人工智慧投資分析

Artificial Intelligence for Investment Analysis

Python Scikit-Learn 機器學習

(Machine Learning with Scikit-Learn in Python)

1071AIIA08

EMBA, IMTKU (M2399) (8540)

Thu 12,13,14 (19:20-22:10) (D503)

Min-Yuh Day

戴敏育

Assistant Professor

專任助理教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

http://mail. tku.edu.tw/myday/

2018-11-08

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2018/09/13	人工智慧投資分析課程介紹 (Course Orientation on Artificial Intelligence for Investment Analysis)
2	2018/09/20	AI 金融科技: 金融服務創新應用 (AI in FinTech: Financial Services Innovation and Application)
3	2018/09/27	機器人理財顧問與AI交談機器人 (Robo-Advisors and AI Chatbots)
4	2018/10/04	投資心理學與行為財務學 (Investing Psychology and Behavioral Finance)
5	2018/10/11	財務金融事件研究法 (Event Studies in Finance)
6	2018/10/18	人工智慧投資分析個案研究 I (Case Study on Artificial Intelligence for Investment Analysis I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2018/10/25	Python AI投資分析基礎 (Foundations of AI Investment Analysis in Python)
8	2018/11/01	Python Pandas量化投資分析 (Quantitative Investing with Pandas in Python)
9	2018/11/08	Python Scikit-Learn 機器學習 (Machine Learning with Scikit-Learn in Python)
10	2018/11/15	期中報告 (Midterm Project Report)
11	2018/11/22	TensorFlow 深度學習財務時間序列預測 I (Deep Learning for Financial Time Series Forecasting with TensorFlow I)
12	2018/11/29	TensorFlow 深度學習財務時間序列預測 II (Deep Learning for Financial Time Series Forecasting with TensorFlow II)

課程大綱 (Syllabus)

週次 (Week)　日期 (Date)　內容 (Subject/Topics)

13 2018/12/06 人工智慧投資分析個案研究 II
(Case Study on Artificial Intelligence for Investment Analysis II)

14 2018/12/13 TensorFlow 深度學習財務時間序列預測 III
(Deep Learning for Financial Time Series Forecasting with TensorFlow III)

15 2018/12/20 投資組合最佳化與程式交易
(Portfolio Optimization and Algorithmic Trading)

16 2018/12/27 自然語言處理 (Natural Language Processing)

17 2019/01/03 期末報告 I (Final Project Presentation I)

18 2019/01/10 期末報告 II (Final Project Presentation II)

Machine Learning with Scikit-Learn in Python

Outline

- Machine Learning with Scikit-Learn in Python
  - Machine Learning
  - Scikit-Learn

Artificial Intelligence (A.I.) Timeline

A.I. TIMELINE

1950
TURING TEST
Computer scientist Alan Turing proposes a test for machine intelligence. If a machine can trick humans into thinking it is human, then it has intelligence

1955
A.I. BORN
Term 'artificial intelligence' is coined by computer scientist, John McCarthy to describe "the science and engineering of making intelligent machines"

1961
UNIMATE
First industrial robot, Unimate, goes to work at GM replacing humans on the assembly line

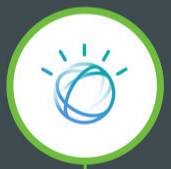
1964
ELIZA
Pioneering chatbot developed by Joseph Weizenbaum at MIT holds conversations with humans

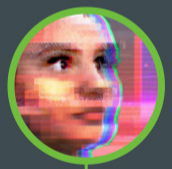
1966
SHAKEY
The 'first electronic person' from Stanford, Shakey is a general-purpose mobile robot that reasons about its own actions

A.I. WINTER
Many false starts and dead-ends leave A.I. out in the cold

1997
DEEP BLUE
Deep Blue, a chess-playing computer from IBM defeats world chess champion Garry Kasparov

1998
KISMET
Cynthia Breazeal at MIT introduces KISmet, an emotionally intelligent robot insofar as it detects and responds to people's feelings





1999
AIBO
Sony launches first consumer robot pet dog AiBO (AI robot) with skills and personality that develop over time

2002
ROOMBA
First mass produced autonomous robotic vacuum cleaner from iRobot learns to navigate and clean homes

2011
SIRI
Apple integrates Siri, an intelligent virtual assistant with a voice interface, into the iPhone 4S

2011
WATSON
IBM's question answering computer Watson wins first place on popular $1M prize television quiz show *Jeopardy*

2014
EUGENE
Eugene Goostman, a chatbot passes the Turing Test with a third of judges believing Eugene is human

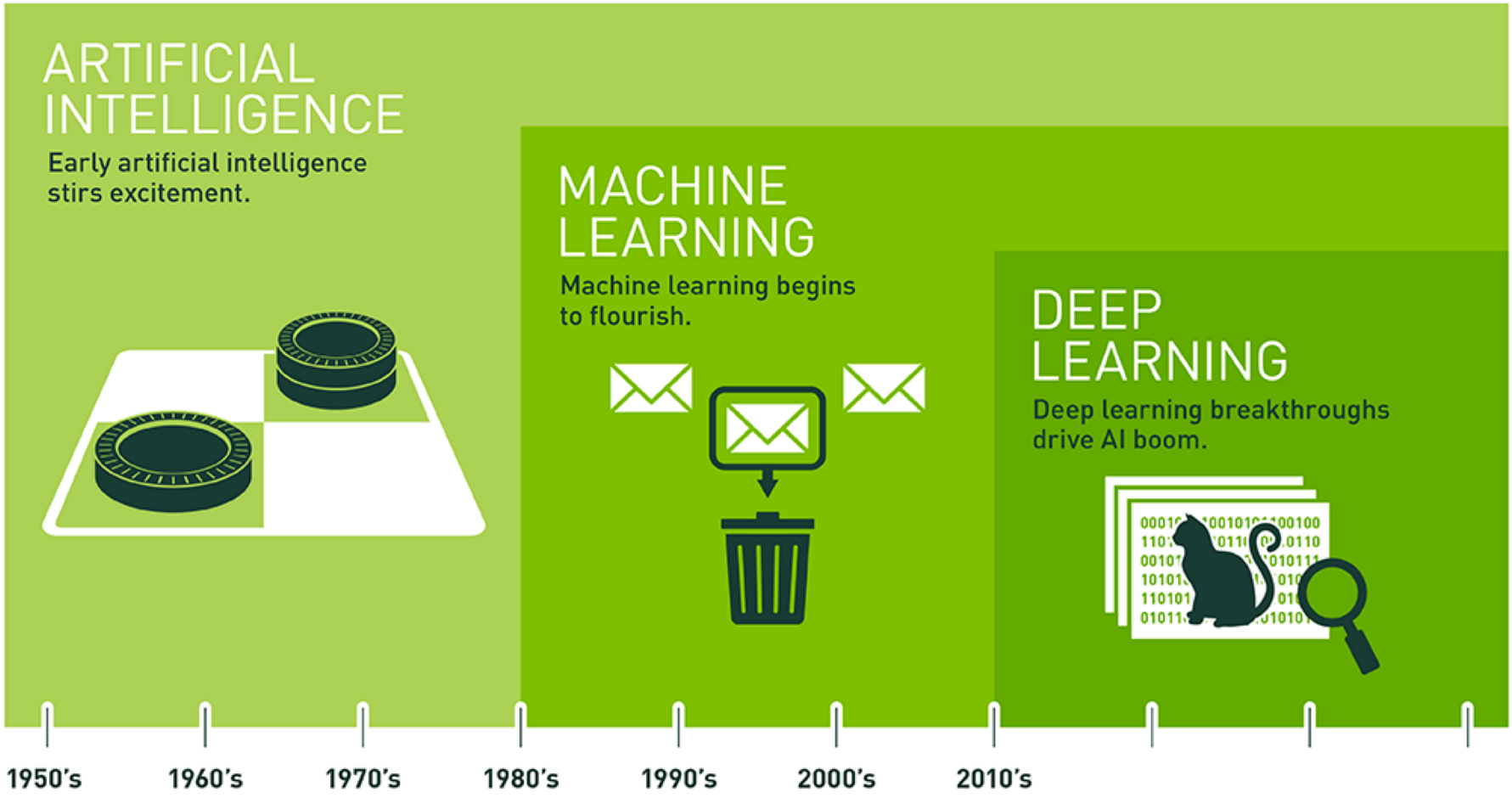
2014
ALEXA
Amazon launches Alexa, an intelligent virtual assistant with a voice interface that completes shopping tasks

2016
TAY
Microsoft's chatbot Tay goes rogue on social media making inflammatory and offensive racist comments

2017
ALPHAGO
Google's A.I. AlphaGo beats world champion Ke Jie in the complex board game of Go, notable for its vast number (2^{170}) of possible positions

Source: https://digitalintelligencetoday.com/artificial-intelligence-timeline-infographic-from-eliza-to-tay-and-beyond/

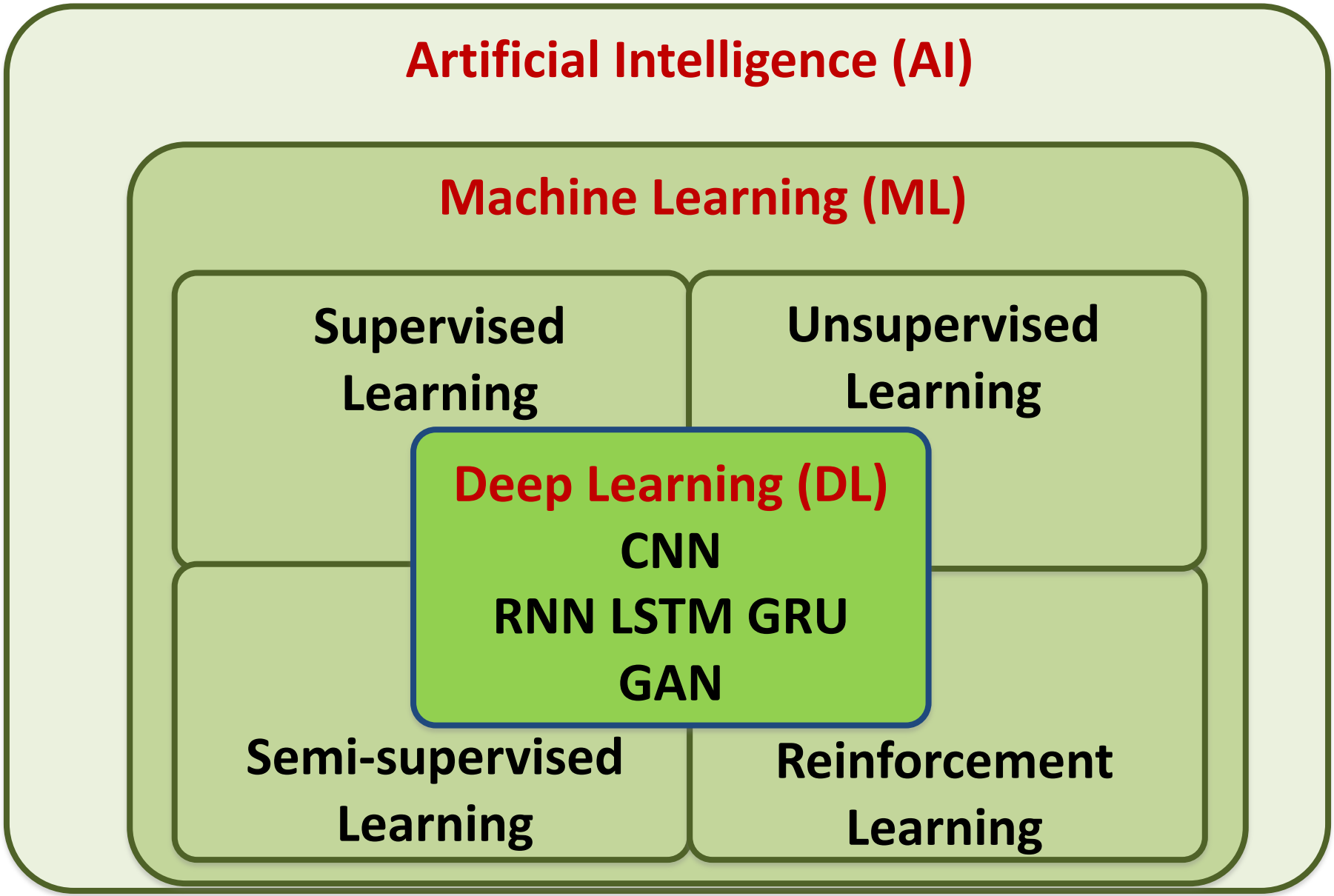
Artificial Intelligence
Machine Learning & Deep Learning



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.

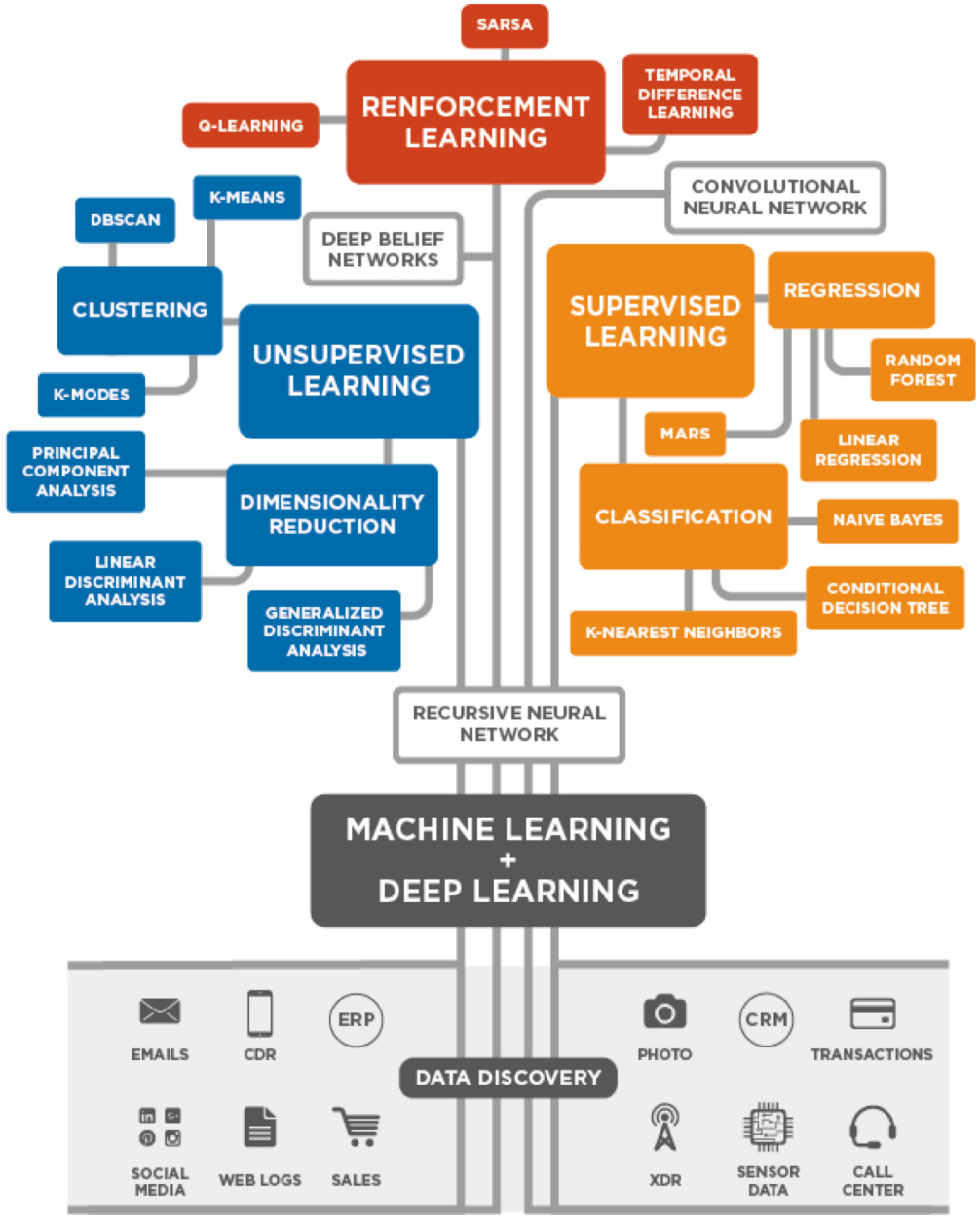
Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

AI, ML, DL



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

3 Machine Learning Algorithms



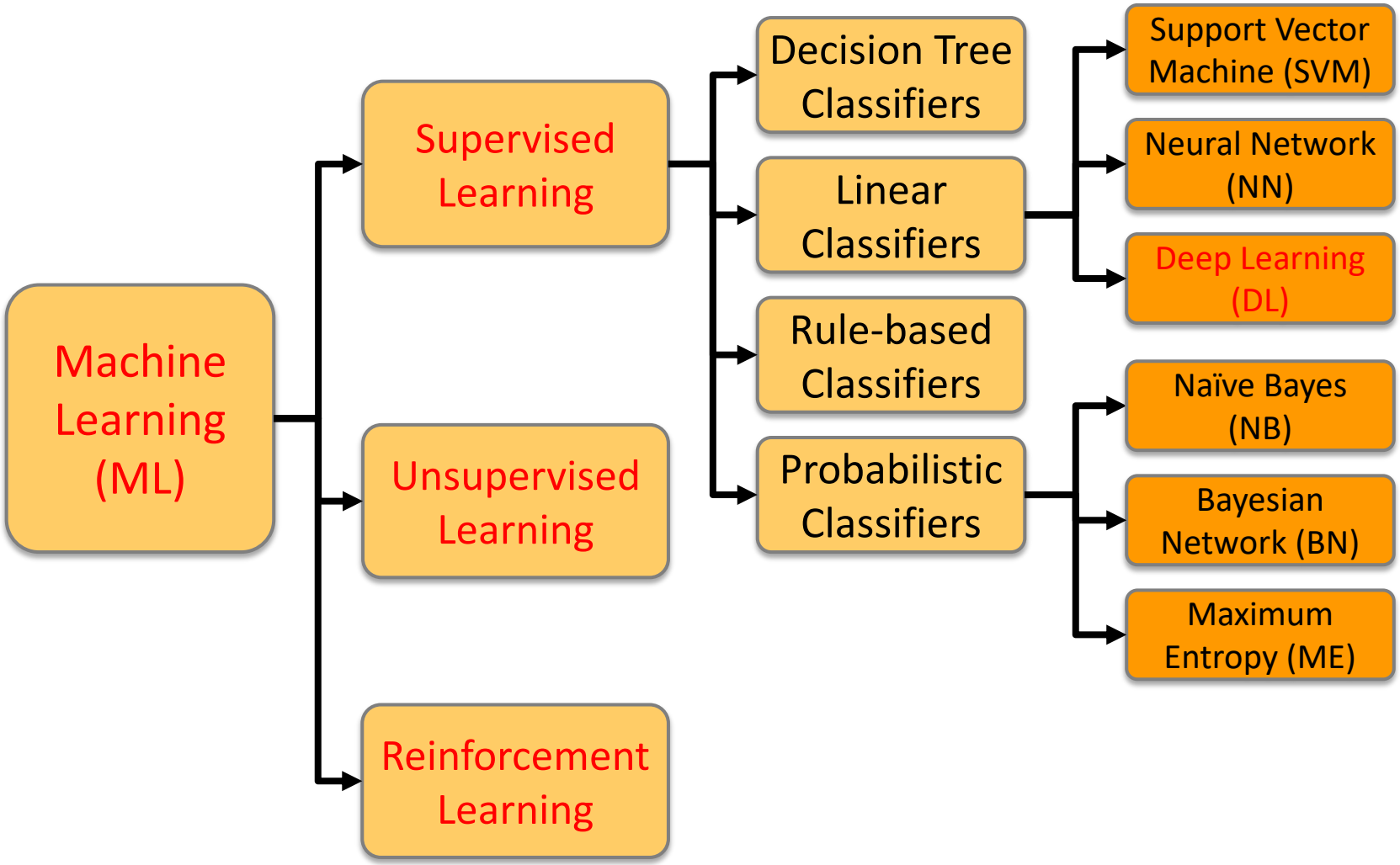
Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

Machine Learning Models

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

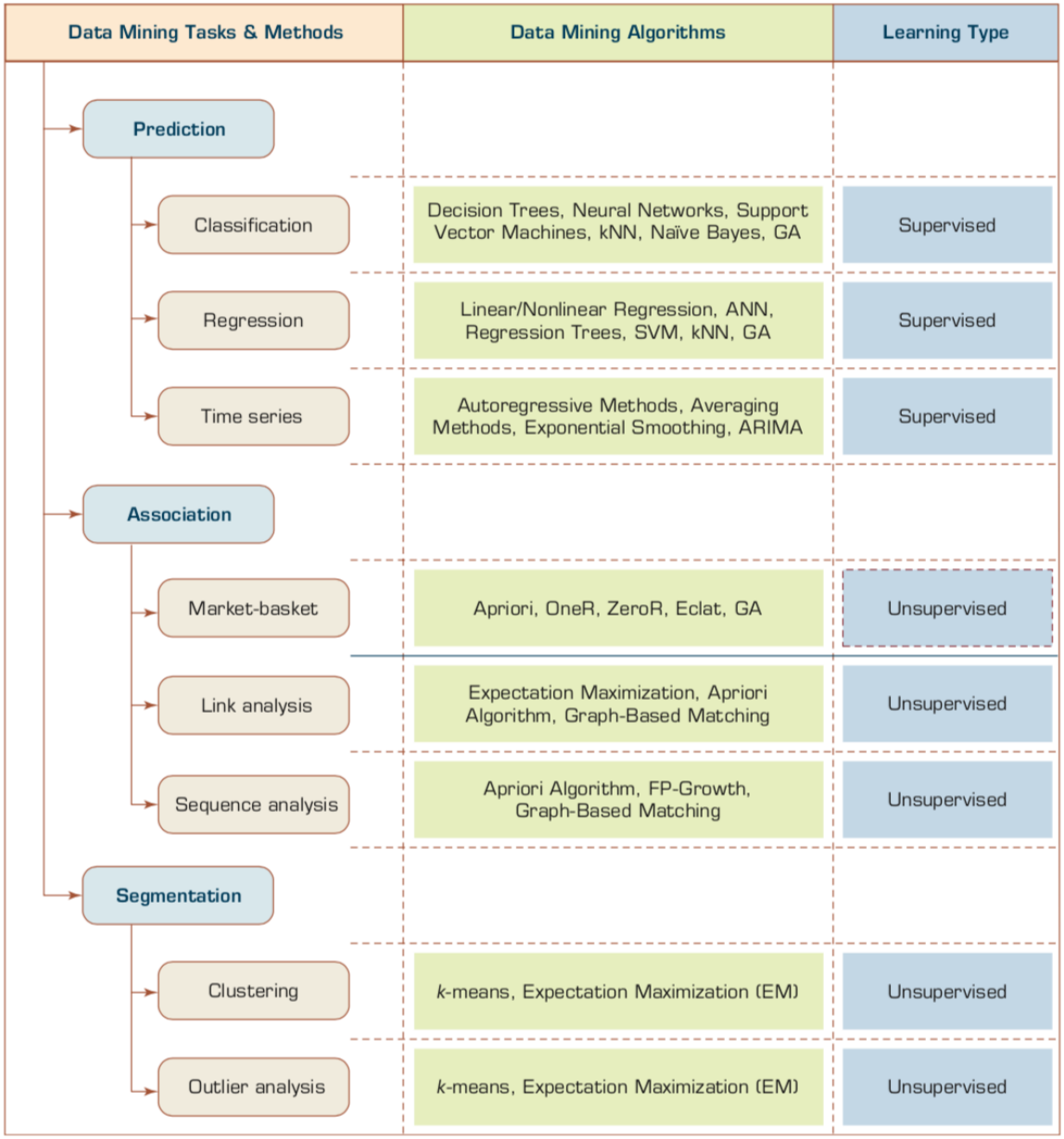
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

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.

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

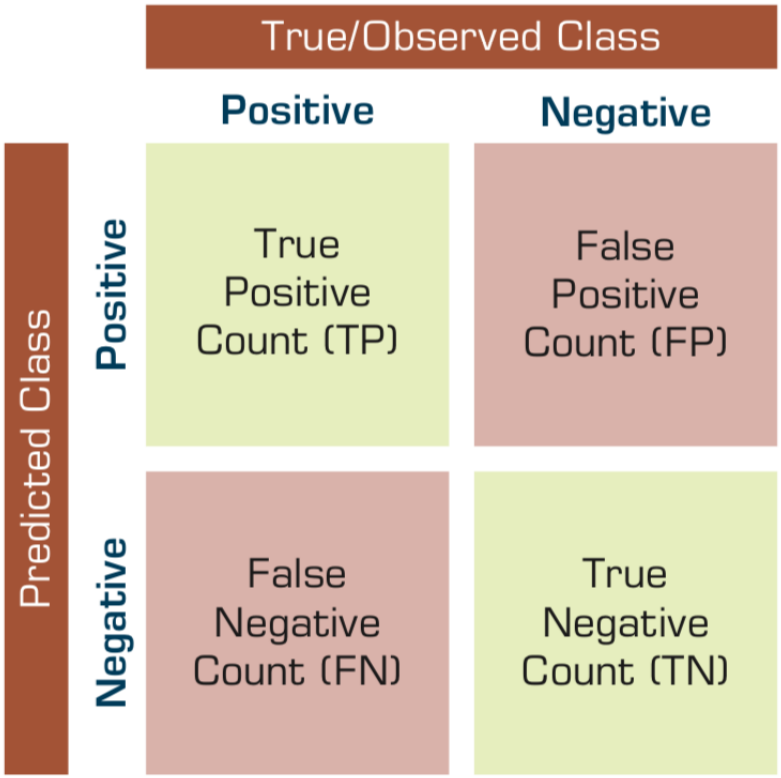
Data Mining Methods

- Classification
  - –Classification
    - Class Label Prediction
  - –Regression
    - Numeric Value Prediction
- Clustering
- Association

Assessing the Classification Model

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

Confusion Matrix for Tabulation of Two-Class Classification Results



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

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

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

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

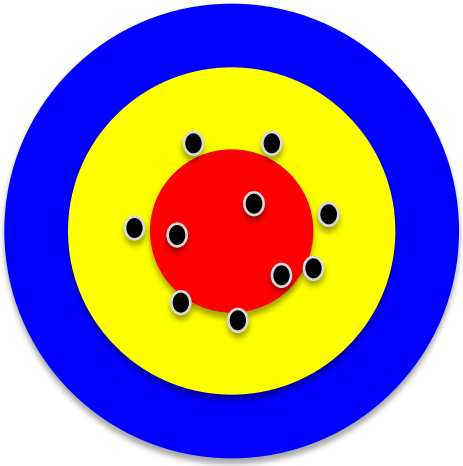
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

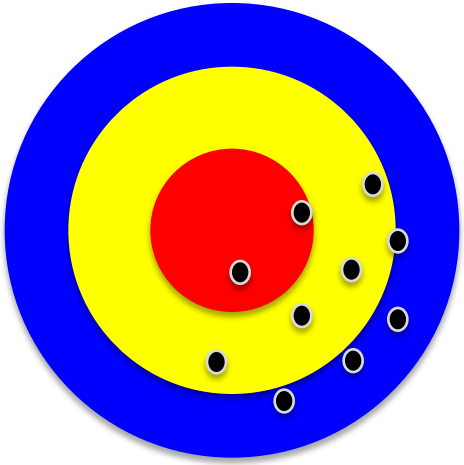
Accuracy

Validity

Precision

Reliability





Accuracy vs. Precision

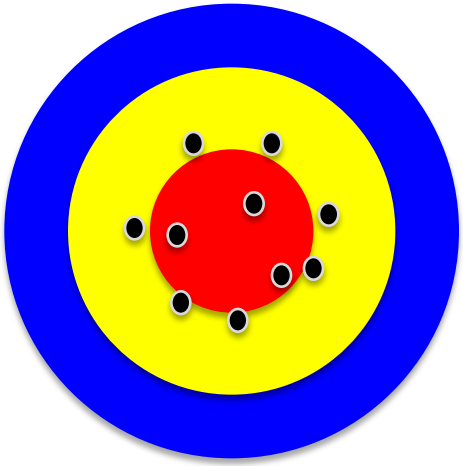
A

High Accuracy
High Precision

B

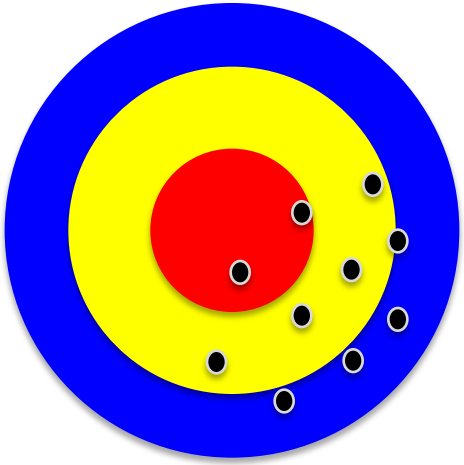
Low Accuracy
High Precision

C



High Accuracy
Low Precision

D



Low Accuracy
Low Precision

Accuracy vs. Precision

A

High Accuracy
High Precision

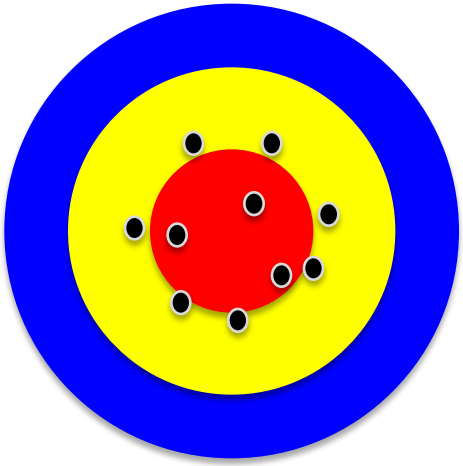
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

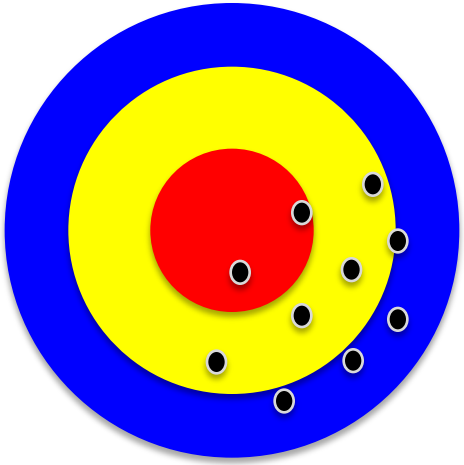
C



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

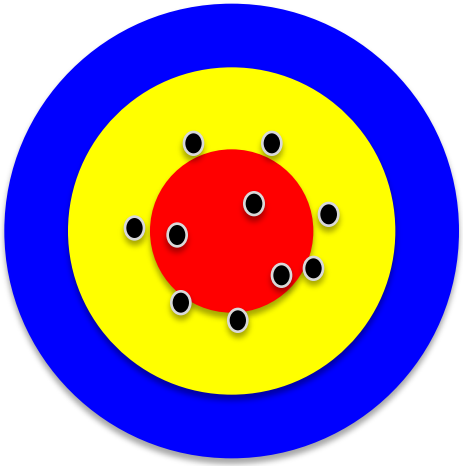
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

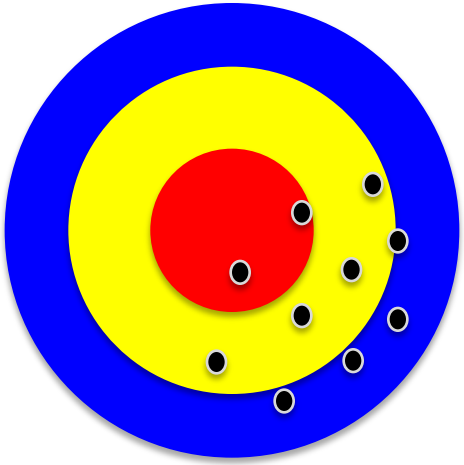
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Sensitivity =True Positive Rate

Specificity =True Negative Rate

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	

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

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

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

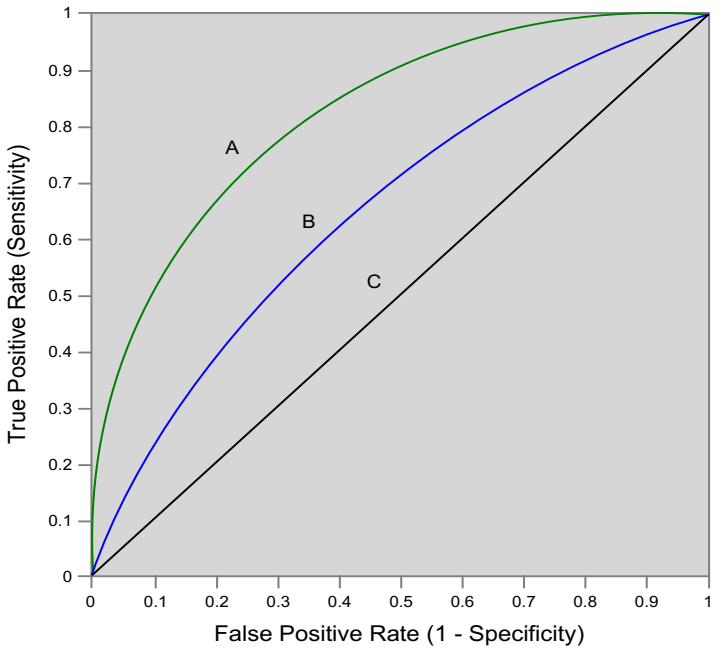
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

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

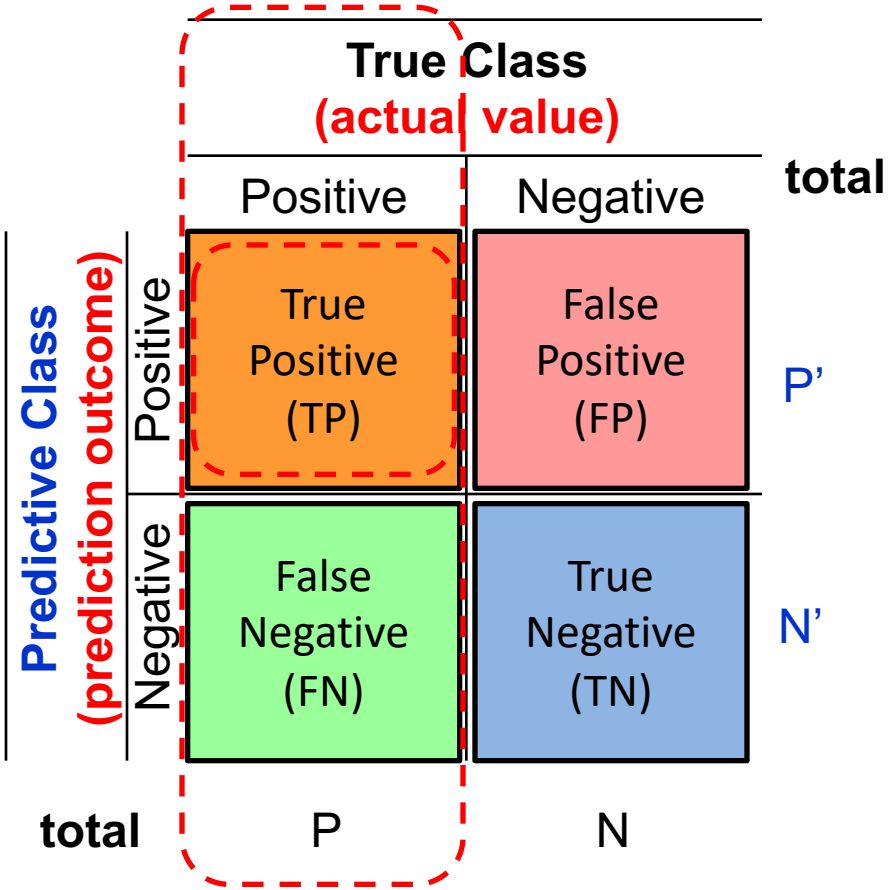
$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

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

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



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



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

Sensitivity

= True Positive Rate

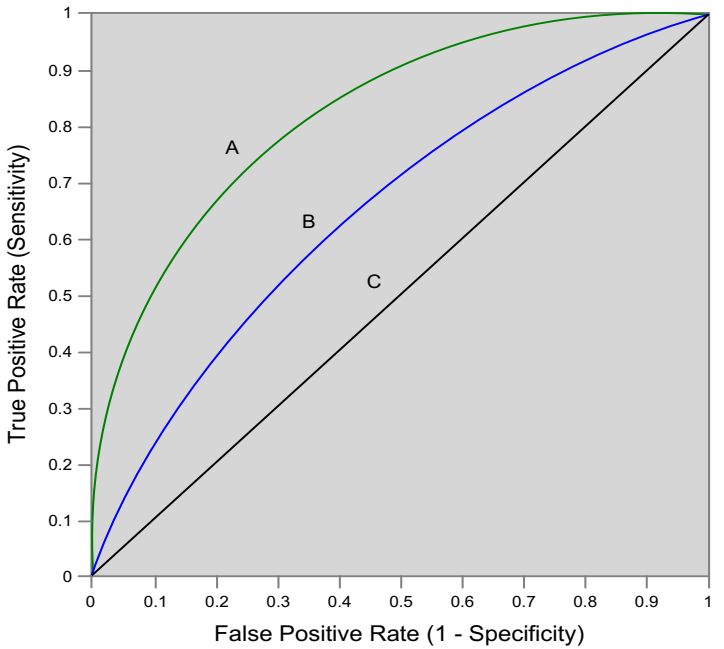
= Recall

= Hit rate

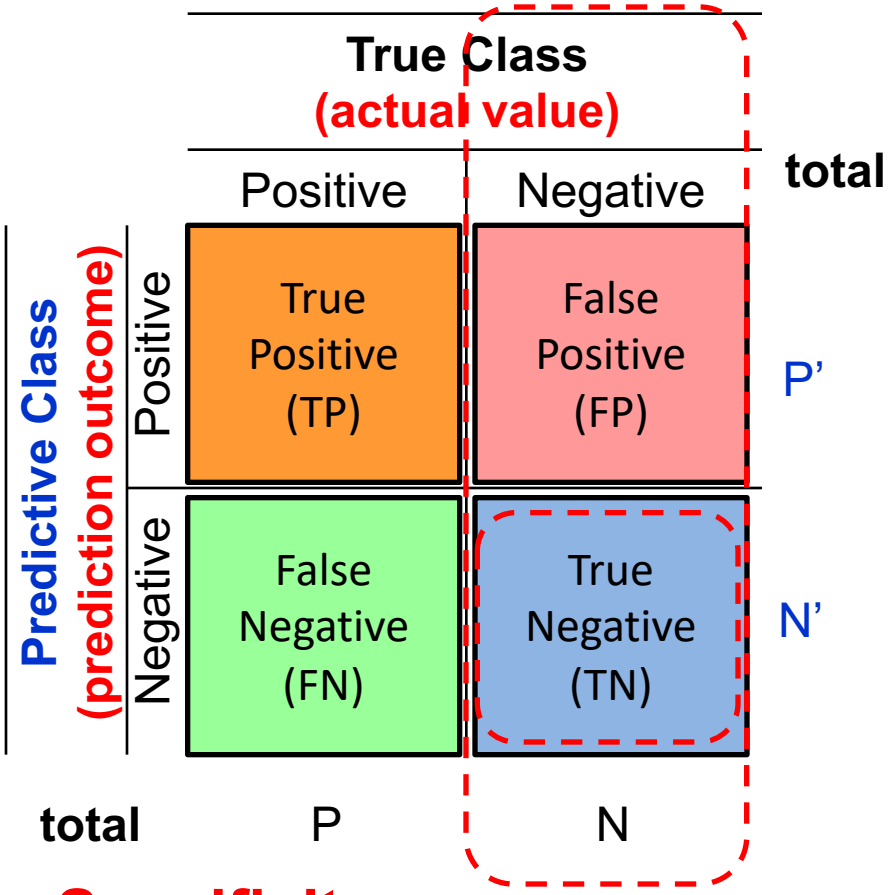
= TP / (TP + FN)

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

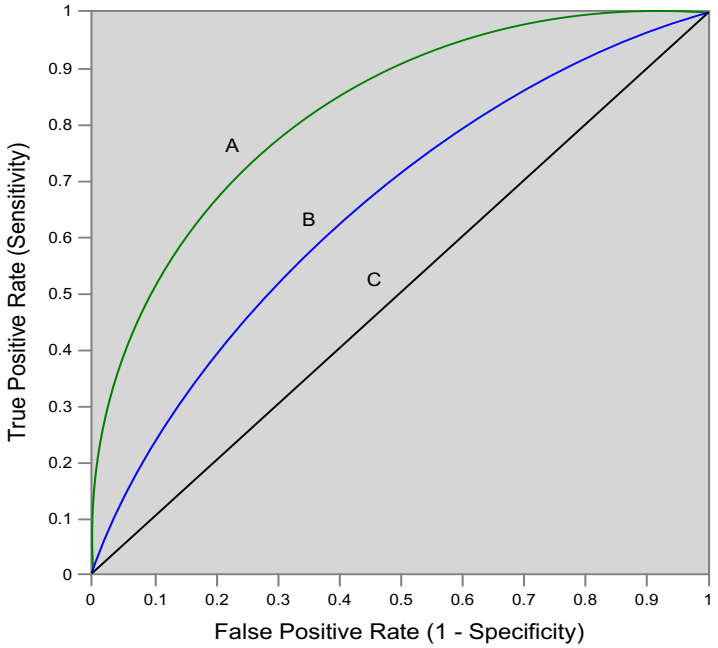
$$Recall = \frac{TP}{TP + FN}$$



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



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



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

Specificity

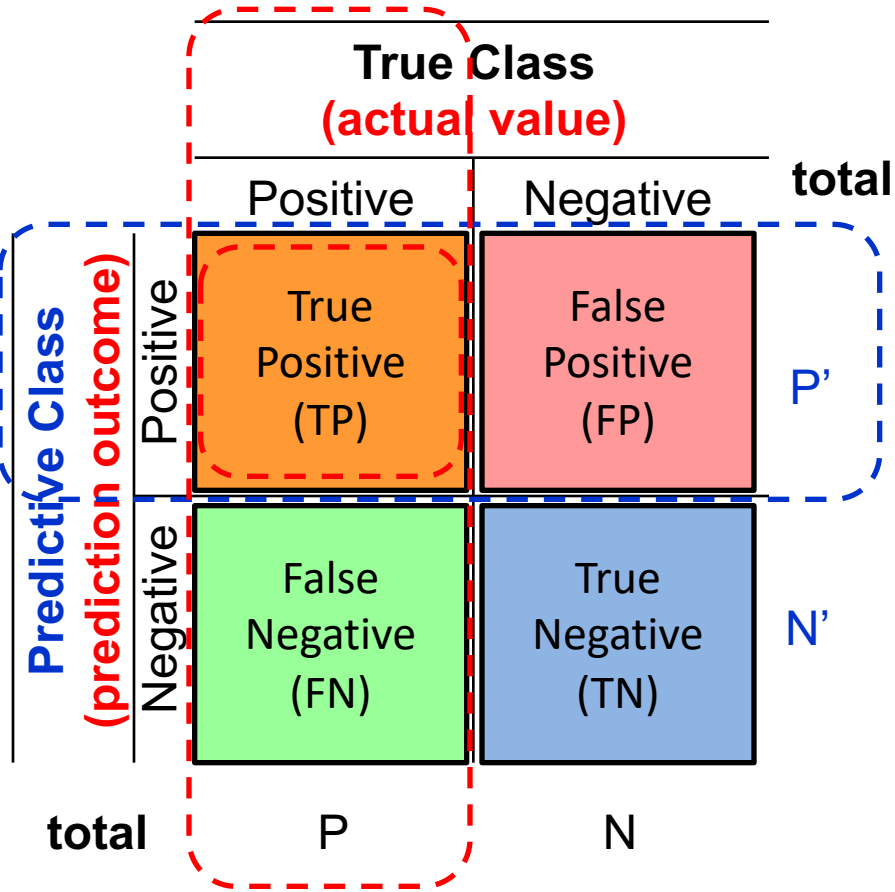
= True Negative Rate

= TN / N

= TN / (TN+ FP)

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

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



Precision
= Positive Predictive Value (PPV)

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

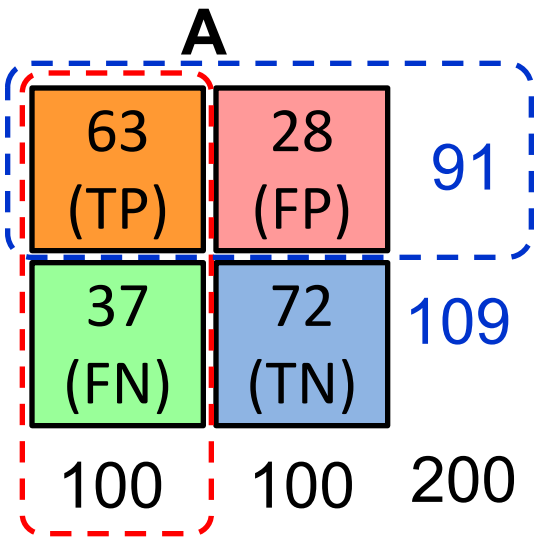
Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

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

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

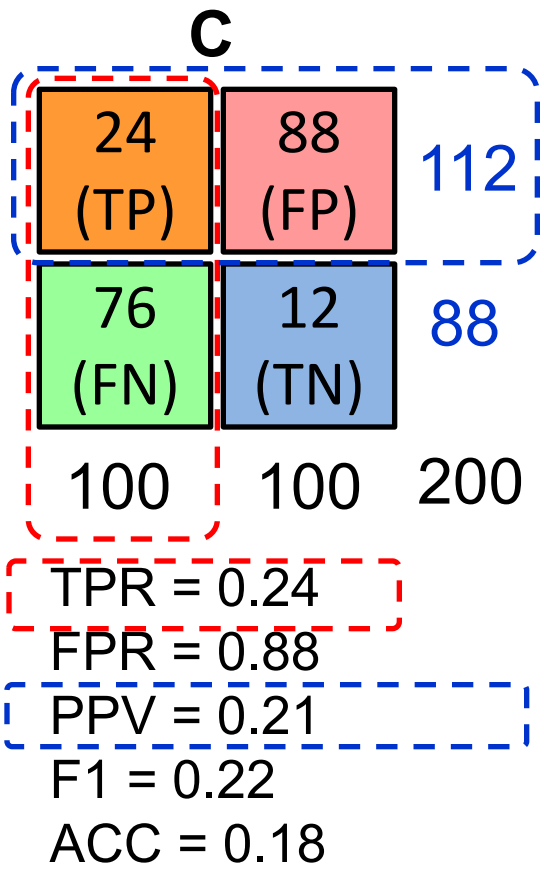
Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

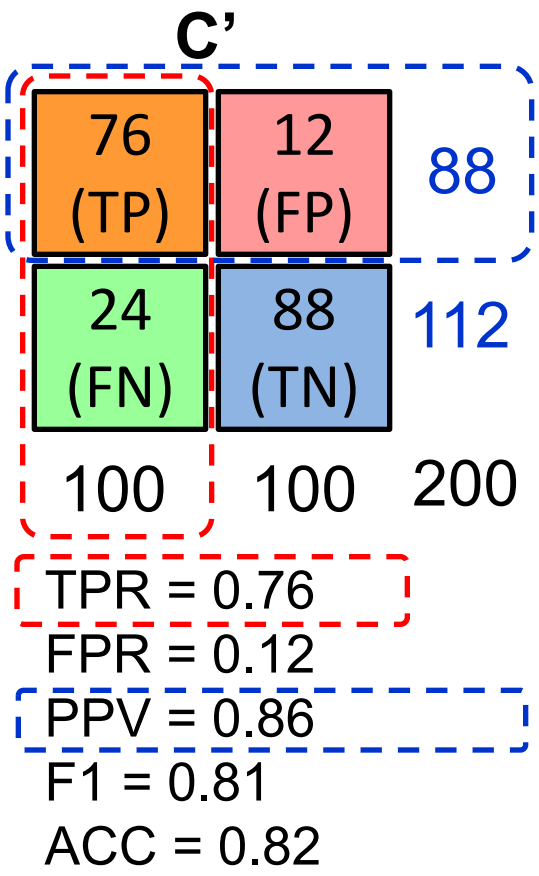
$$Recall = \frac{TP}{TP + FN}$$

Precision
= Positive Predictive Value (PPV)

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

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





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

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

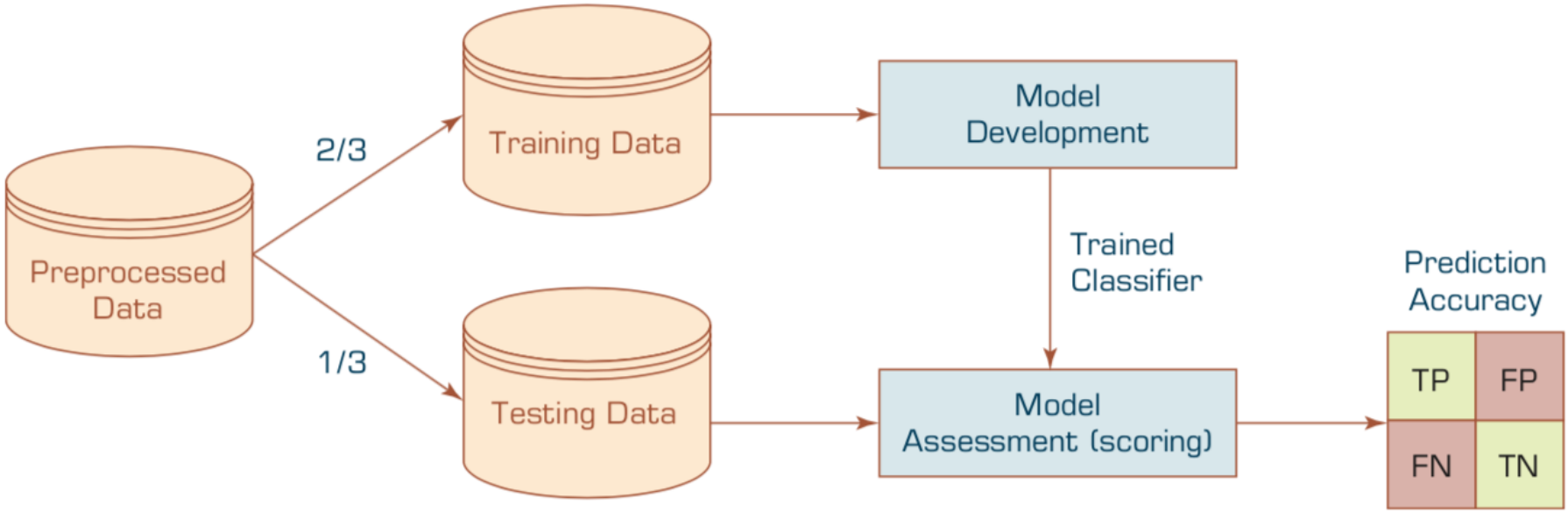
Precision
= Positive Predictive Value (PPV)

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

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

Estimation Methodologies for Classification

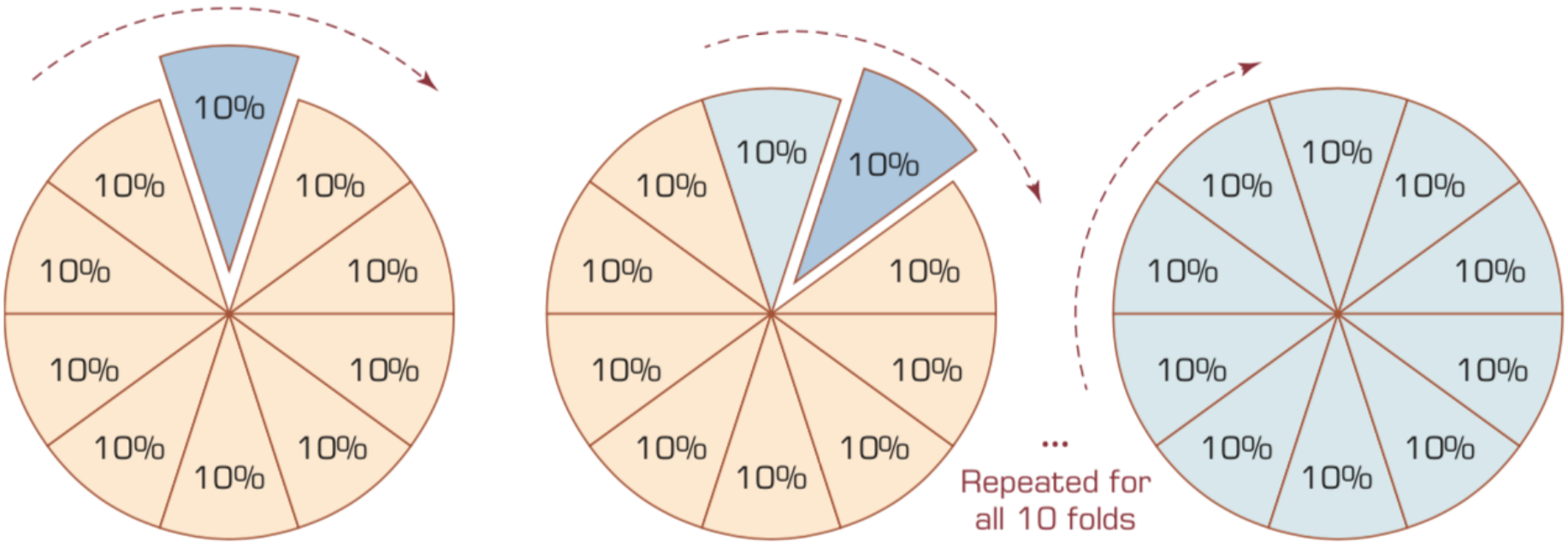
- Simple split (or holdout or test sample estimation)
  - Split the data into 2 mutually exclusive sets training (~70%) and testing (30%)



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

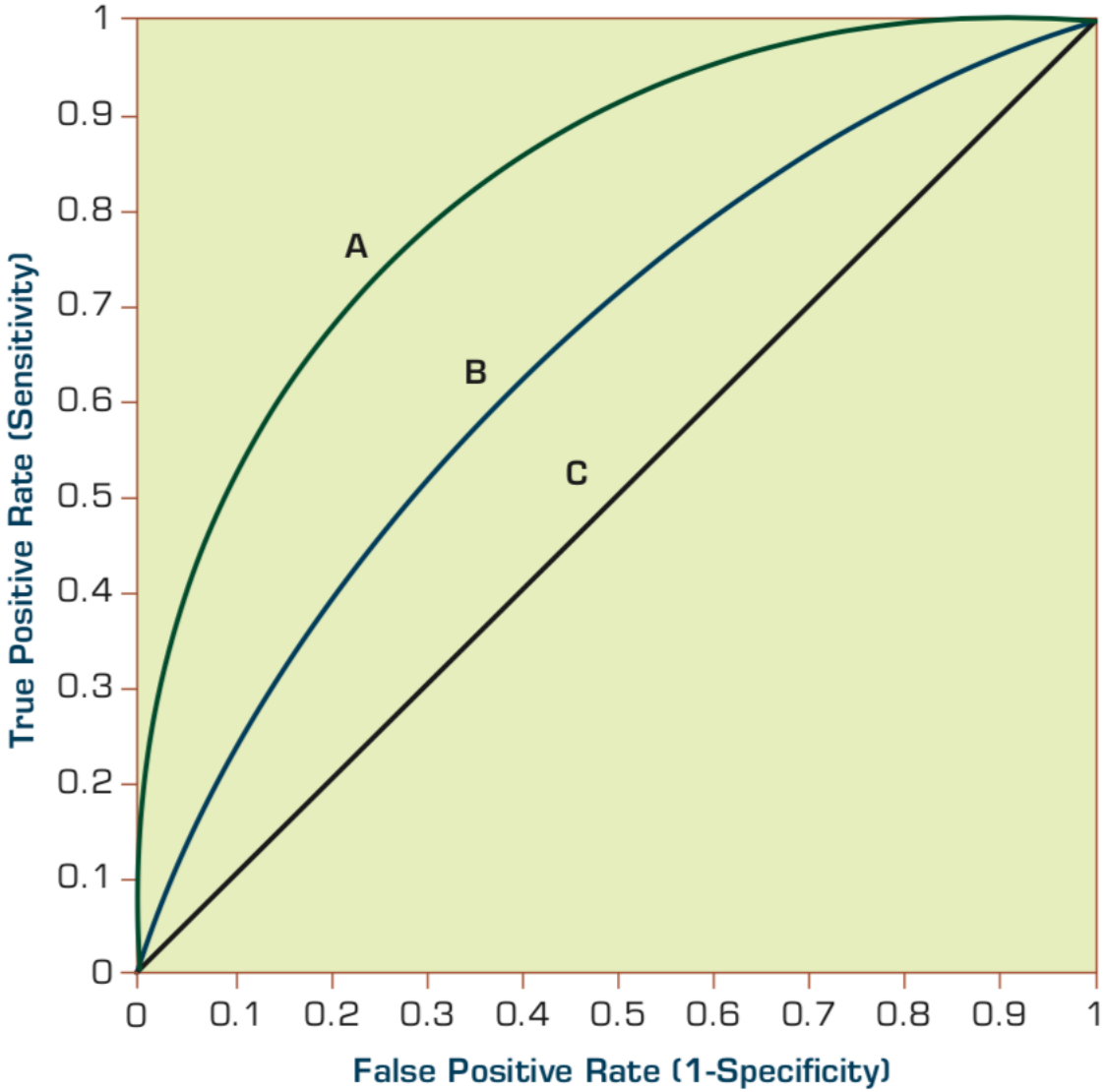
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

k-Fold Cross-Validation



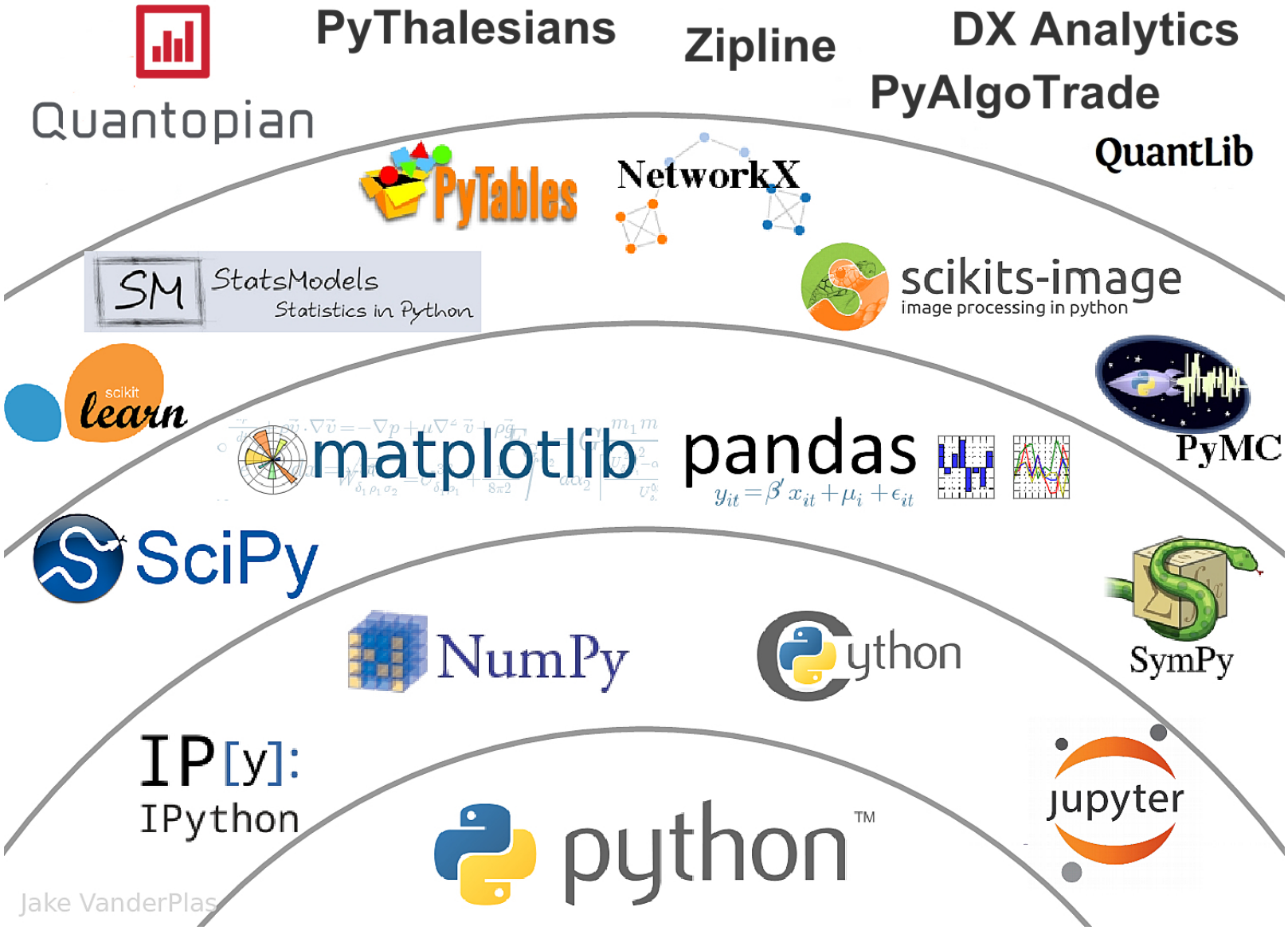
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

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

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Scikit-Learn

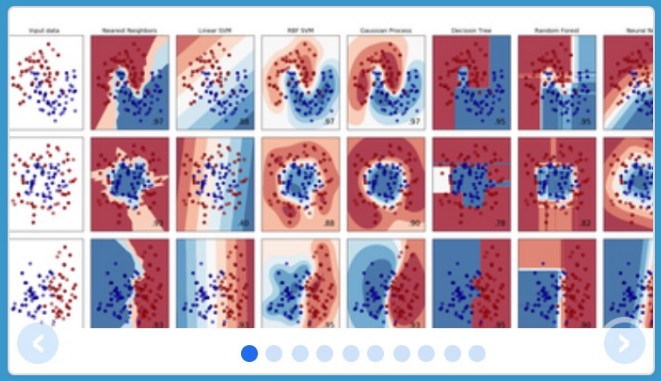
Machine Learning in Python

Scikit-Learn

scikit learn

Home Installation Documentation Examples

Google Custom Search



scikit-learn

Machine Learning in Python

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

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

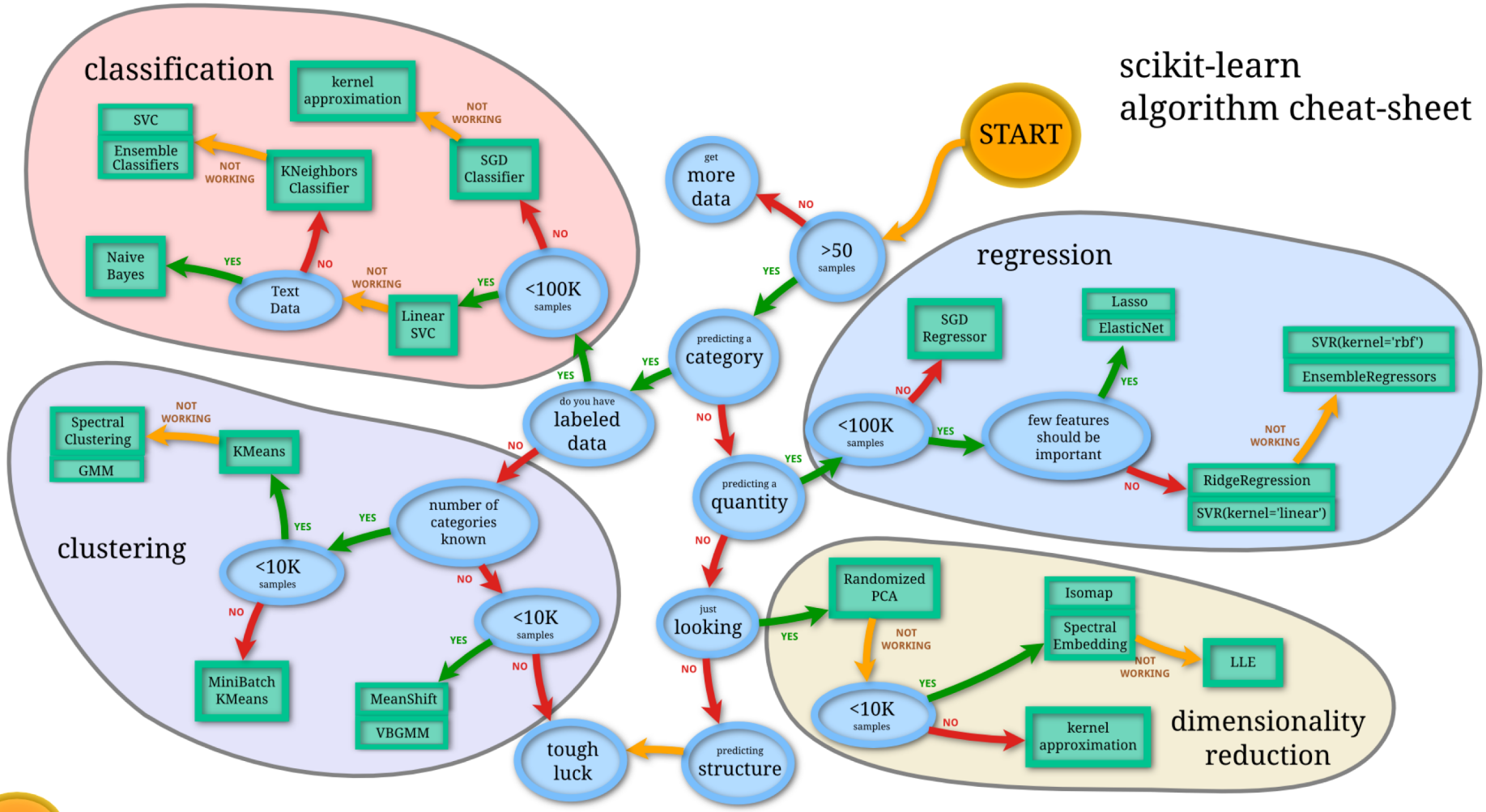
Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

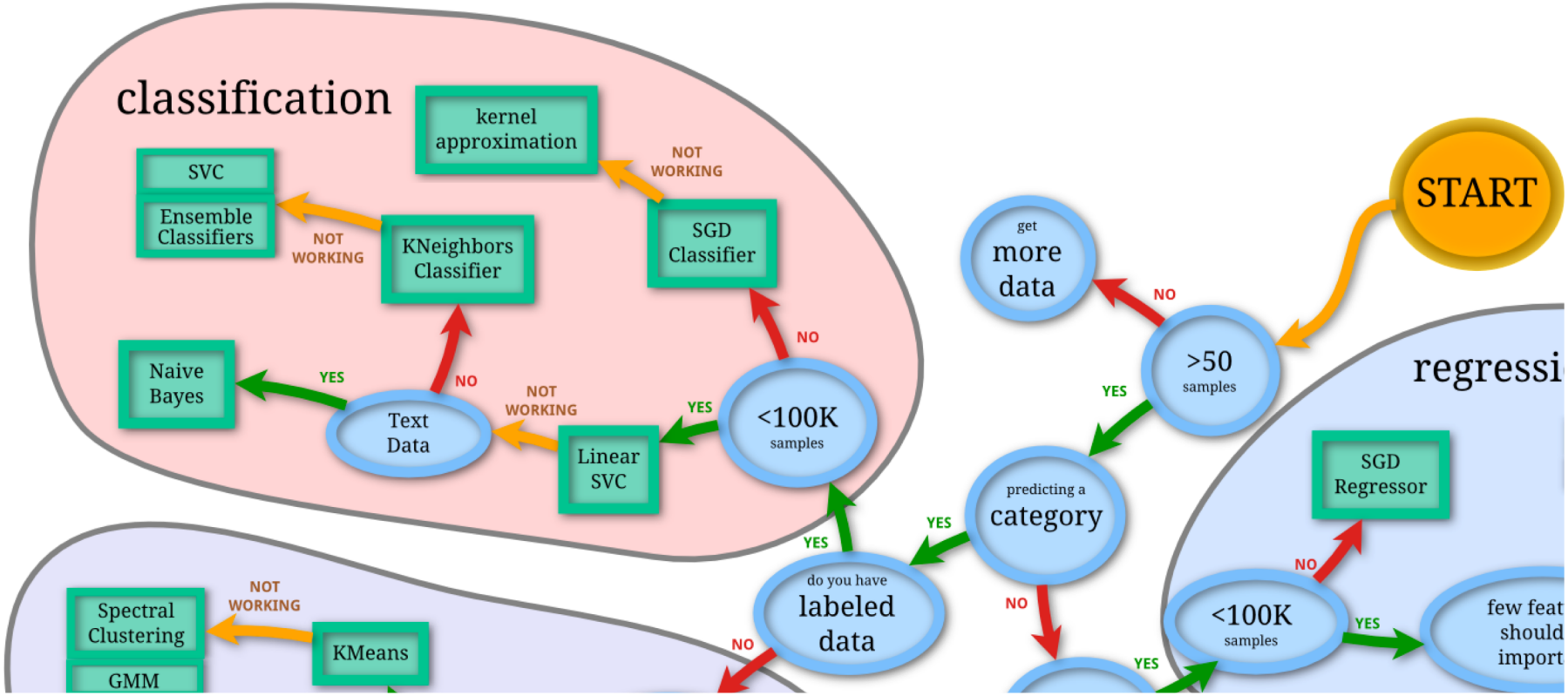
Source: http://scikit-learn.org/

Scikit-Learn Machine Learning Map



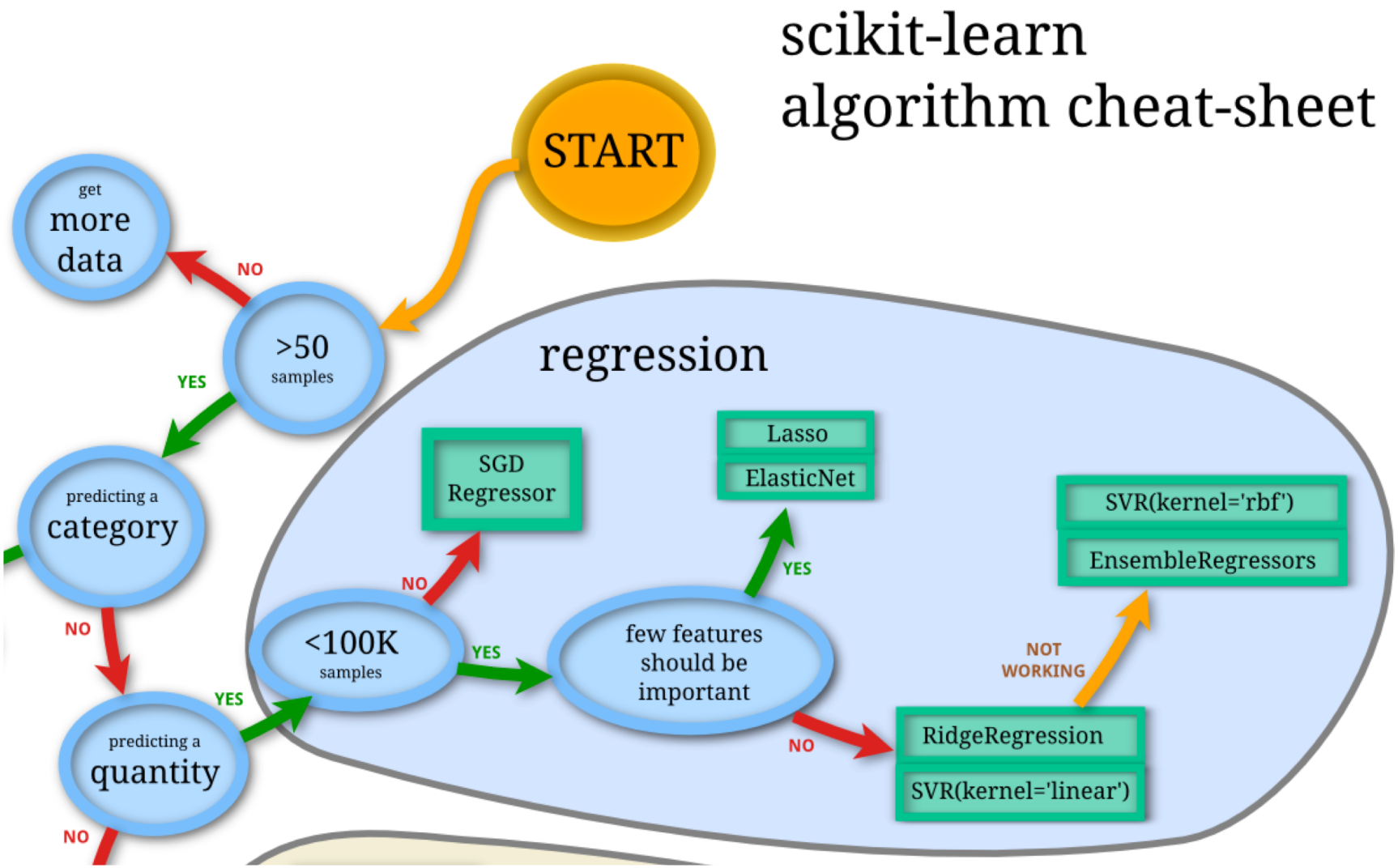
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



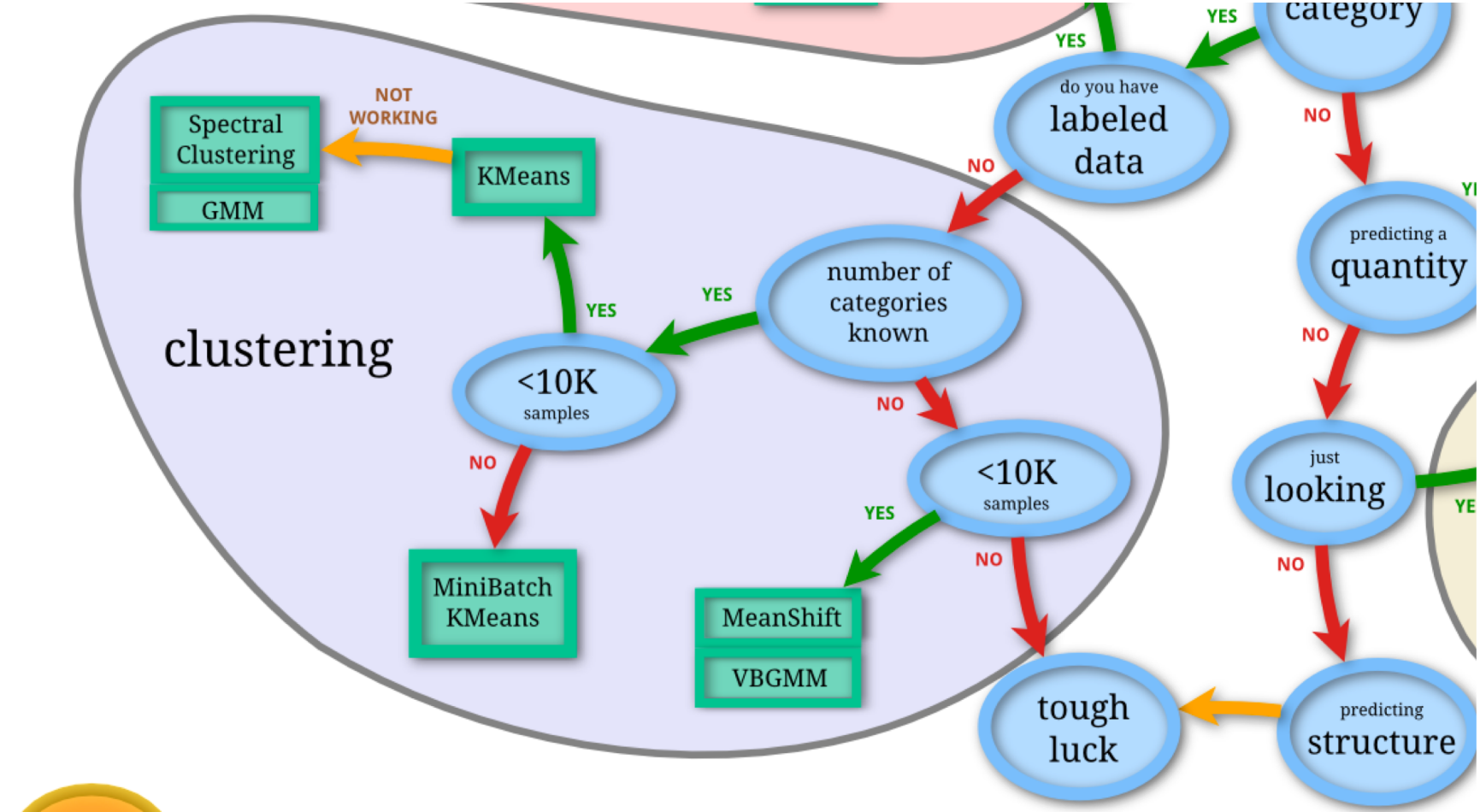
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

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

Iris flower data set

setosa

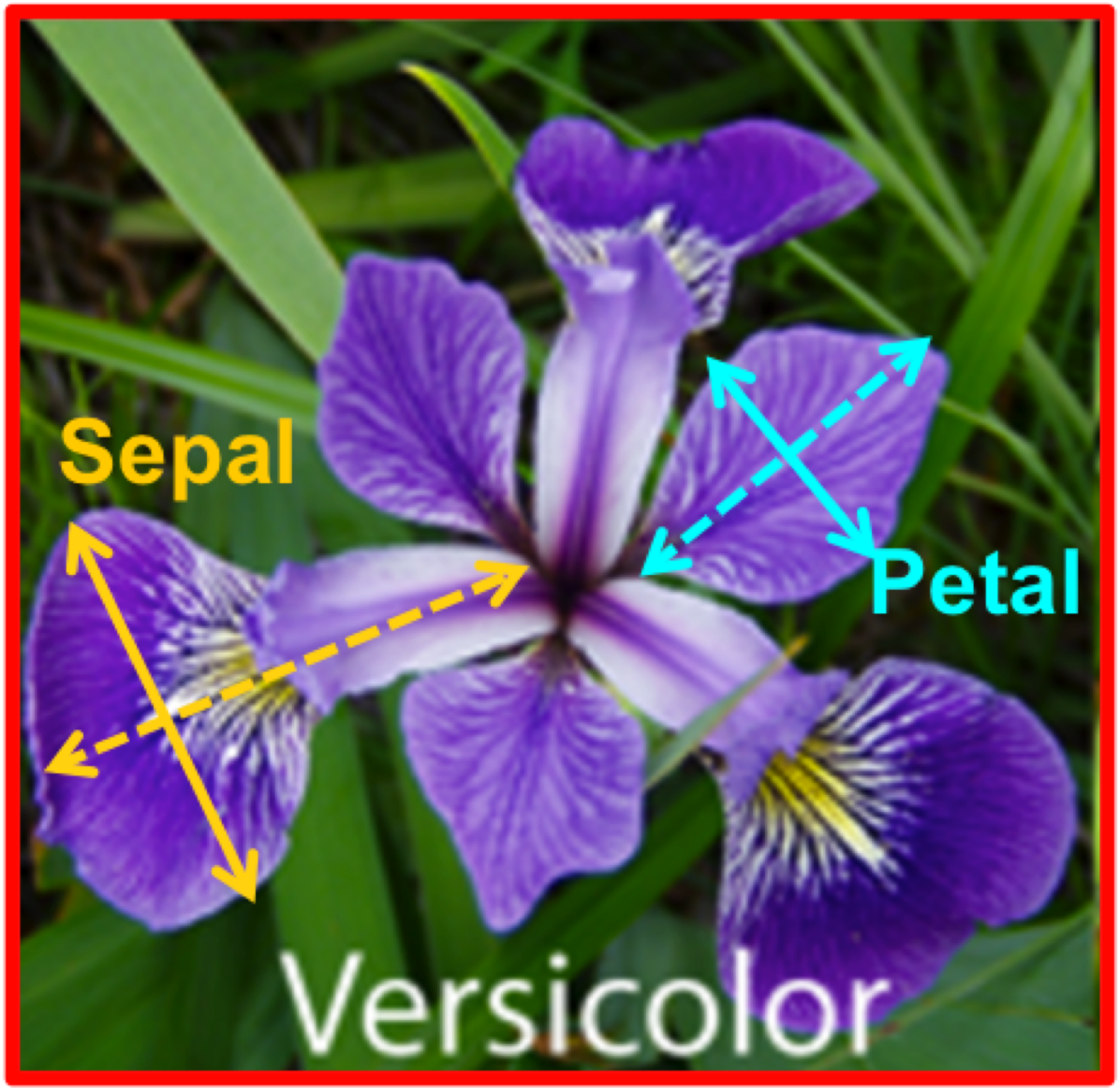
versicolor

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

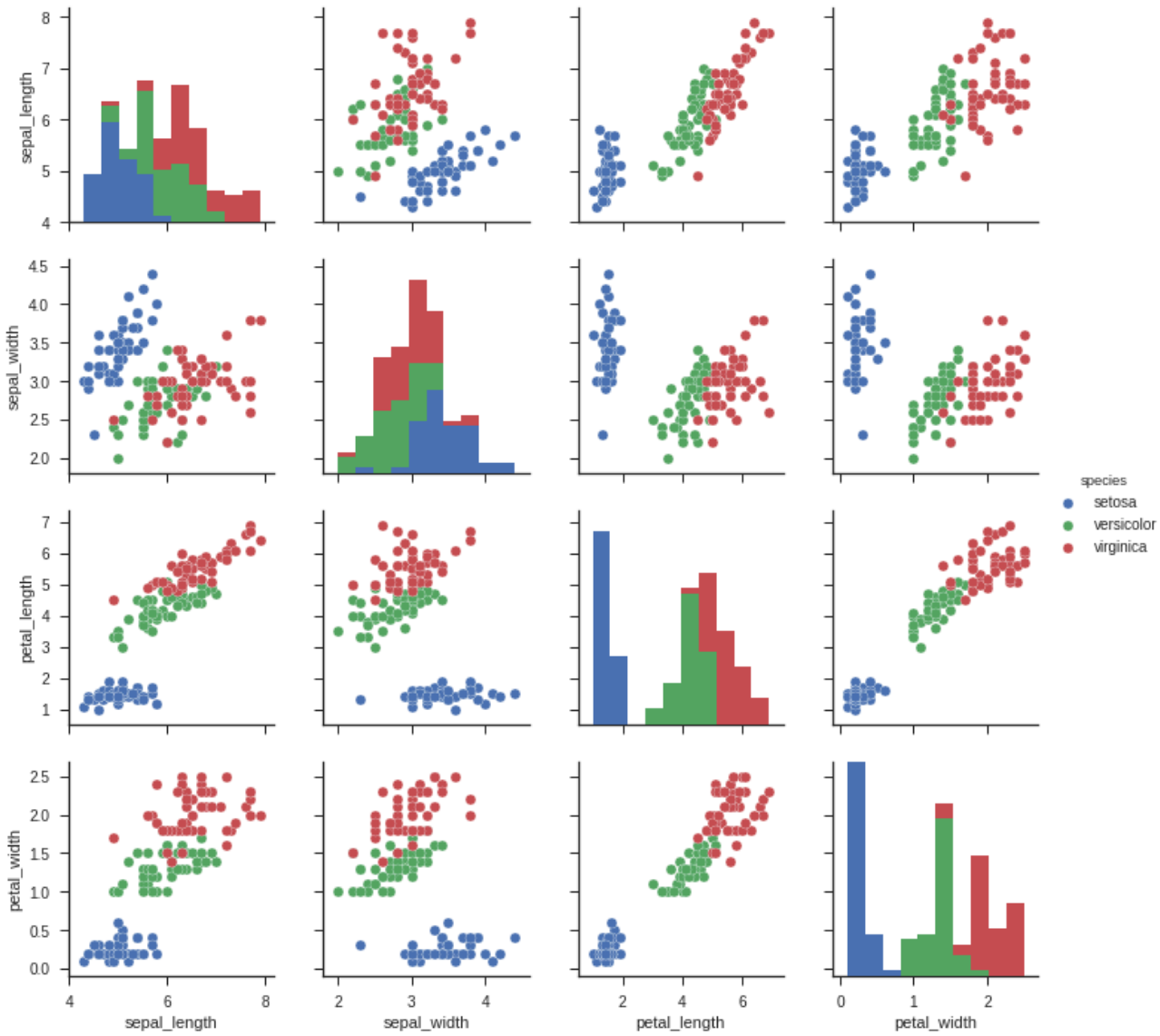
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
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

Iris Data Visualization



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

Data Visualization in Google Colab

datav.ipynb - Colaboratory

https://colab.research.google.com/drive/1KRqtEUd2Hg4dM2au9bfVQKrxWnWN3O9-?authuser=2

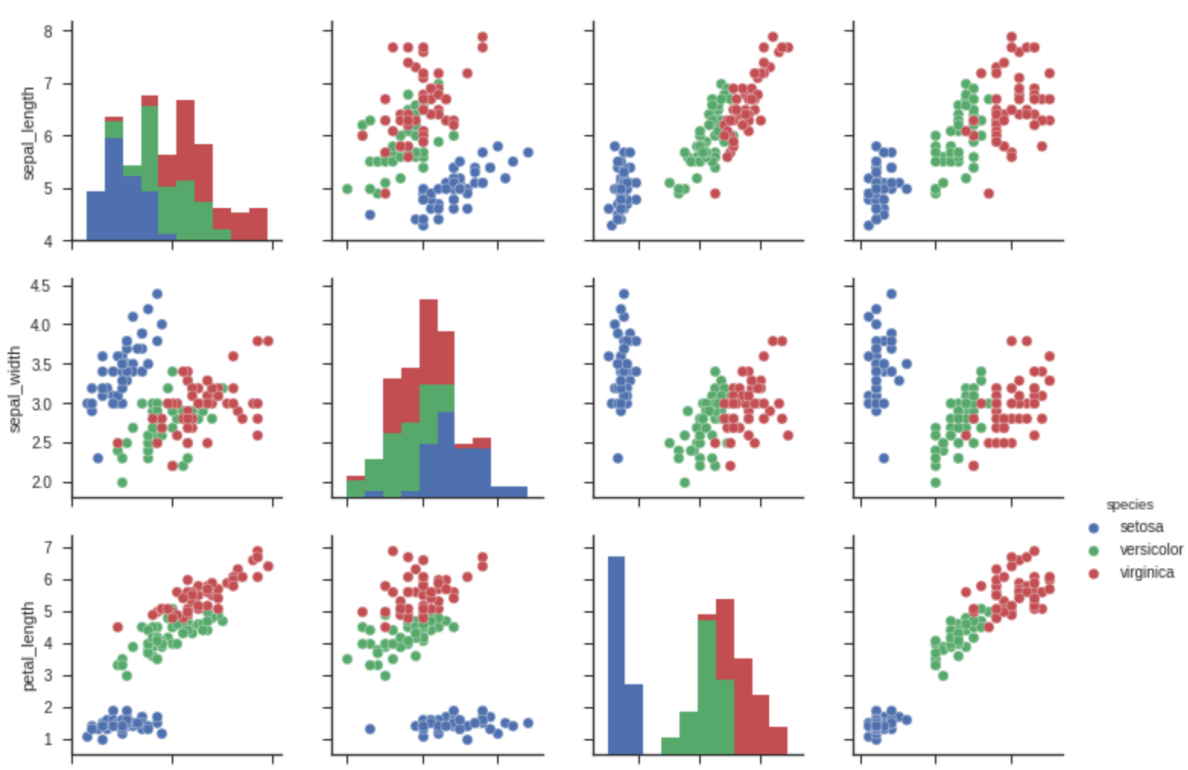
datav.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

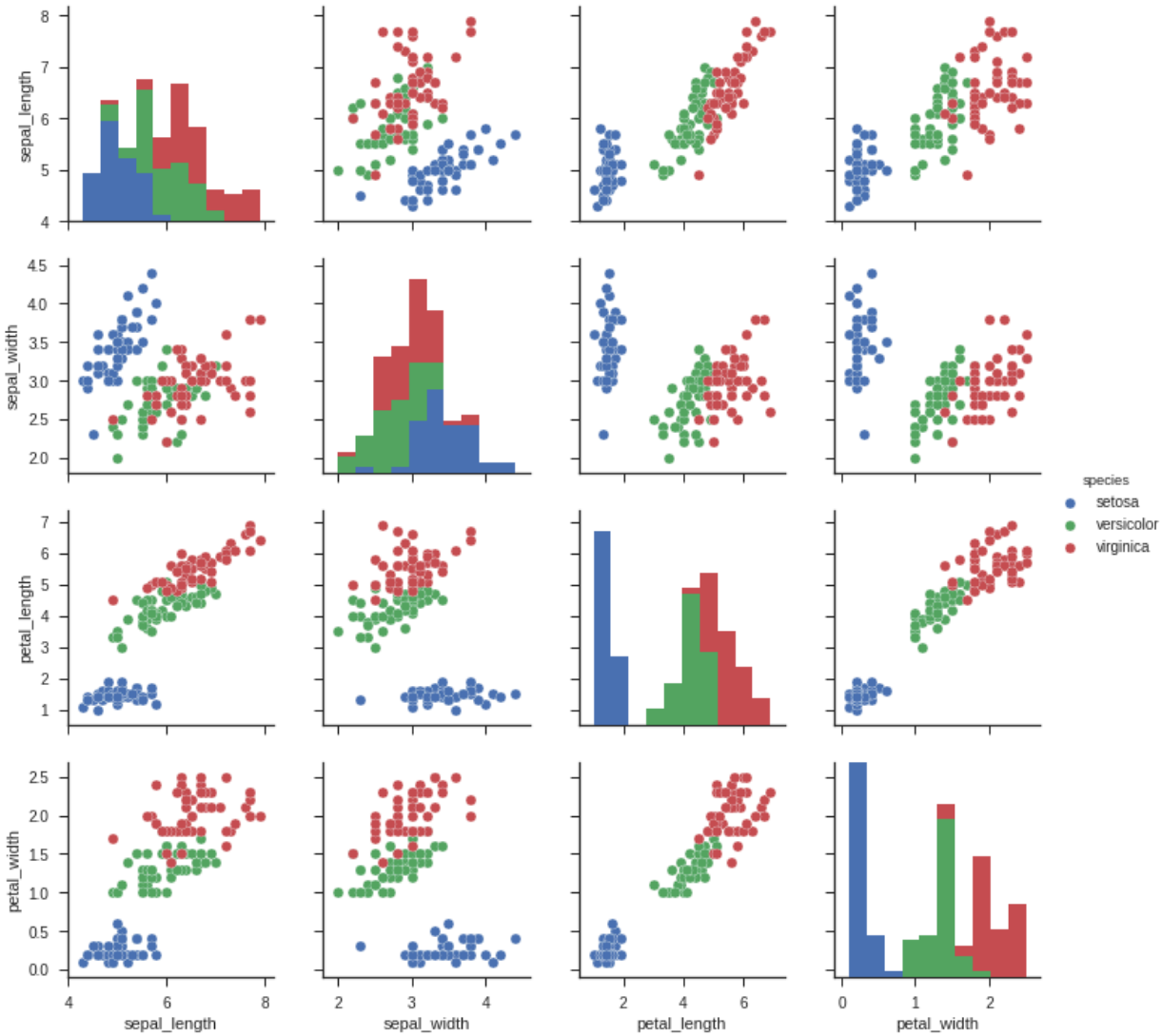
CODE TEXT CELL CELL CONNECTED EDITING

```
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 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

https://colab.research.google.com/drive/1KRqtEUd2Hg4dM2au9bfVQKrxWnWN3O9-

```
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

# 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)
```

Source: https://machinelearningmastery.com/machine-learning-in-python-step-by-step/

```
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.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
```

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

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)
```

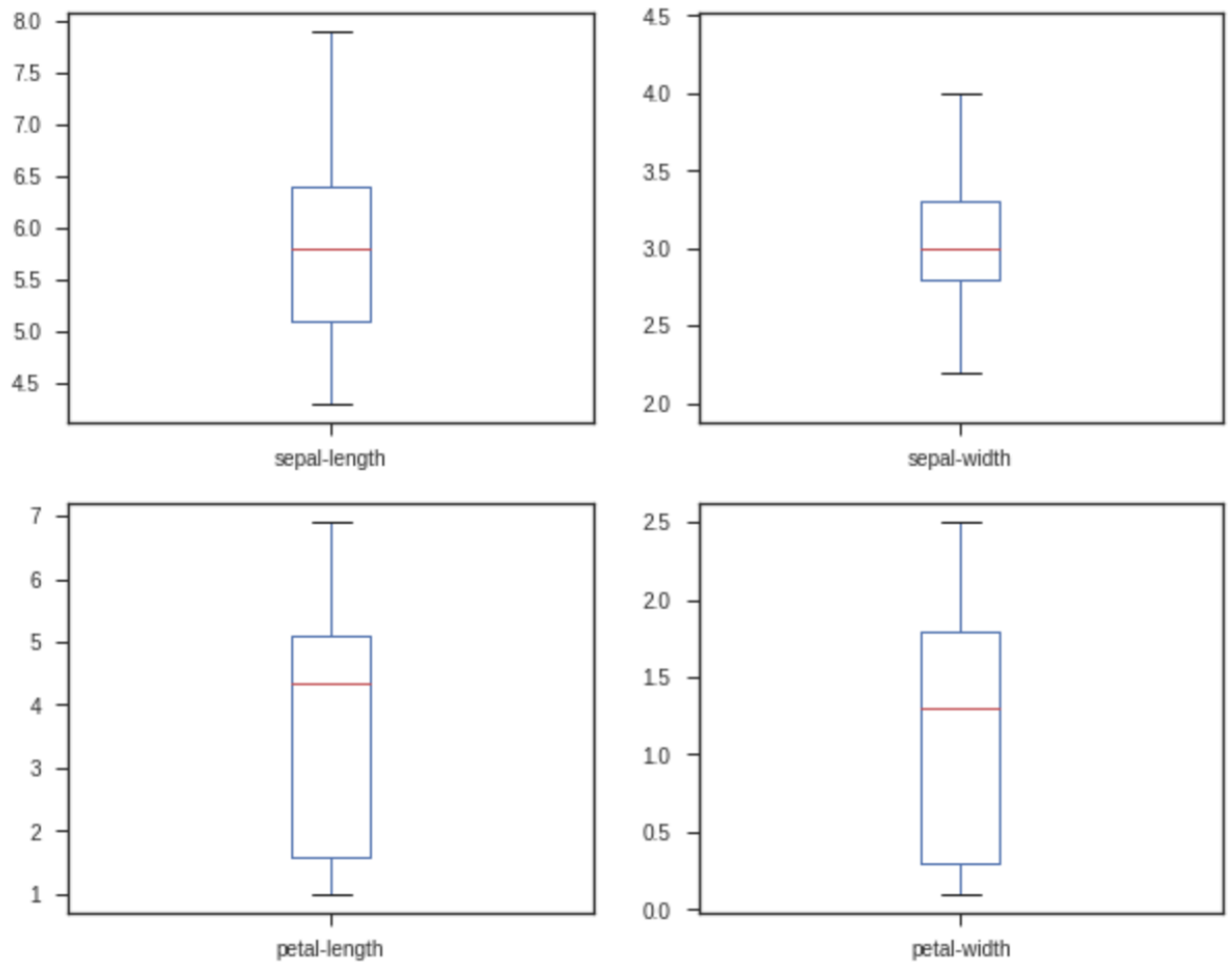
```
(150, 5)
```

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

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

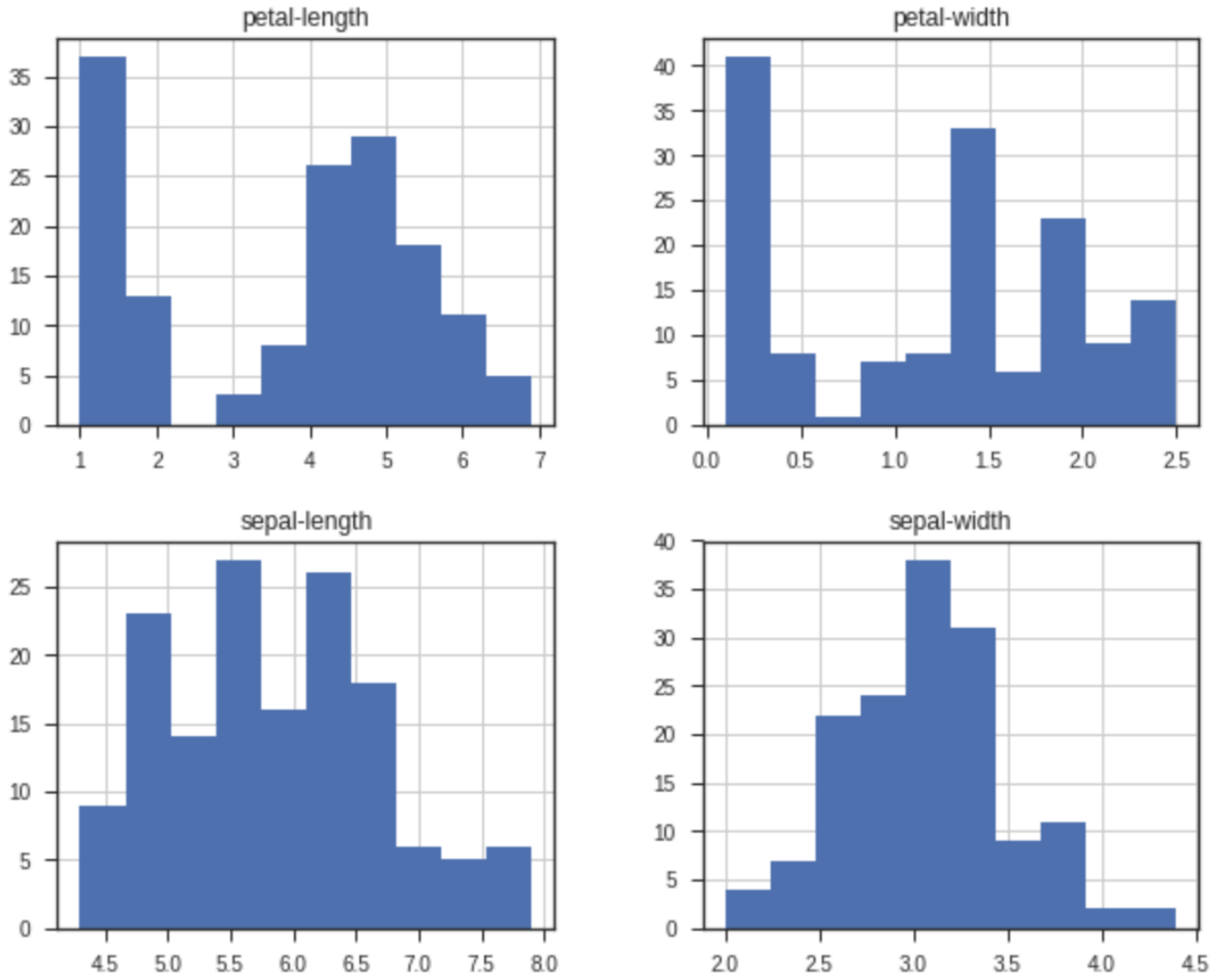
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

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



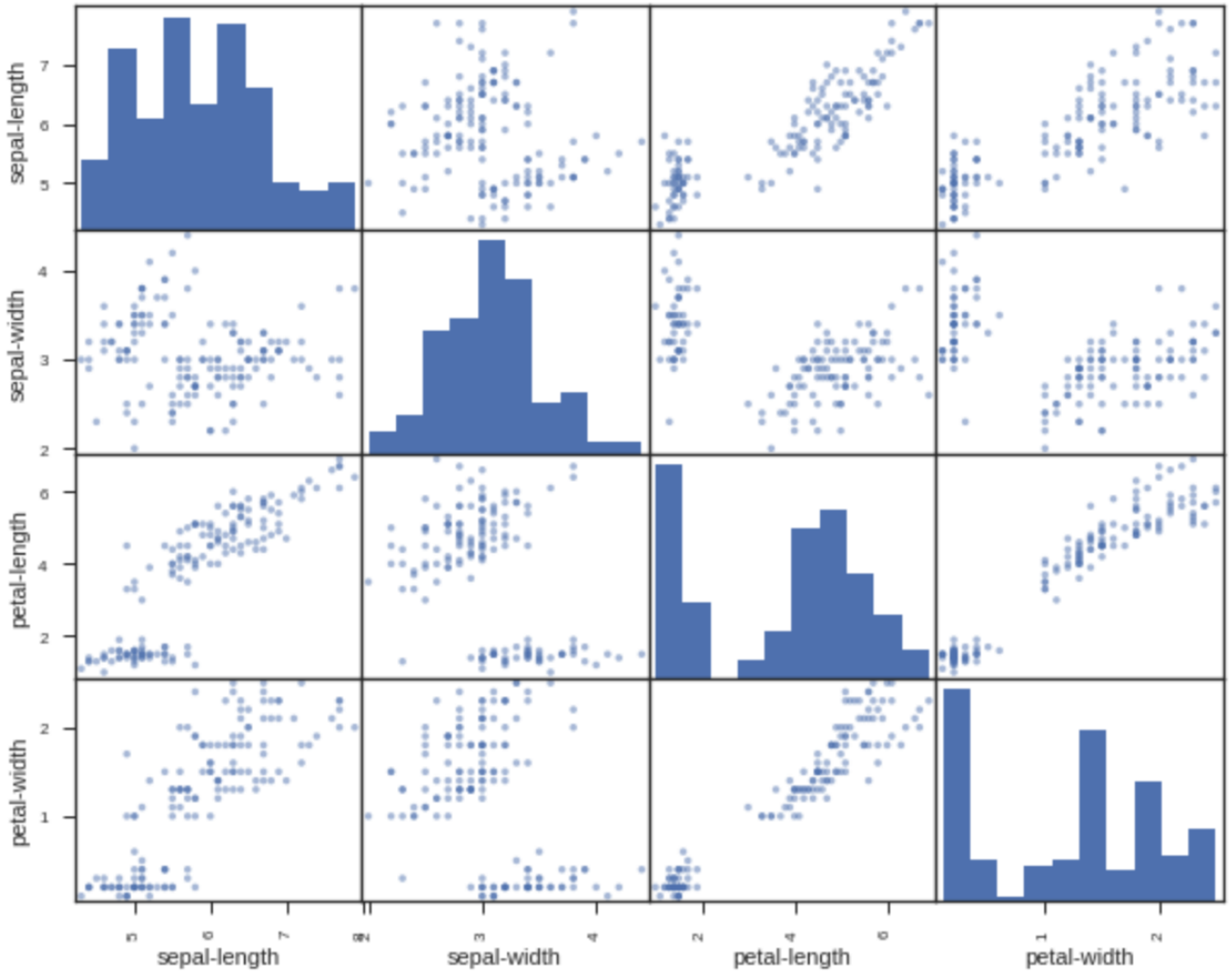
df.hist()
plt.show()

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



scatter_matrix(df)
plt.show()

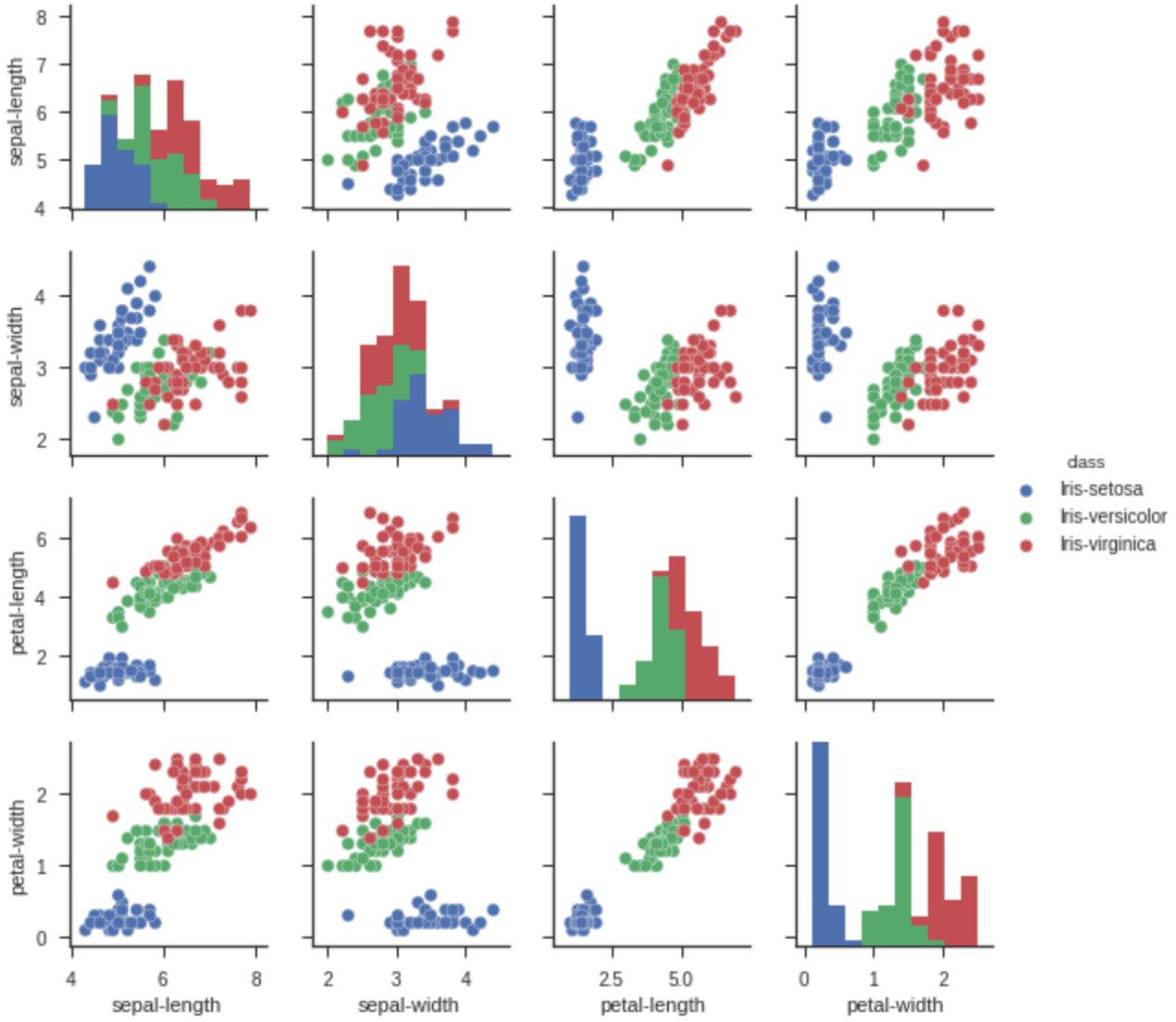
```
scatter_matrix(df)
plt.show()
```



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

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

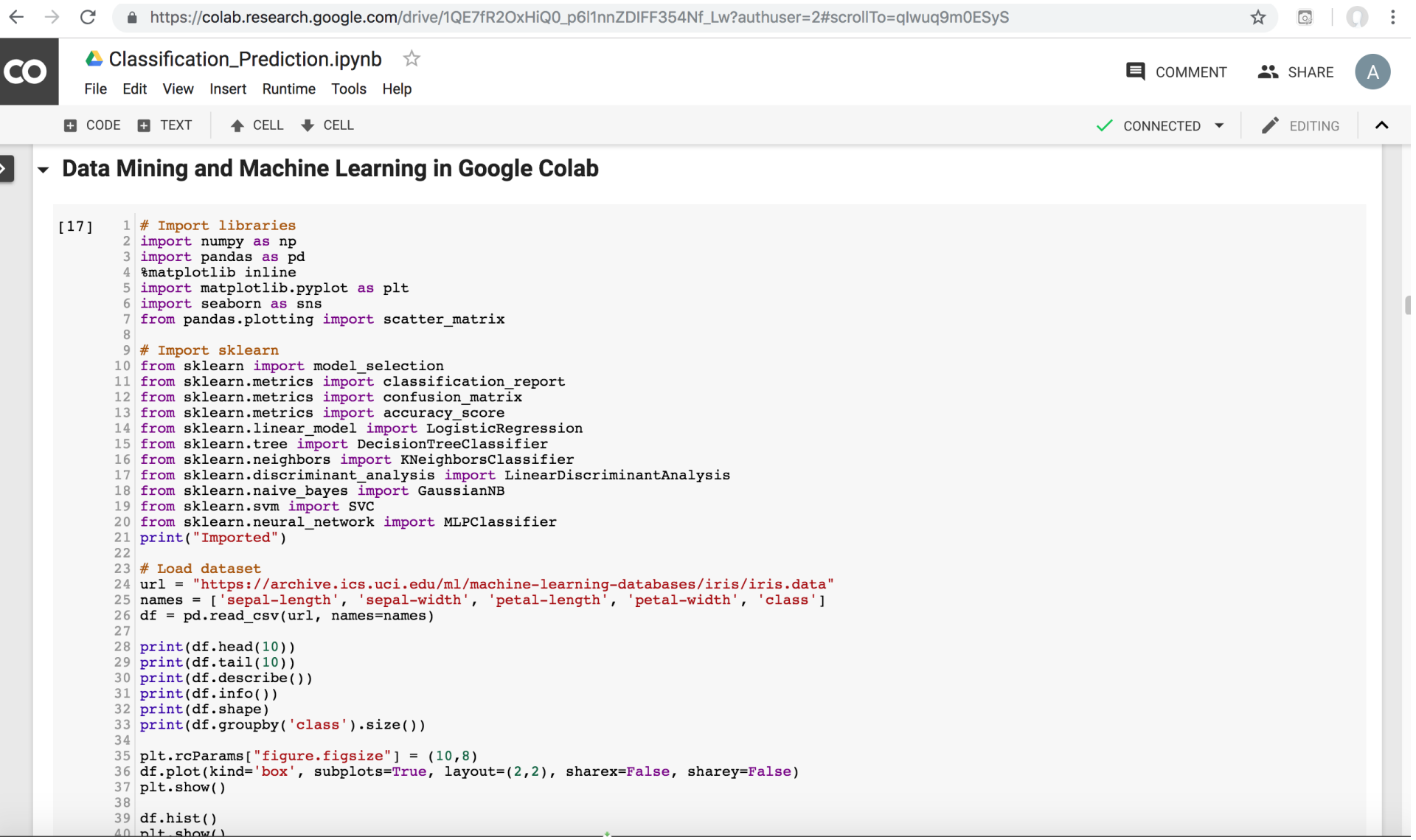
```
<seaborn.axisgrid.PairGrid at 0x7f1d21267390>
```



Machine Learning
Supervised Learning
Classification
and
Prediction

Classification and Prediction

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw



https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
# 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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

Machine Learning

Unsupervised Learning

Cluster Analysis

K-Means Clustering

K-Means Clustering

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
#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()
```

```
#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]
```

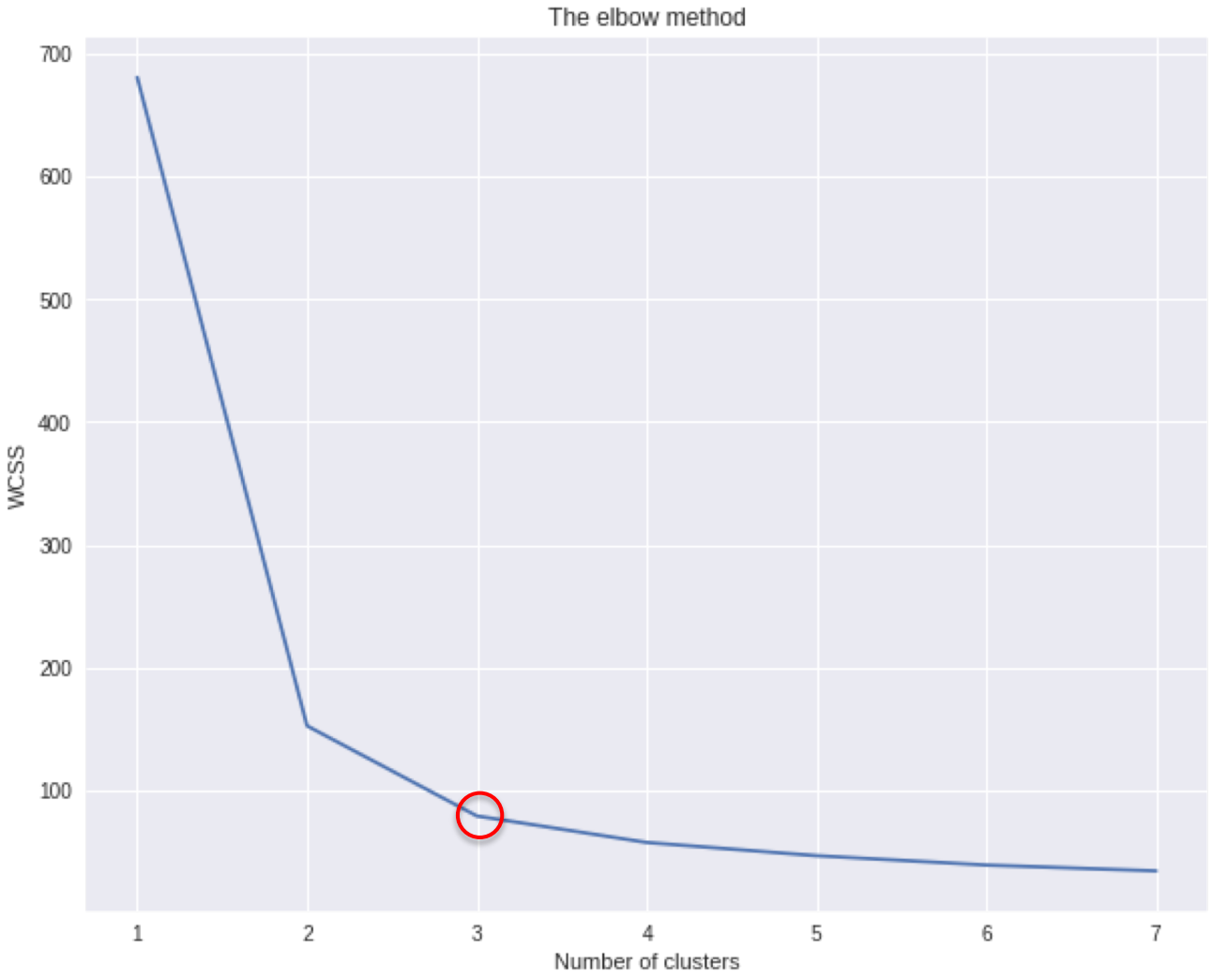
https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
#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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

K-Means Clustering

The elbow method (*k*=3)



https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
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)
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
#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()
```

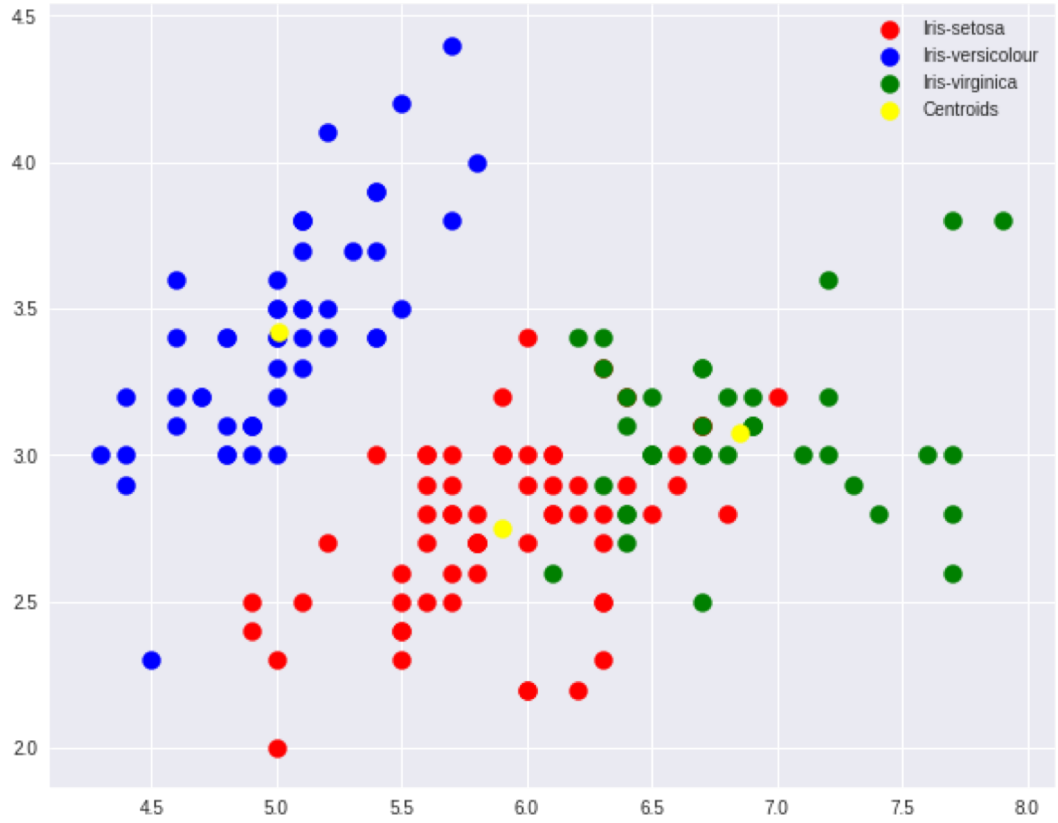
```
1 #Visualising the clusters
2 plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
3 plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
4 plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')
5
6 #Plotting the centroids of the clusters
7 plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')
8
9 plt.legend()
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

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://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

Python in Google Colab

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

python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECTED EDITING

```
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```


AAPL
Adj Close
MA05
MA20
MA60
220
200
180
160
140
120
100
80
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10
Date

Python Data Science Handbook in Google Colab

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

Index.ipynb

File Edit View Insert Runtime Tools Help

SHARE

CODE TEXT CELL CELL COPY TO DRIVE CONNECT EDITING

Python Data Science Handbook

Jake VanderPlas

powered by jupyter

Jake VanderPlas

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

Python Data Science Handbook in Google Colab

Table of Contents

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

Python Data Science Handbook in Google Colab

2. Introduction to NumPy

- Understanding Data Types in Python
- The Basics of NumPy Arrays
- Computation on NumPy Arrays: Universal Functions
- Aggregations: Min, Max, and Everything In Between
- Computation on Arrays: Broadcasting
- Comparisons, Masks, and Boolean Logic
- Fancy Indexing
- Sorting Arrays
- Structured Data: NumPy's Structured Arrays

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

Python Data Science Handbook in Google Colab

3. Data Manipulation with Pandas

- Introducing Pandas Objects
- Data Indexing and Selection
- Operating on Data in Pandas
- Handling Missing Data
- Hierarchical Indexing
- Combining Datasets: Concat and Append
- Combining Datasets: Merge and Join
- Aggregation and Grouping
- Pivot Tables
- Vectorized String Operations
- Working with Time Series
- High-Performance Pandas: eval() and query()
- Further Resources

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

Python Data Science Handbook in Google Colab

4. Visualization with Matplotlib

- Simple Line Plots
- Simple Scatter Plots
- Visualizing Errors
- Density and Contour Plots
- Histograms, Binnings, and Density
- Customizing Plot Legends
- Customizing Colorbars
- Multiple Subplots
- Text and Annotation
- Customizing Ticks
- Customizing Matplotlib: Configurations and Stylesheets
- Three-Dimensional Plotting in Matplotlib
- Geographic Data with Basemap
- Visualization with Seaborn
- Further Resources

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

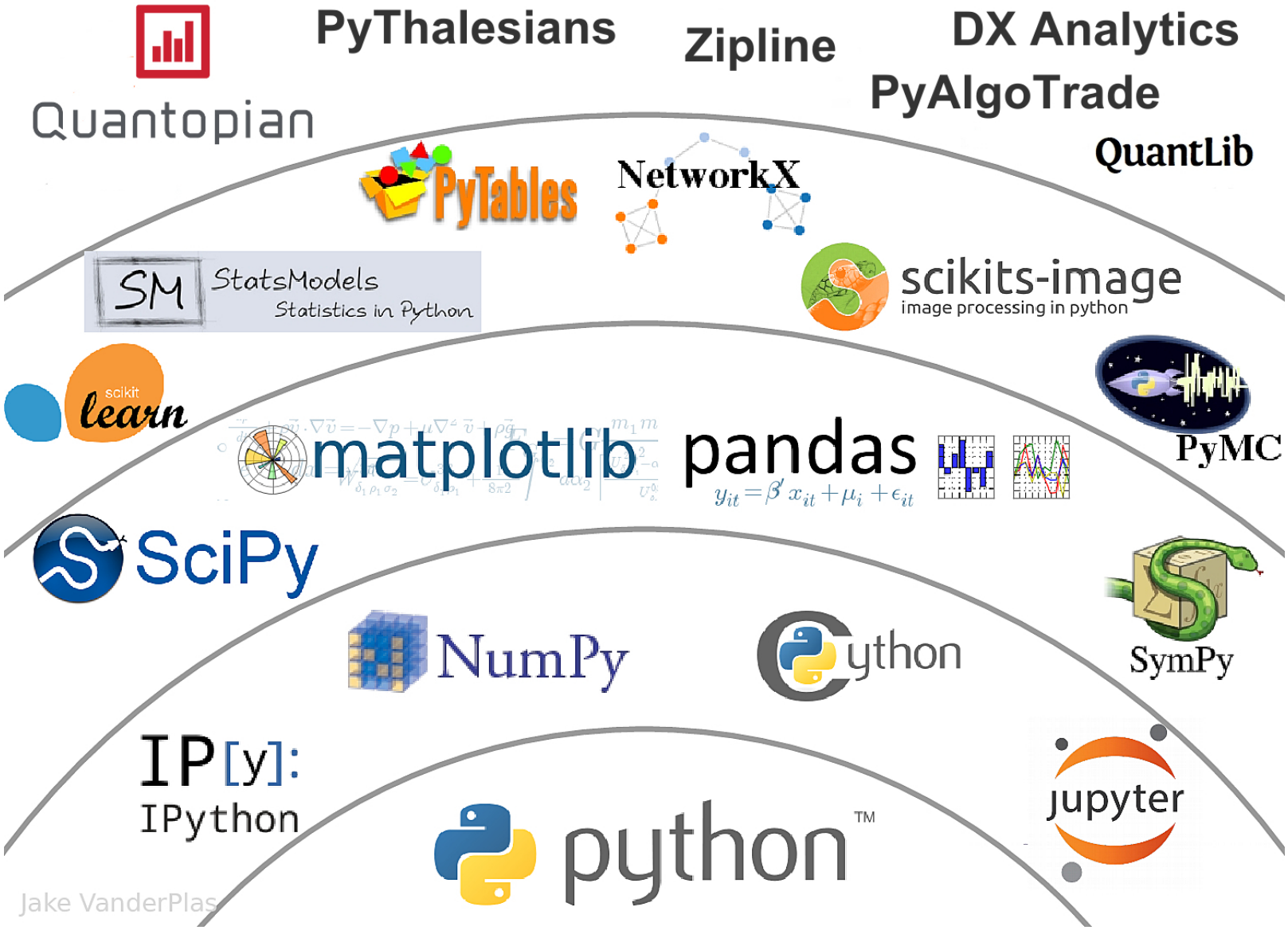
Python Data Science Handbook in Google Colab

5. Machine Learning

- What Is Machine Learning?
- Introducing Scikit-Learn
- Hyperparameters and Model Validation
- Feature Engineering
- In Depth: Naive Bayes Classification
- In Depth: Linear Regression
- In-Depth: Support Vector Machines
- In-Depth: Decision Trees and Random Forests
- In Depth: Principal Component Analysis
- In-Depth: Manifold Learning
- In Depth: k-Means Clustering
- In Depth: Gaussian Mixture Models
- In-Depth: Kernel Density Estimation
- Application: A Face Detection Pipeline
- Further Machine Learning Resources

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Summary

- Machine Learning with Scikit-Learn in Python
  - Machine Learning
  - Scikit-Learn

References

- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.
  https://github.com/wesm/pydata-book
- Avinash Jain (2017), Introduction To Python Programming, Udemy,
  https://www.udemy.com/pythonforbeginnersintro/
- Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
- Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
- Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
- James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
- Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
- Skikit-learn, http://scikit-learn.org/
- Data School (2015), Machine learning in Python with scikit-learn,
  https://www.youtube.com/playlist?list=PL5-da3qGB5ICeMbQuqbbCOQWcS6OYBr5A
- Jason Brownlee (2016), Your First Machine Learning Project in Python Step-By-Step,
  https://machinelearningmastery.com/machine-learning-in-python-step-by-step/
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.