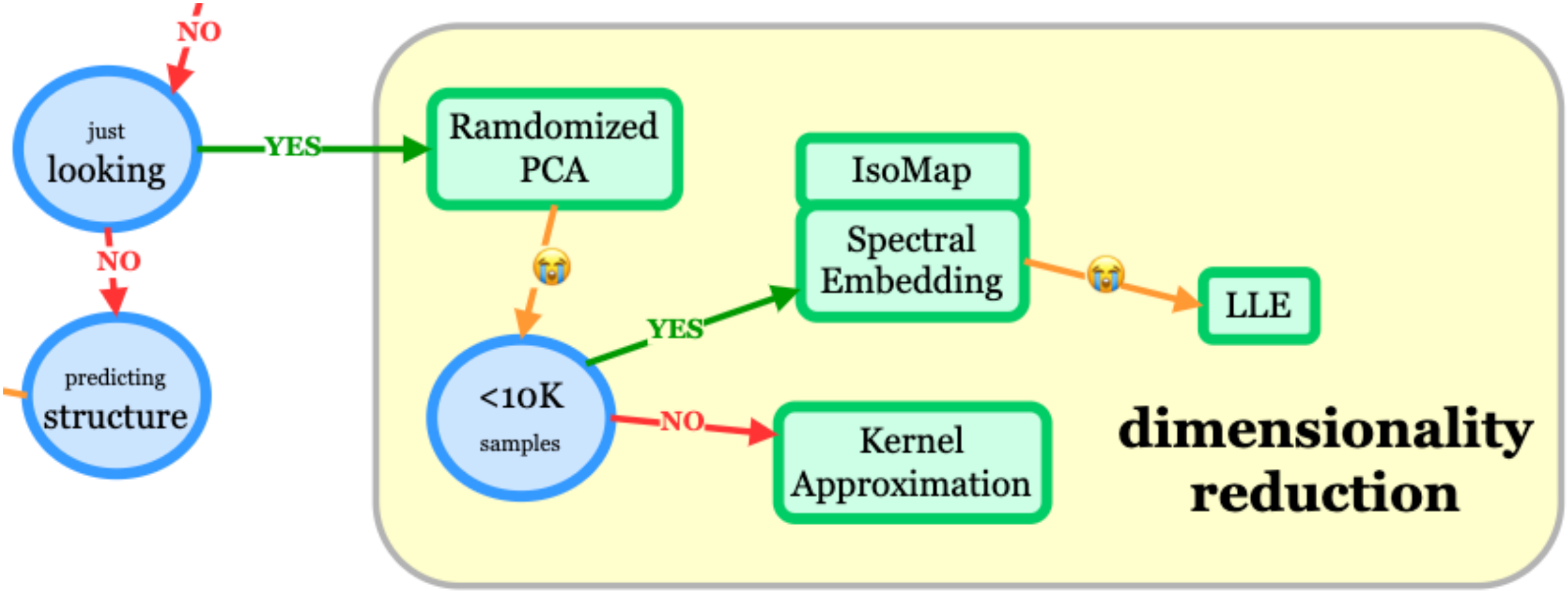
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

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

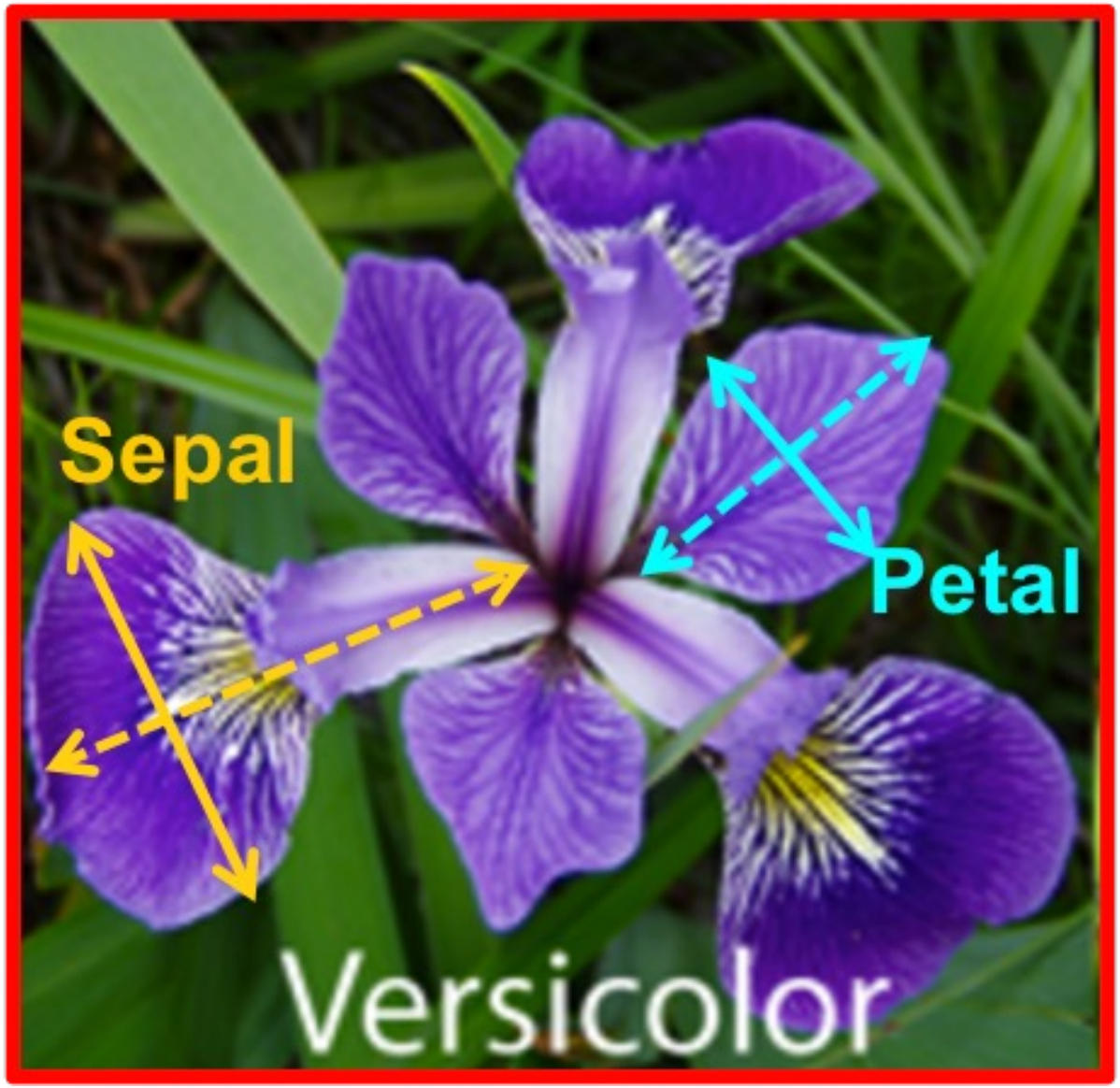
versicolor

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

$y = f(x)$

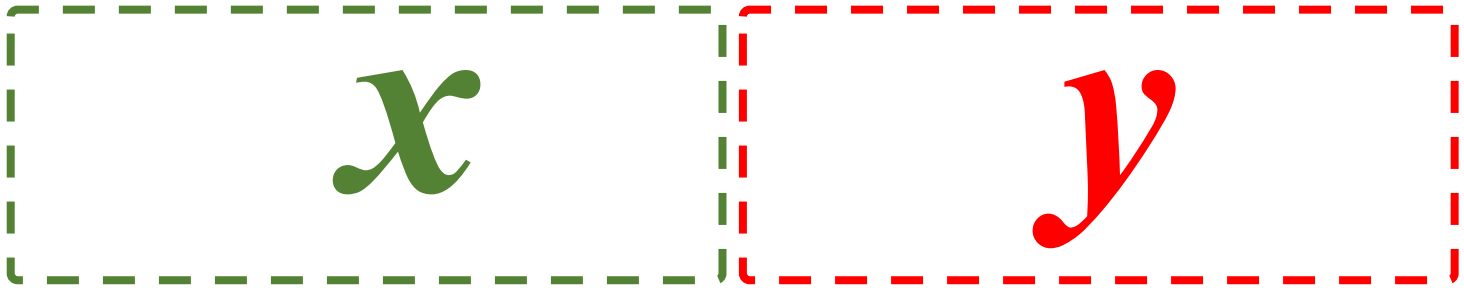
x

```
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
```

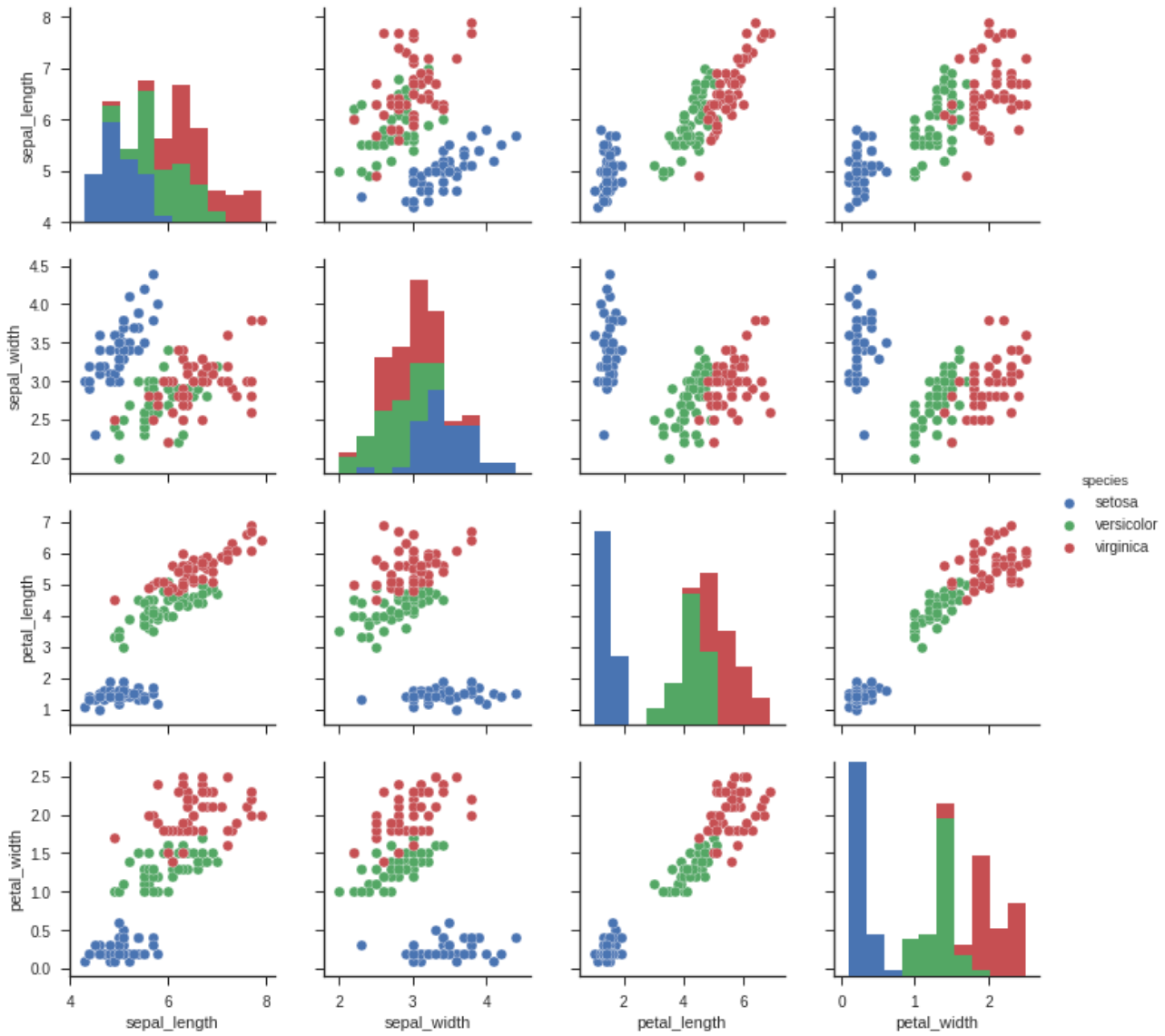
y

Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$



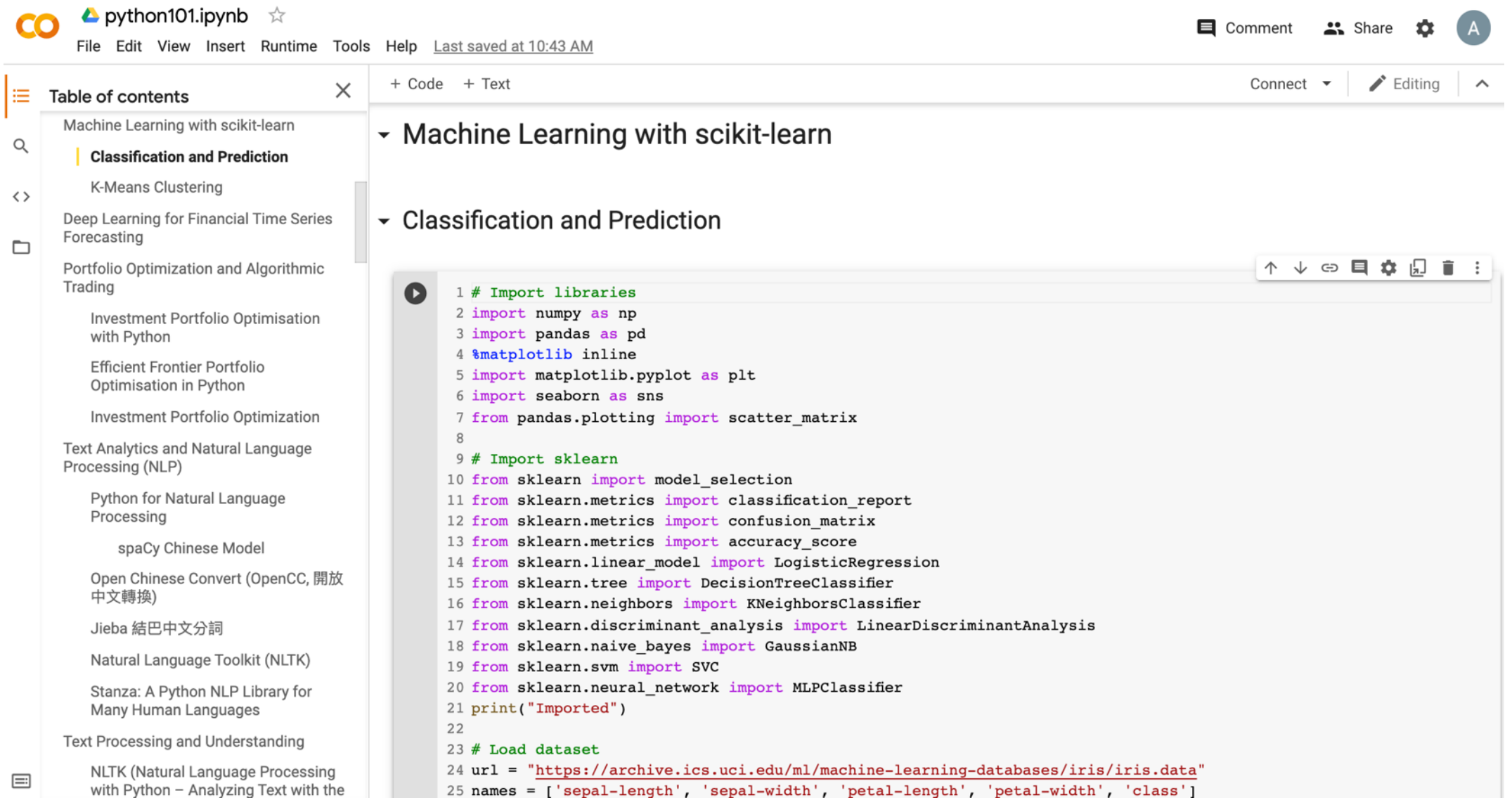
Iris Data Visualization



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

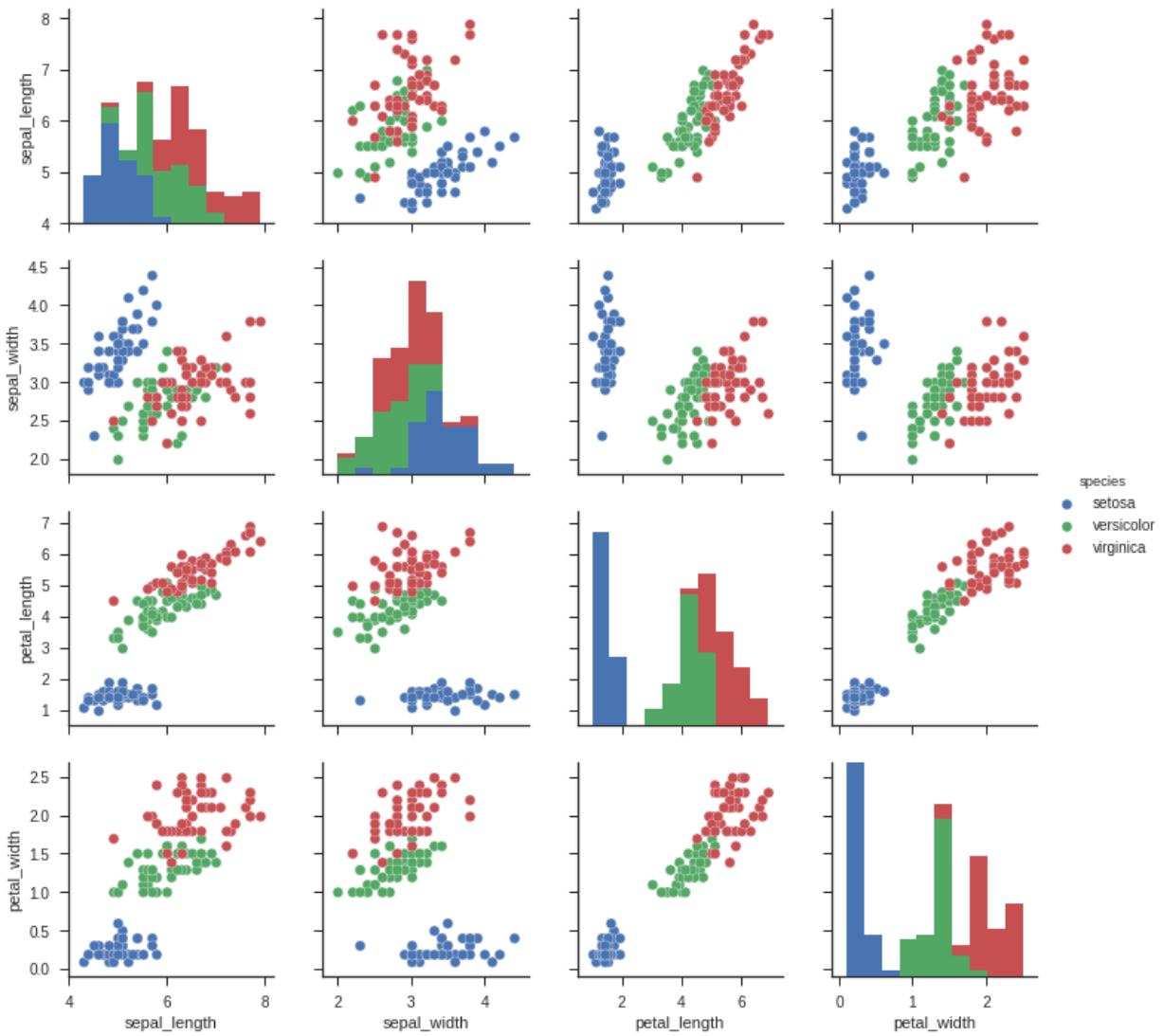
Python in Google Colab (Python101)

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



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)
```

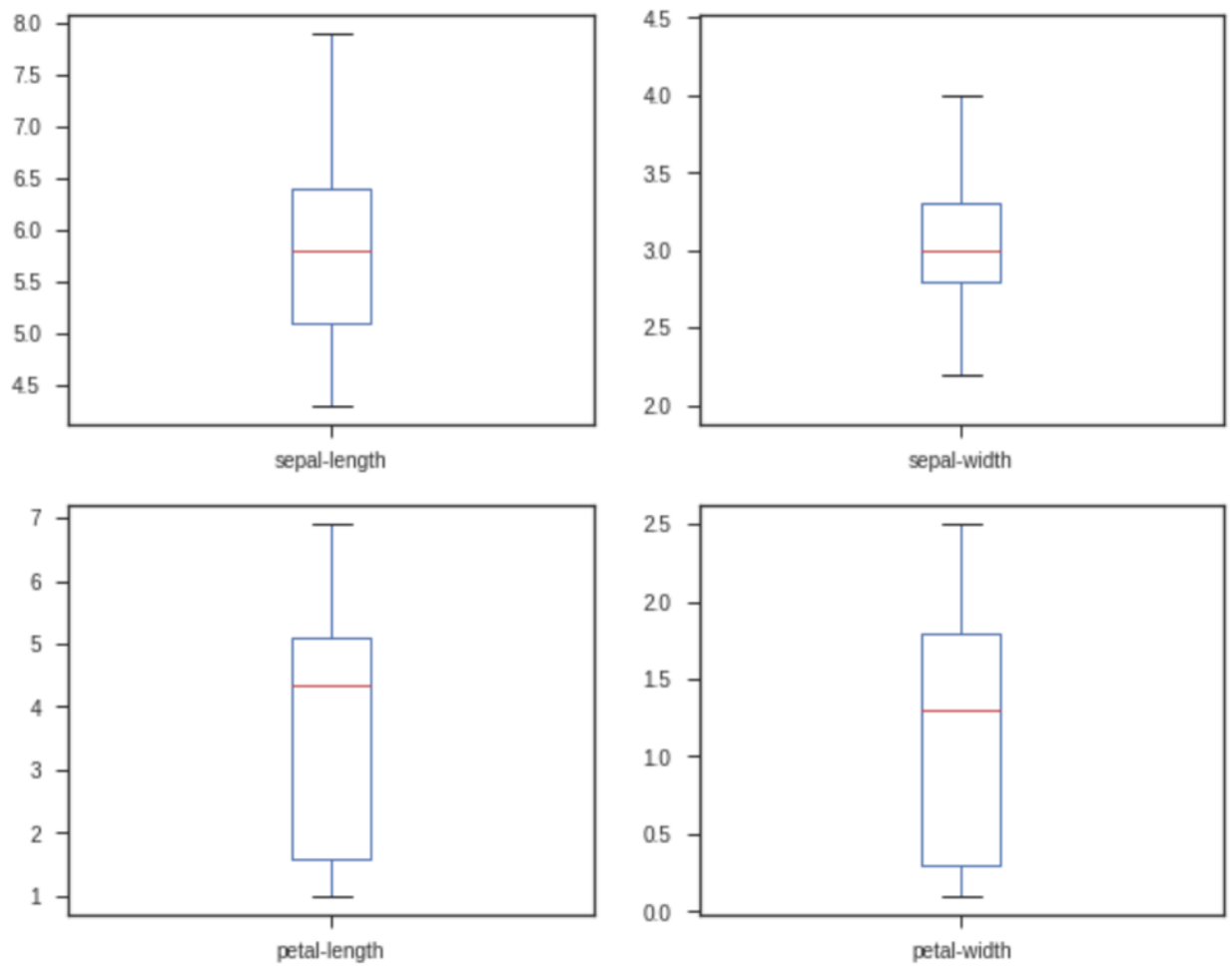
```
(150, 5)
```

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

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

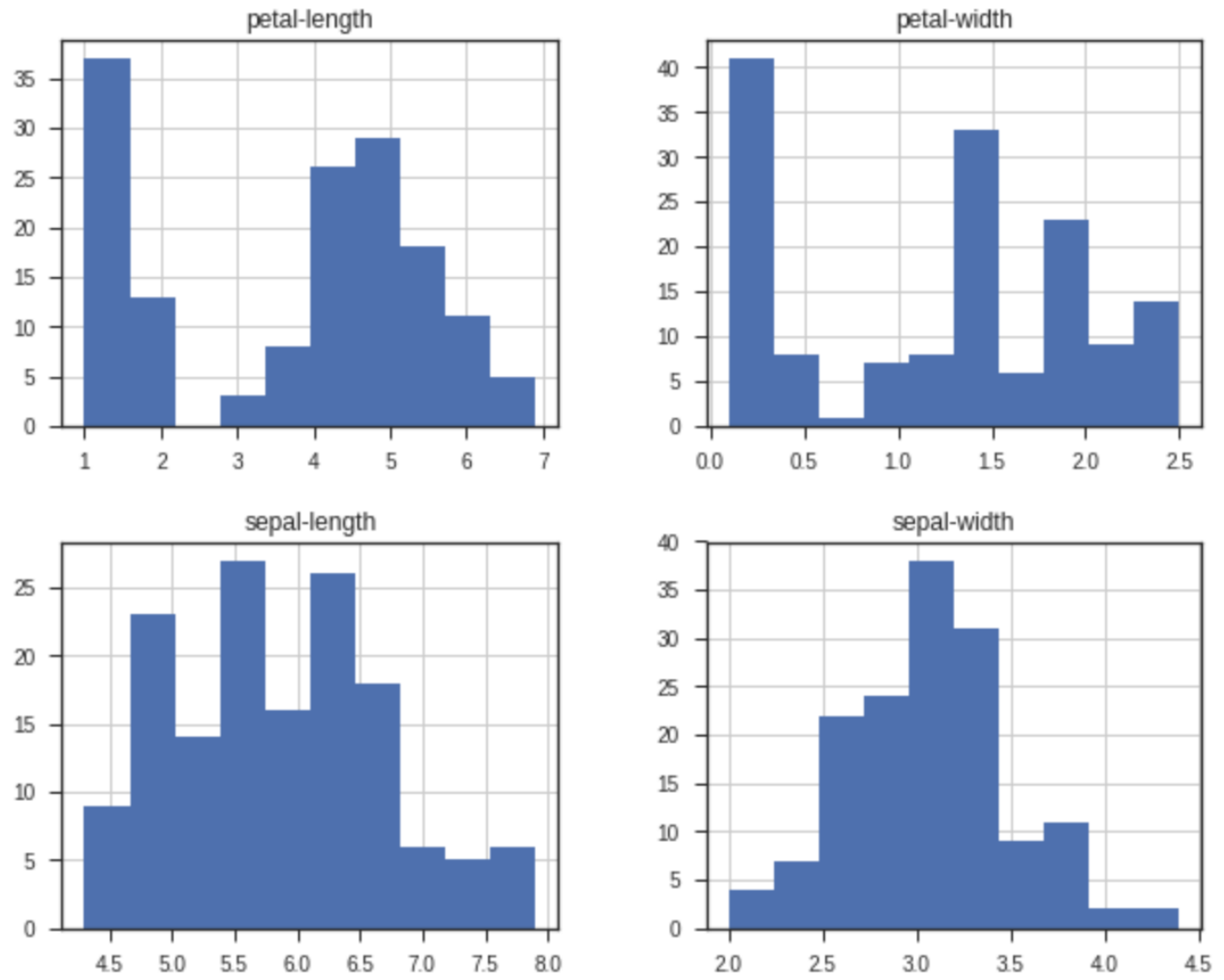
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

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



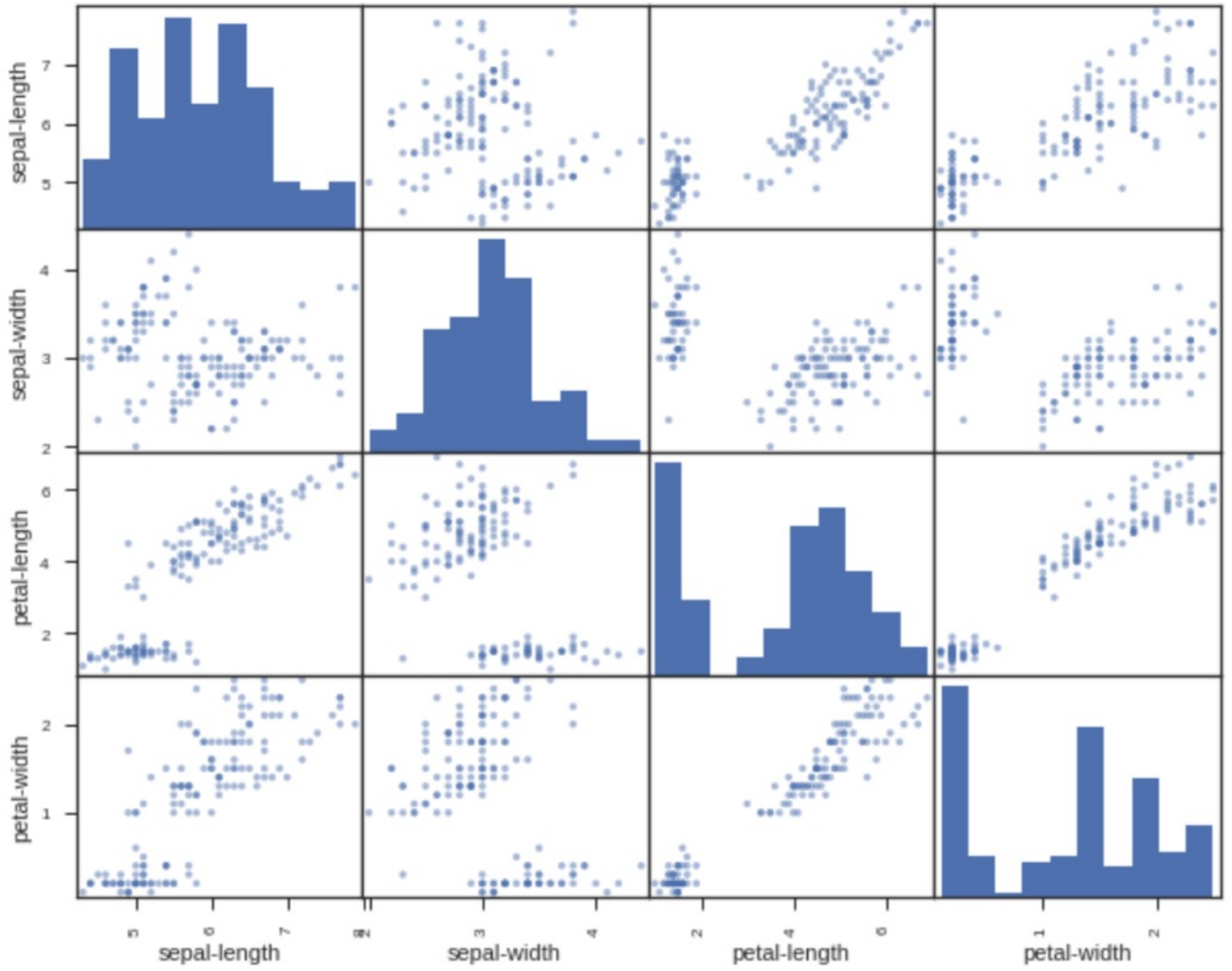
df.hist()
plt.show()

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



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

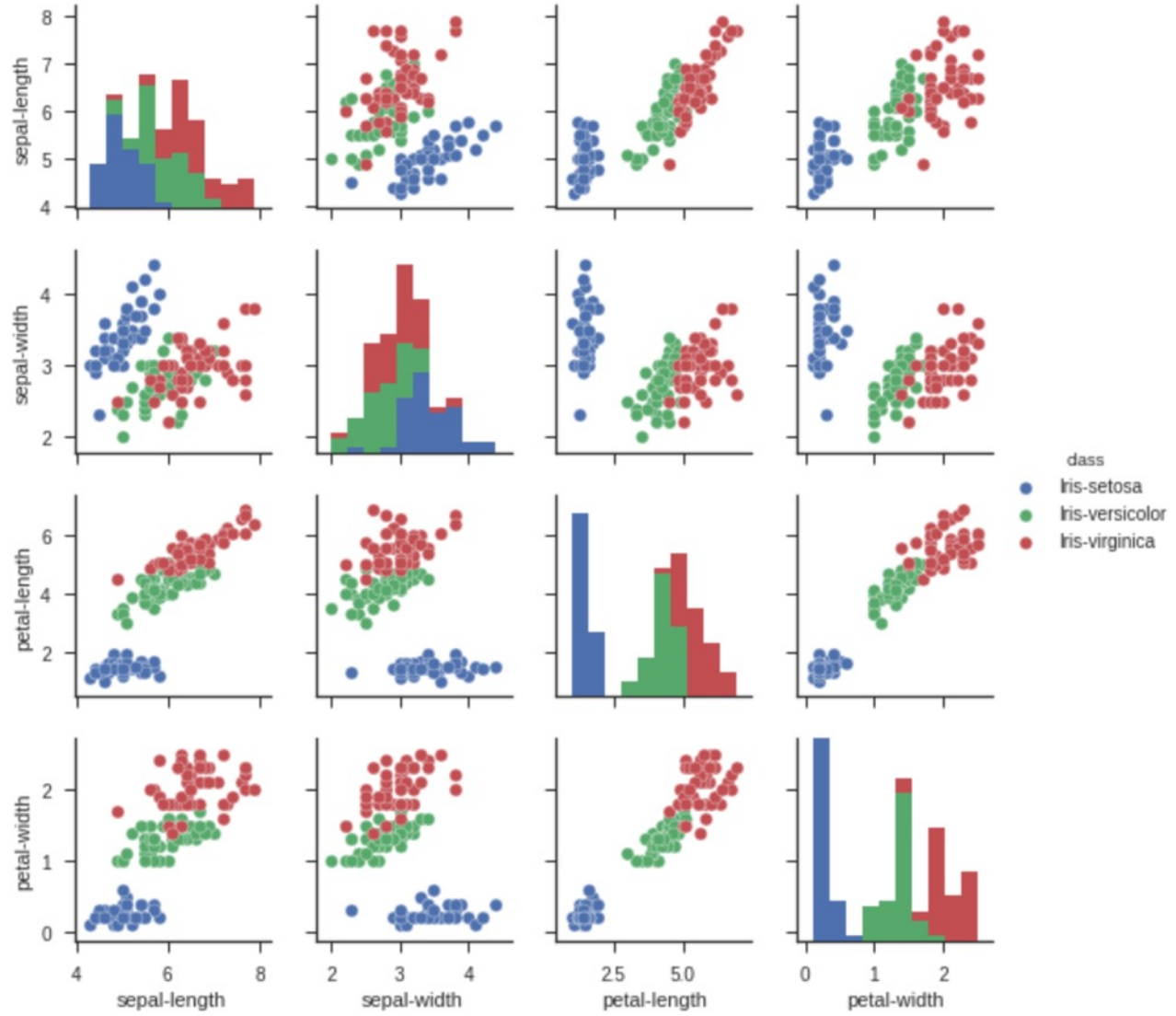
```
scatter_matrix(df)
plt.show()
```



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

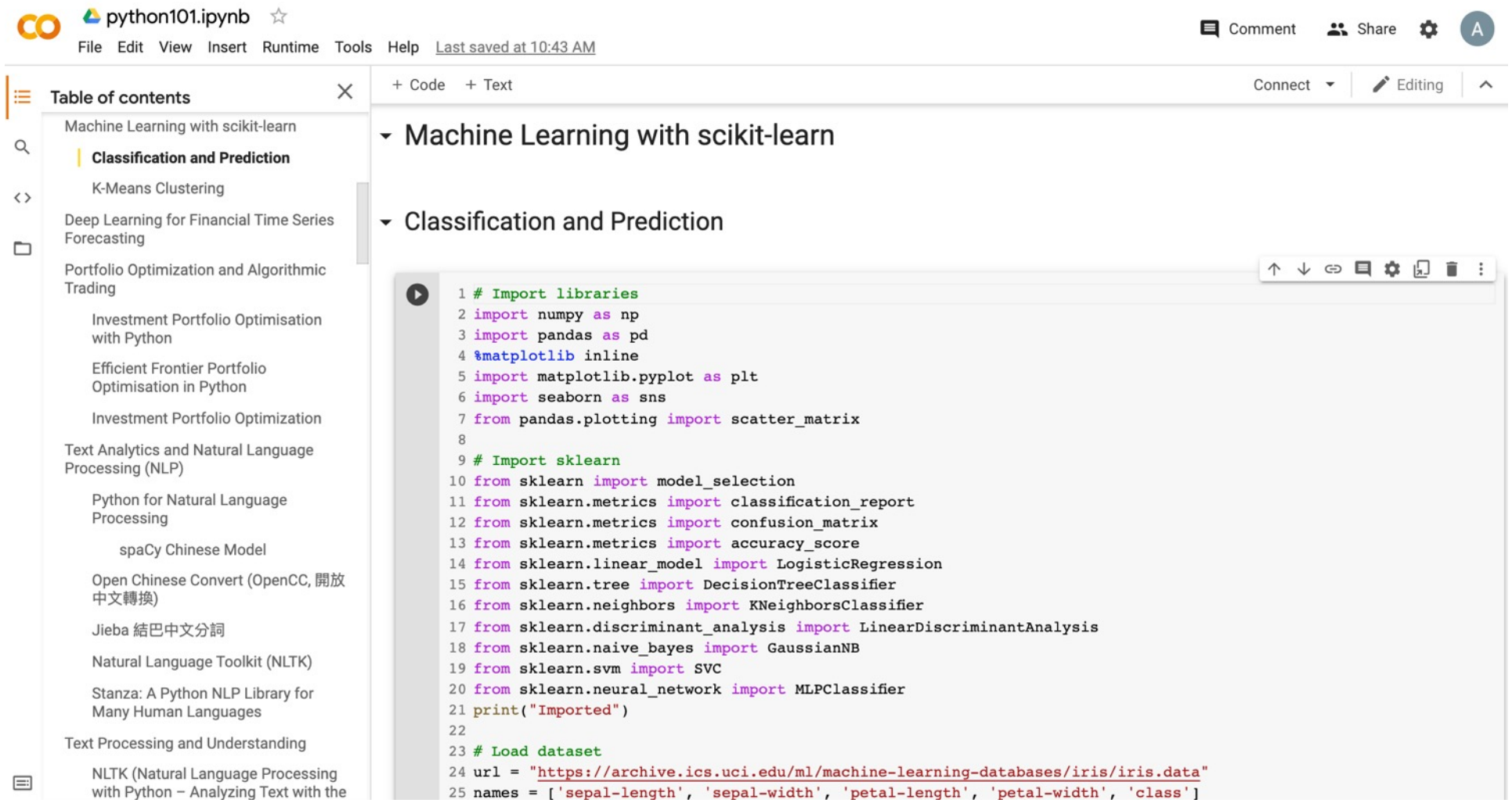
Prediction
Classification

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



https://tinyurl.com/aintpupython101

```
# 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")
```

https://tinyurl.com/aintpupython101

```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()

```
1 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

https://tinyurl.com/aintpupython101

```
# 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
```

https://tinyurl.com/aintpupython101

```
# 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()))
```

https://tinyurl.com/aintpupython101

```
# 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)
```

https://tinyurl.com/aintpupython101

```
# 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

```
# 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)
```

```
1 # Make predictions on validation dataset
2 model = SVC()
3 model.fit(X_train, Y_train)
4 predictions = model.predict(X_validation)
5 print("%.4f" % accuracy_score(Y_validation, predictions))
6 print(confusion_matrix(Y_validation, predictions))
7 print(classification_report(Y_validation, predictions))
8 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

```
# 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

```
1 # Make predictions on validation dataset
2 model = GaussianNB()
3 model.fit(X_train, Y_train)
4 predictions = model.predict(X_validation)
5 print("%.4f" % accuracy_score(Y_validation, predictions))
6 print(confusion_matrix(Y_validation, predictions))
7 print(classification_report(Y_validation, predictions))
8 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)
```

https://tinyurl.com/aintpupython101

```
# 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

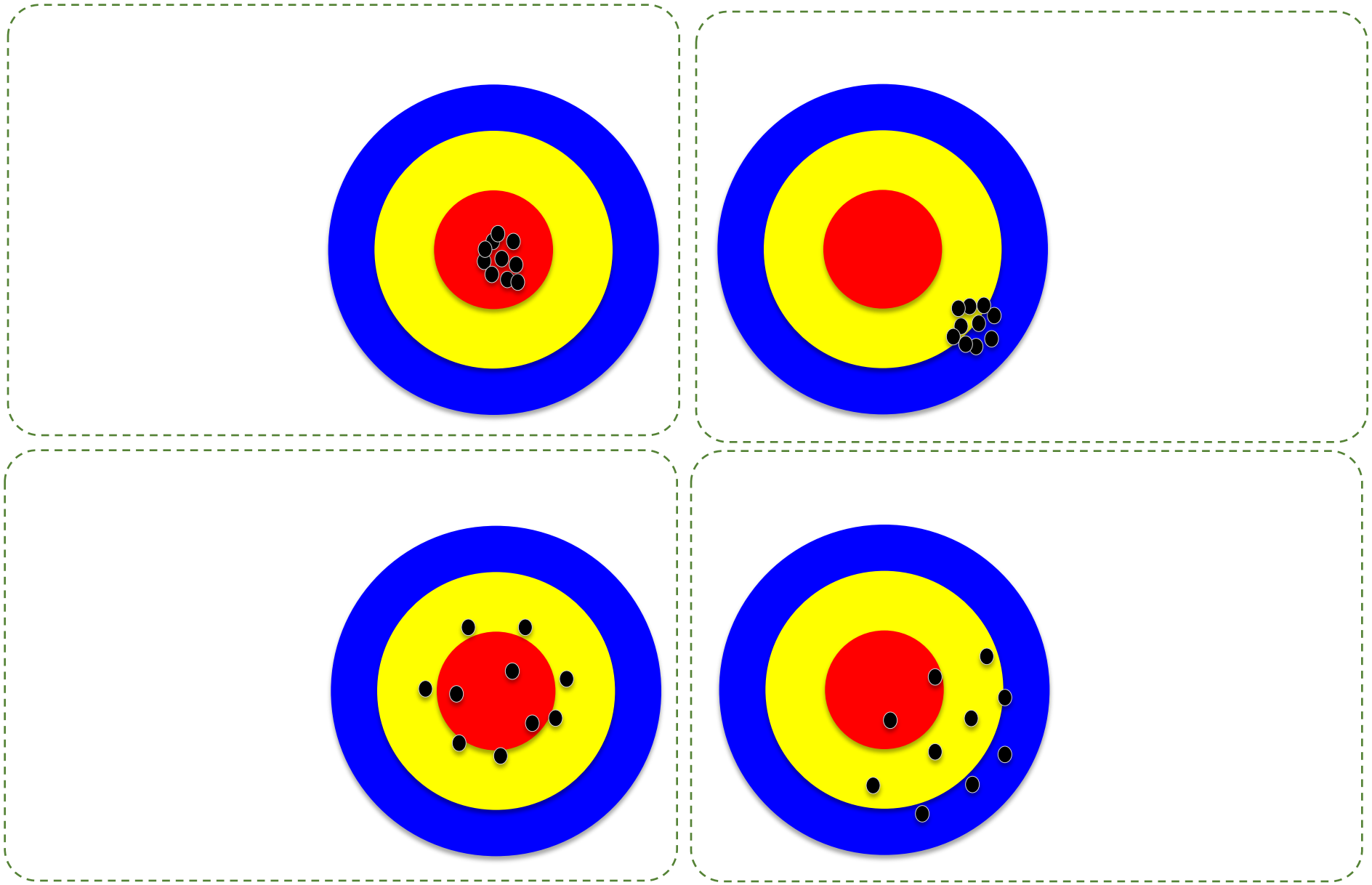
Evaluation
(Accuracy of Classification Model)

Assessing the Classification Model

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

Accuracy **Validity**

Precision **Reliability**



Accuracy vs. Precision

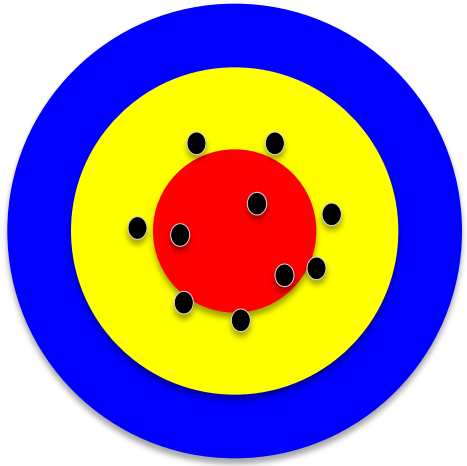
A

High Accuracy
High Precision

B

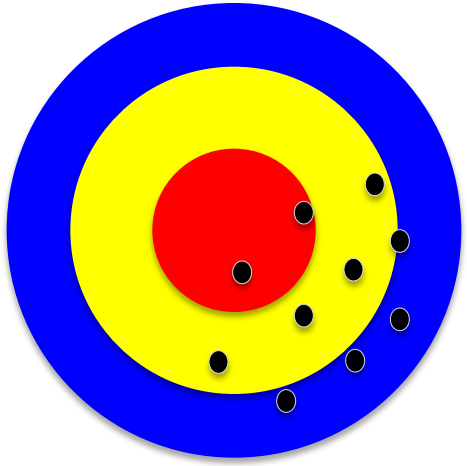
Low Accuracy
High Precision

C



High Accuracy
Low Precision

D



Low Accuracy
Low Precision

Accuracy vs. Precision

A

High Accuracy
High Precision

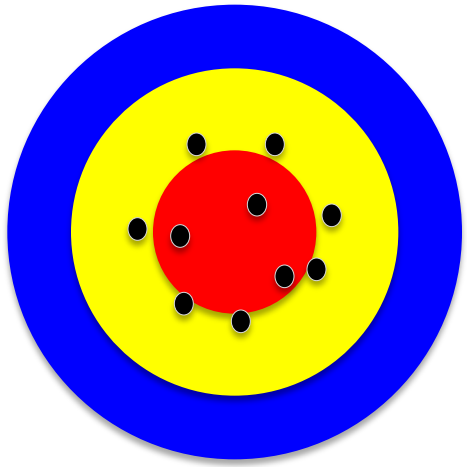
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

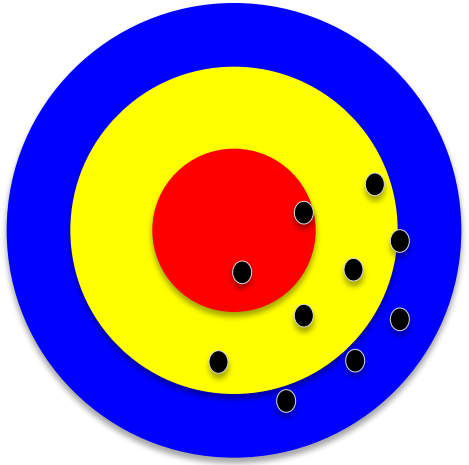
C



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

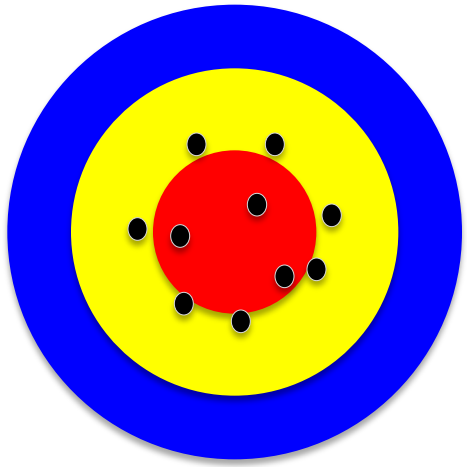
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

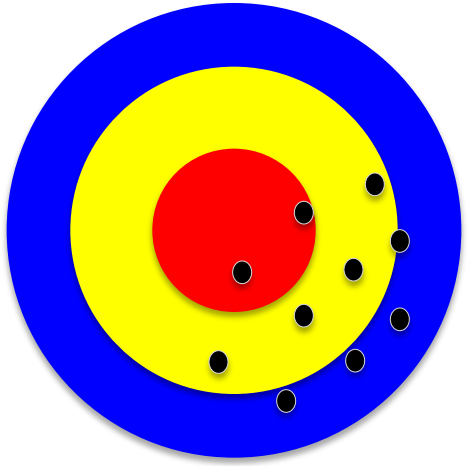
C



High Accuracy
Low Precision

High Validity
Low Reliability

D

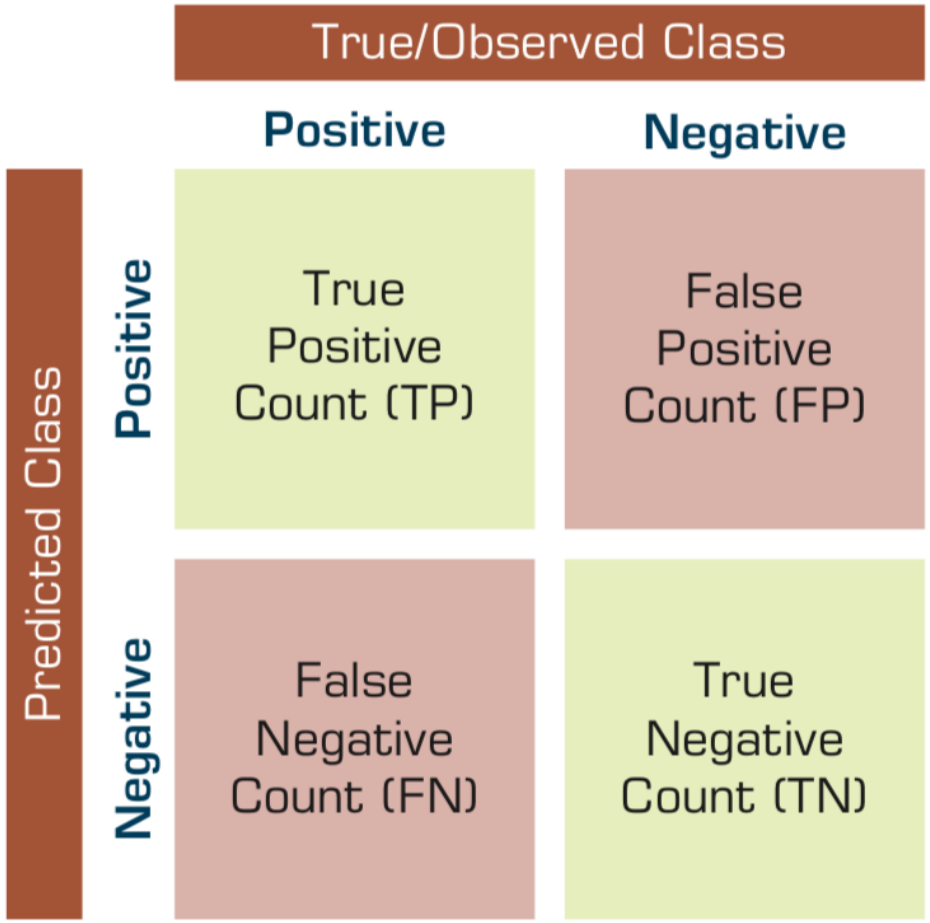


Low Accuracy
Low Precision

Low Validity
Low Reliability

Confusion Matrix

for Tabulation of Two-Class Classification Results



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

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

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

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

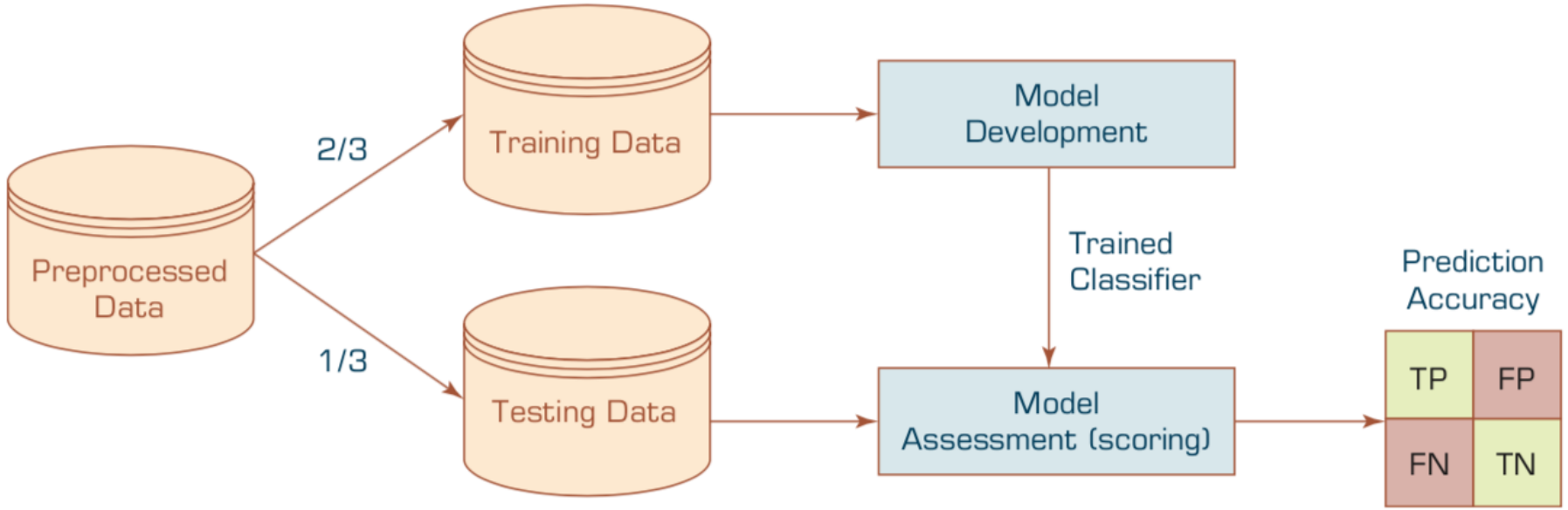
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Sensitivity =True Positive Rate

Specificity =True Negative Rate

Estimation Methodologies for Classification

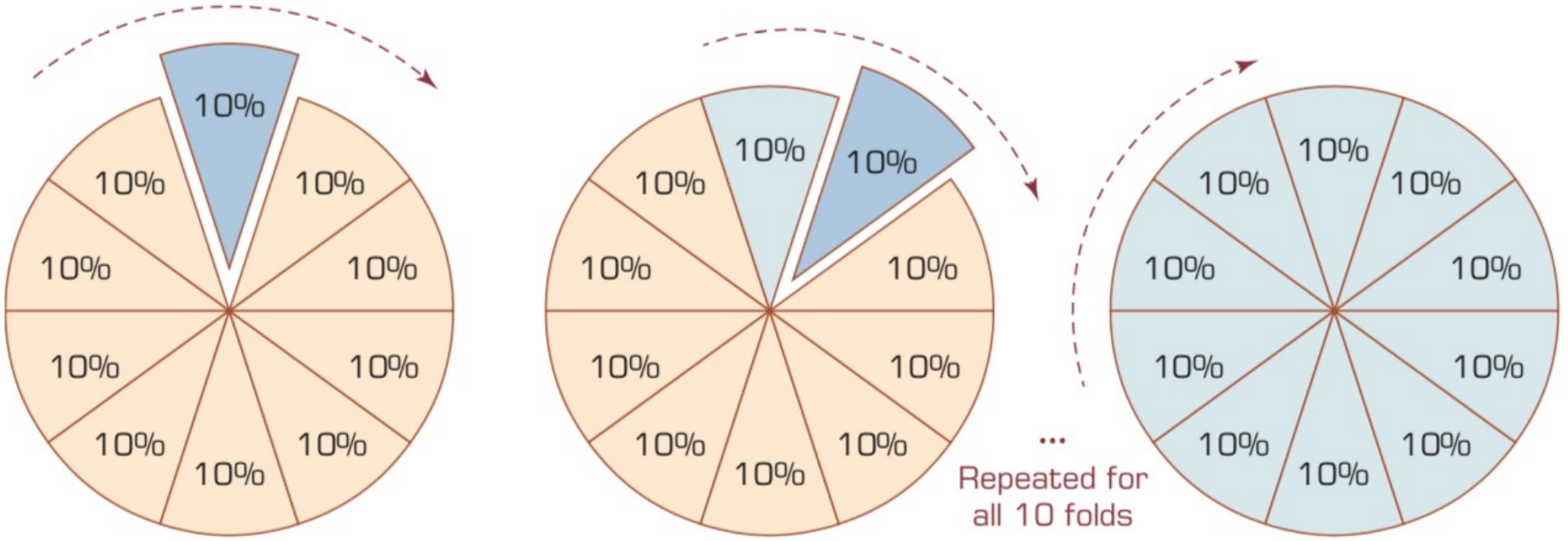
- **Simple split (or holdout or test sample estimation)**
  - **Split the data into 2 mutually exclusive sets training (~70%) and testing (30%)**



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

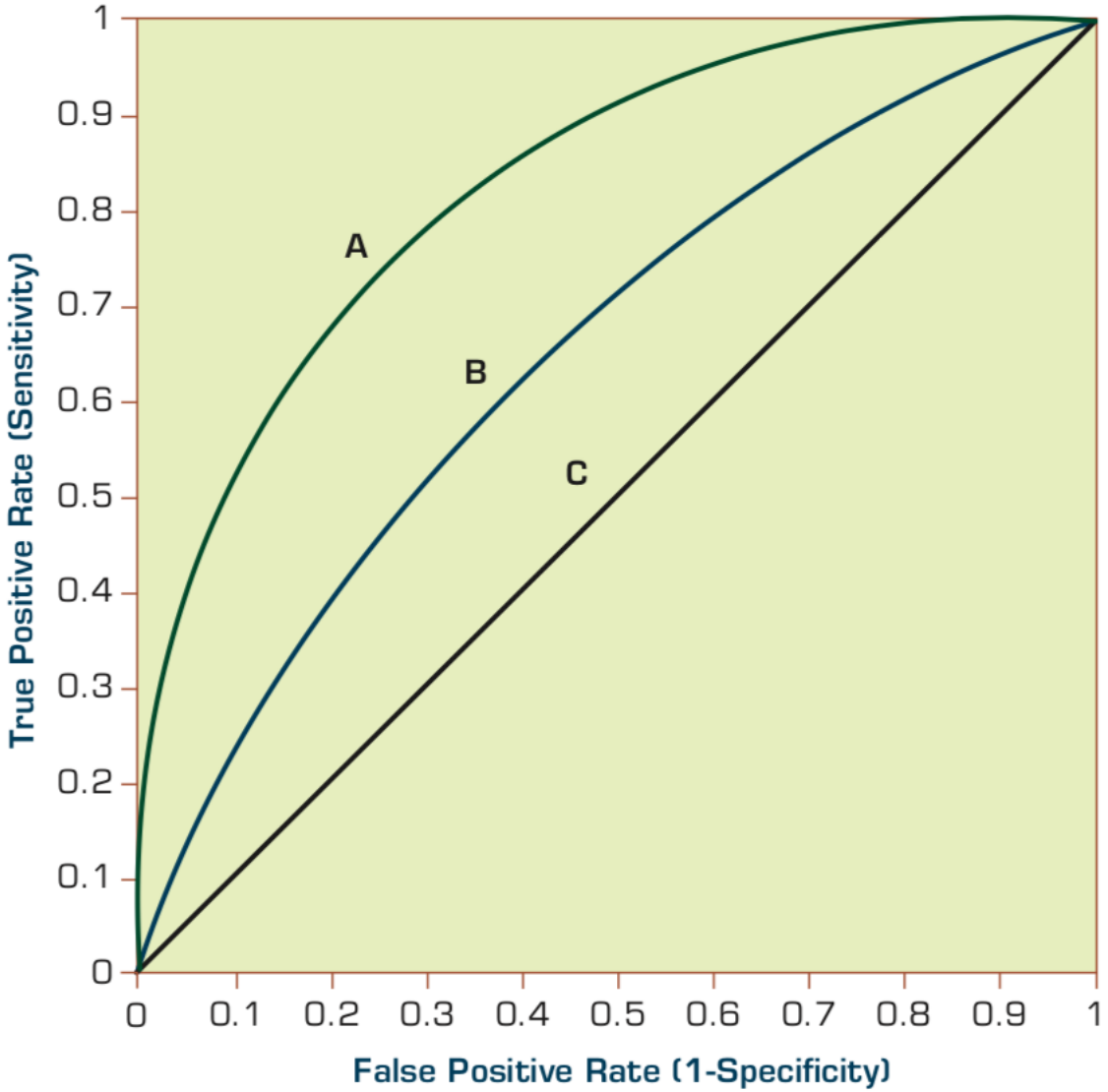
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

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

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

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

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

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

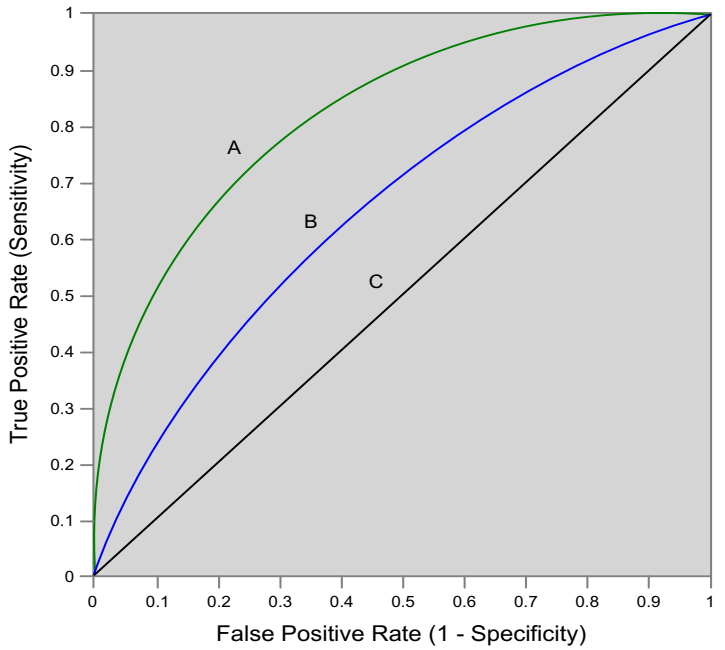
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

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

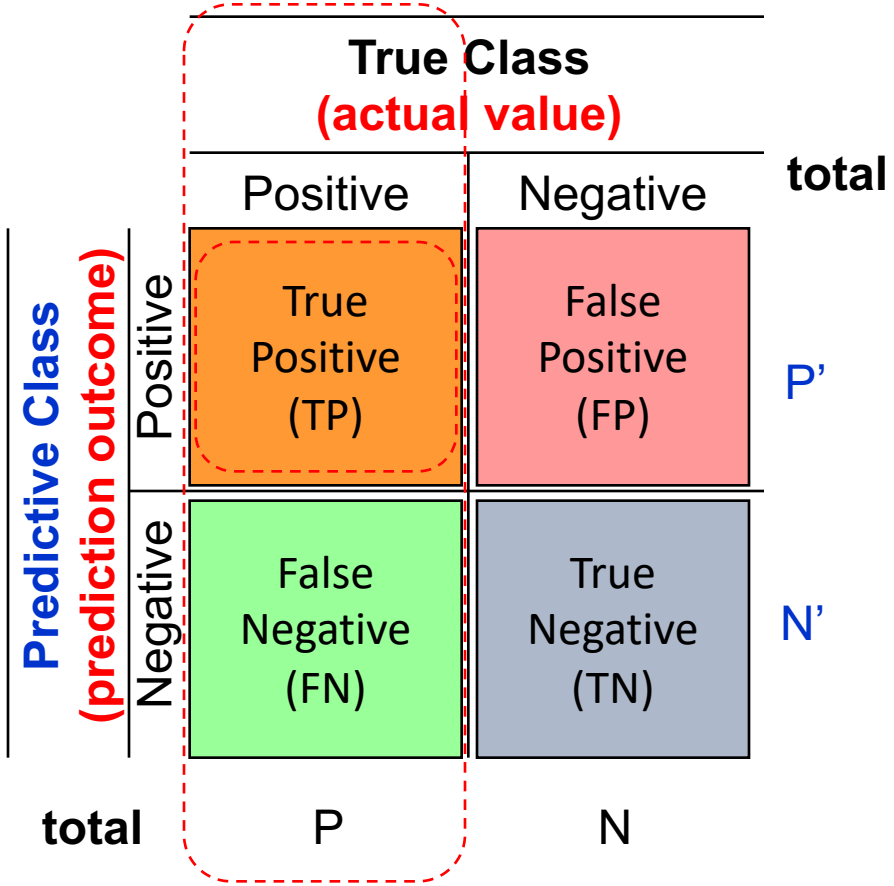
$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

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

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



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

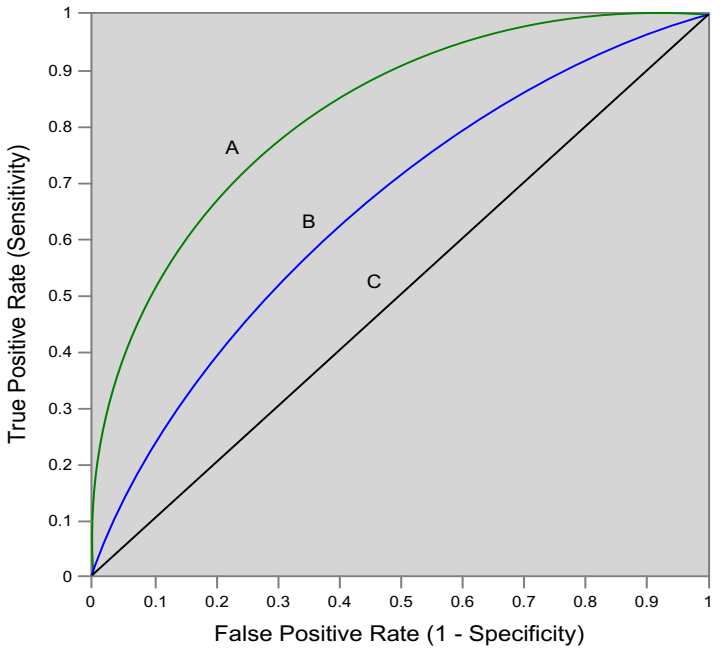


$$True\ Positive\ Rate\ (\text{Sensitivity}) = \frac{TP}{TP + FN}$$

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	

Specificity

= True Negative Rate

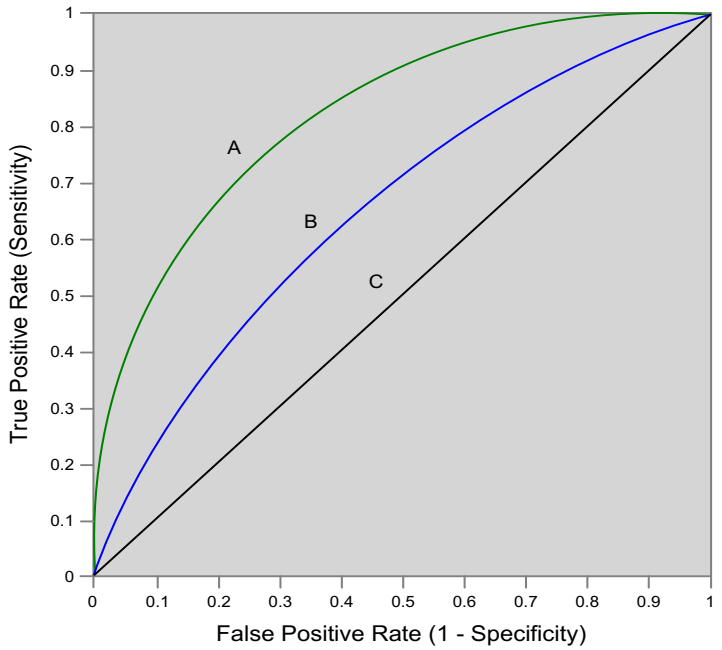
= TN / N

= TN / (TN+ FP)

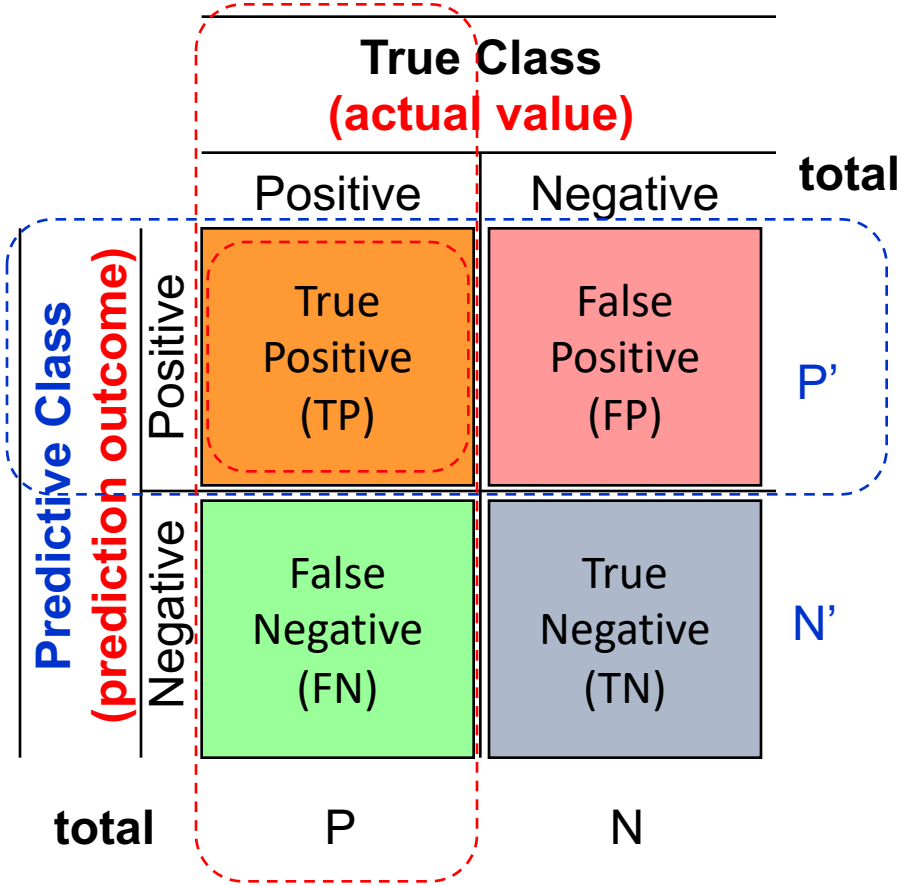
$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

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

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



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



Precision
= Positive Predictive Value (PPV)

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

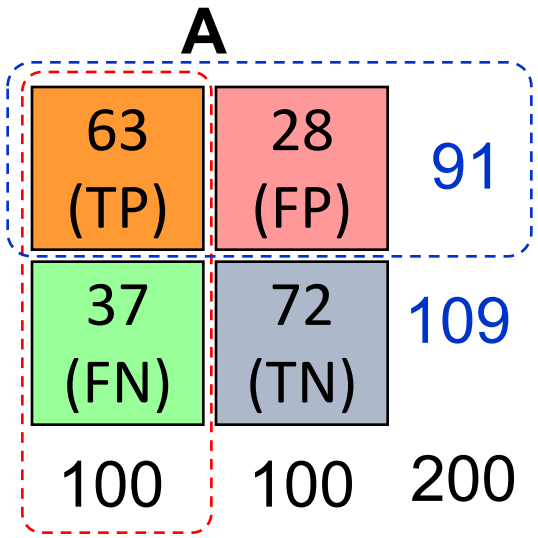
Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

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

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



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)

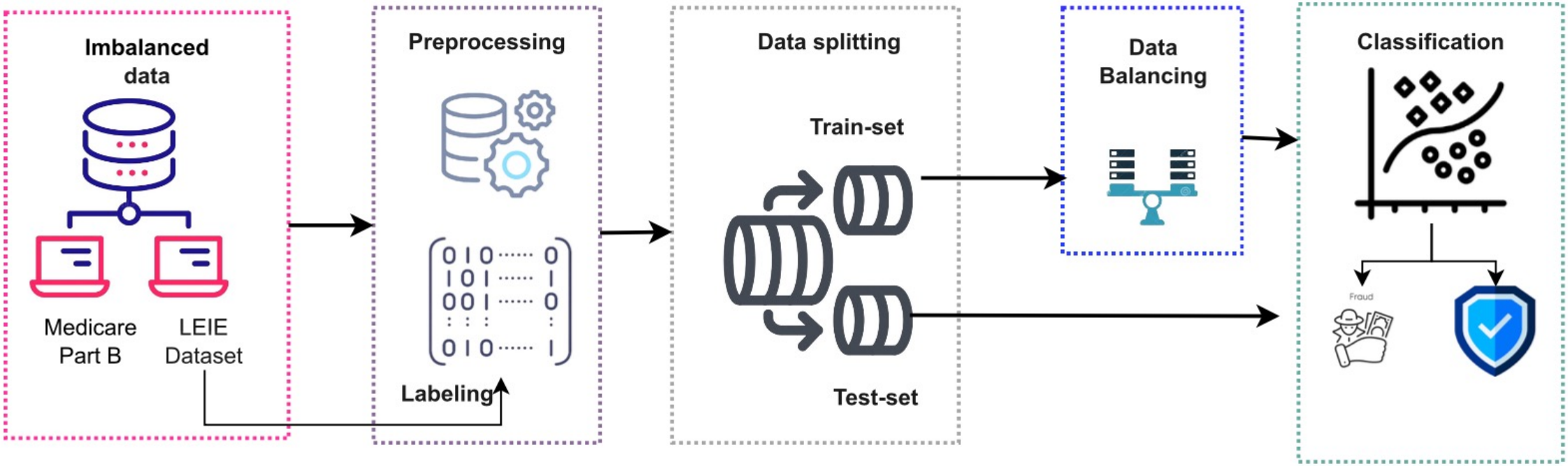
$$Precision = \frac{TP}{TP + FP}$$

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

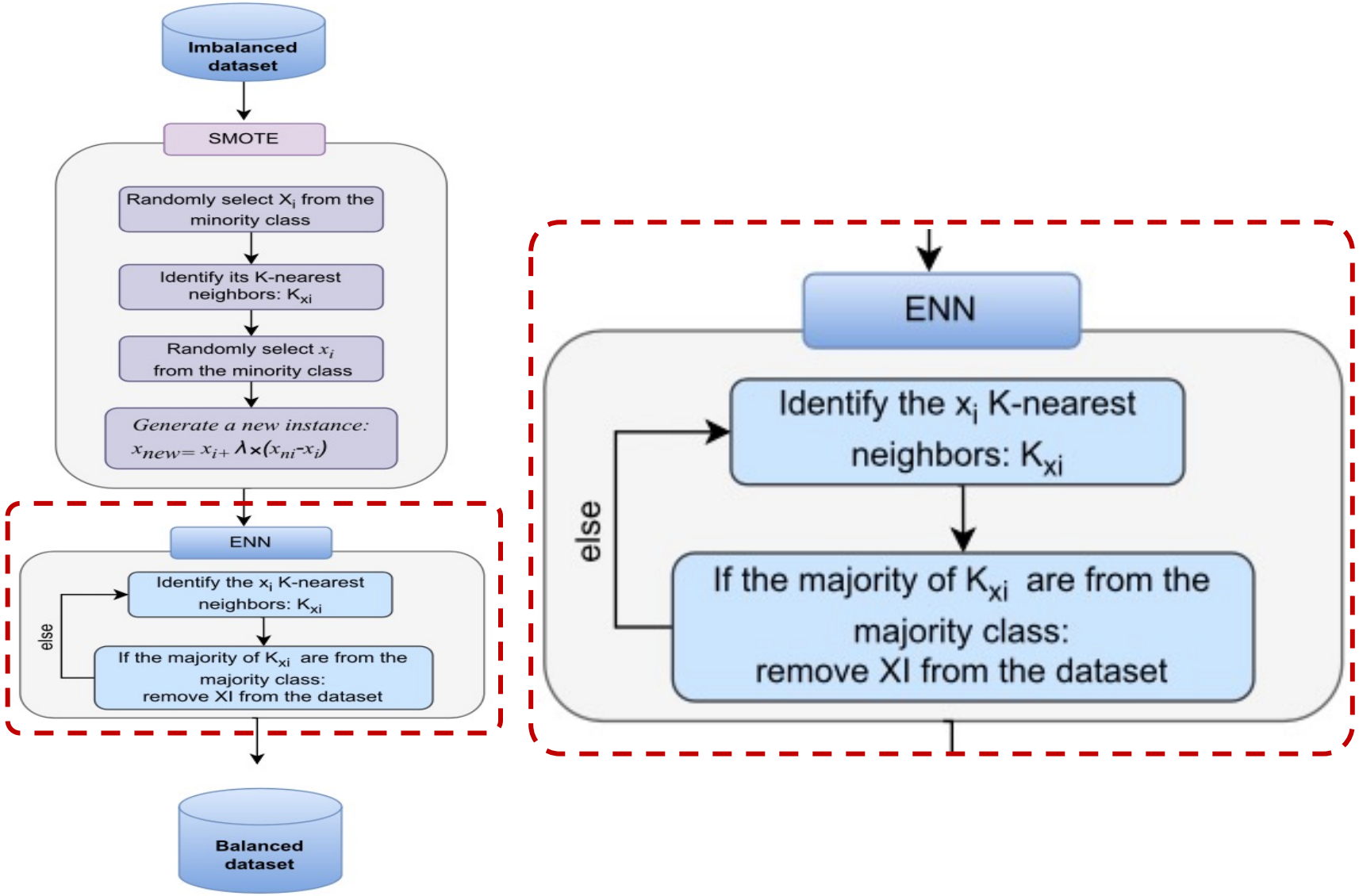
Ref	Dataset	ML Methods	Data Balancing Method	Evaluation
[21]	Medicare	Logistic Regression(LR), Random Forest (RF), Gradient Boosting Trees (GBT)	ROS, RUS, SMOTE, SMOTE variants, ADASYN	Area Under the Curve (AUC)= 0.82
[20]	Medicare Part B	Naive Bayes(NB), LR, Decision Trees (DT), K-Nearest Neighbors (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

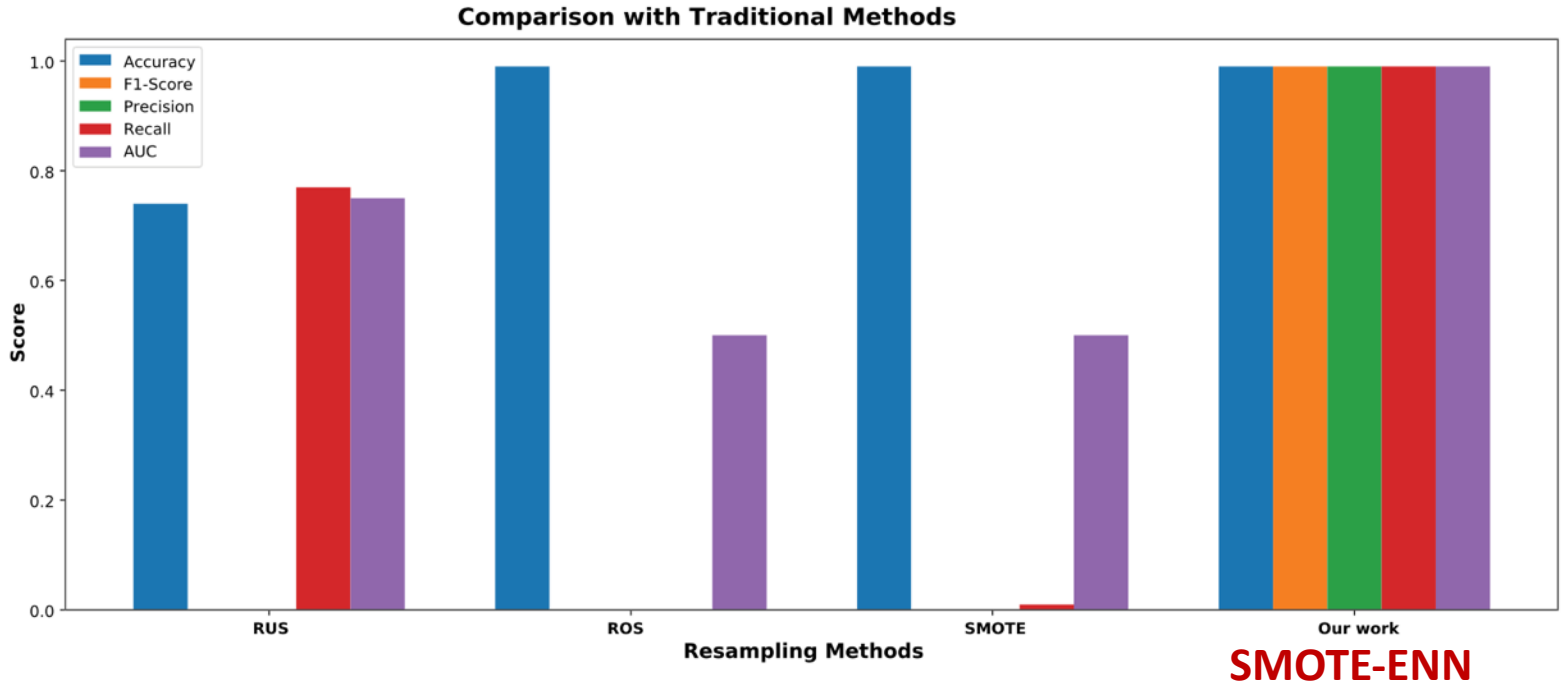
Synthetic Minority Over-sampling technique with Edited Nearest Neighbors (SMOTE-ENN)

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)

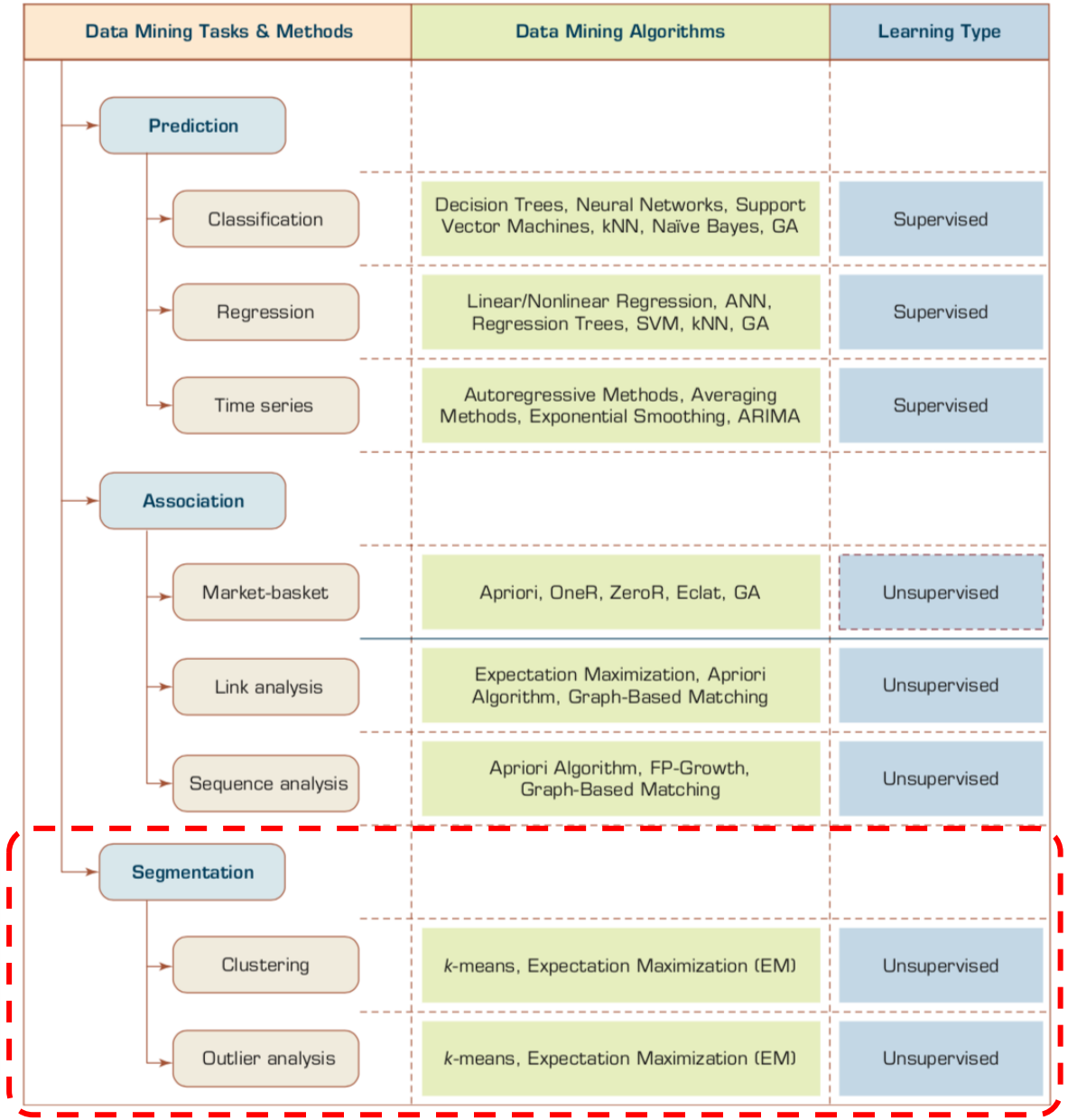
Data augmentation	Ensemble model	Accuracy (Best, Std)	F1 (Best, Std)	AUC (Best, Std)
Yeast-v6				
Borderline-SMOTE	LightGBM	98.490(98.653, 0.150)	99.230(99.314, 0.077)	76.668(76.751, 0.077)
No augmentation	Stacking-I	98.185(98.653, 0.320)	99.076(99.315, 0.165)	69.757(76.579, 3.658)
SVM-SMOTE	AdaBoost	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



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

Segmentation

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)

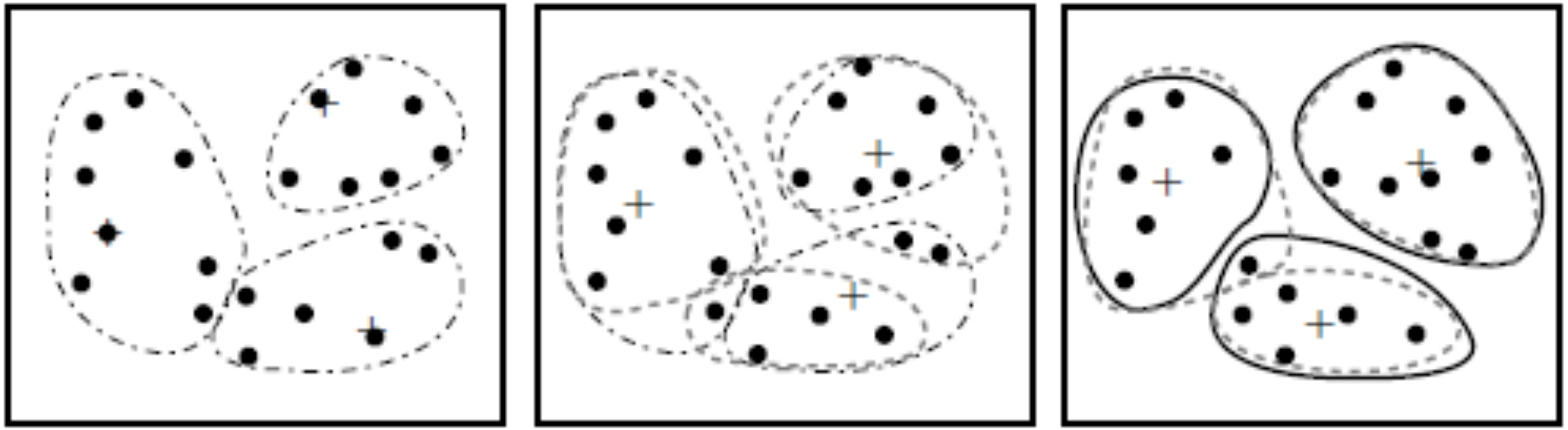
Cluster Analysis

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

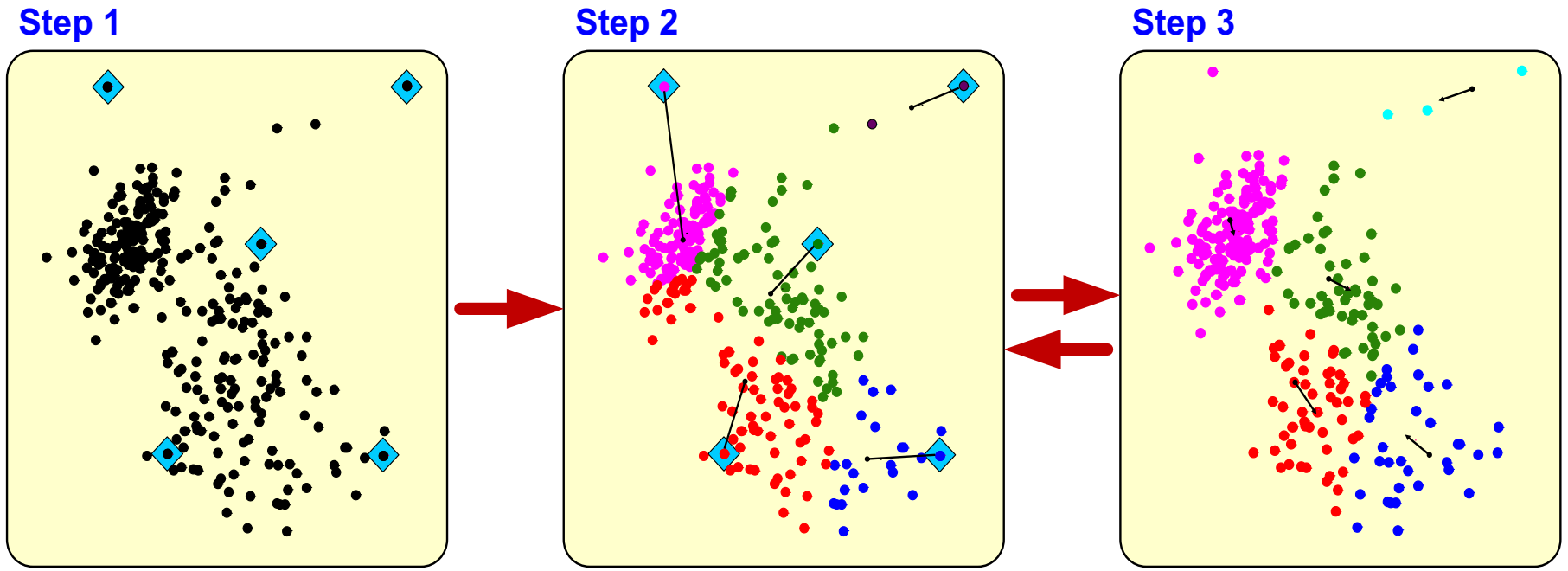
Step 2: Assign each point to the nearest cluster center

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*:**

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + ... + |x_{ip} - x_{jp}|^q)}$$

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

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

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ip} - x_{jp}|$$

Source: Han & Kamber (2006)

Similarity and Dissimilarity Between Objects (Cont.)

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

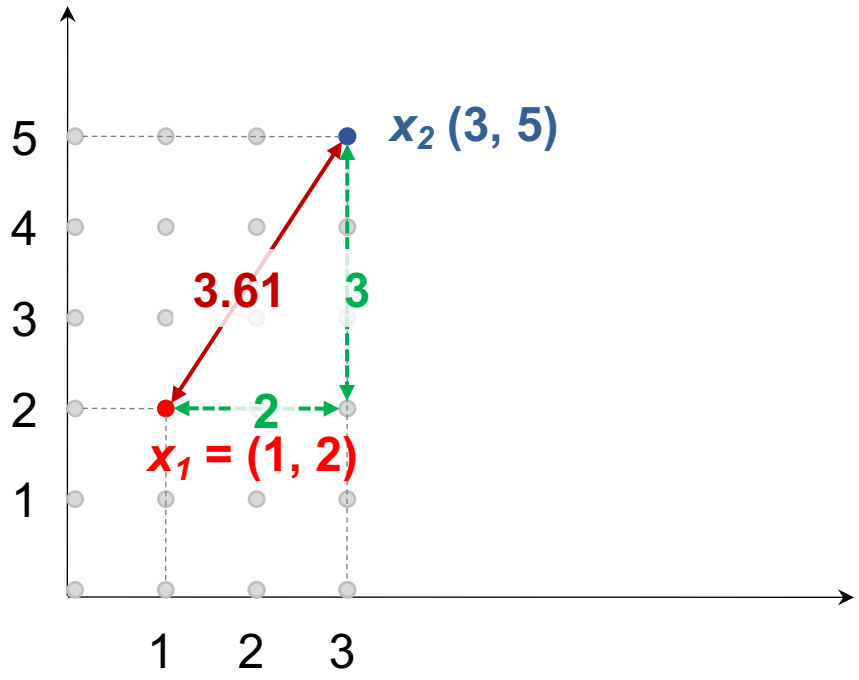
$$d(i,j)=\sqrt{(|x_{i1}-x_{j1}|^2+|x_{i2}-x_{j2}|^2+...+|x_{ip}-x_{jp}|^2)}$$

  - **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:

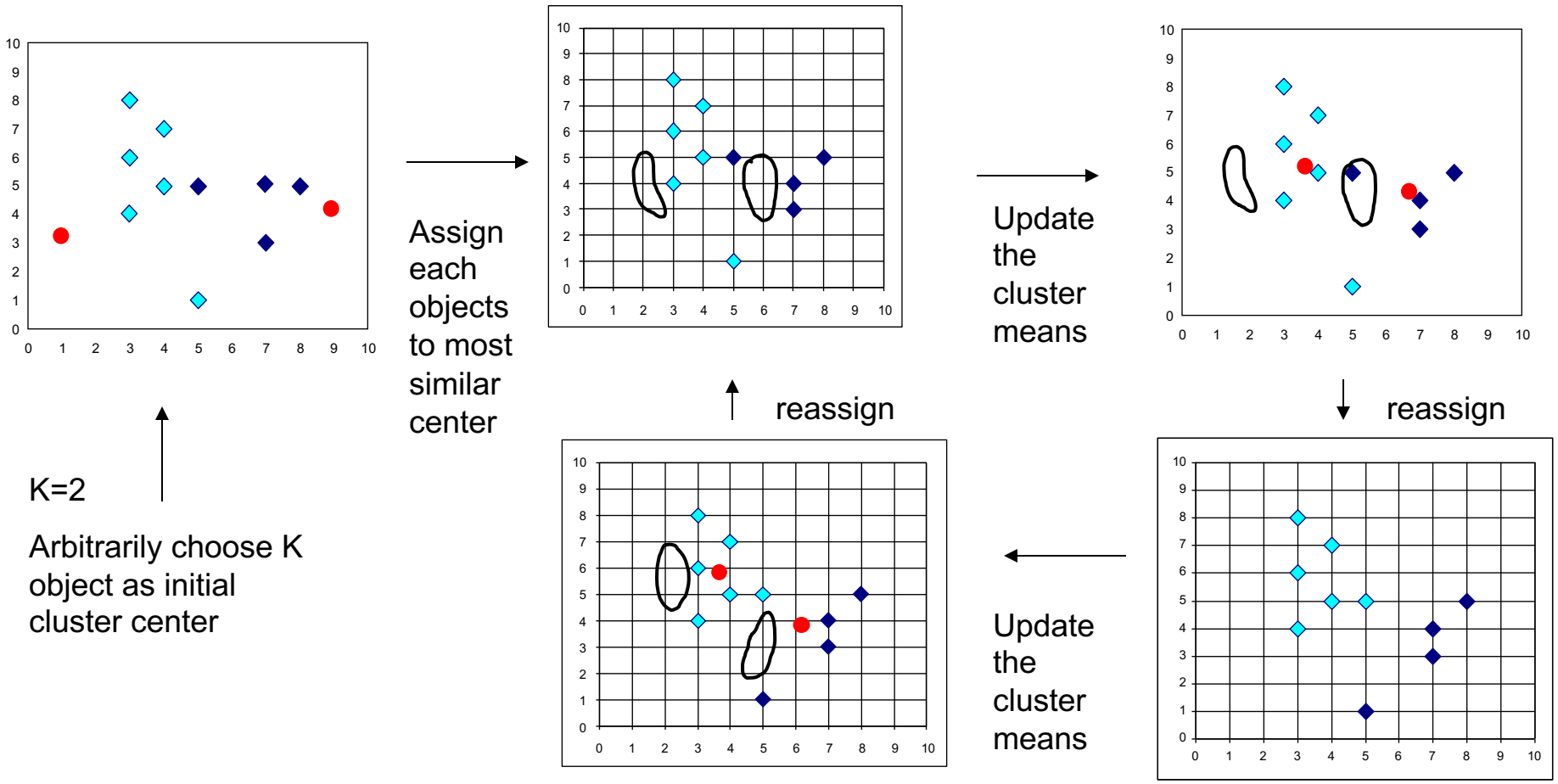
$= (3-1) + (5-2)$

$= 2 + 3$

$= 5$

The *K-Means* Clustering Method

- **Example**



Source: Han & Kamber (2006)

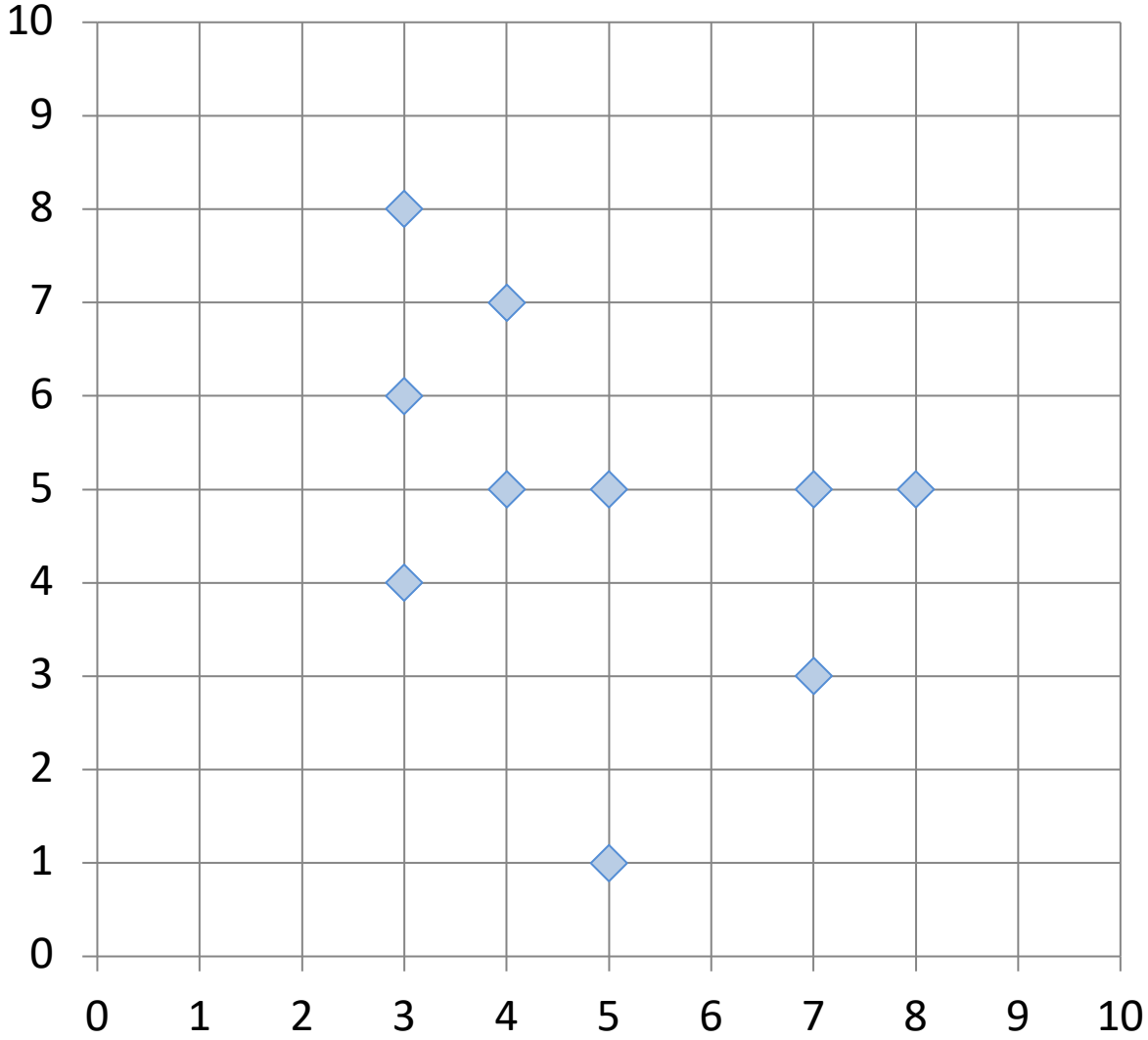
K-Means Clustering

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

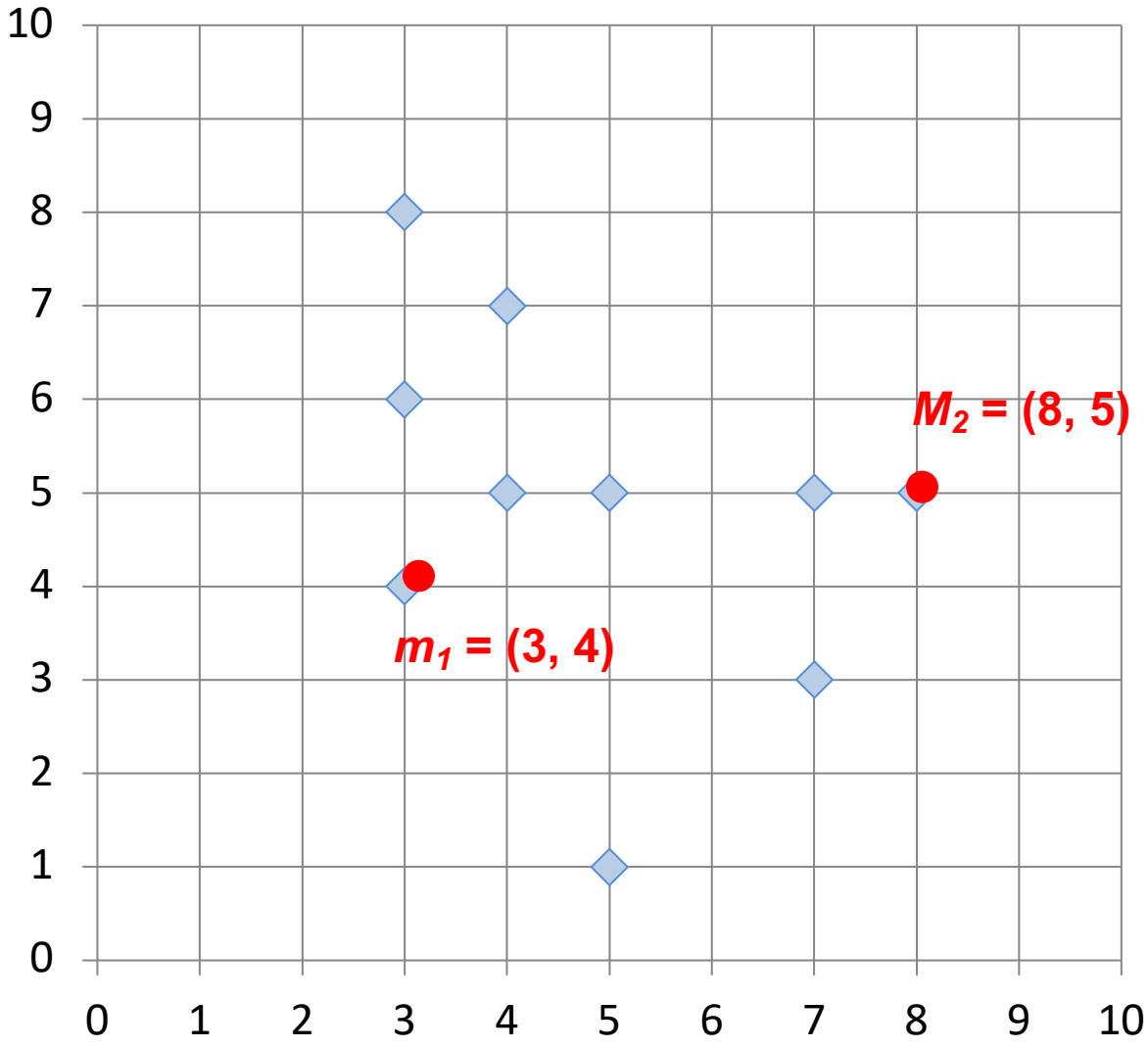
Step by Step



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

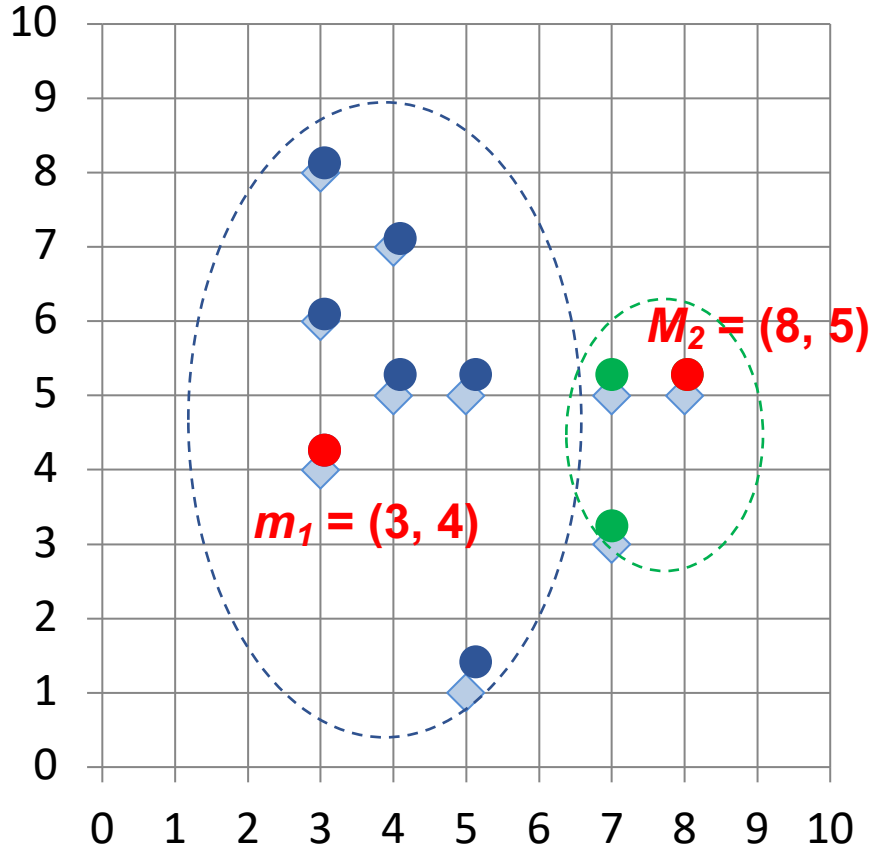
Step 1: K=2, Arbitrarily choose K object as initial cluster center



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

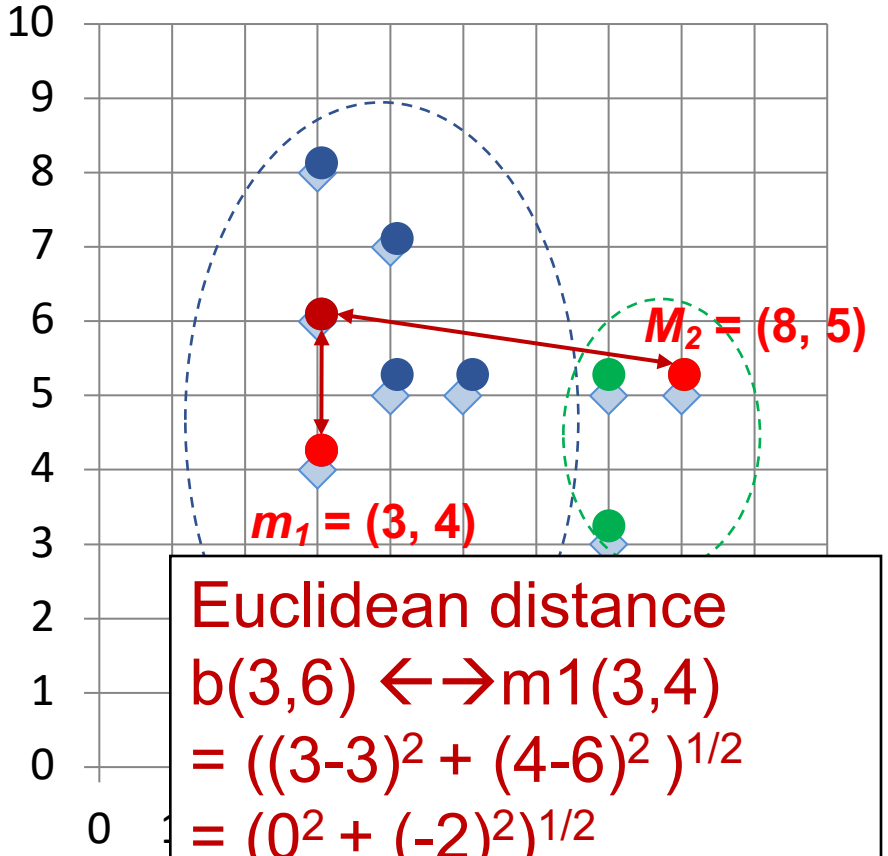
K-Means Clustering

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					ster1
p06					ster1
p07					ster1
p08					ster2
p09					ster2
p10					ster2

Euclidean distance
b(3,6) ←→m1(3,4)
$= ((3-3)^2 + (4-6)^2)^{1/2}$
$= (0^2 + (-2)^2)^{1/2}$
$= (0 + 4)^{1/2}$
$= (4)^{1/2}$
$= 2.00$

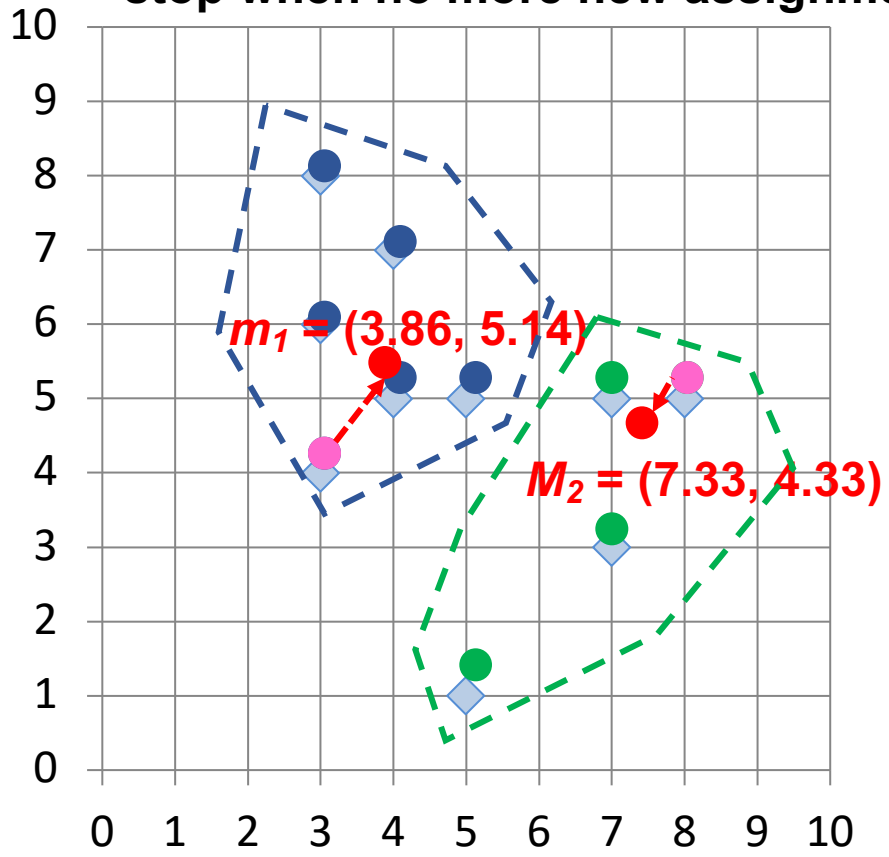
Euclidean distance
b(3,6) ←→m2(8,5)
$= ((8-3)^2 + (5-6)^2)^{1/2}$
$= (5^2 + (-1)^2)^{1/2}$
$= (25 + 1)^{1/2}$
$= (26)^{1/2}$
$= 5.10$

Initial m1 (3, 4)

Initial m2 (8, 5)

K-

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



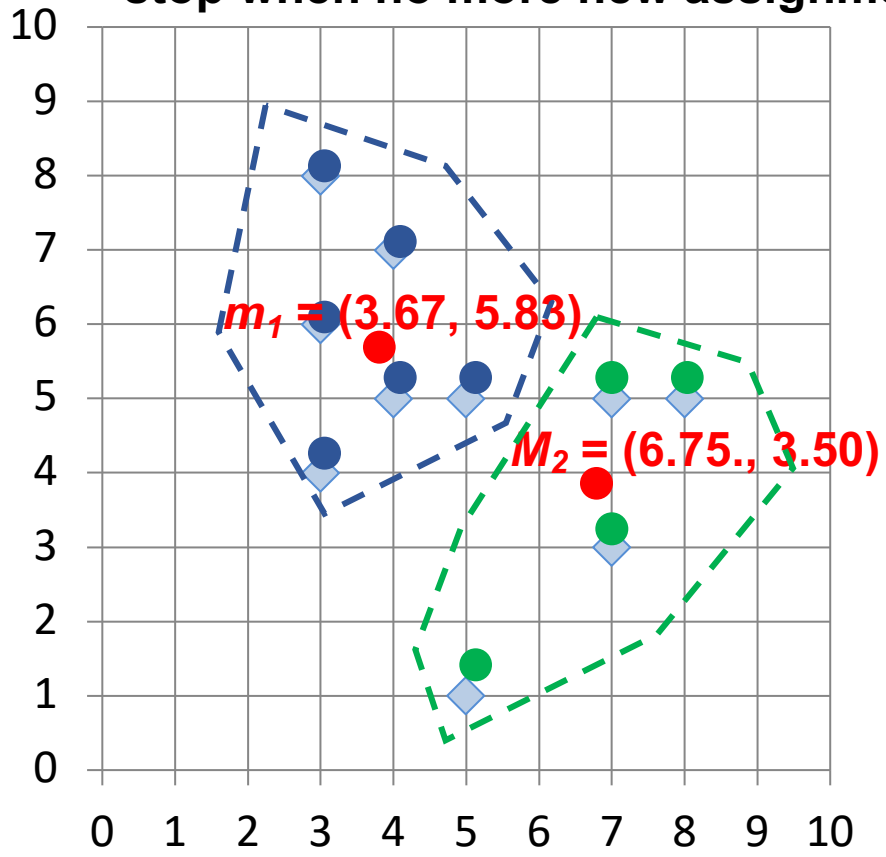
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.43	4.34	Cluster1
p02	b	(3, 6)	1.22	4.64	Cluster1
p03	c	(3, 8)	2.99	5.68	Cluster1
p04	d	(4, 5)	0.20	3.40	Cluster1
p05	e	(4, 7)	1.87	4.27	Cluster1
p06	f	(5, 1)	4.29	4.06	Cluster2
p07	g	(5, 5)	1.15	2.42	Cluster1
p08	h	(7, 3)	3.80	1.37	Cluster2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

K-Means Clustering

m1 (3.86, 5.14)

m2 (7.33, 4.33)

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



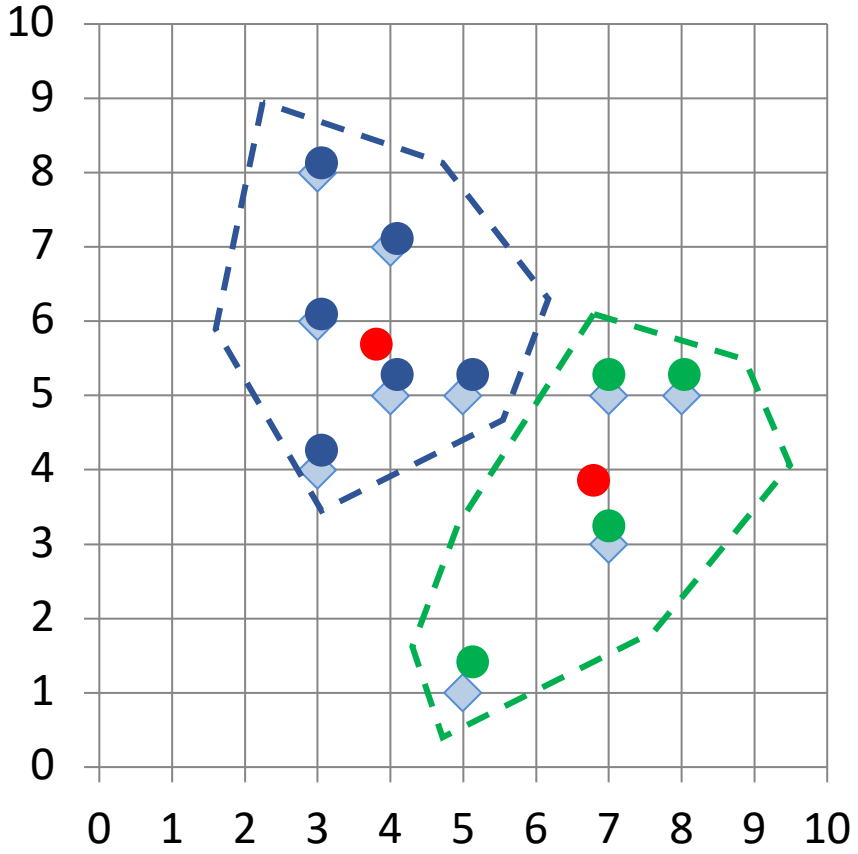
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2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

K-Means Clustering

m1 (3.67, 5.83)

m2 (6.75, 3.50)

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

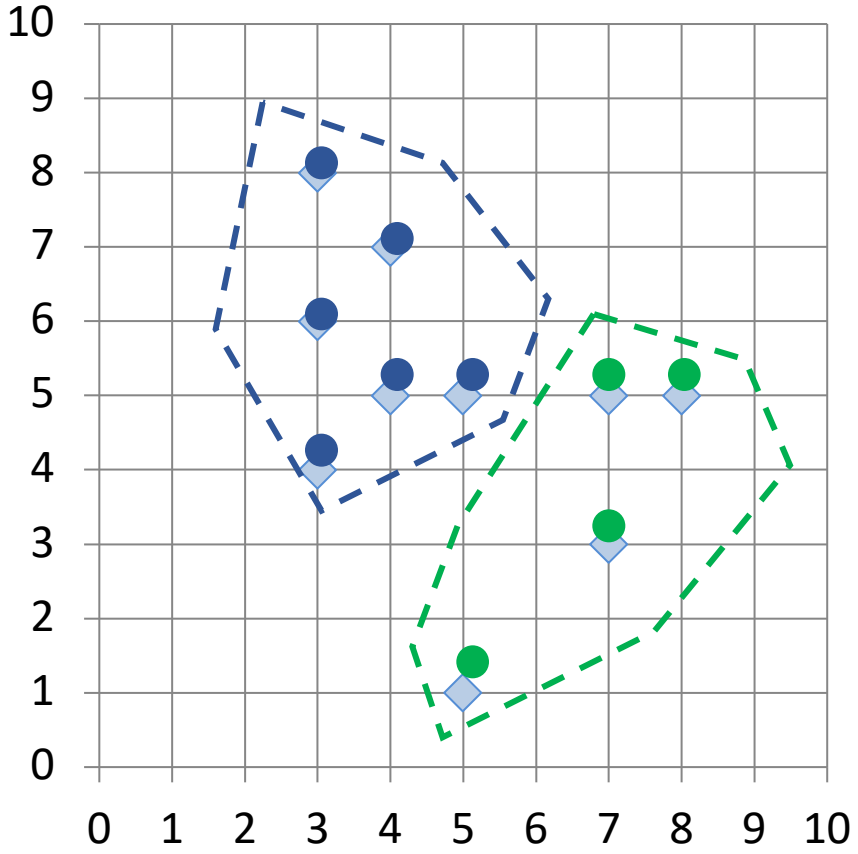
K-Means Clustering

m1 (3.67, 5.83)

m2 (6.75, 3.50)

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

***K-Means* Clustering**

m1 (3.67, 5.83)

m2 (6.75, 3.50)

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

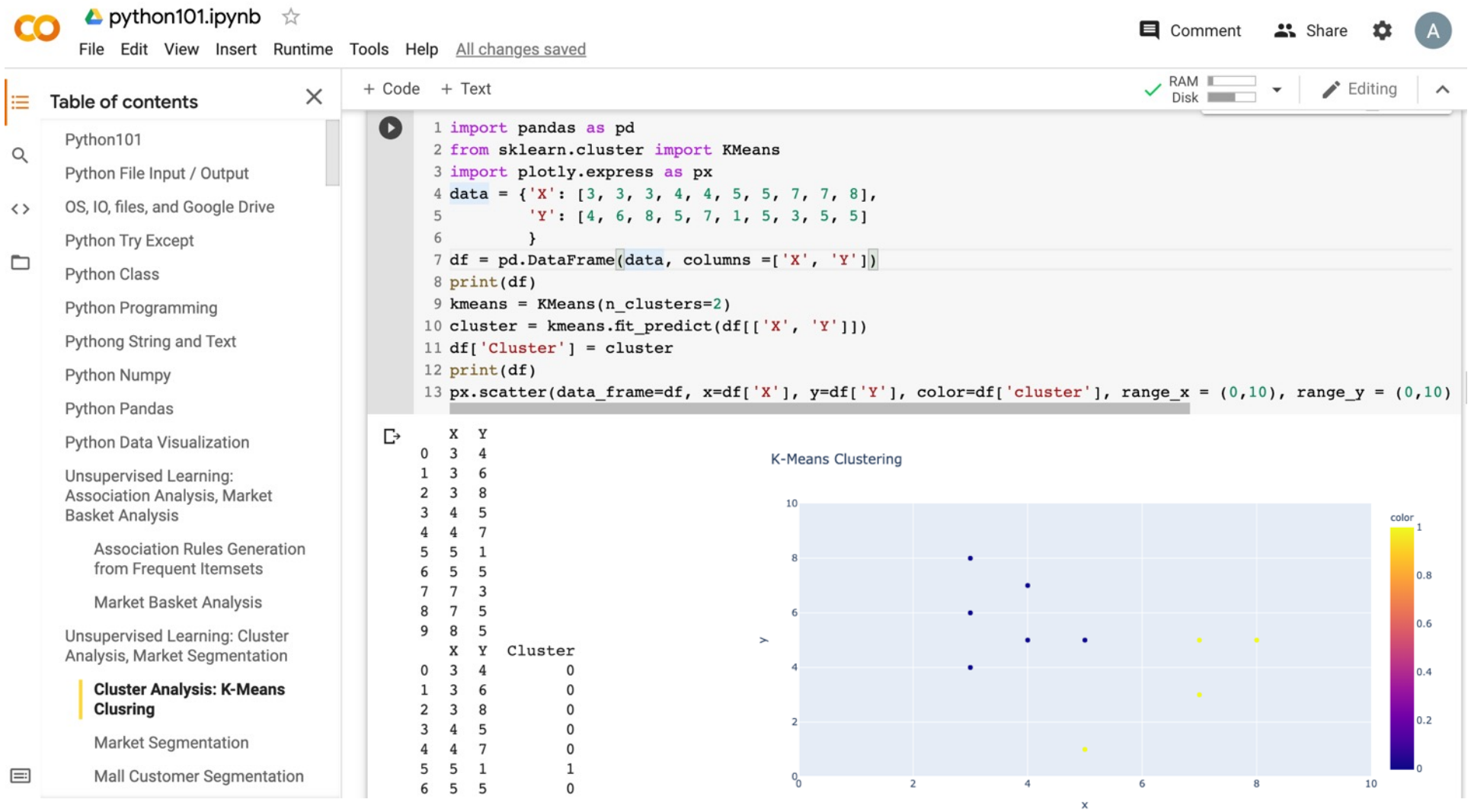
m1 (3.67, 5.83)

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



https://tinyurl.com/aintpupython101

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

```
1 import pandas as pd
2 from sklearn.cluster import KMeans
3 import plotly.express as px
4 data = {'X': [3, 3, 3, 4, 4, 5, 5, 7, 7, 8],
5         'Y': [4, 6, 8, 5, 7, 1, 5, 3, 5, 5]
6         }
7 df = pd.DataFrame(data, columns =['X', 'Y'])
8 print(df)
9 kmeans = KMeans(n_clusters=2)
10 cluster = kmeans.fit_predict(df[['X', 'Y']])
11 df['Cluster'] = cluster
12 print(df)
13 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()
```

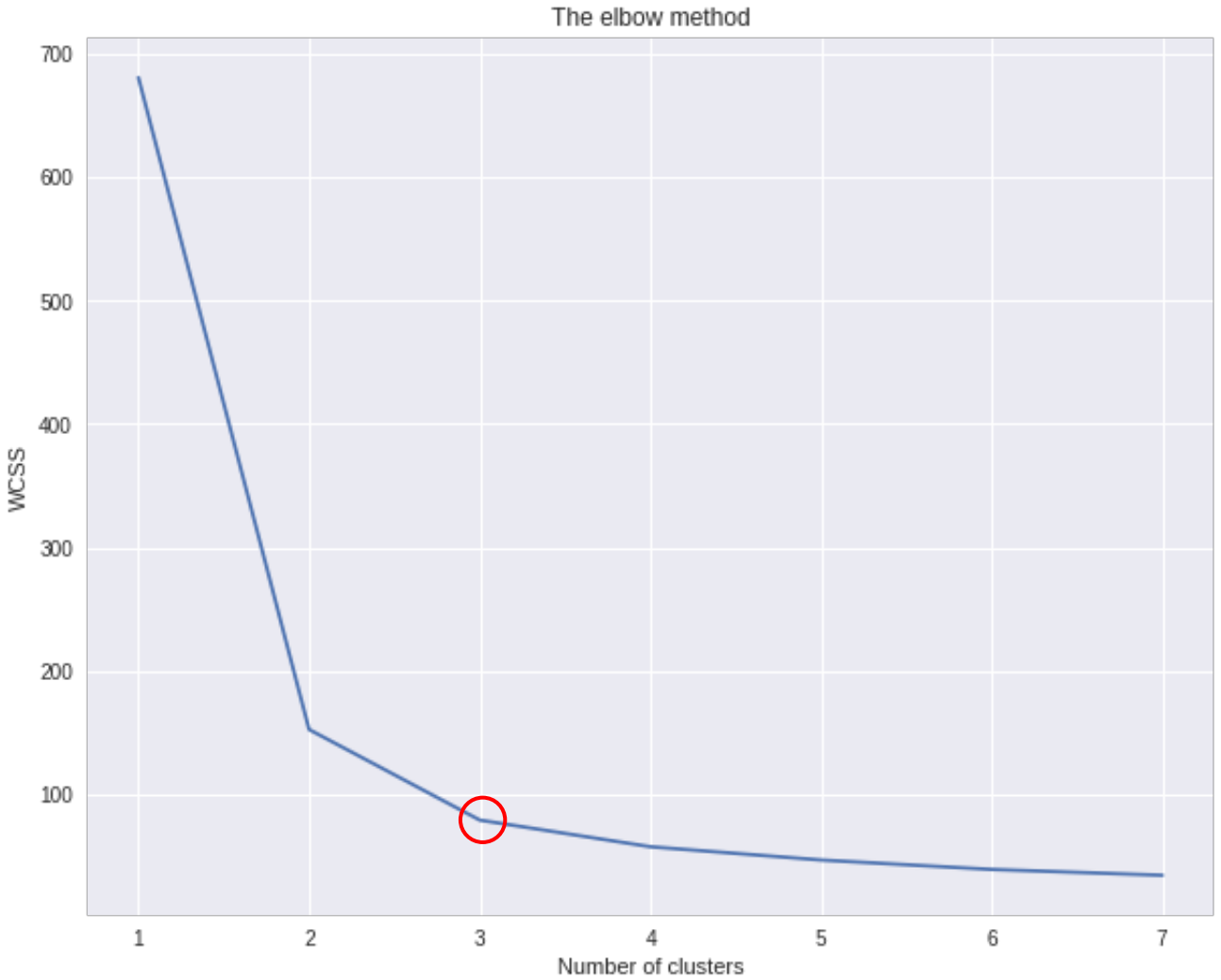
https://tinyurl.com/aintpupython101

```
#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)
```

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

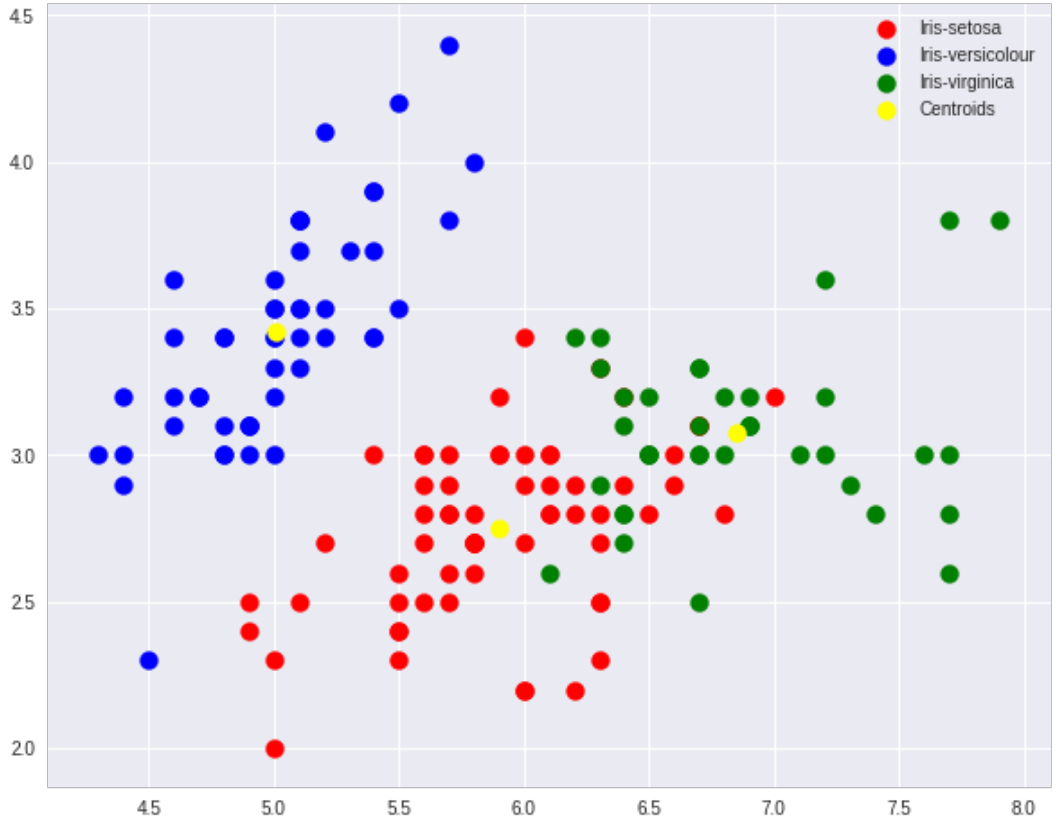
```
#Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100,
c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100,
c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100,
c = 'green',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

K-Means Clustering

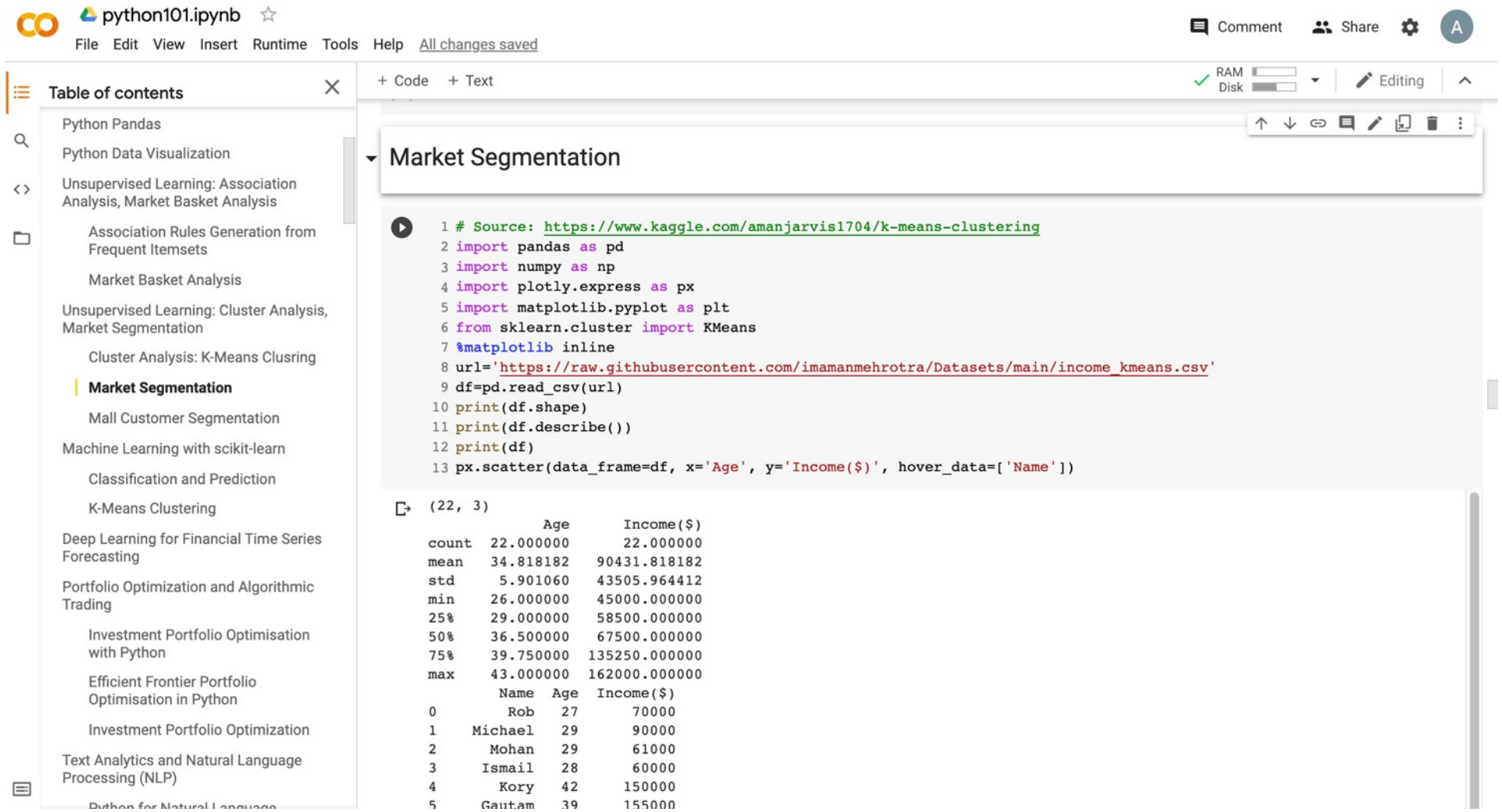
```
#Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
```

```
#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()
```



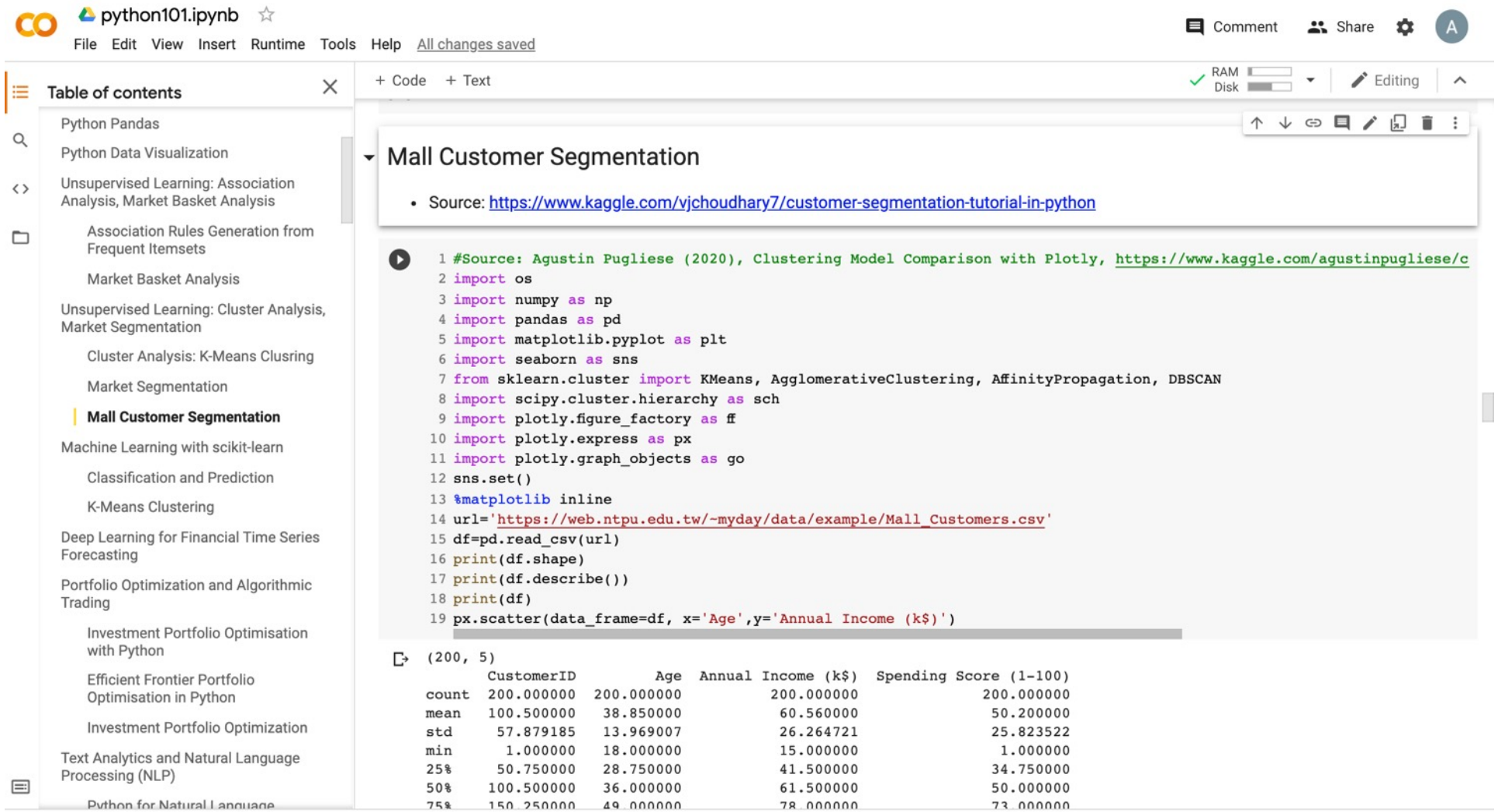
https://tinyurl.com/aintpupython101

Market Segmentation



https://tinyurl.com/aintpupython101

Mall Customer Segmentation

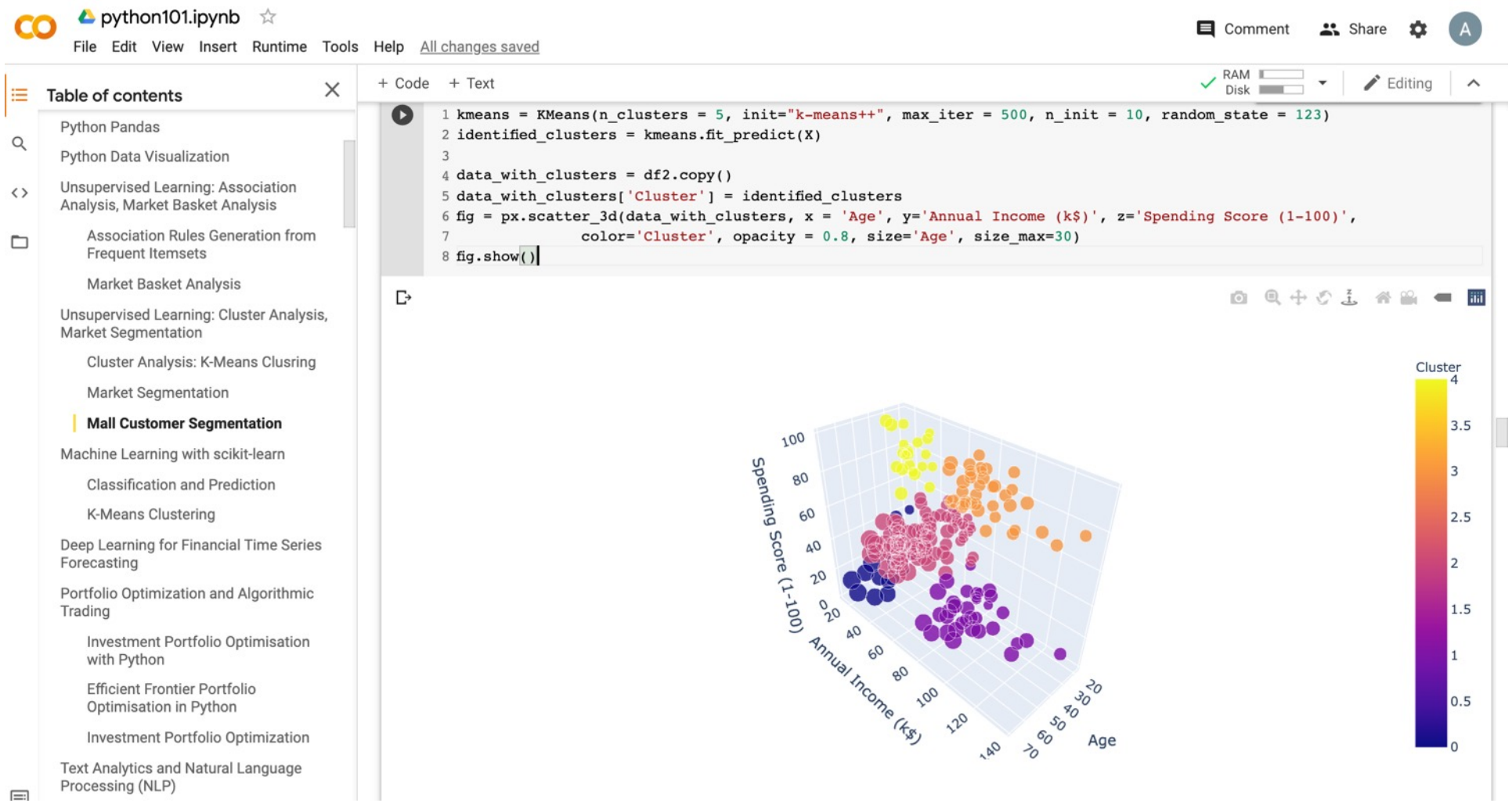


https://tinyurl.com/aintpupython101

Mall Customer Segmentation

https://tinyurl.com/aintpupython101

Mall Customer Segmentation



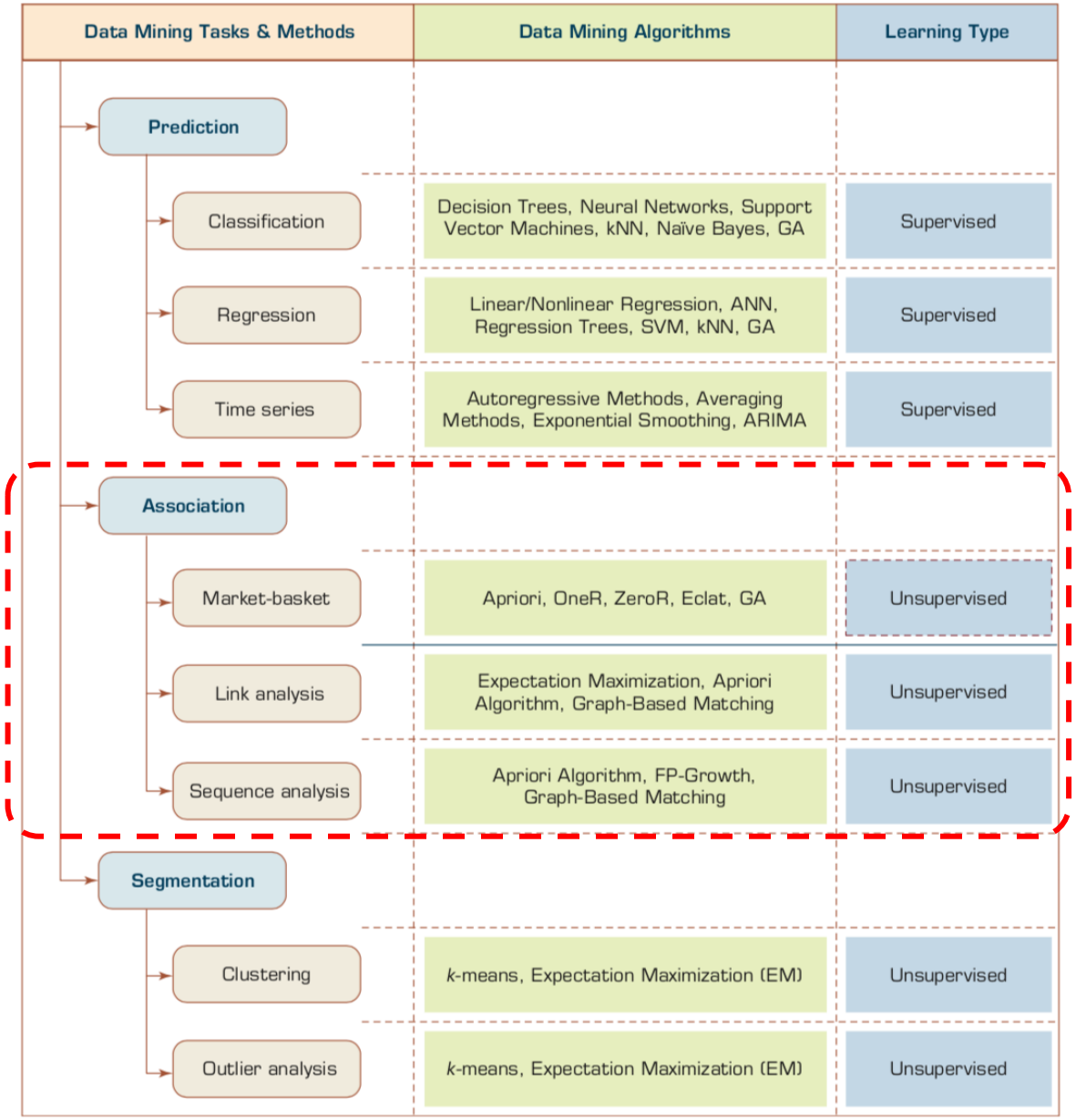
https://tinyurl.com/aintpupython101

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

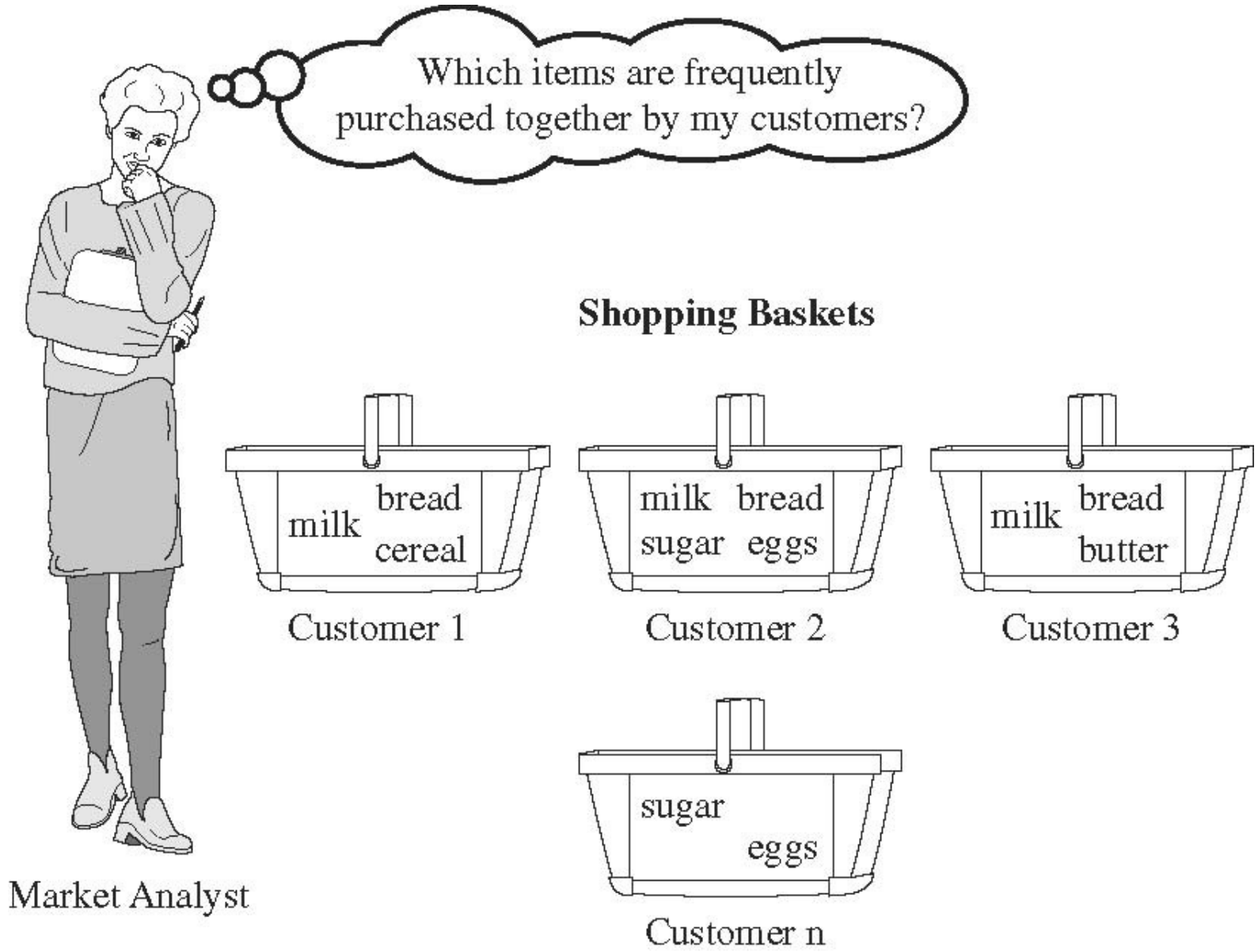
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

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)
```

https://tinyurl.com/aintpupython101

Python mlxtend Association Rules

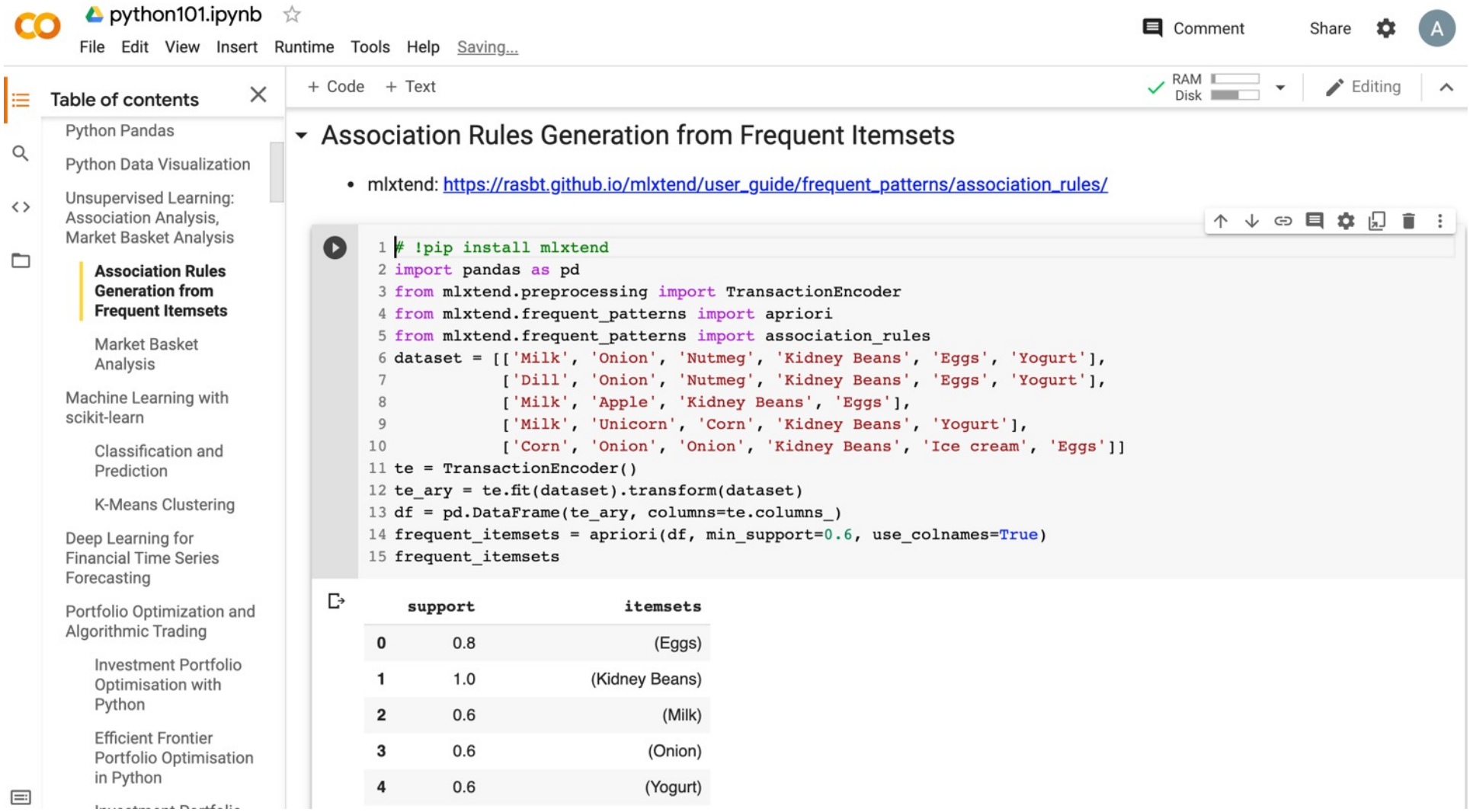
```
# !pip install mlxtend
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
dataset = [['A', 'B', 'D'],
           ['A', 'C', 'D'],
           ['B', 'C', 'D', 'E'],
           ['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)
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



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)
```

```
1  # ! pip install mlxtend
2  import pandas as pd
3  from mlxtend.preprocessing import TransactionEncoder
4  from mlxtend.frequent_patterns import apriori
5  from mlxtend.frequent_patterns import association_rules
6
7  dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
8             ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
9             ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
10            ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
11            ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]
12
13 te = TransactionEncoder()
14 te_ary = te.fit(dataset).transform(dataset)
15 df = pd.DataFrame(te_ary, columns=te.columns_)
16 frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
17
18 frequent_itemsets
```

```
# ! 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)

https://tinyurl.com/aintpupython101

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

```
1 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
```

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

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

```
1 rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
2 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) ]
```

```
1 rules[ (rules['antecedent_len'] >= 2) &
2        (rules['confidence'] > 0.75) &
3        (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'}]
```

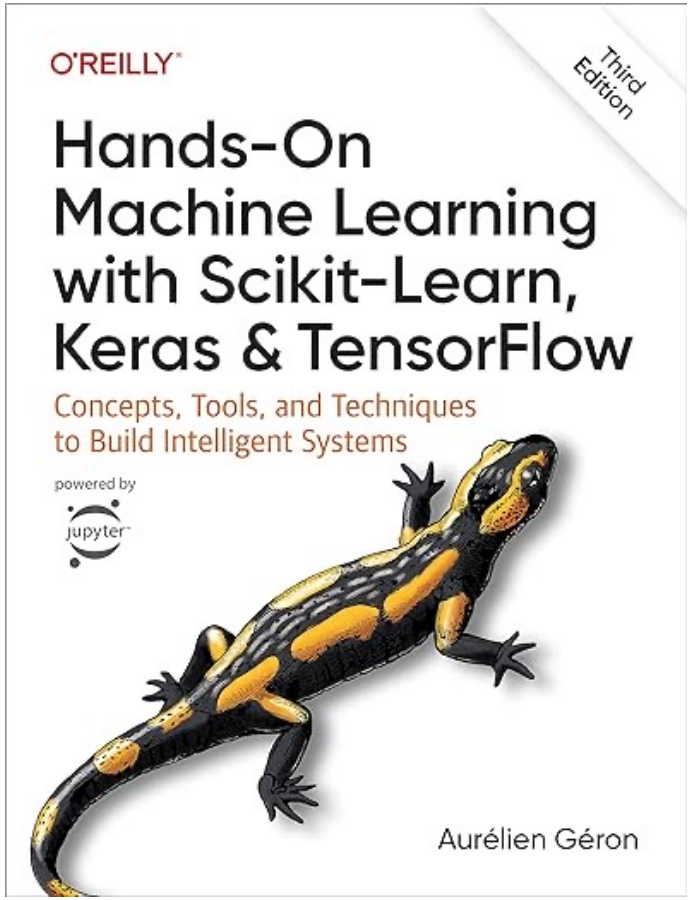
```
1 rules[rules['antecedents'] == {'Eggs', 'Kidney Beans'}]
```

	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

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

Notebooks

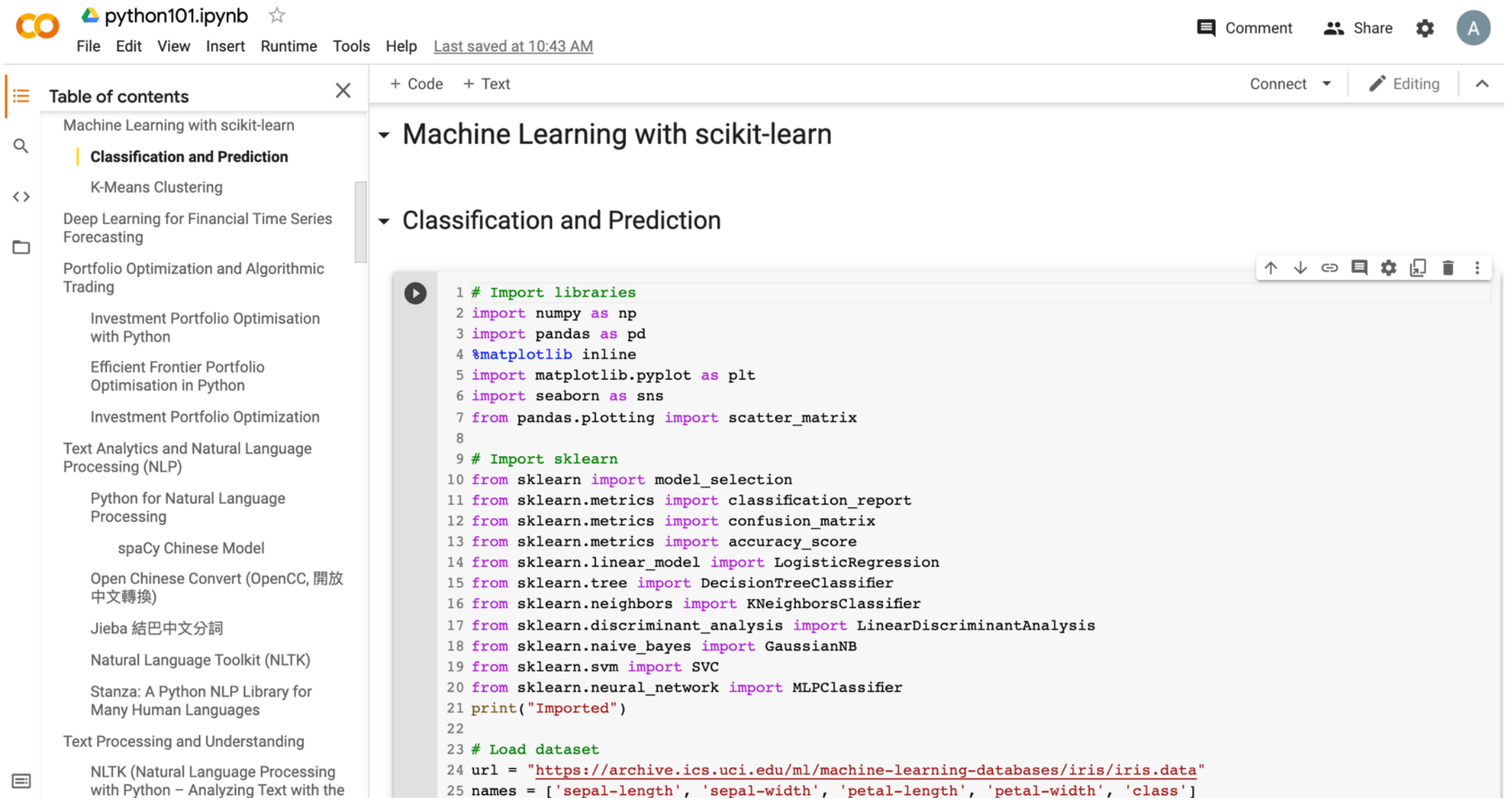
1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. 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

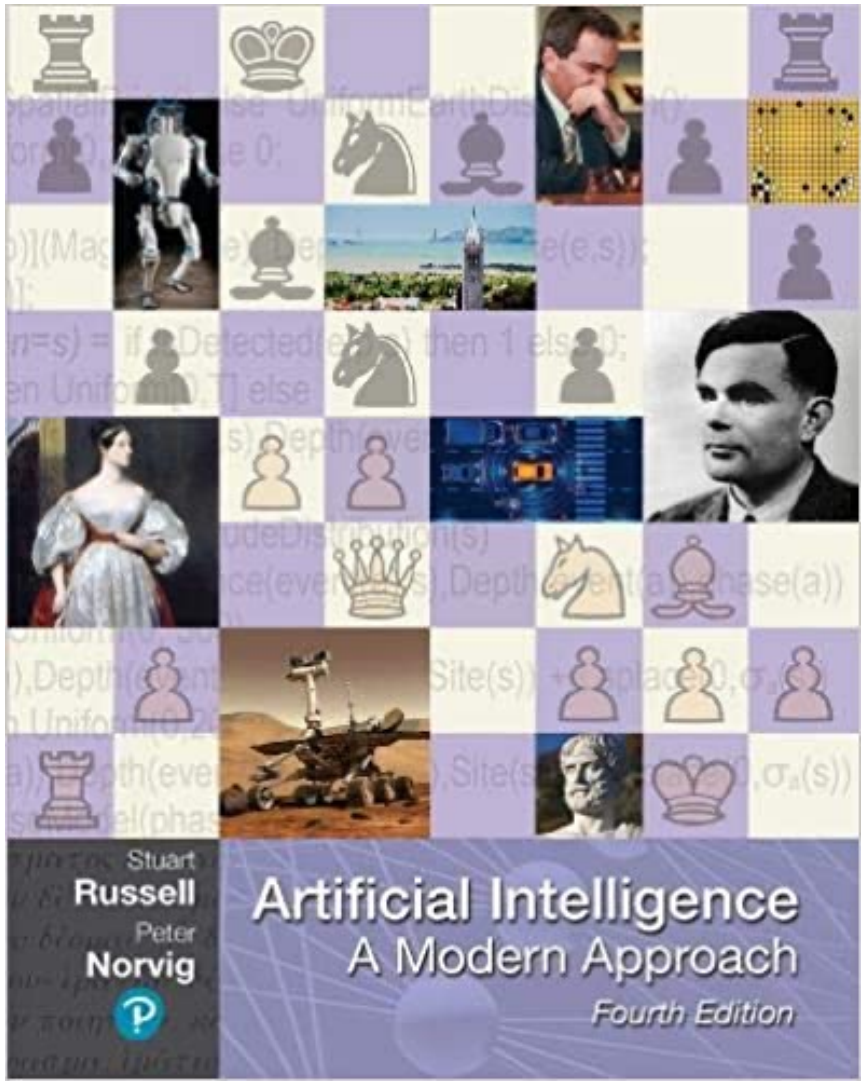


https://tinyurl.com/aintpupython101

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 (AIMA)

- **Artificial Intelligence: A Modern Approach (AIMA)**
  - http://aima.cs.berkeley.edu/
- **AIMA Python**
  - http://aima.cs.berkeley.edu/python/readme.html
  - https://github.com/aimacode/aima-python
- **Learning**
  - http://aima.cs.berkeley.edu/python/learning.html

Artificial Intelligence: A Modern Approach (AIMA)

Artificial Intelligence: A Modern Approach, 4th US ed.

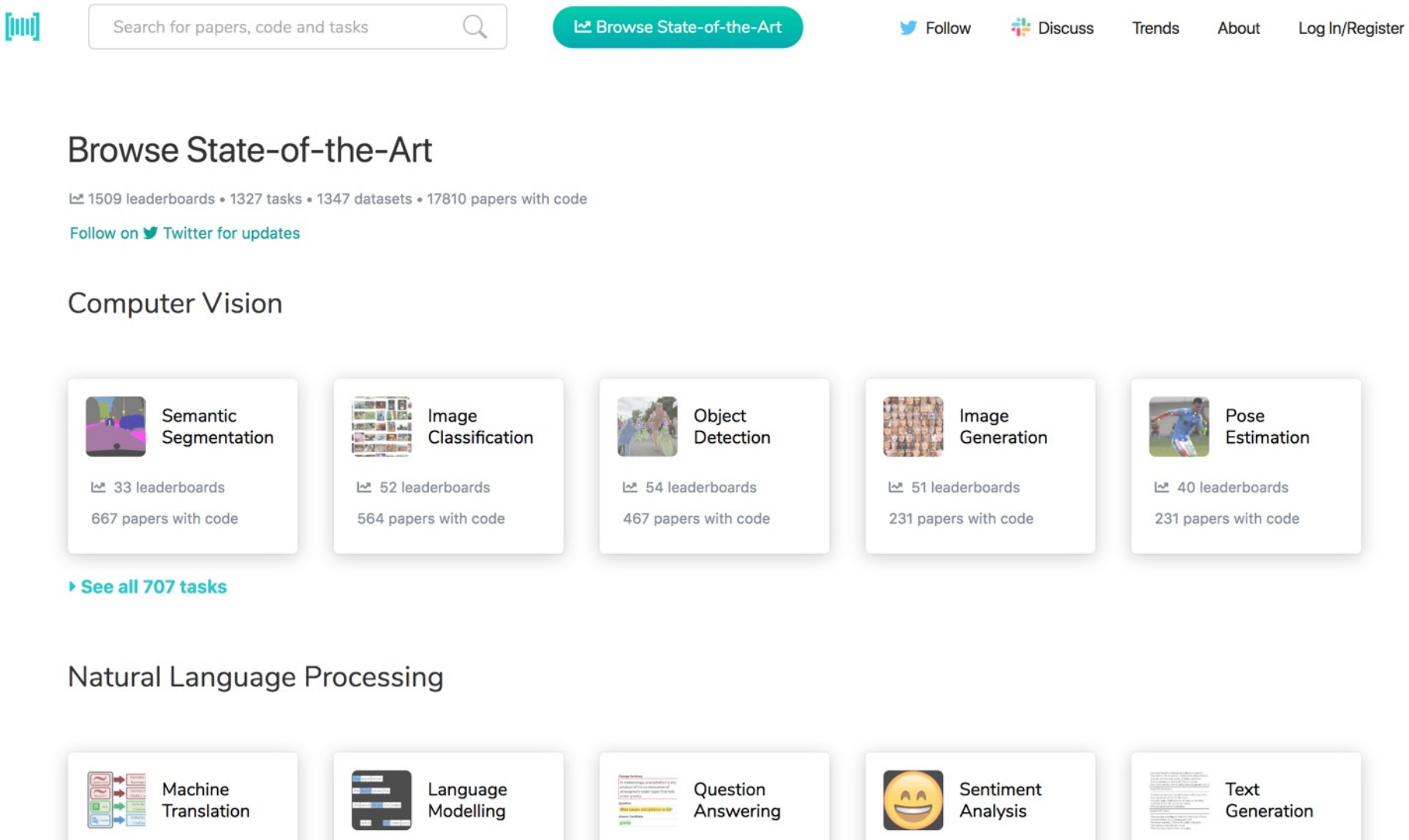
by Stuart Russell and Peter Norvig

The authoritative, most-used AI textbook, adopted by over **1500** schools.

Table of Contents for the US Edition (or see the Global Edition)

http://aima.cs.berkeley.edu/

Papers with Code
State-of-the-Art (SOTA)



https://paperswithcode.com/sota

Summary

- **Machine Learning**
- **Supervised Learning**
- **Unsupervised Learning**

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
- 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.