Tamkang University
淡江大學

Big Data Mining
巨量資料探勘

Tamkang University

機器學習與深度學習
(Machine Learning and Deep Learning)

1082DM10
MI4 (M2244) (2744)
Tue 3, 4 (10:10-12:00) (B218)

Min-Yuh Day
戴敏育
Associate Professor
副教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

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

2020-05-19

課程大綱 (Syllabus)

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

1　2020/03/03　巨量資料探勘課程介紹
(Course Orientation for Big Data Mining)

2　2020/03/10　AI人工智慧與大數據分析
(Artificial Intelligence and Big Data Analytics)

3　2020/03/17　分群分析 (Cluster Analysis)

4　2020/03/24　個案分析與實作一 (SAS EM 分群分析)：
Case Study 1 (Cluster Analysis - K-Means using SAS EM)

5　2020/03/31　關連分析 (Association Analysis)

6　2020/04/07　個案分析與實作二 (SAS EM 關連分析)：
Case Study 2 (Association Analysis using SAS EM)

7　2020/04/14　分類與預測 (Classification and Prediction)

8　2020/04/21　期中報告 (Midterm Project Presentation)

課程大綱 (Syllabus)

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

9 2020/04/28 期中考試週

10 2020/05/05 個案分析與實作三 (SAS EM 決策樹、模型評估)：
Case Study 3 (Decision Tree, Model Evaluation using SAS EM)

11 2020/05/12 個案分析與實作四 (SAS EM 迴歸分析、類神經網路)：
Case Study 4 (Regression Analysis,
Artificial Neural Network using SAS EM)

12 2020/05/19 機器學習與深度學習
(Machine Learning and Deep Learning)

13 2020/05/26 期末報告 (Final Project Presentation)

14 2020/06/02 畢業考試週

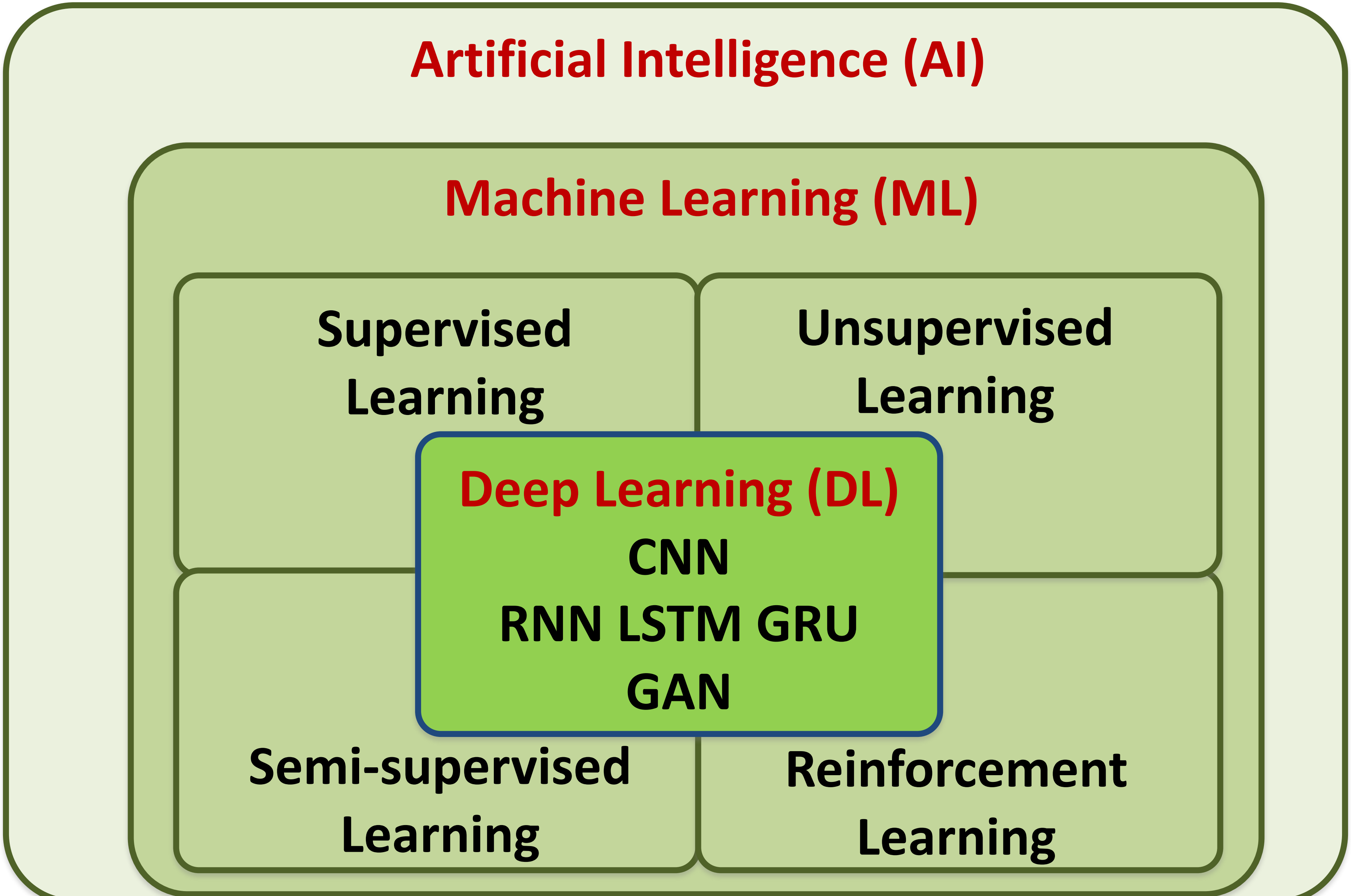
15 2020/06/09 教師彈性補充教學

Machine Learning
and
Deep Learning

Outline

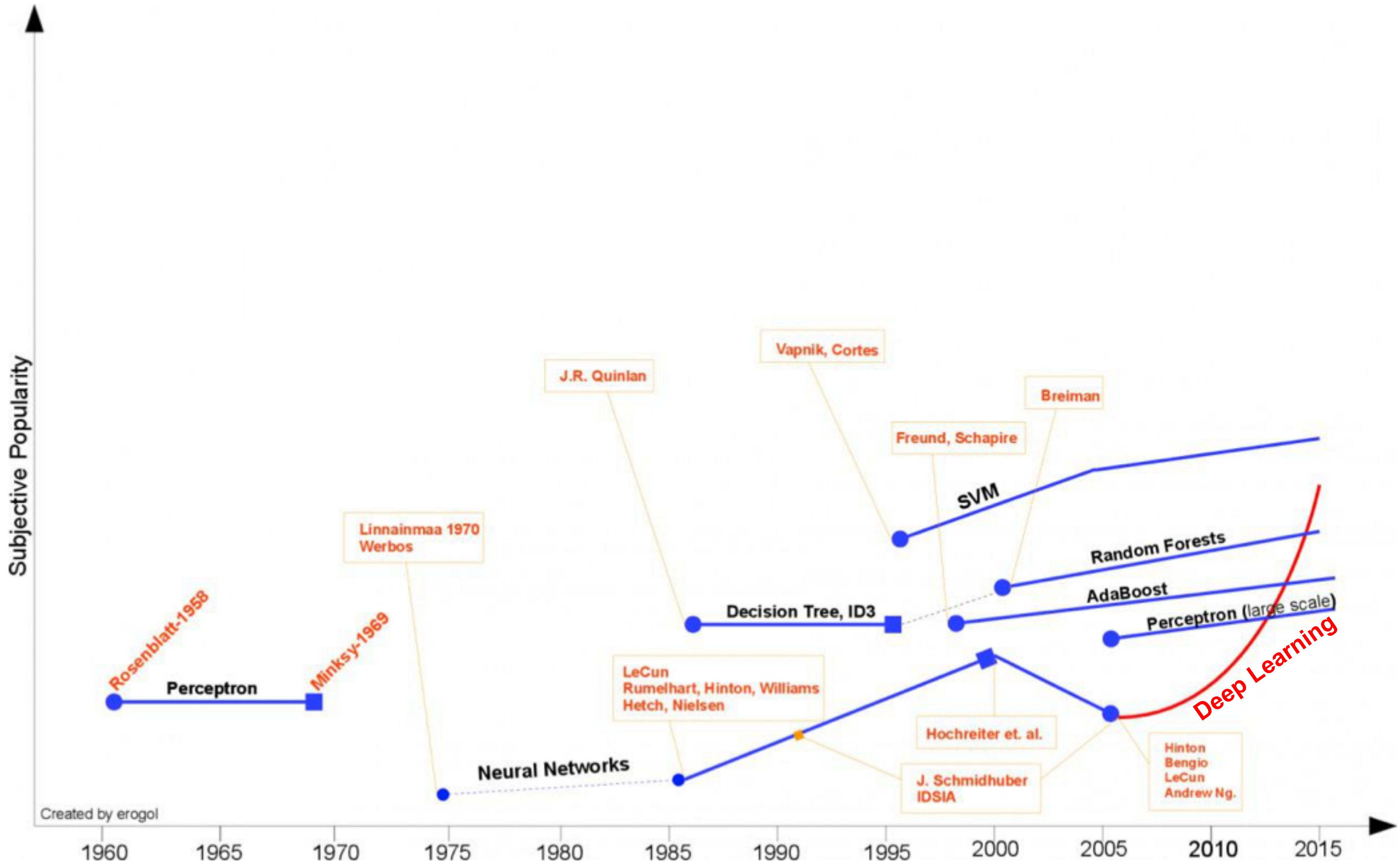
- Machine Learning
- Deep Learning

AI, ML, DL



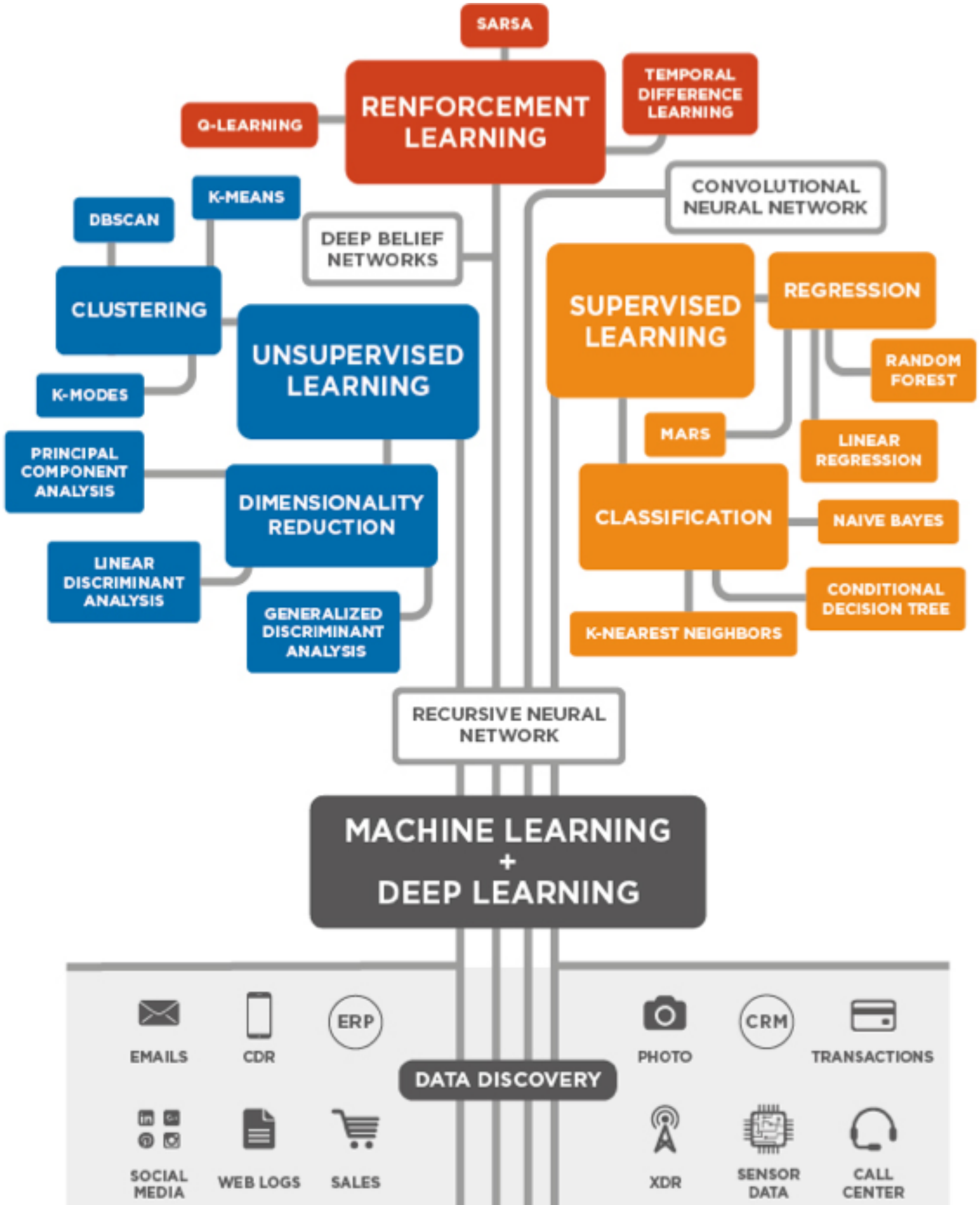
Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

Deep Learning Evolution



Source: http://www.erogol.com/brief-history-machine-learning/

3 Machine Learning Algorithms



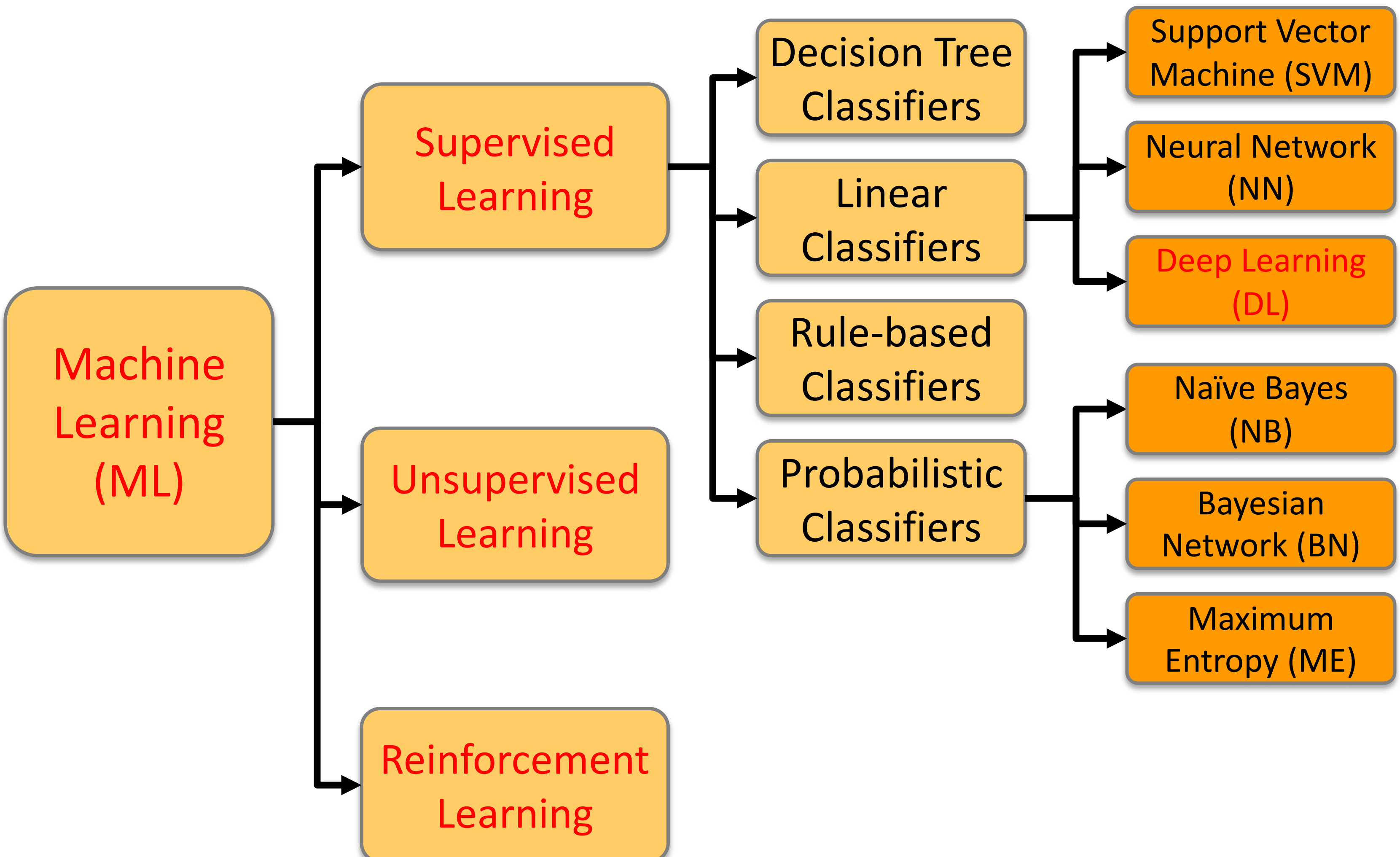
Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

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

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.

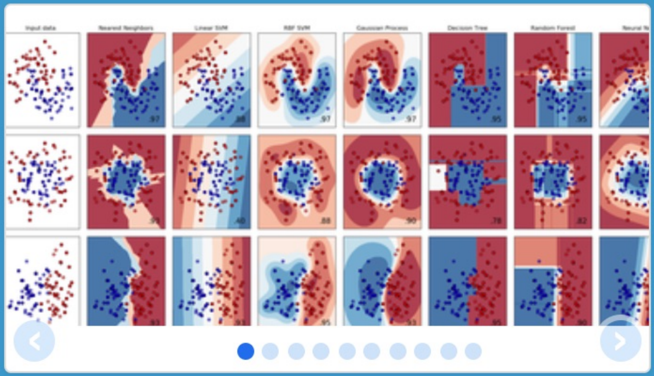
Scikit-Learn

Machine Learning in Python

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

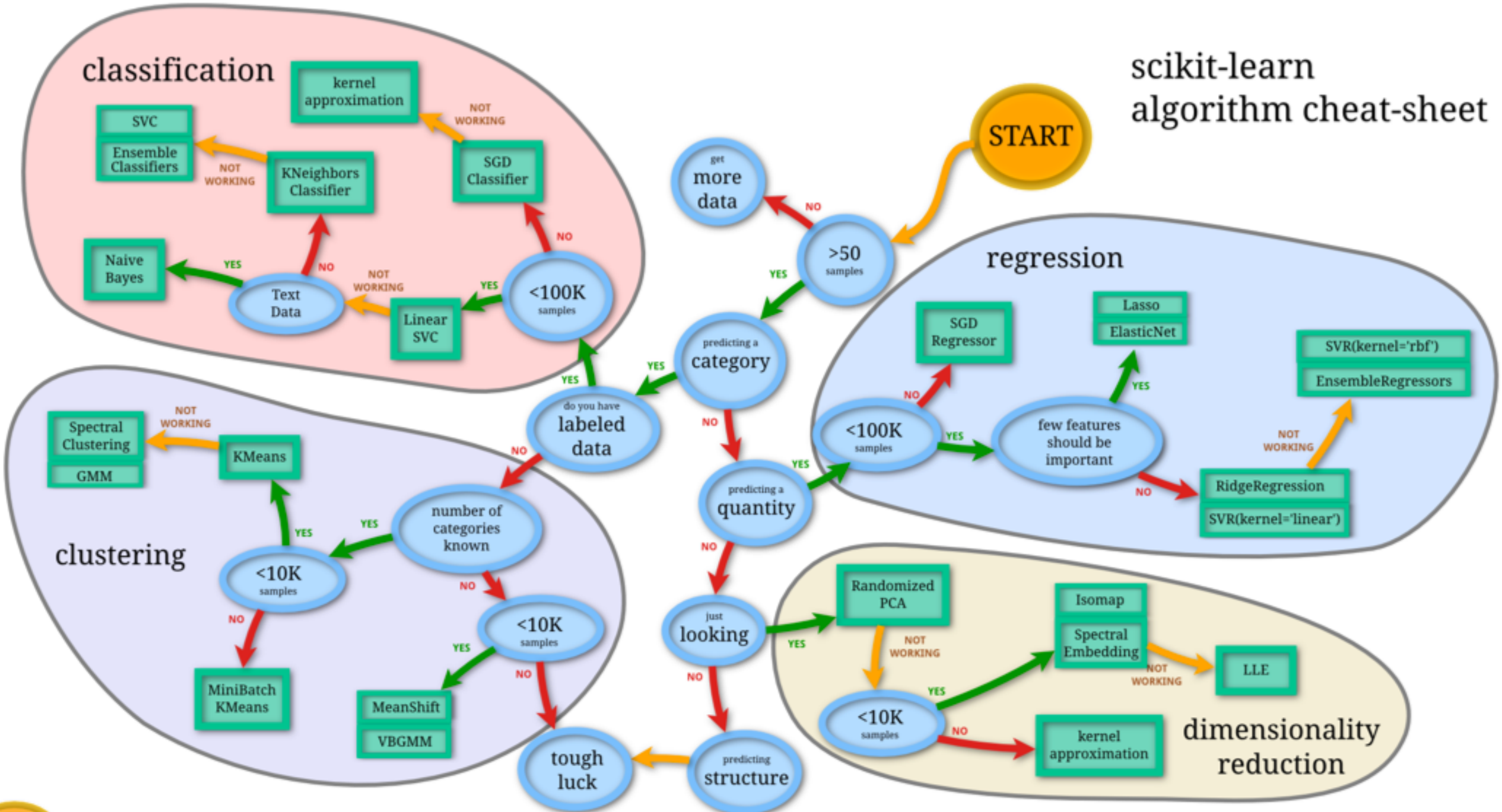
Preprocessing

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

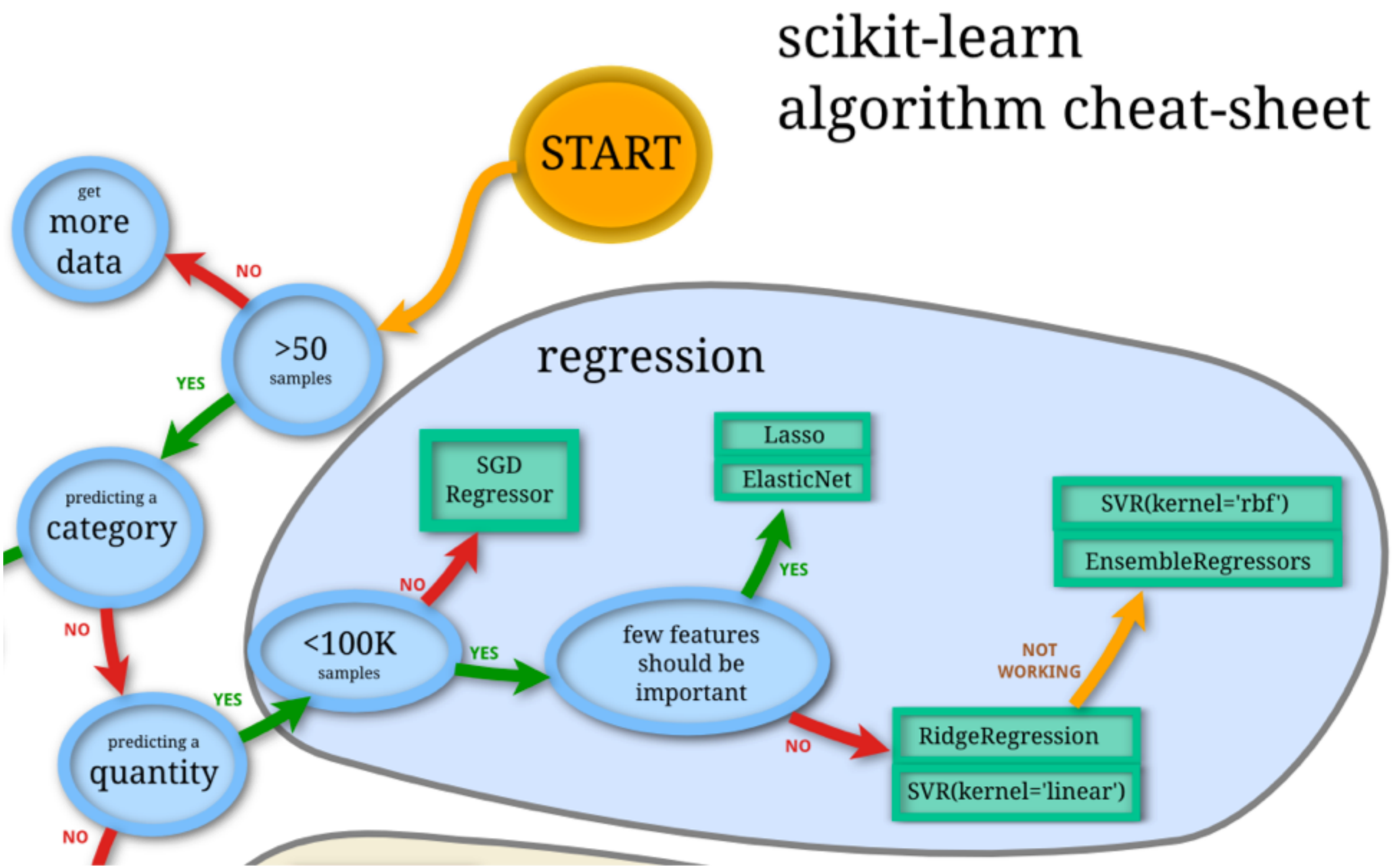


Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map

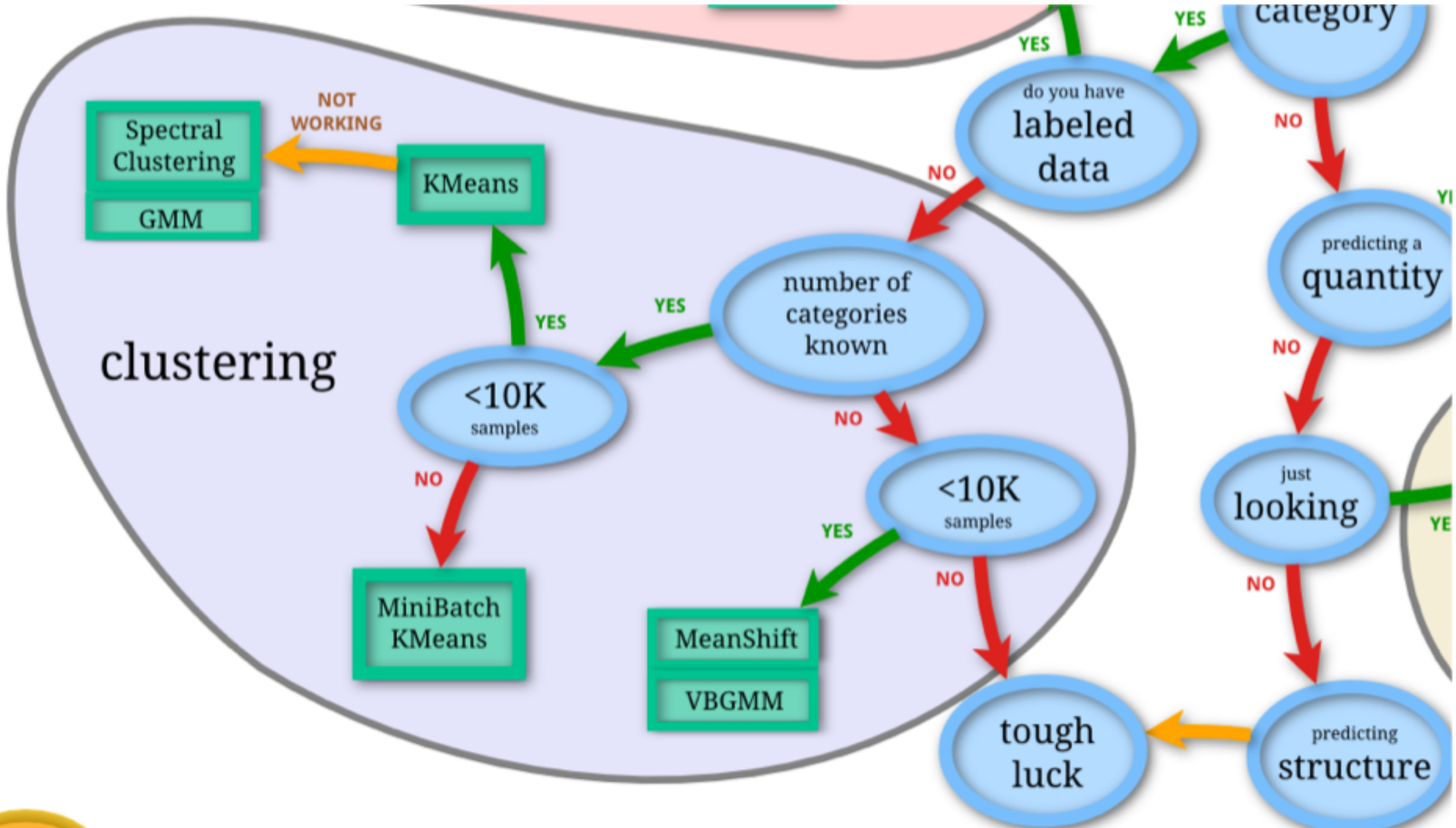
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



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

Scikit-Learn Machine Learning Map



Back

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

Iris flower data set

setosa

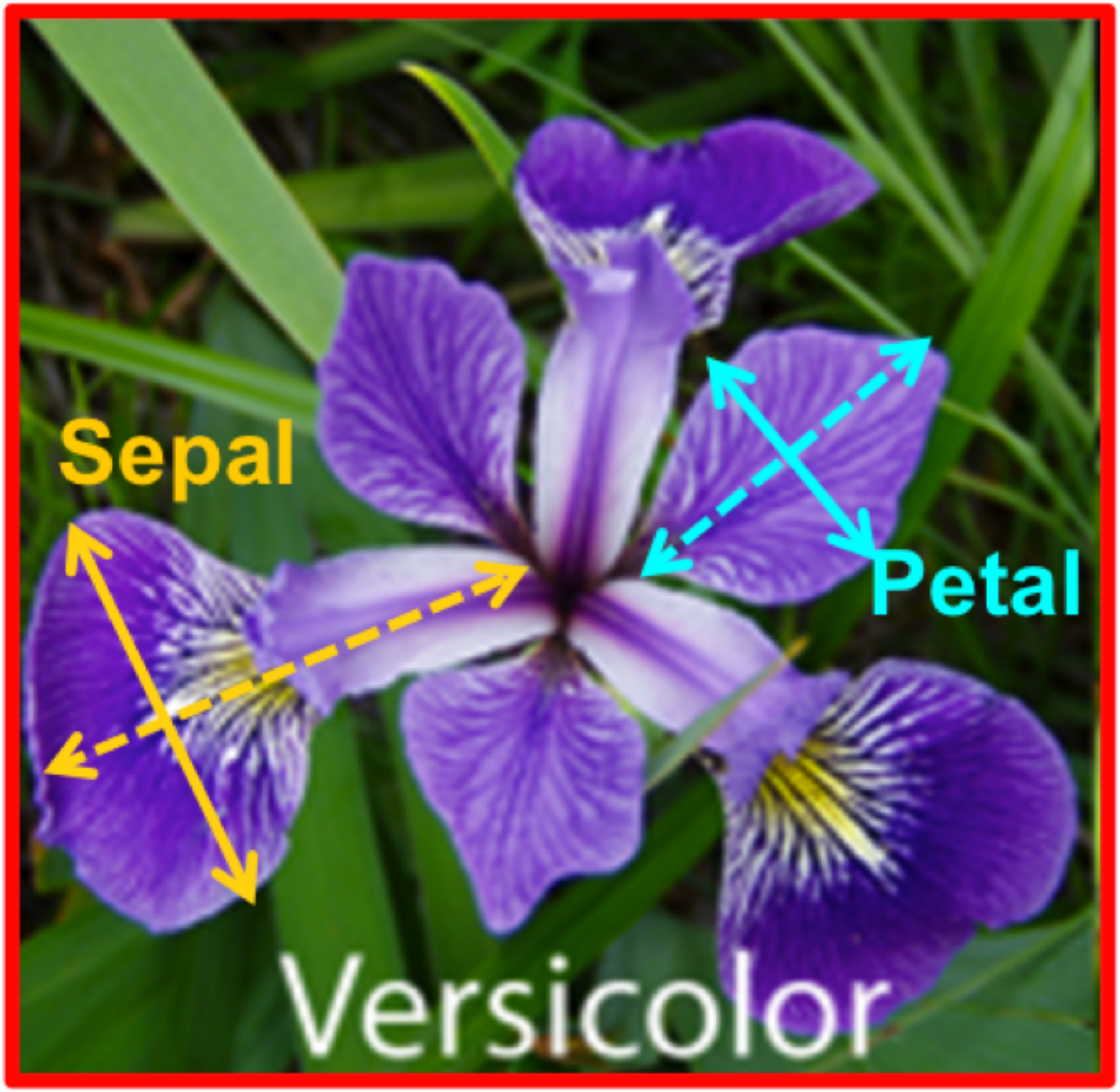
versicolor

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

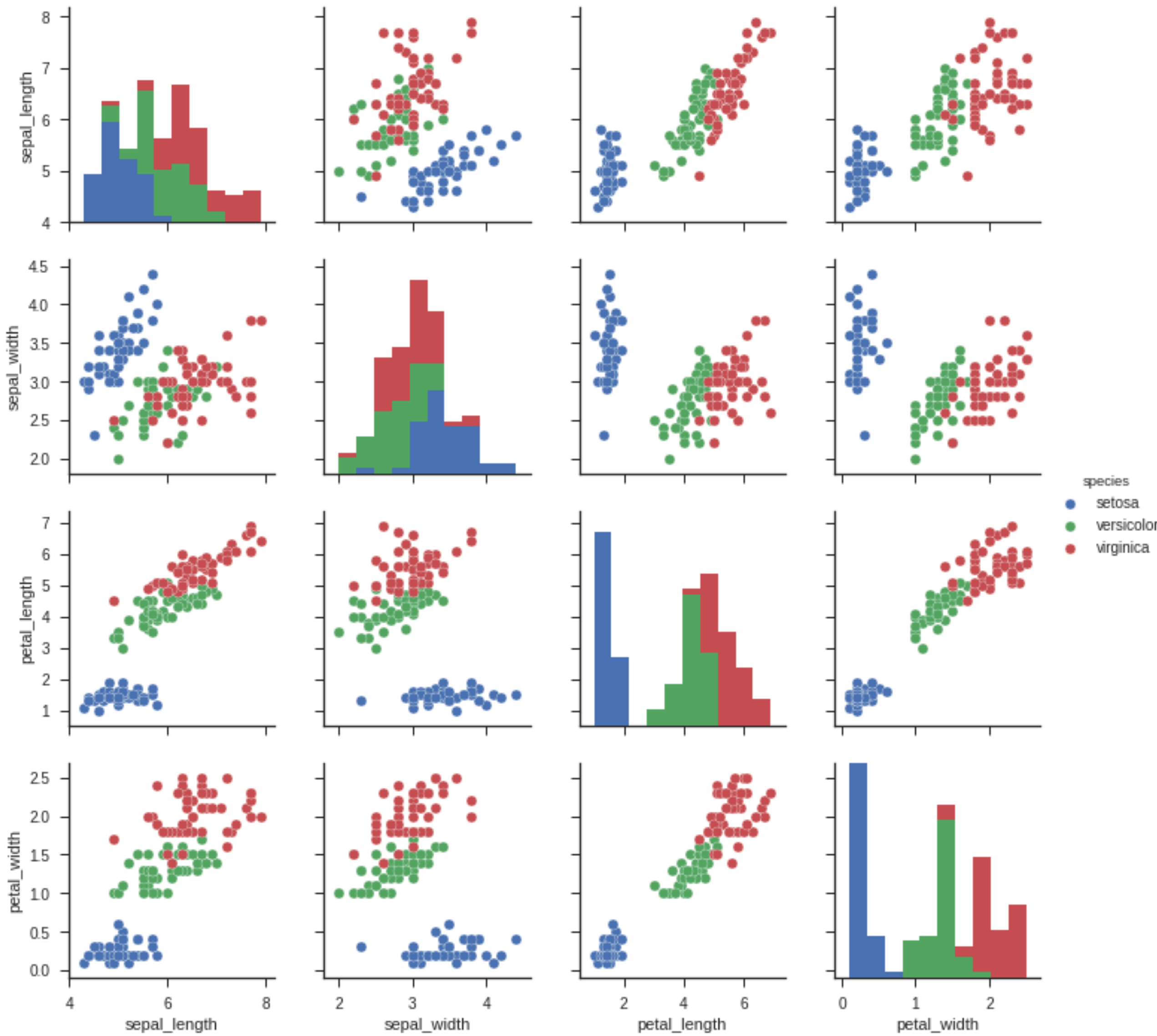
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa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

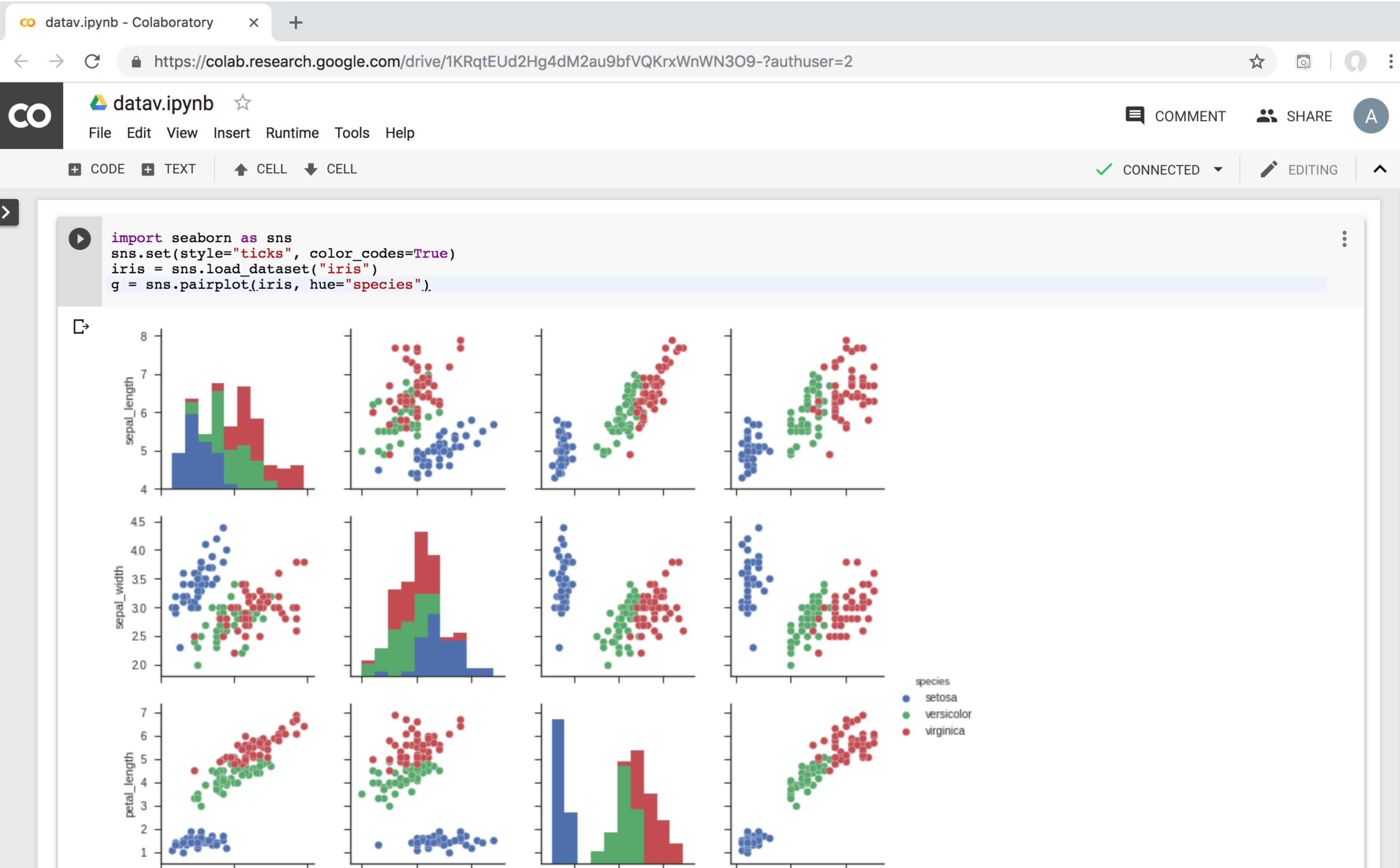
versicolor

Iris Data Visualization



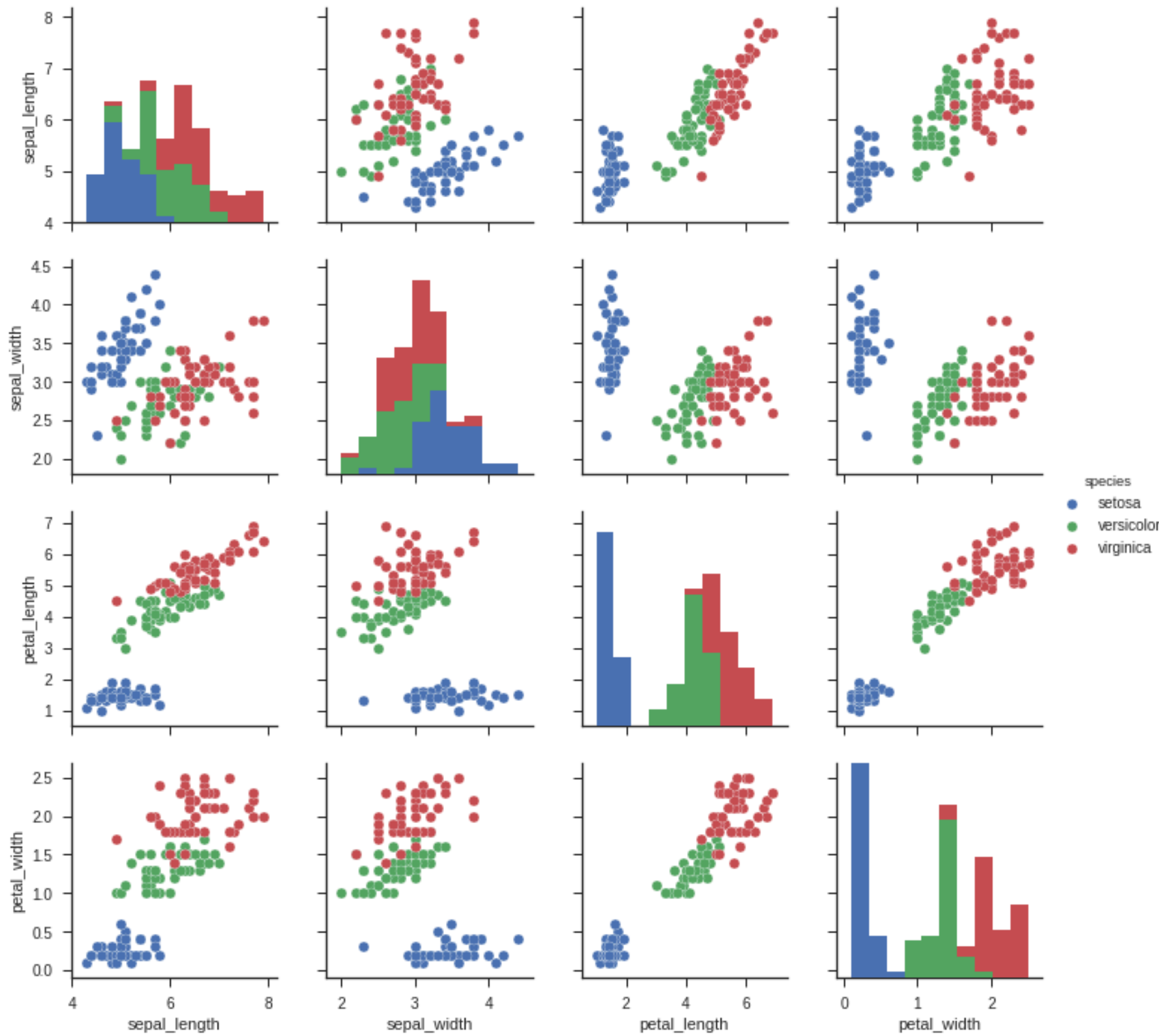
Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

Data Visualization in Google Colab



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

```python
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
```

```python
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))
```

```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))
```

	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()

```python
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)
```

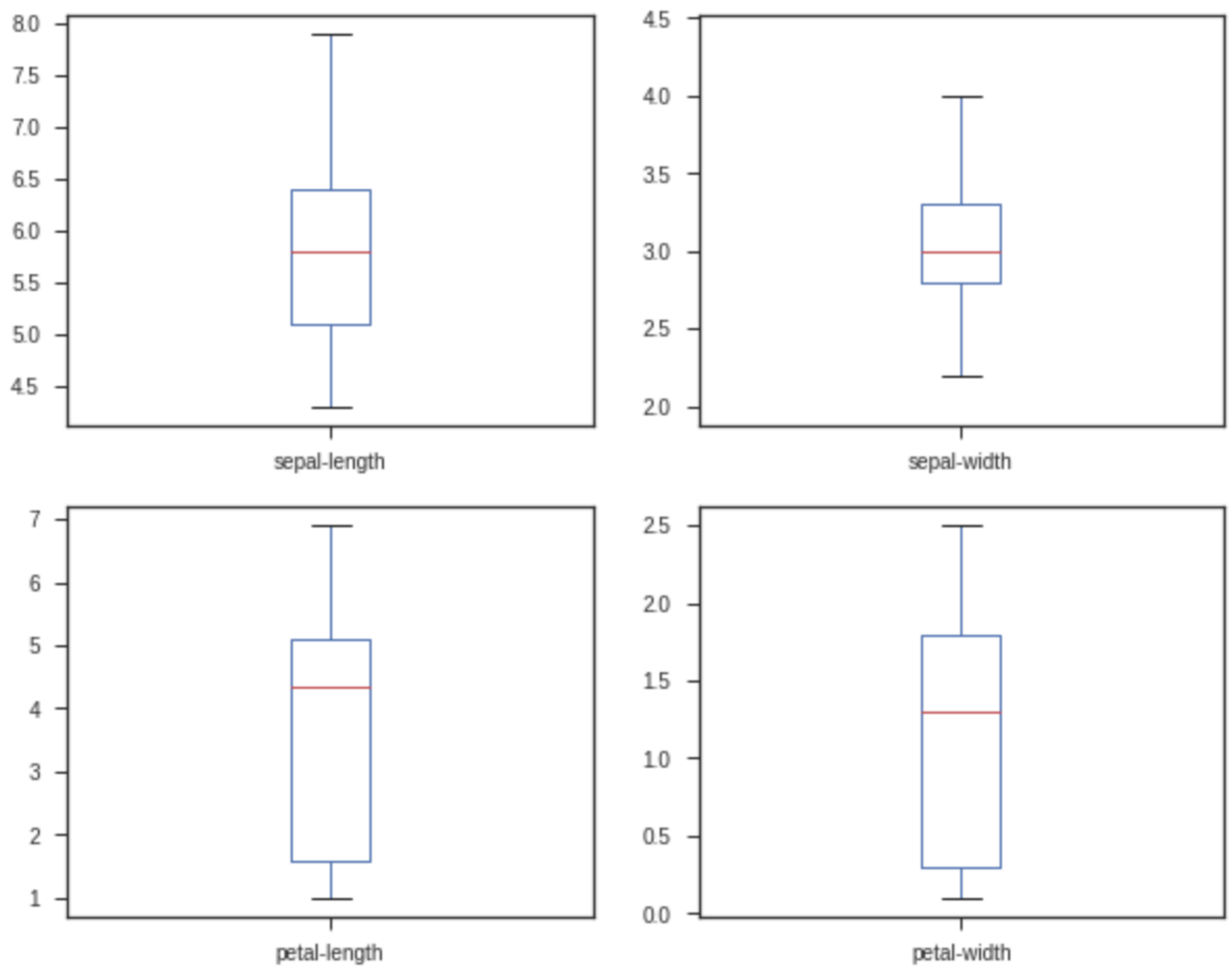
df.groupby('class').size()

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

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

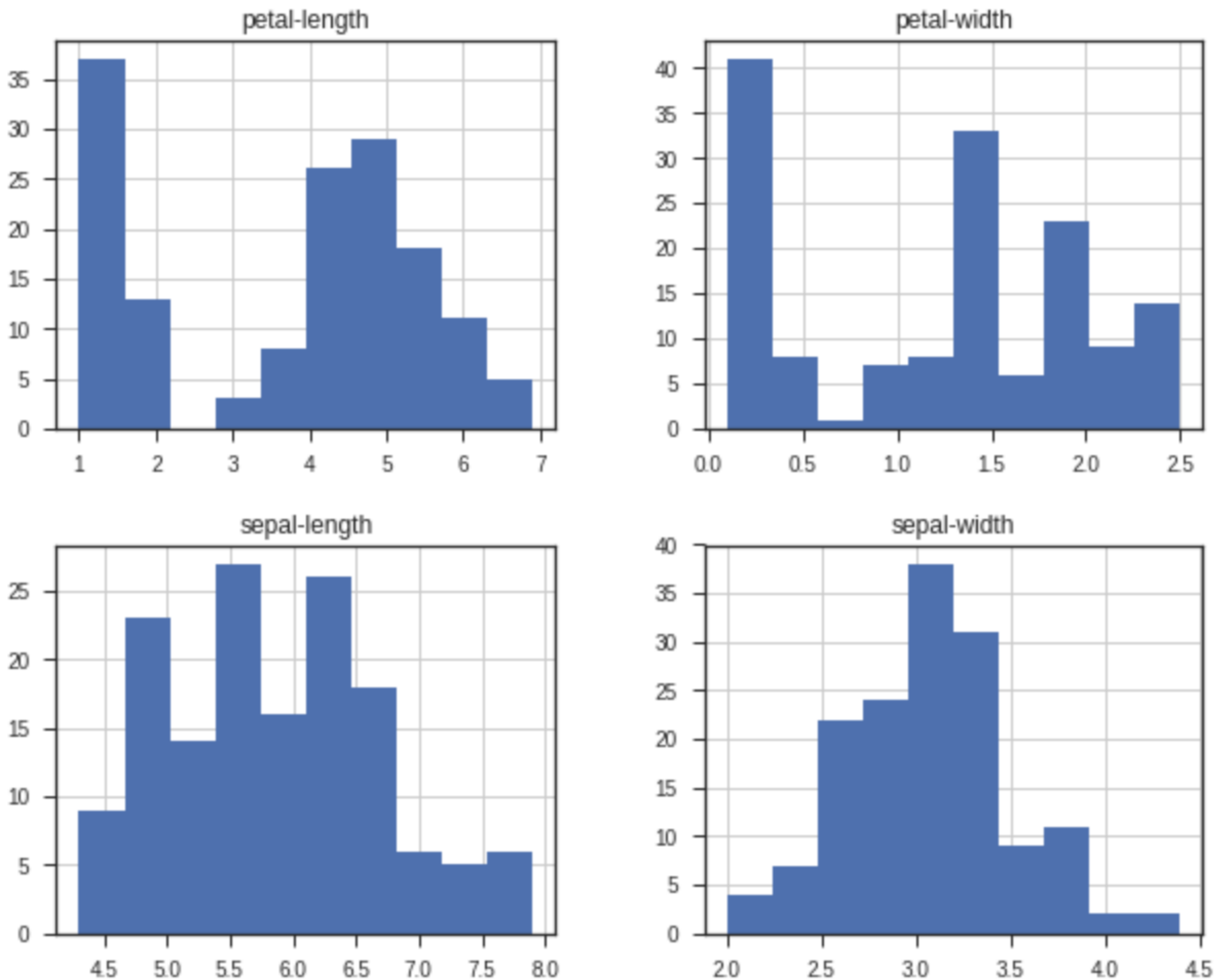
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```


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

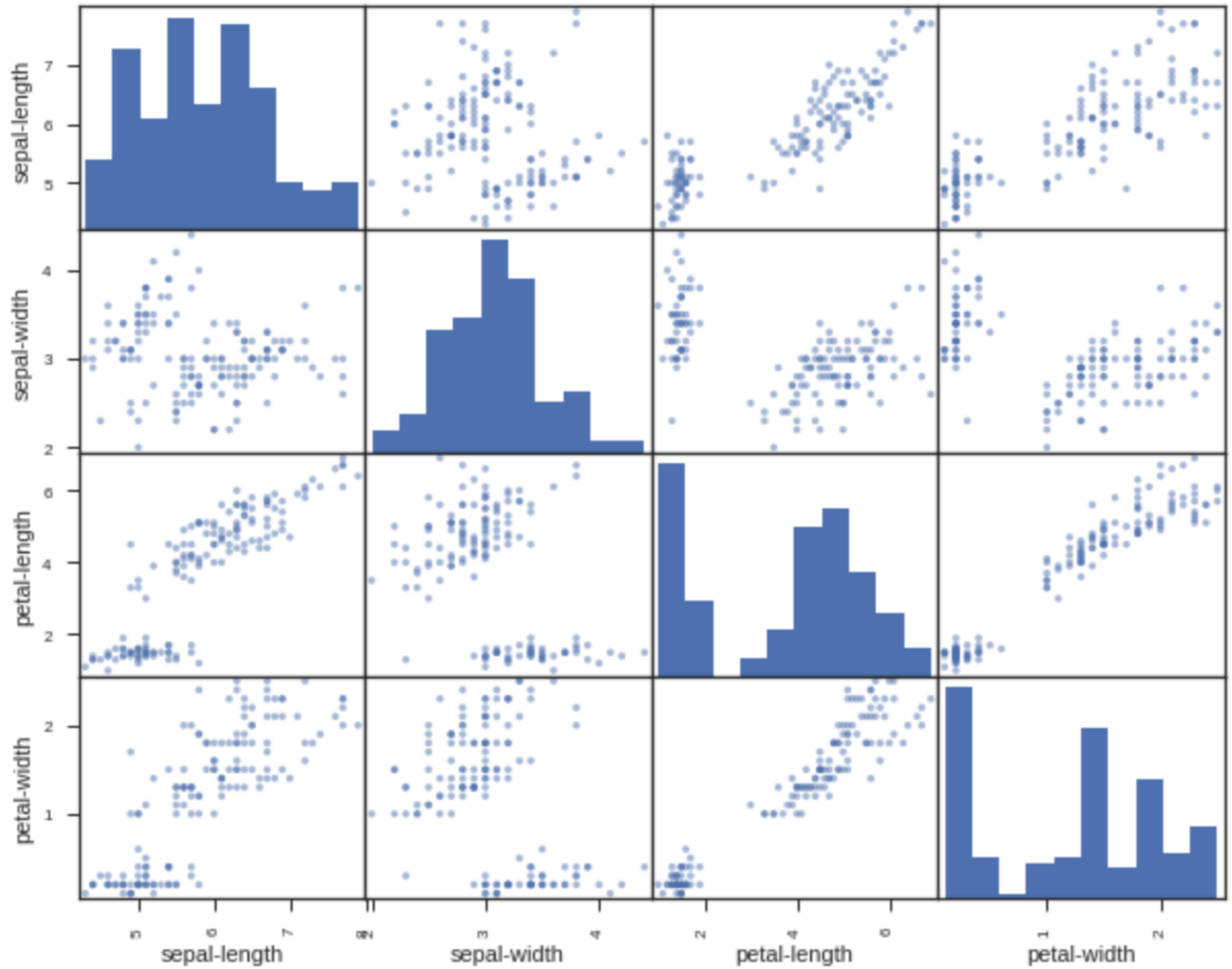
```
df.hist()
plt.show()
```

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



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

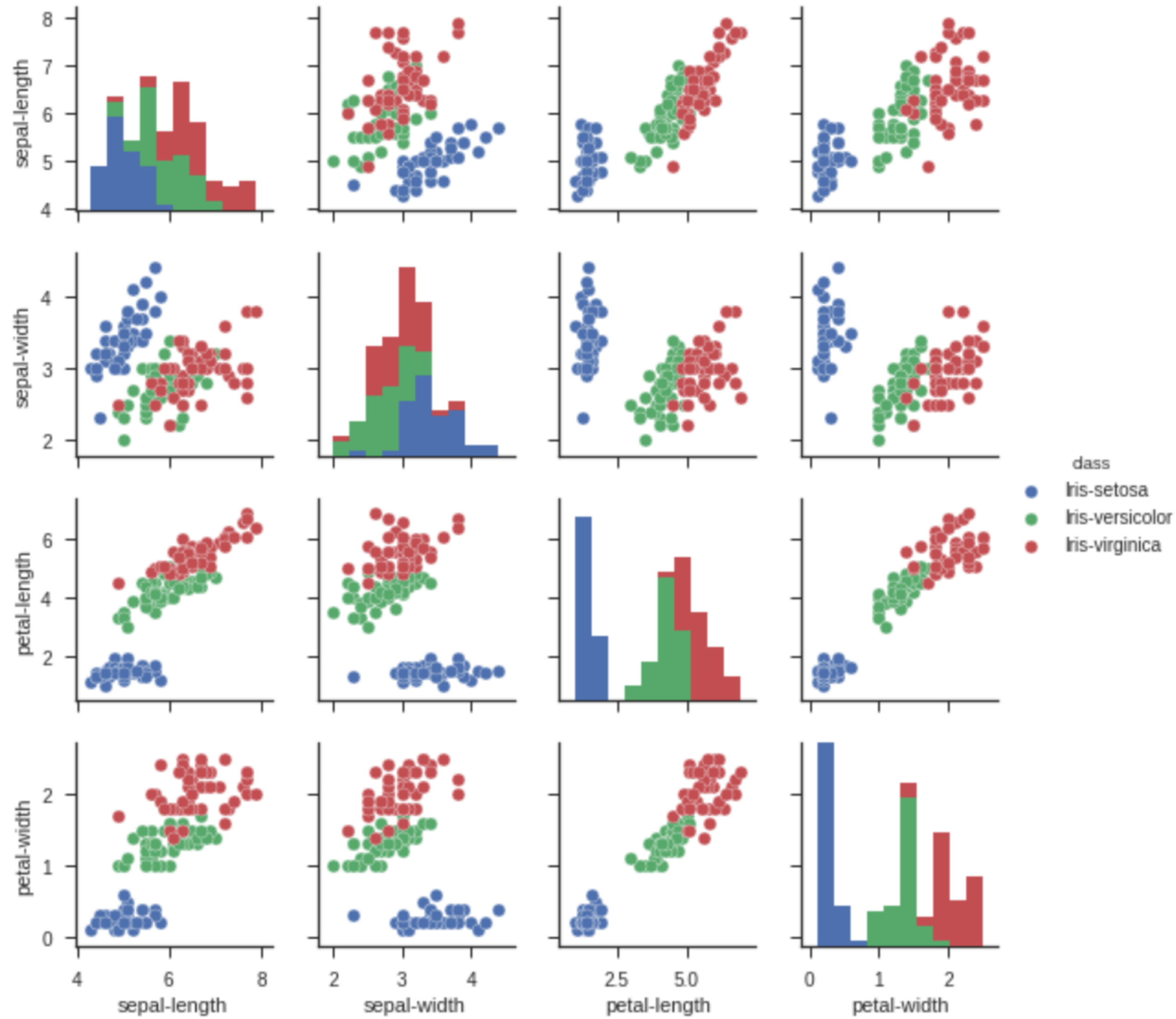

```
scatter_matrix(df)
plt.show()
```

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_p6I1nnZDIFF354Nf_Lw

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw?authuser=2#scrollTo=qlwuq9m0ESyS

Classification_Prediction.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

Data Mining and Machine Learning in Google Colab

[17]

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

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

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

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6I1nnZDIFF354Nf_Lw

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

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

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

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

df.corr()

```
df.corr()
```

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

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

```python
# 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'
```

```python
# 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'
```

```python
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

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

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

https://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_p6I1nnZDIFF354Nf_Lw

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

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

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

    avg / total       0.90      0.90      0.90        30

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

https://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)
```

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

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

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

    avg / total       0.94      0.93      0.93        30

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

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

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

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

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

    avg / total       0.90      0.90      0.90        30

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

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

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

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

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

    avg / total       0.84      0.83      0.83        30

GaussianNB(priors=None)
```

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

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

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

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

    avg / total       0.83      0.80      0.80        30

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

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

```python
# 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()
```

```python
#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]
```

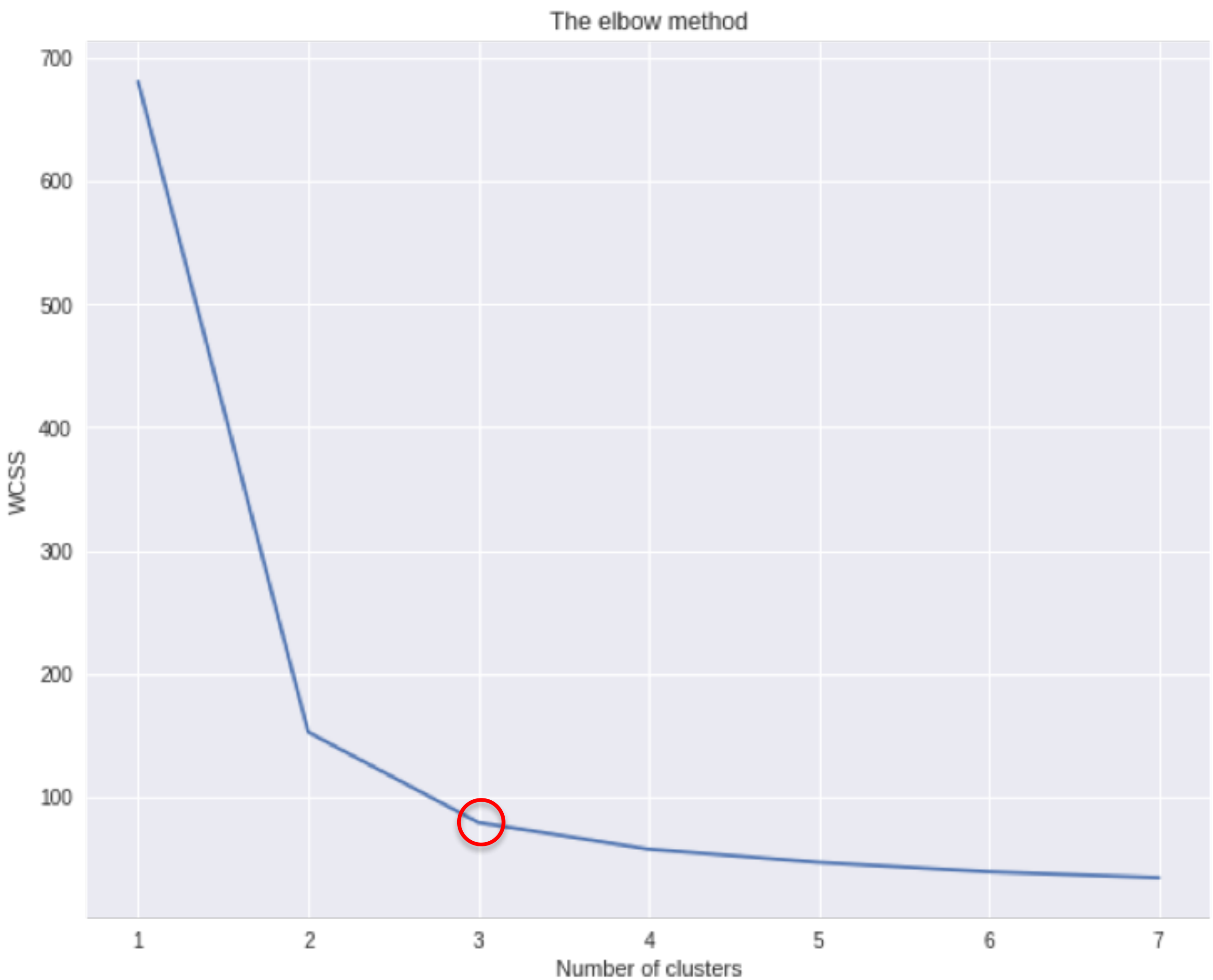
https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

```
#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_p6I1nnZDIFF354Nf_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()
```

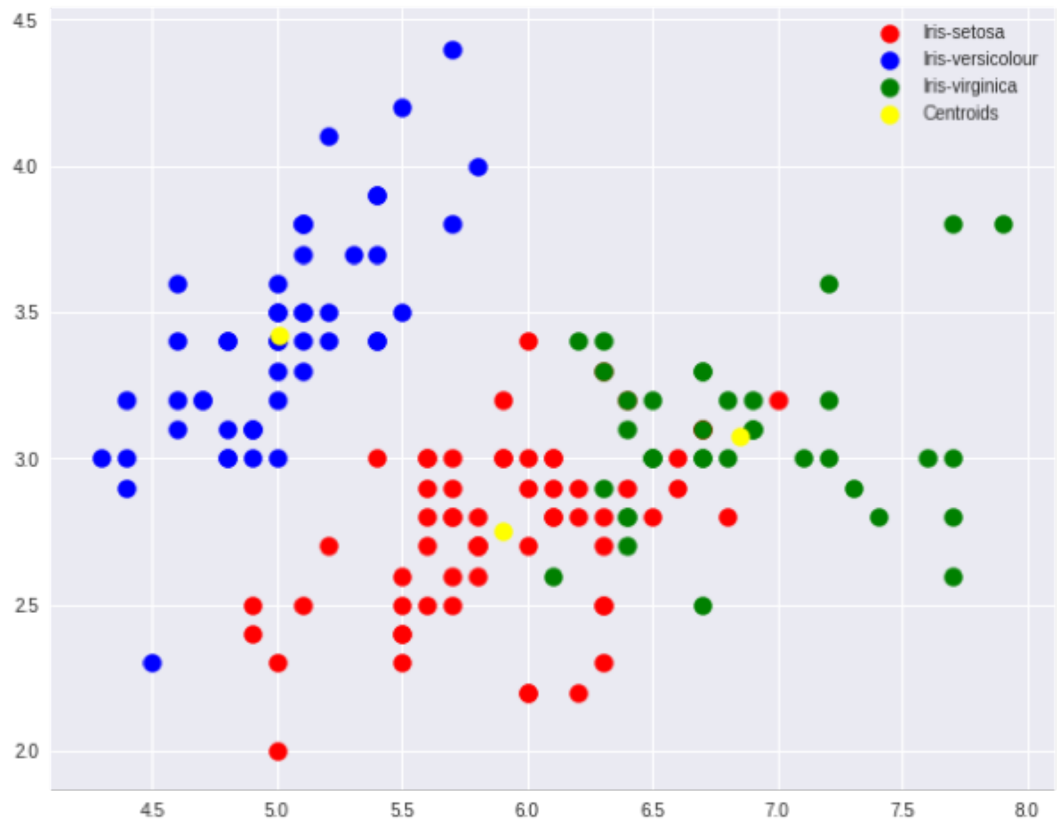

```
#Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green',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

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()
```

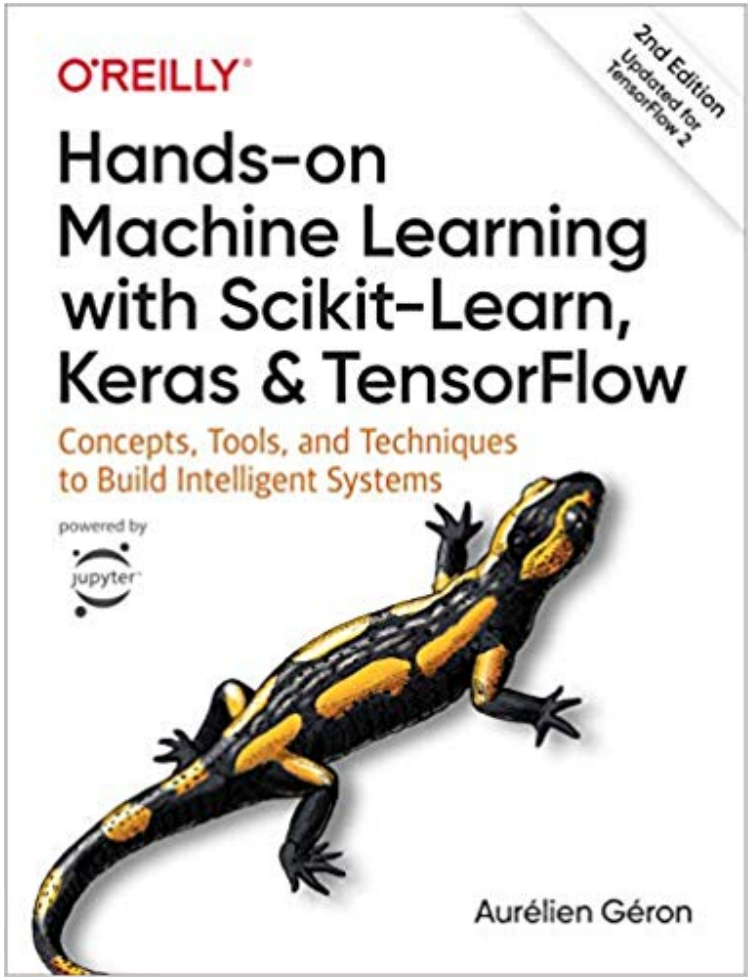


https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

Aurélien Géron (2019),

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition

O'Reilly Media, 2019



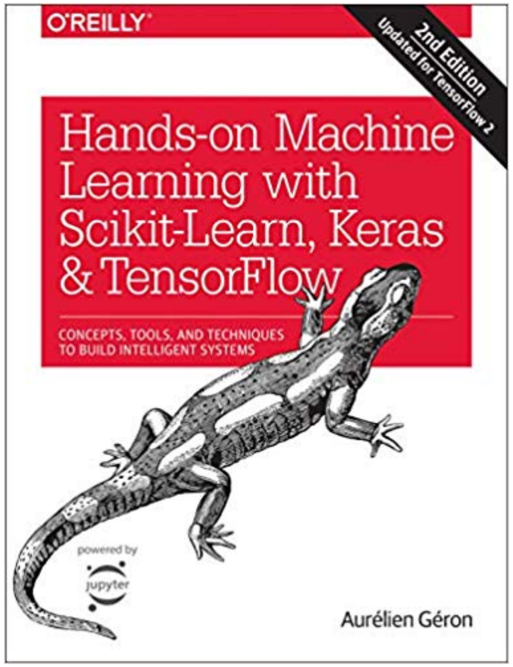
https://github.com/ageron/handson-ml2

Source: https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1492032646/

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

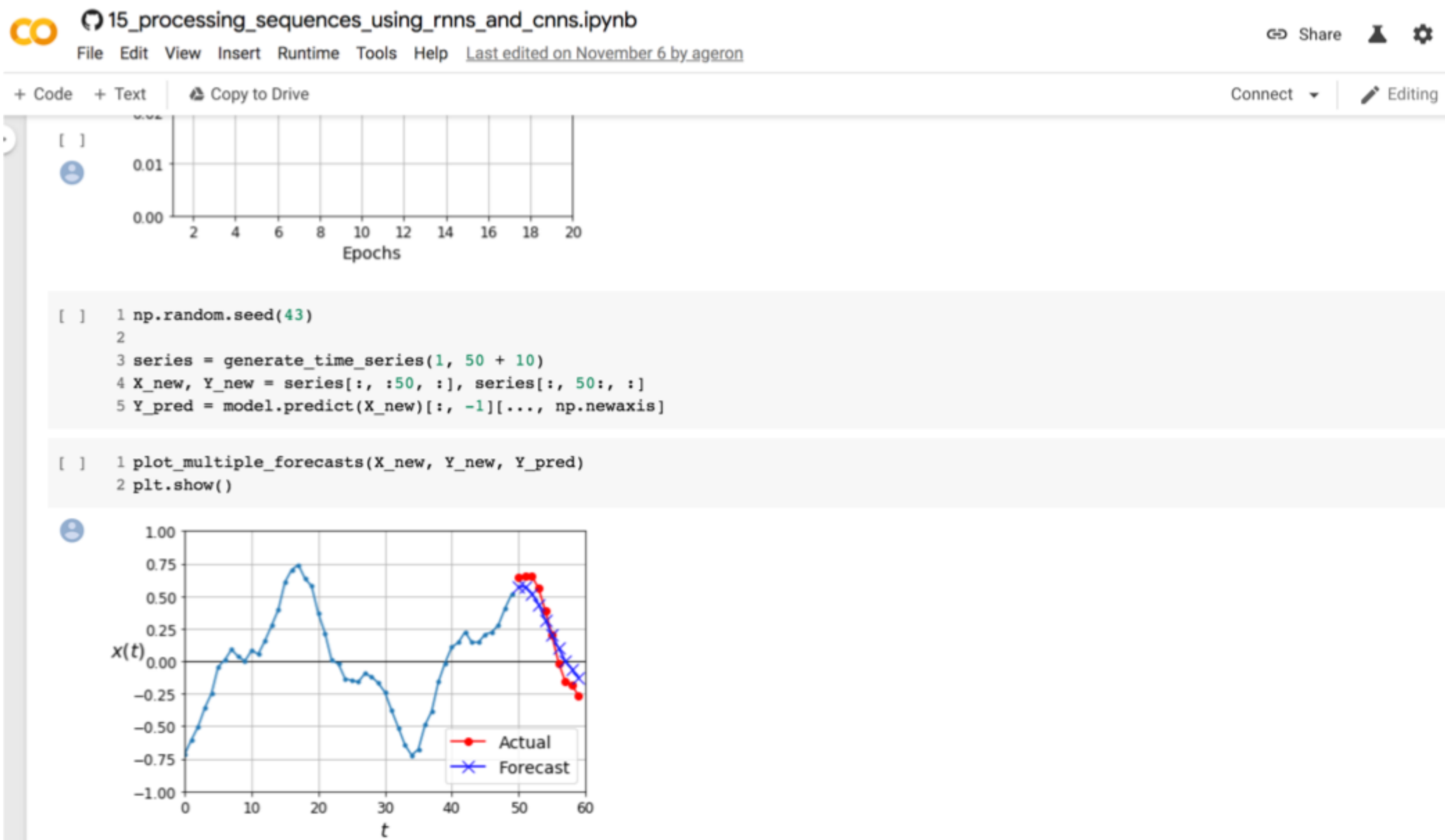
Notebooks

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. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml2

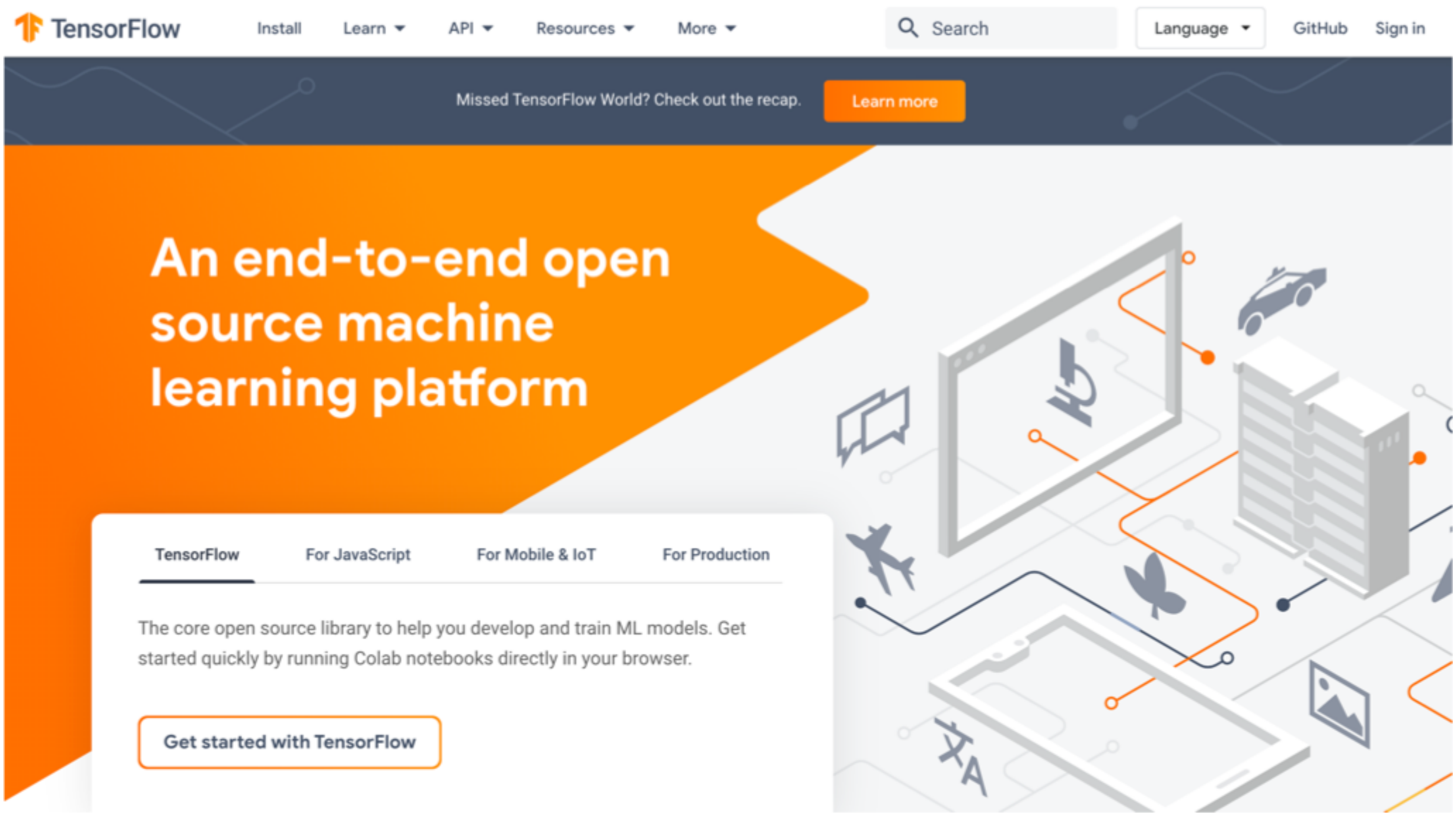
Sequences using RNNs and CNNs



https://colab.research.google.com/github/ageron/handson-

TensorFlow

TensorFlow



https://www.tensorflow.org/

TensorFlow

- An end-to-end open source machine learning platform.
- The core open source library to help you develop and train ML models.
- Get started quickly by running Colab notebooks directly in your browser.

https://www.tensorflow.org/

TensorFlow

Why TensorFlow 2.0

Why TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

About →

Easy model building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

Robust ML production anywhere

Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

TensorFlow

TensorFlow 2.0 vs. 1.X

```
# TensorFlow 2.0
outputs = f(input)

# TensorFlow 1.X
outputs = session.run(f(placeholder), feed_dict={placeholder: input})
```

Source: https://www.tensorflow.org/guide/effective_tf2

TensorFlow

TensorFlow 2.0

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/overview/

TensorFlow

TensorFlow 2 Quick Start

beginner.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 2 by lamberta

Share

\+ Code + Text Copy to Drive

Connect Editing

TensorFlow 2 quickstart for beginners

View on TensorFlow.org | Run in Google Colab | View source on GitHub | Download notebook

This short introduction uses Keras to:

1. Build a neural network that classifies images.
2. Train this neural network.
3. And, finally, evaluate the accuracy of the model.

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

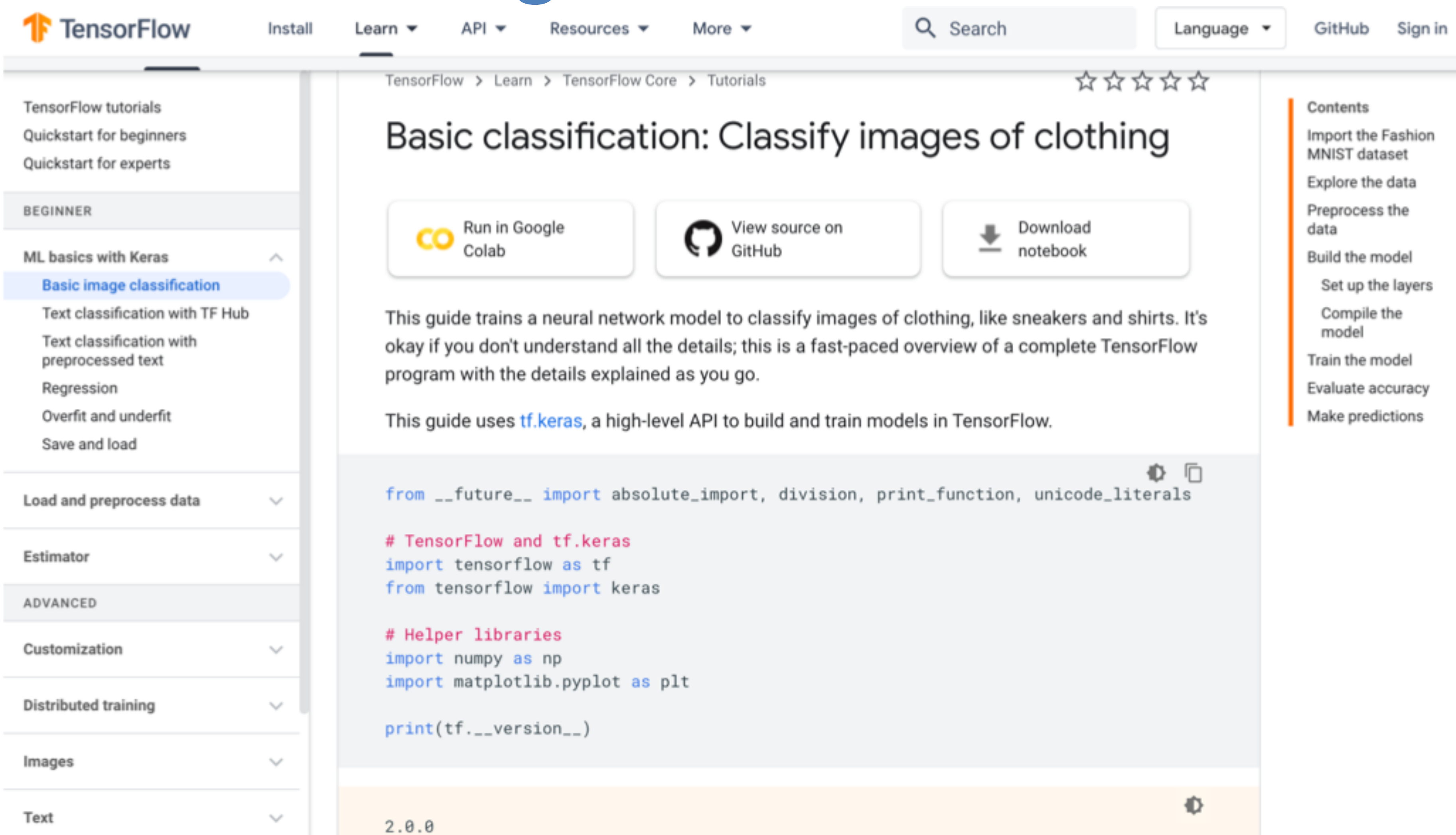
Download and install the TensorFlow 2 package. Import TensorFlow into your program:

```
from __future__ import absolute_import, division, print_function, unicode_literals

# Install TensorFlow
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb

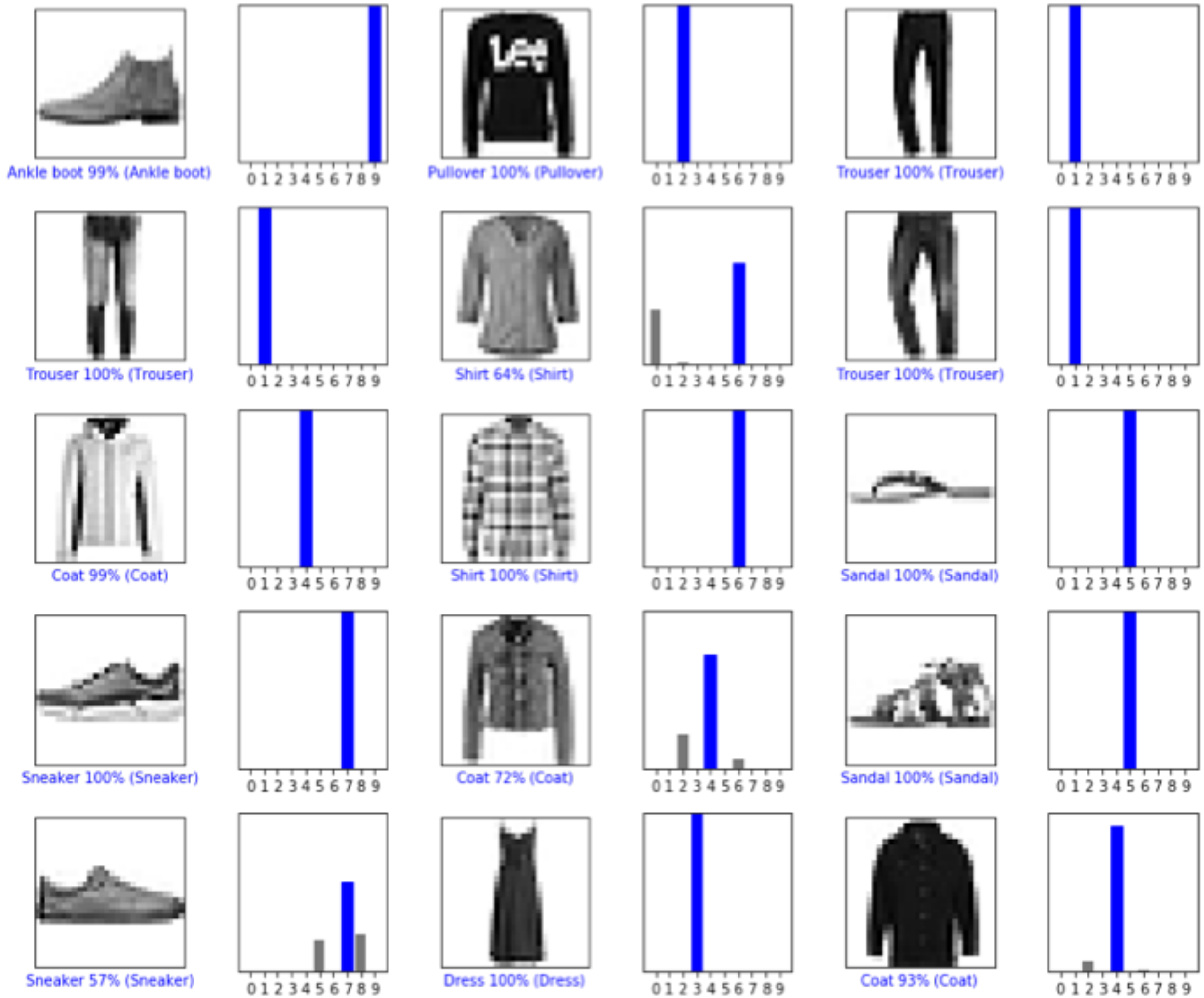
TensorFlow

TensorFlow
Image Classification



https://www.tensorflow.org/tutorials/keras/classification

Image Classification
Fashion MNIST dataset



https://www.tensorflow.org/tutorials/keras/classification

TensorFlow

Text Classification with TF Hub

TensorFlow Install Learn API Resources More Search Language GitHub Sign in

Overview Tutorials Guide TF 1

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data
- CSV
- NumPy
- pandas.DataFrame
- Images
- Text
- Unicode
- TF.Text
- TFRecord and tf.Example
- Additional formats with tf.io

TensorFlow > Learn > TensorFlow Core > Tutorials

Text classification with TensorFlow Hub: Movie reviews

Run in Google Colab | View source on GitHub | Download notebook

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

The tutorial demonstrates the basic application of transfer learning with TensorFlow Hub and Keras.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow, and TensorFlow Hub, a library and platform for transfer learning. For a more advanced text classification tutorial using `tf.keras`, see the MLCC Text Classification Guide.

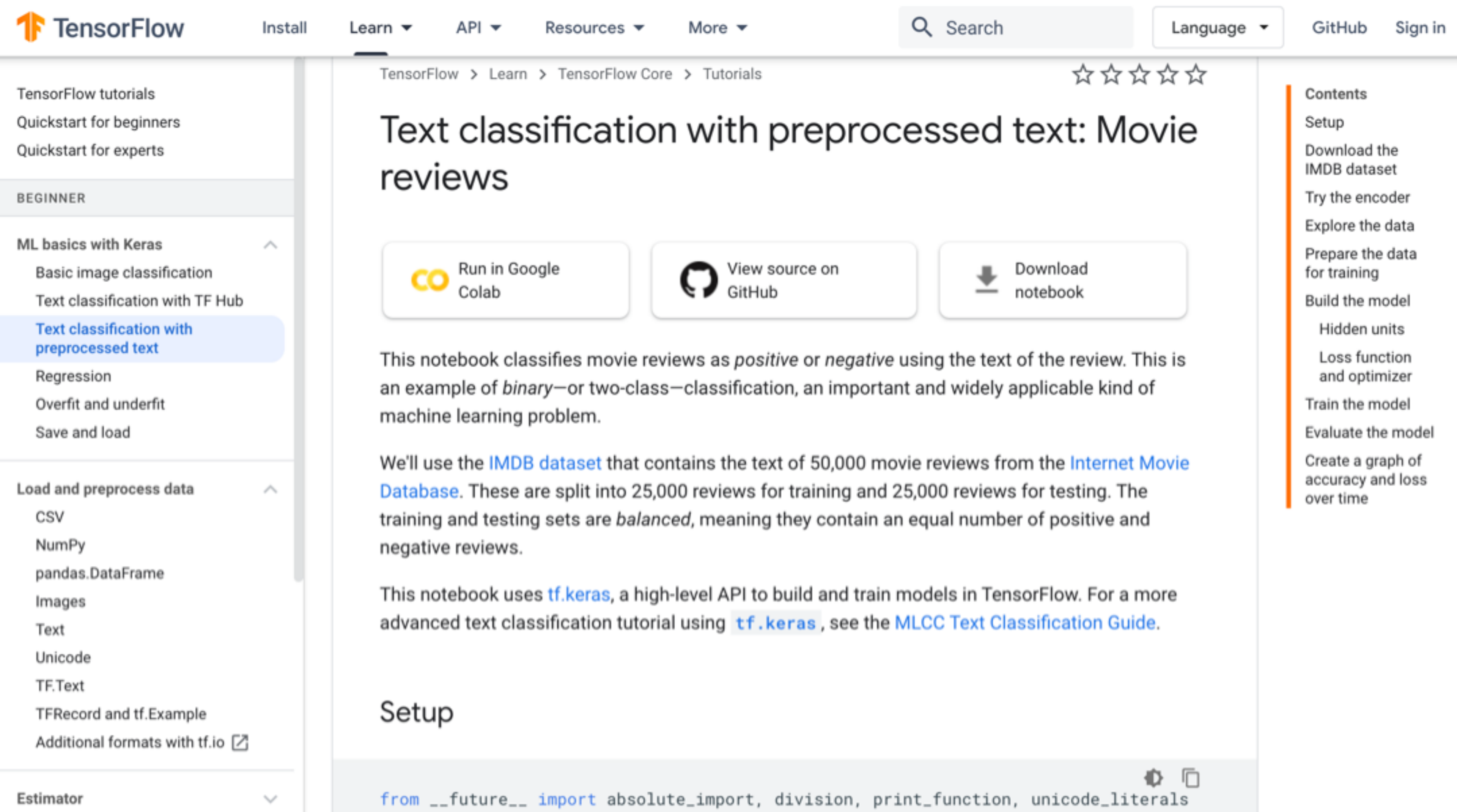
```
from __future__ import absolute_import, division, print_function, unicode_literals
```

Contents
- Download the IMDB dataset
- Explore the data
- Build the model
  - Loss function and optimizer
- Train the model
- Evaluate the model
- Further reading

https://www.tensorflow.org/tutorials/keras/text_classification_with_hub

TensorFlow

Text Classification with Pre Text



TensorFlow | Install | Learn | API | Resources | More | Search | Language | GitHub | Sign in

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data
- CSV
- NumPy
- pandas.DataFrame
- Images
- Text
- Unicode
- TF.Text
- TFRecord and tf.Example
- Additional formats with tf.io

Estimator

TensorFlow > Learn > TensorFlow Core > Tutorials

Text classification with preprocessed text: Movie reviews

Run in Google Colab | View source on GitHub | Download notebook

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the MLCC Text Classification Guide.

Setup

```
from __future__ import absolute_import, division, print_function, unicode_literals
```

Contents
- Setup
- Download the IMDB dataset
- Try the encoder
- Explore the data
- Prepare the data for training
- Build the model
  - Hidden units
  - Loss function and optimizer
- Train the model
- Evaluate the model
- Create a graph of accuracy and loss over time

https://www.tensorflow.org/tutorials/keras/text_classification

TensorFlow

Regression

TensorFlow Install Learn API Resources More Search Language GitHub Sign in

TensorFlow > Learn > TensorFlow Core > Tutorials

Basic regression: Predict fuel efficiency

Run in Google Colab | View source on GitHub | Download notebook

In a *regression* problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a *classification* problem, where we aim to select a class from a list of classes (for example, where a picture contains an apple or an orange, recognizing which fruit is in the picture).

This notebook uses the classic Auto MPG Dataset and builds a model to predict the fuel efficiency of late-1970s and early 1980s automobiles. To do this, we'll provide the model with a description of many automobiles from that time period. This description includes attributes like: cylinders, displacement, horsepower, and weight.

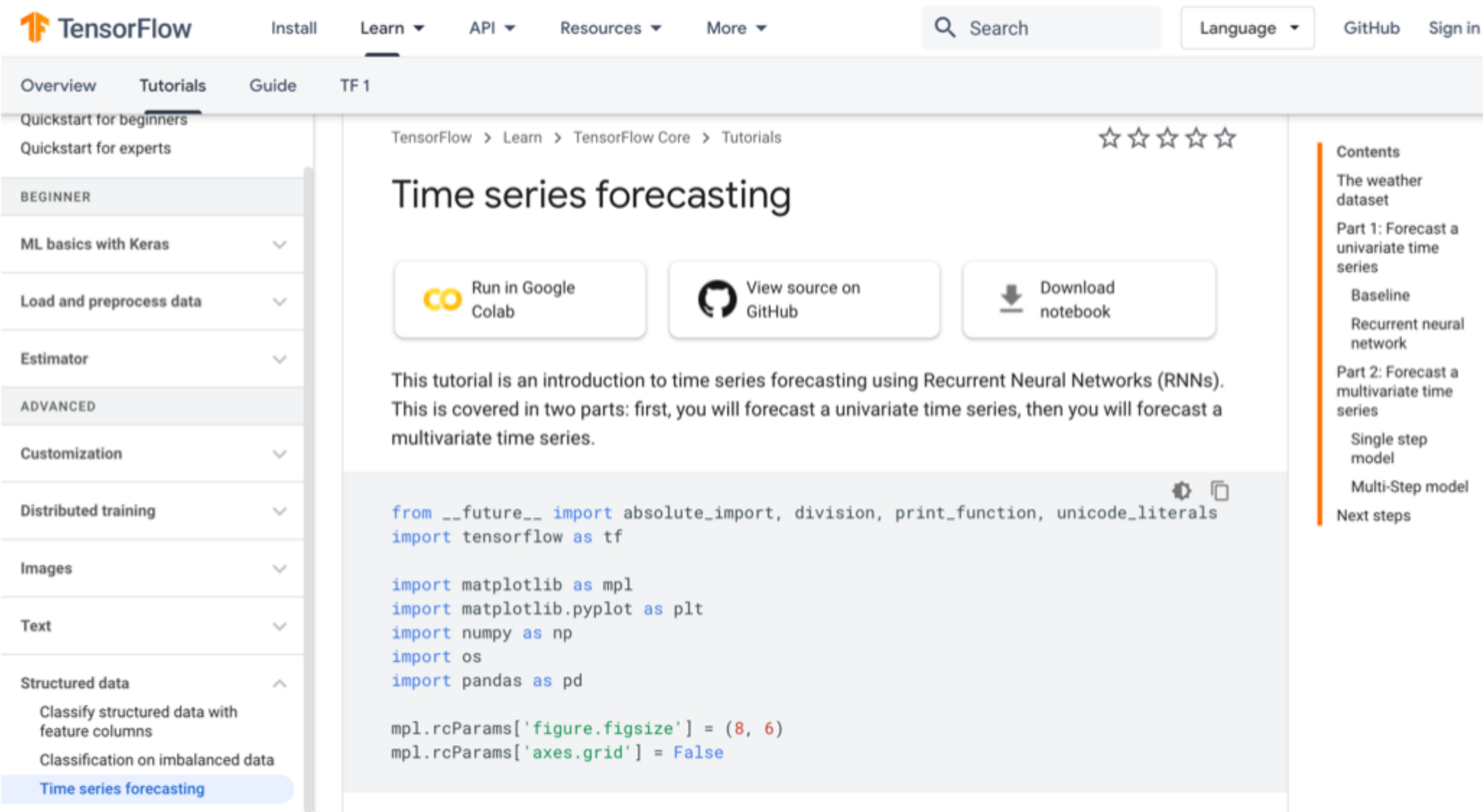
This example uses the `tf.keras` API, see this guide for details.

```
# Use seaborn for pairplot
!pip install -q seaborn
```

```
from __future__ import absolute_import, division, print_function, unicode_literals

import pathlib
```


https://www.tensorflow.org/tutorials/keras/regression

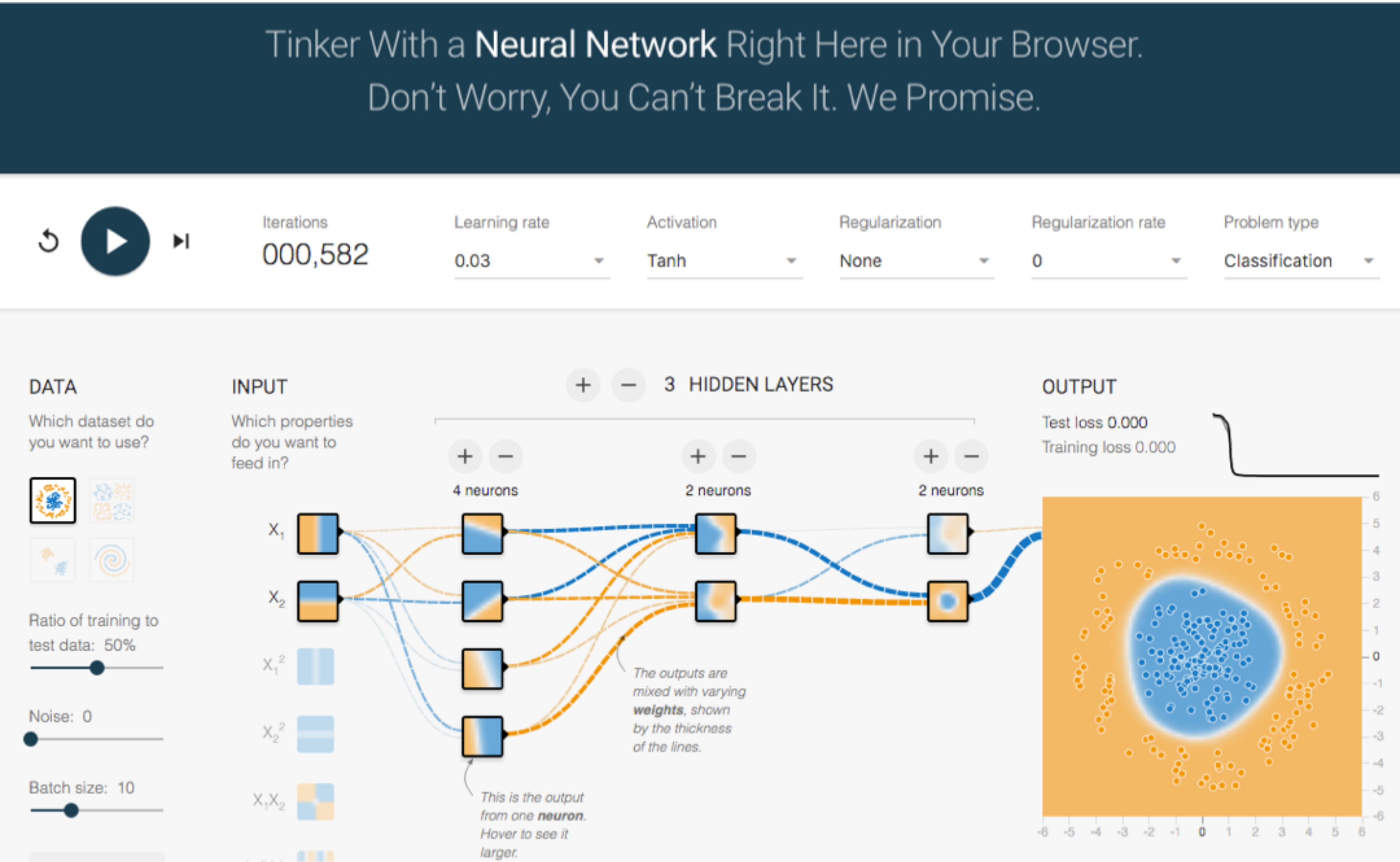
TensorFlow 2.0 Time Series Forecasting




```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf

import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False
```

https://www.tensorflow.org/tutorials/structured_data/time_series

TensorFlow Playground



http://playground.tensorflow.org/

Tensor

- **3**
  - # a rank 0 tensor; this is a **scalar** with shape []
- **[1. ,2., 3.]**
  - # a rank 1 tensor; this is a **vector** with shape [3]
- **[[1., 2., 3.], [4., 5., 6.]]**
  - # a rank 2 tensor; a **matrix** with shape [2, 3]
- **[[[1., 2., 3.]], [[7., 8., 9.]]]**
  - # a rank 3 **tensor** with shape [2, 1, 3]

https://www.tensorflow.org/

Scalar

$$80$$

Vector

$$[50 \quad 60 \quad 70]$$

Matrix

$$\begin{bmatrix} 50 & 60 & 70 \\ 55 & 65 & 75 \end{bmatrix}$$

Tensor

$$\begin{bmatrix} [50 \quad 60 \quad 70] & [70 \quad 80 \quad 90] \\ [55 \quad 65 \quad 75] & [75 \quad 85 \quad 95] \end{bmatrix}$$

Time Series Data

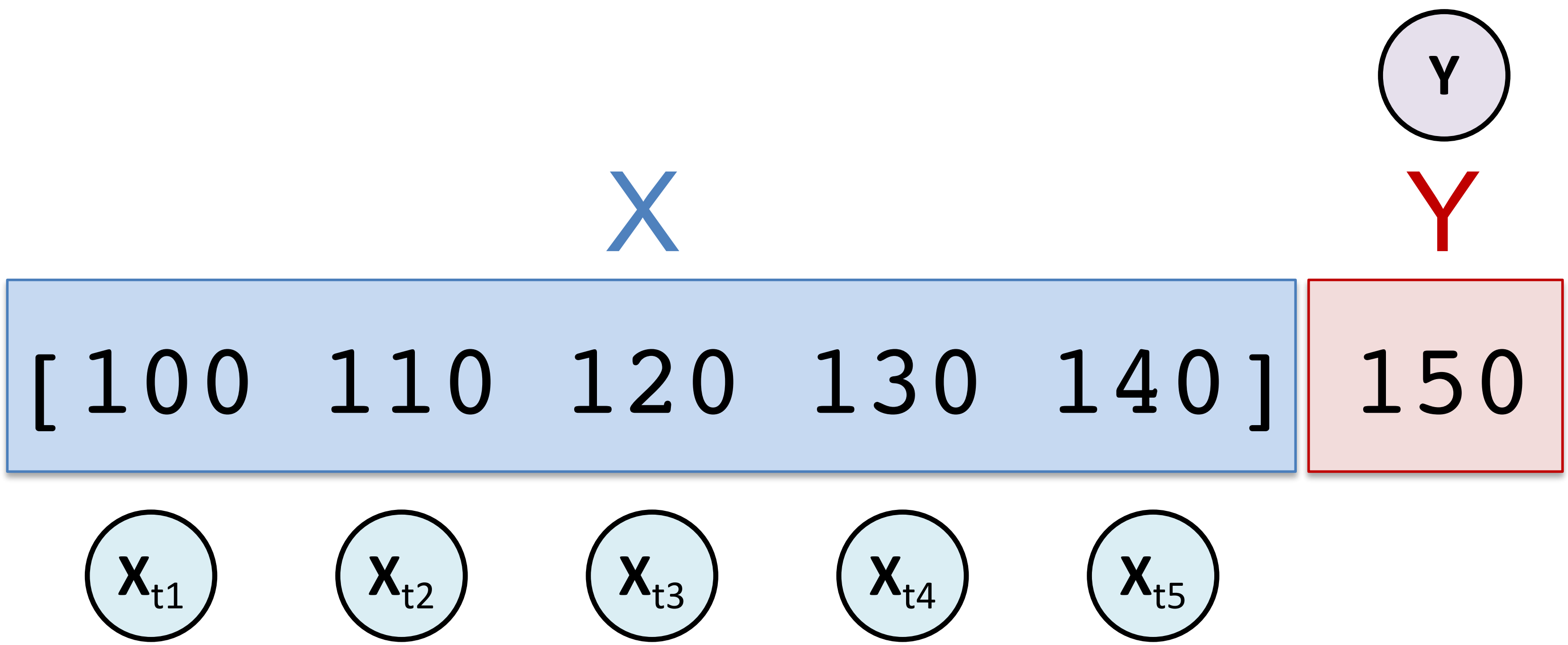
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```

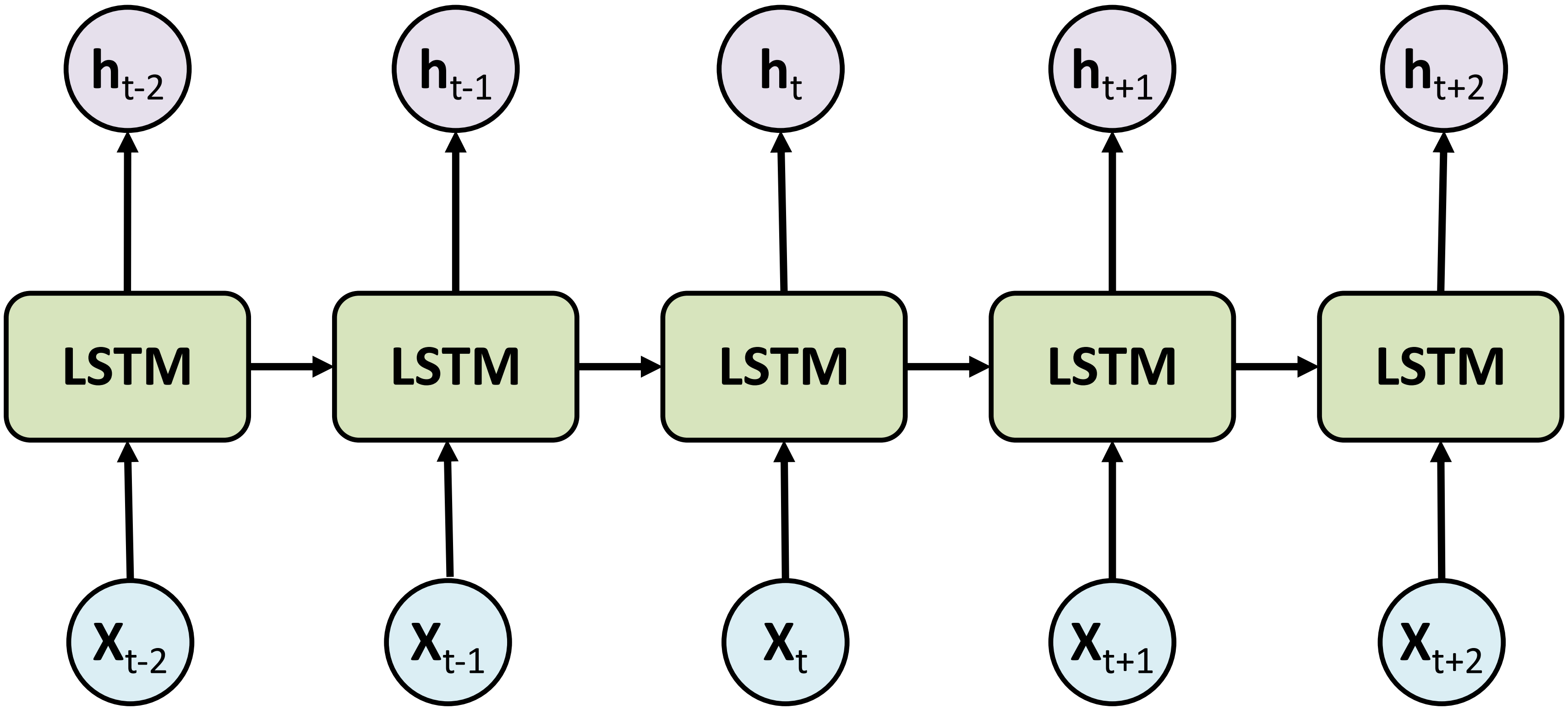
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Time Series Data

[100, 110, 120, 130, 140, 150]



Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

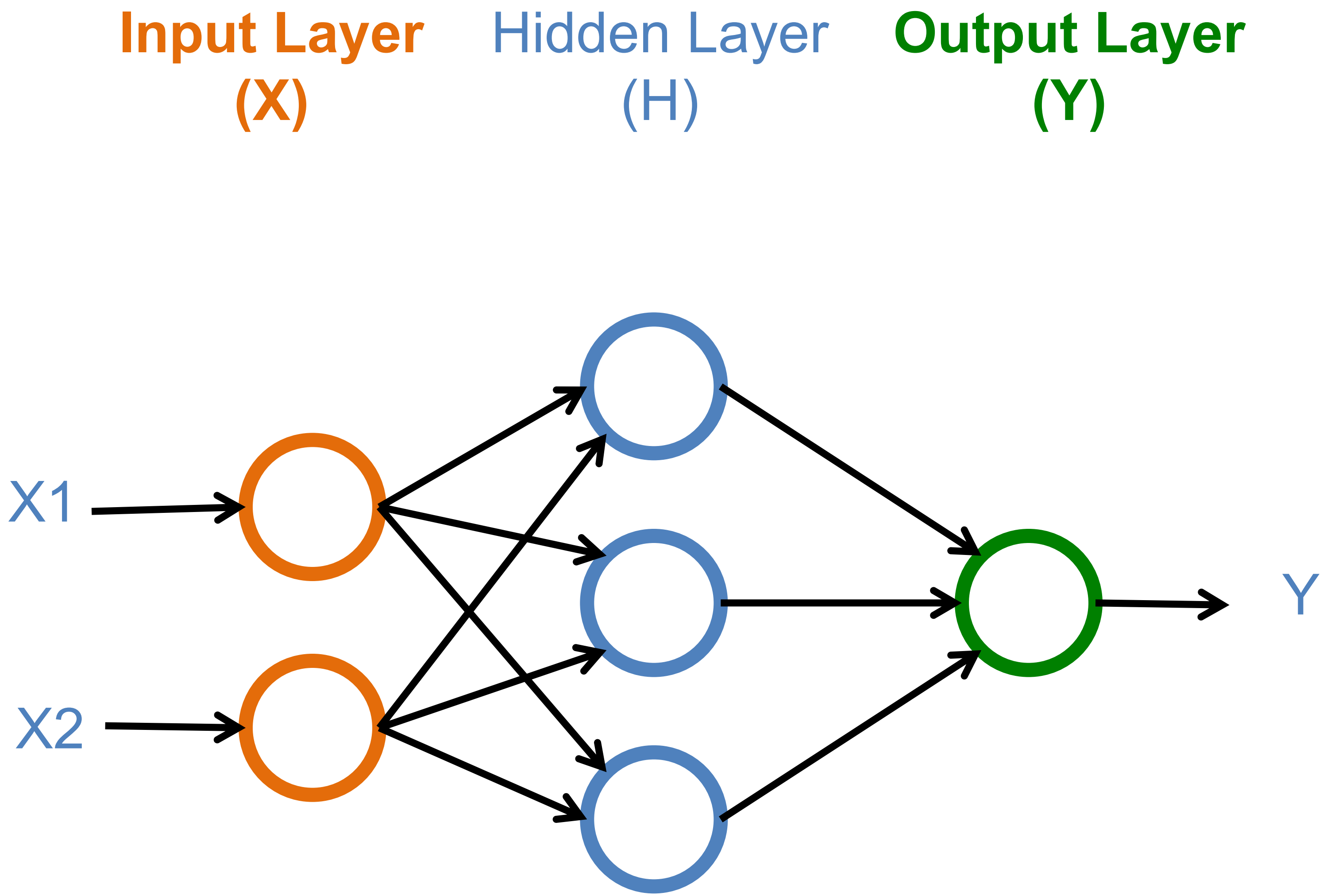
[10, 20, 30, 40, 50, 60, 70, 80, 90]

X	Y
[10 20 30]	40
[20 30 40]	50
[30 40 50]	60
[40 50 60]	70
[50 60 70]	80
[60 70 80]	90

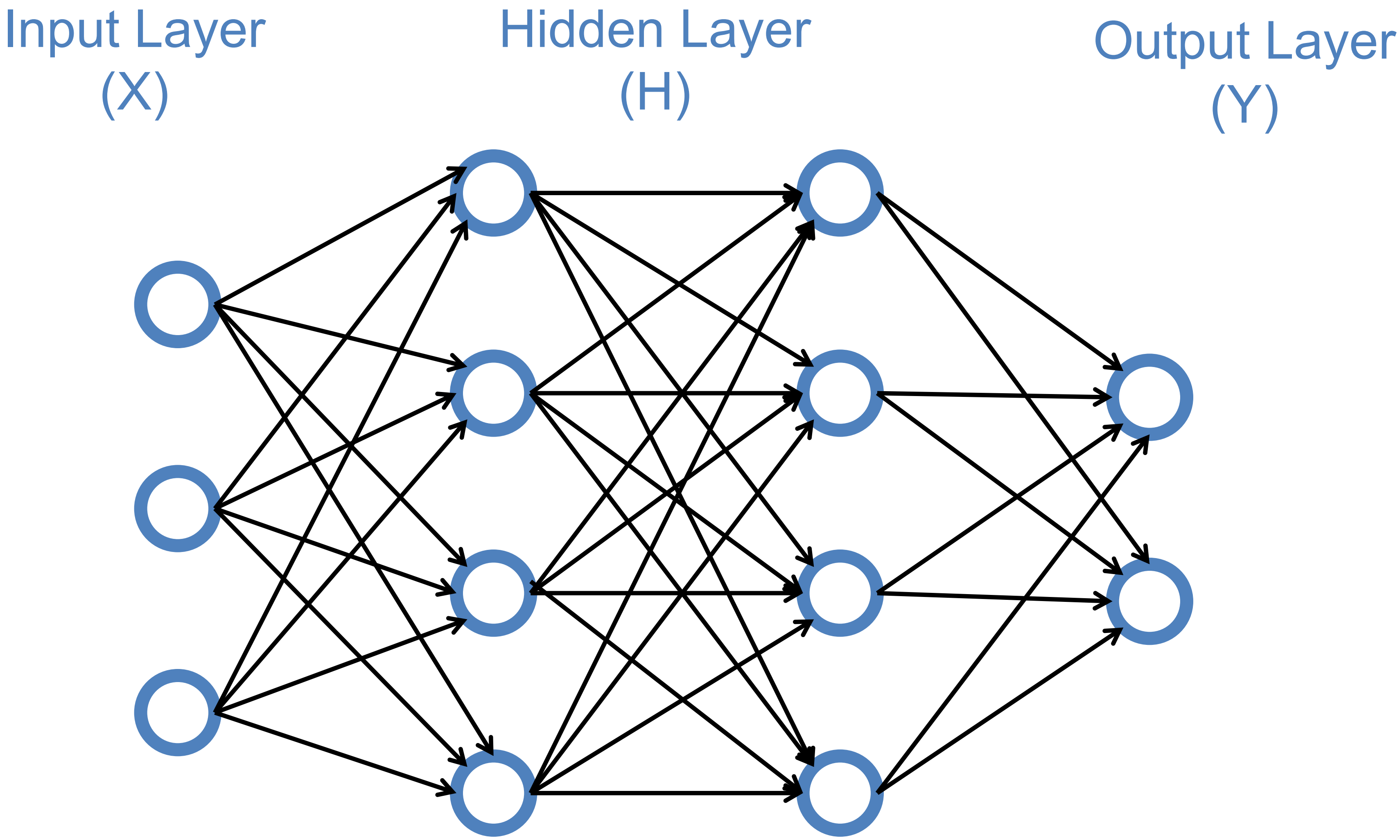
Deep Learning and Neural Networks

Deep Learning Foundations: Neural Networks

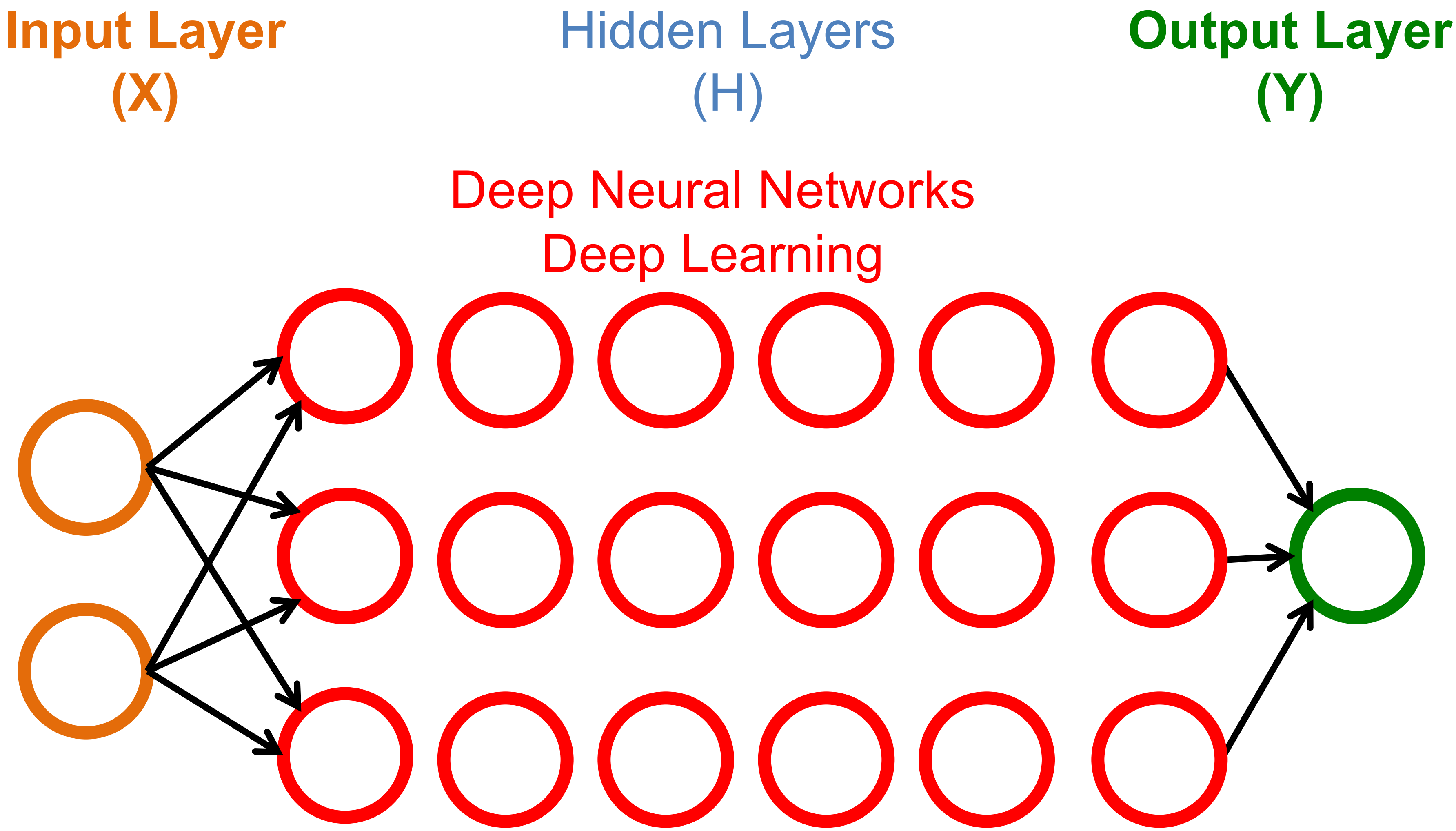
Deep Learning and Neural Networks



Deep Learning and Neural Networks

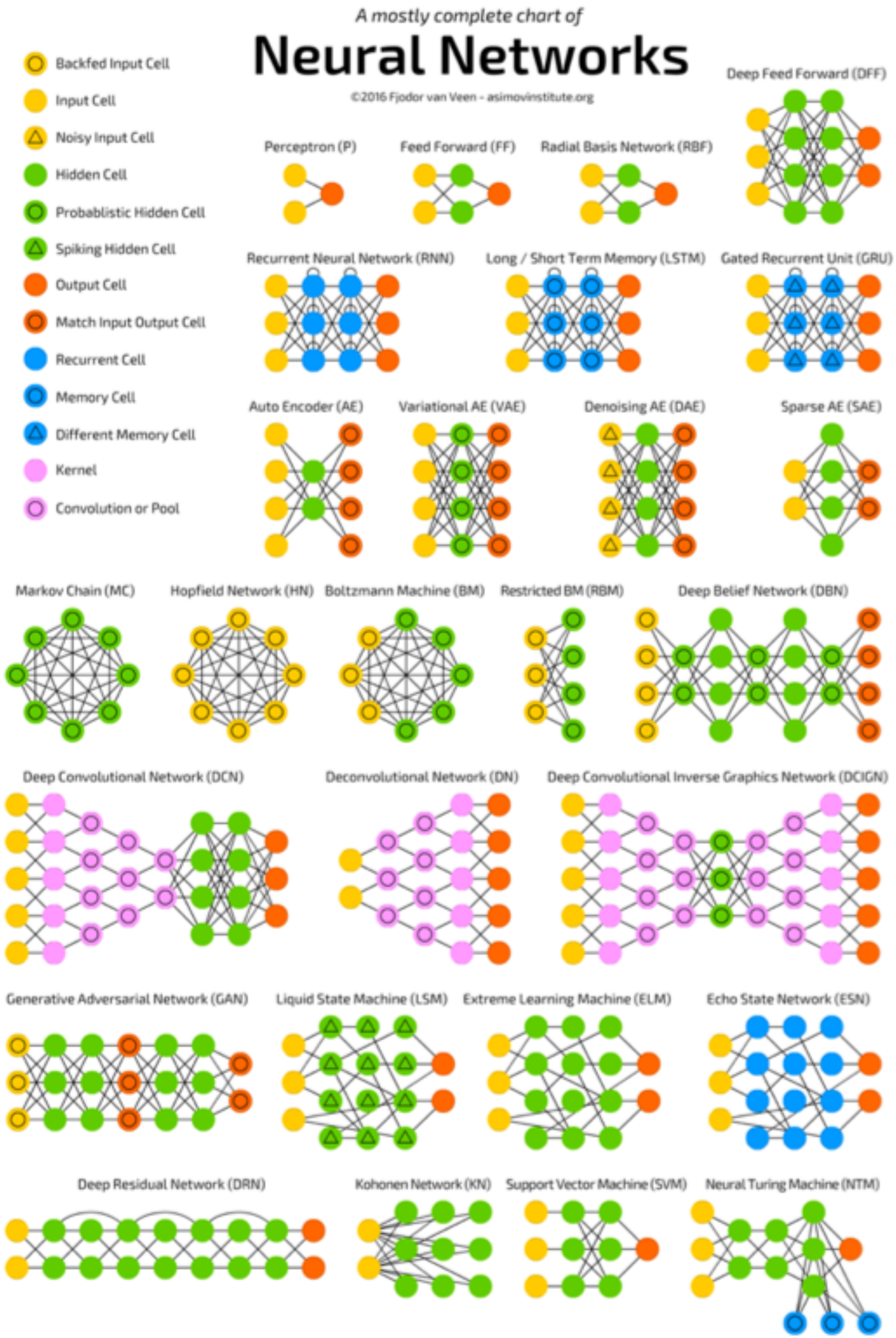


Deep Learning and Neural Networks

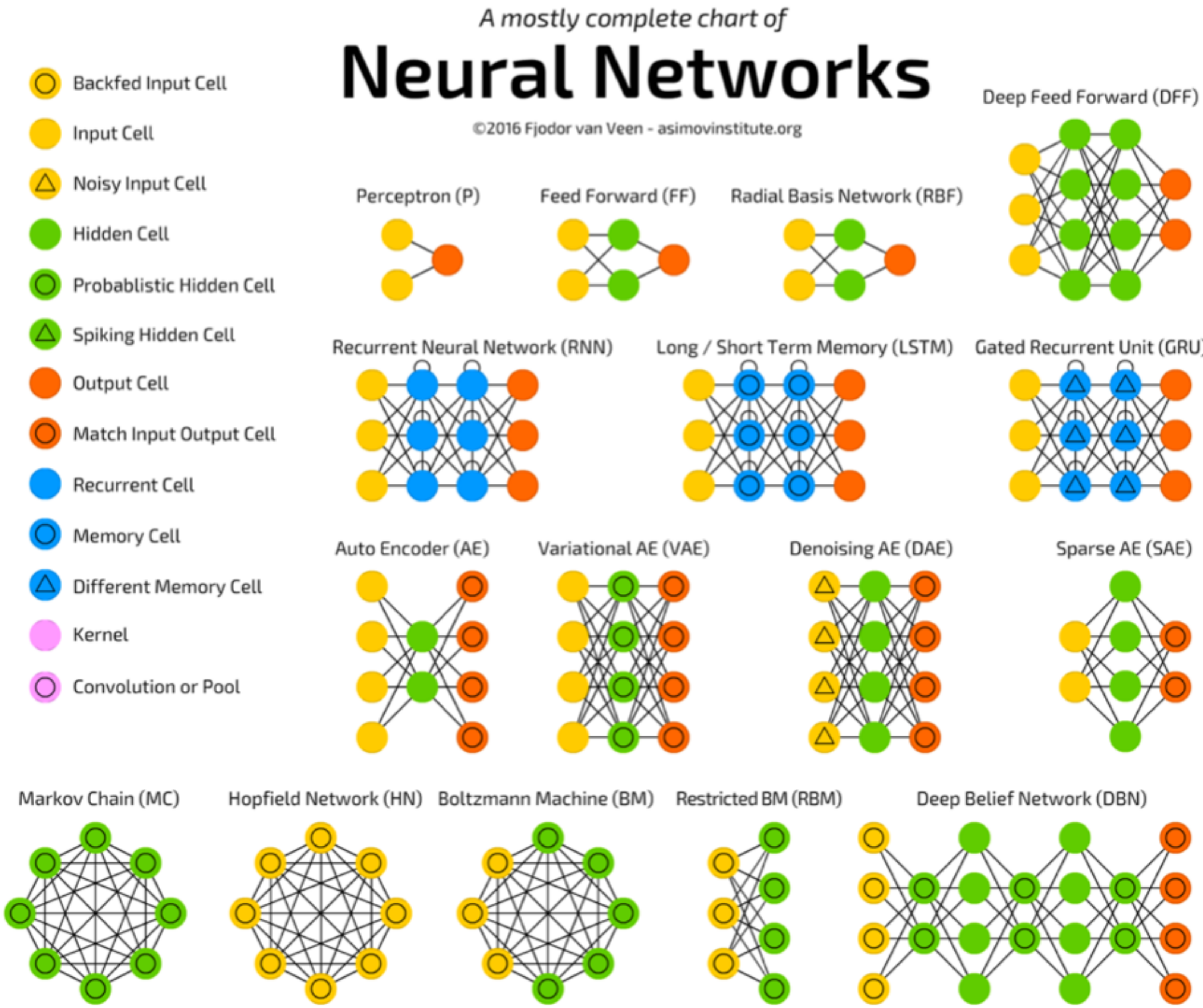


Deep Learning and Deep Neural Networks

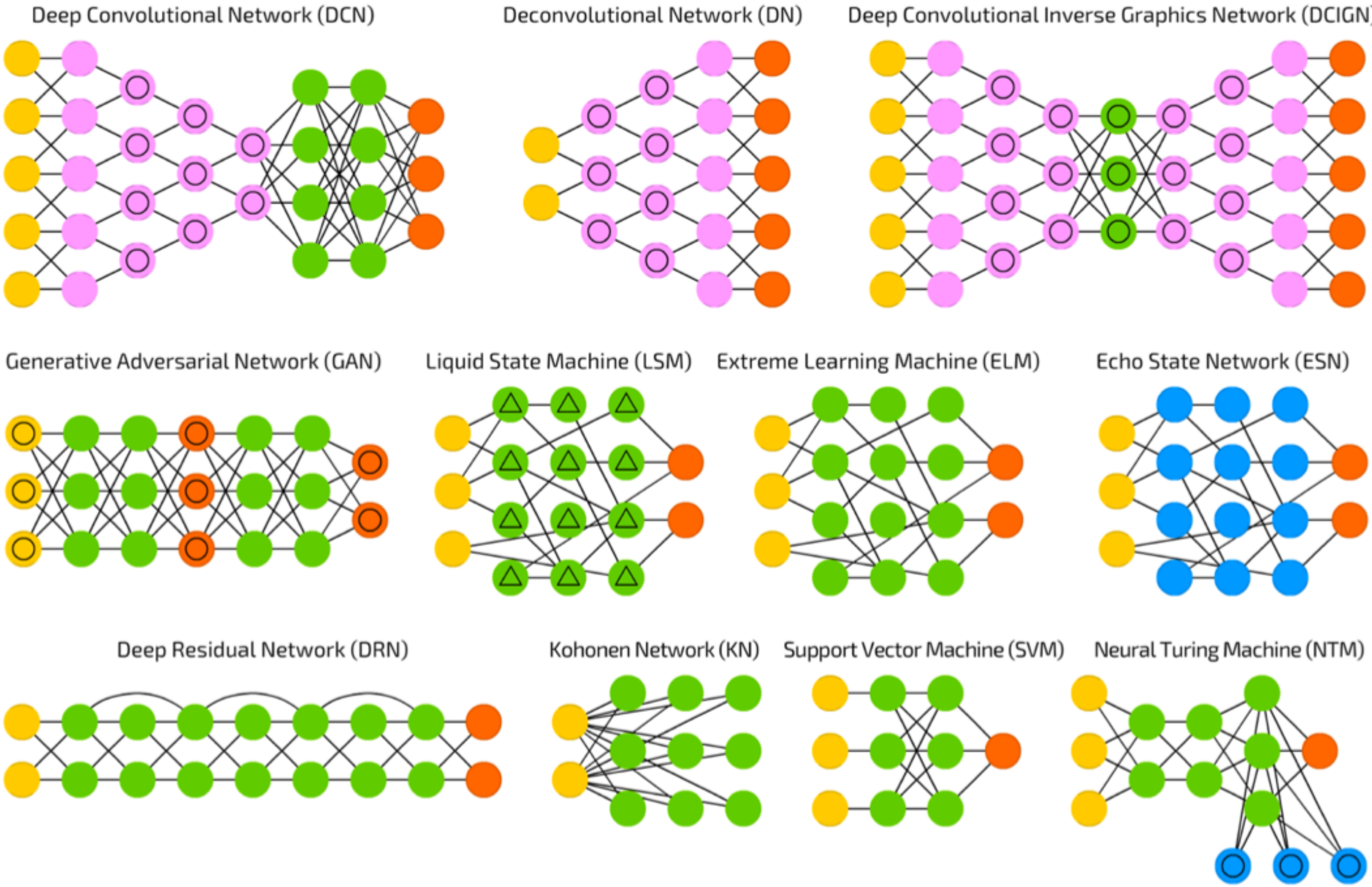
Neural Networks (NN)



Source: http://www.asimovinstitute.org/neural-network-zoo/



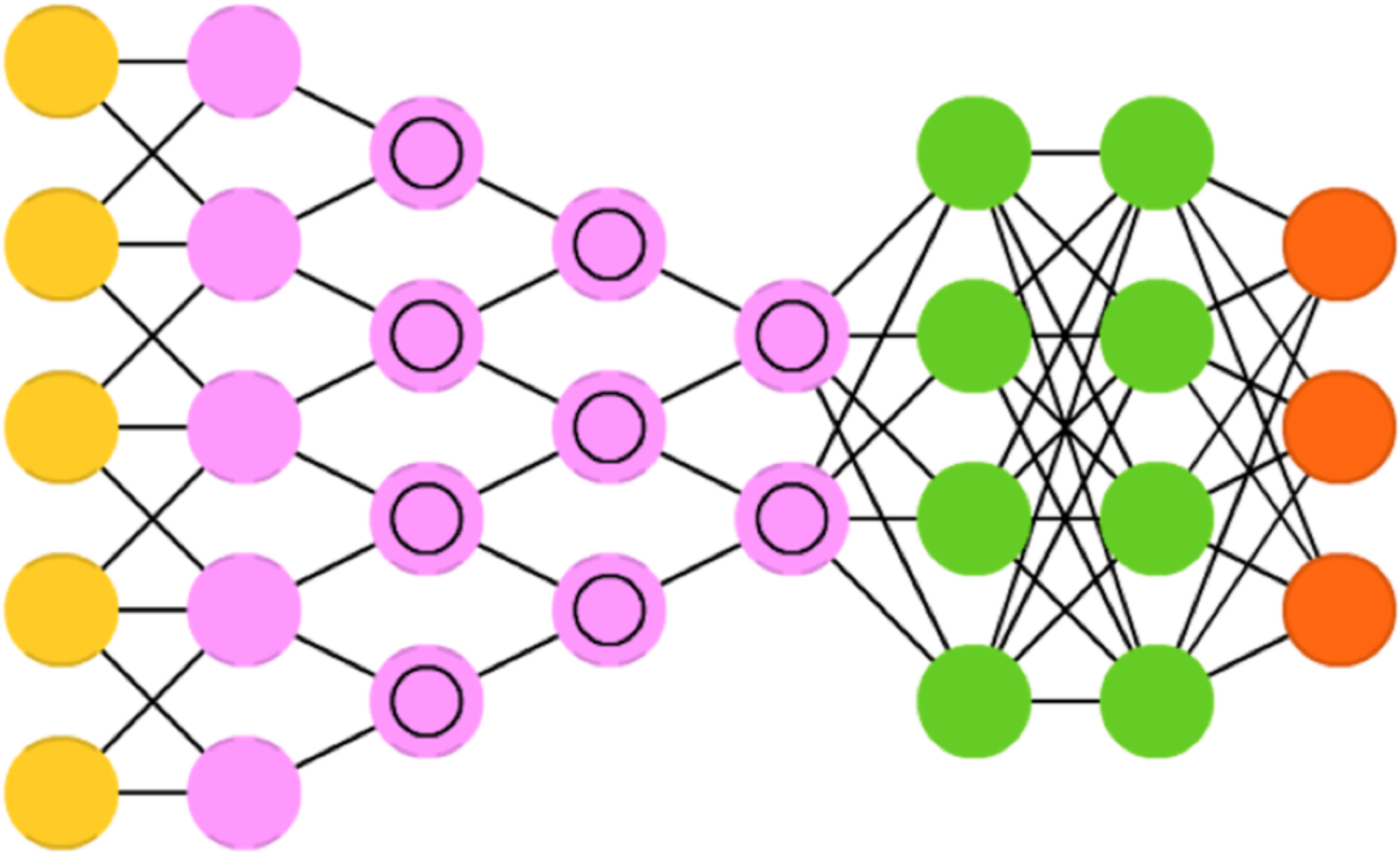
Source: http://www.asimovinstitute.org/neural-network-zoo/



Source: http://www.asimovinstitute.org/neural-network-zoo/

Convolutional Neural Networks

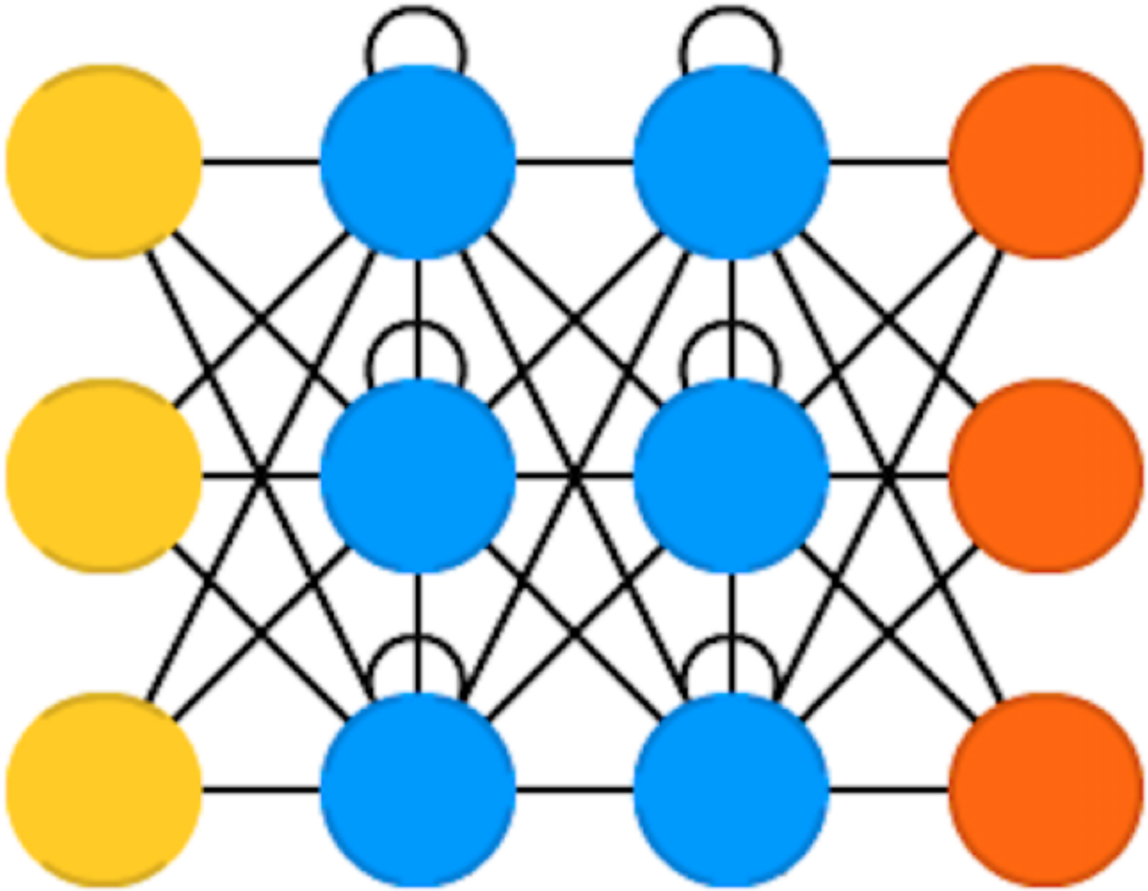
(CNN or Deep Convolutional Neural Networks, DCNN)



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

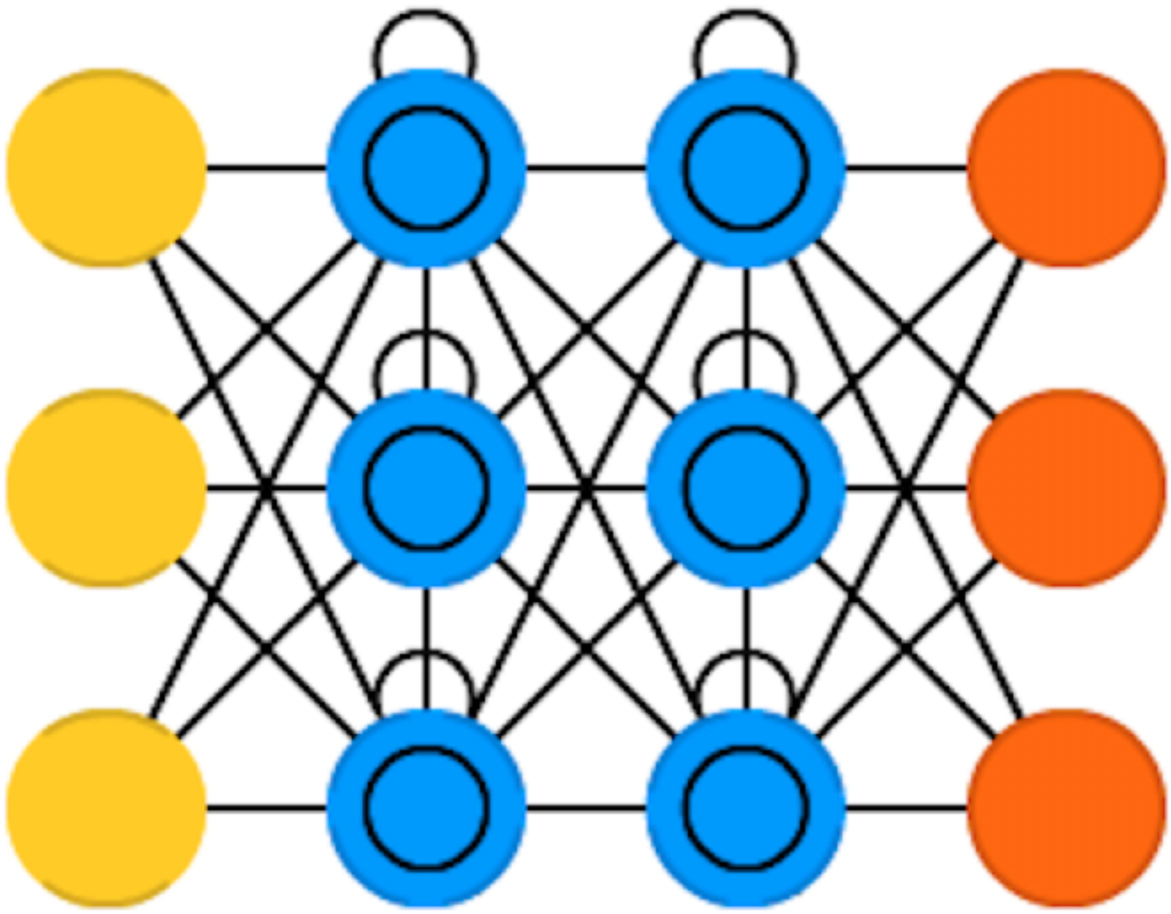
Source: http://www.asimovinstitute.org/neural-network-zoo/

Recurrent Neural Networks (RNN)



Elman, Jeffrey L. "Finding structure in time." Cognitive science 14.2 (1990): 179-211
Source: http://www.asimovinstitute.org/neural-network-zoo/

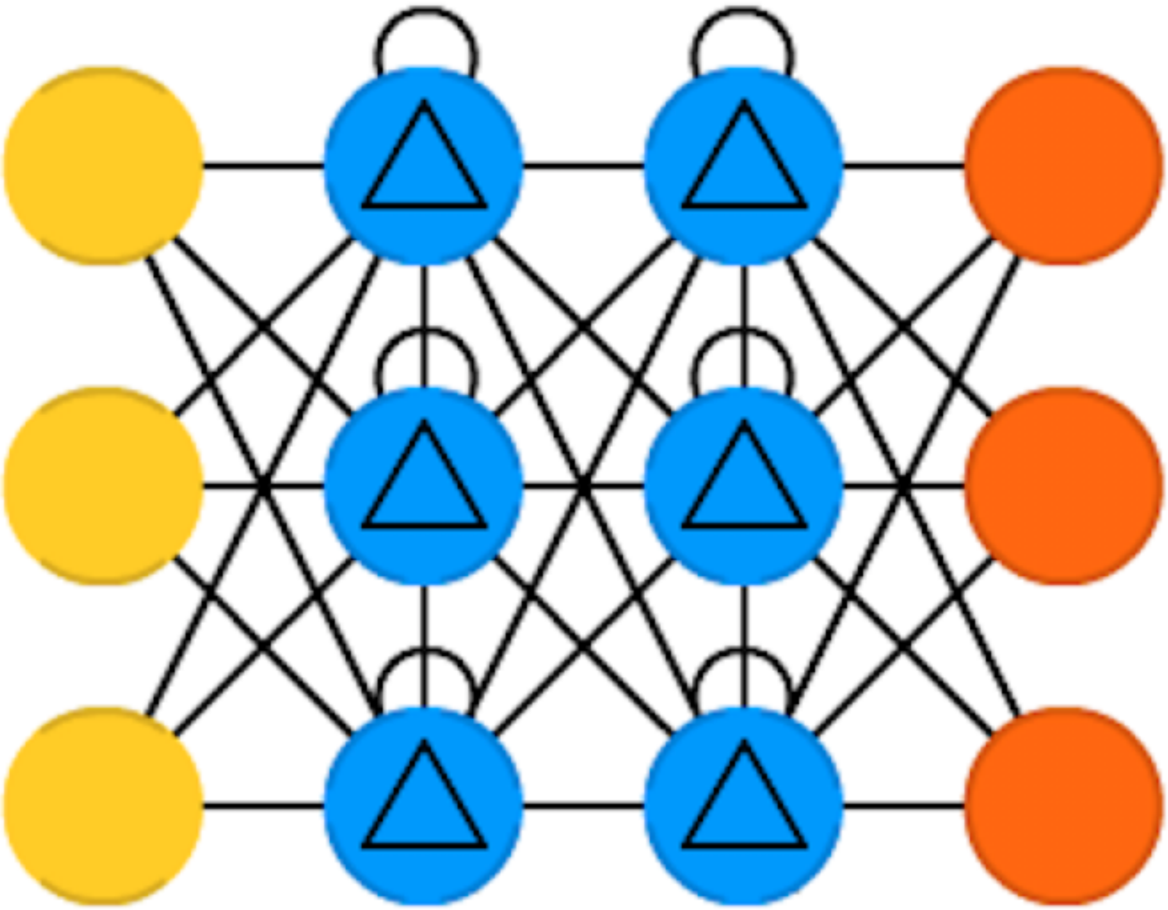
Long / Short Term Memory (LSTM)



Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

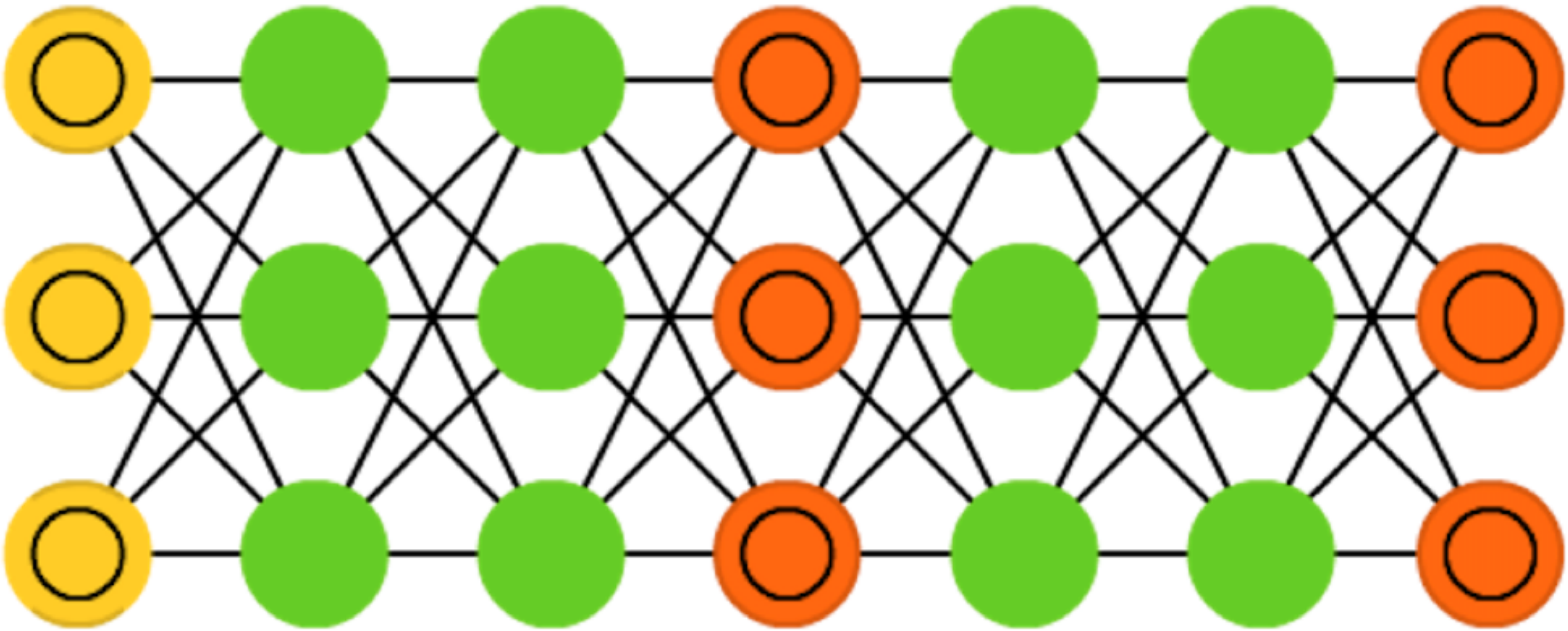
Source: http://www.asimovinstitute.org/neural-network-zoo/

Gated Recurrent Units (GRU)



Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
Source: http://www.asimovinstitute.org/neural-network-zoo/

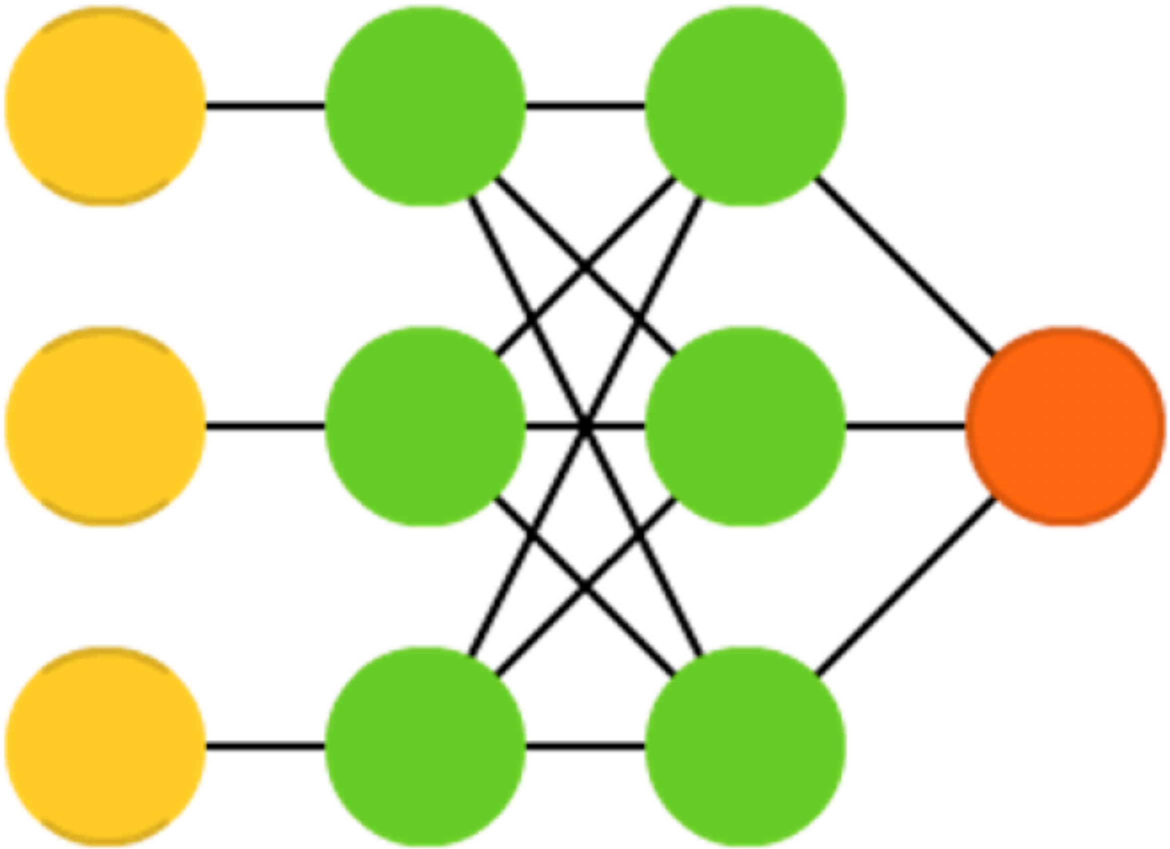
Generative Adversarial Networks (GAN)



Goodfellow, Ian, et al. "Generative adversarial nets." Advances in Neural Information Processing Systems. 2014.

Source: http://www.asimovinstitute.org/neural-network-zoo/

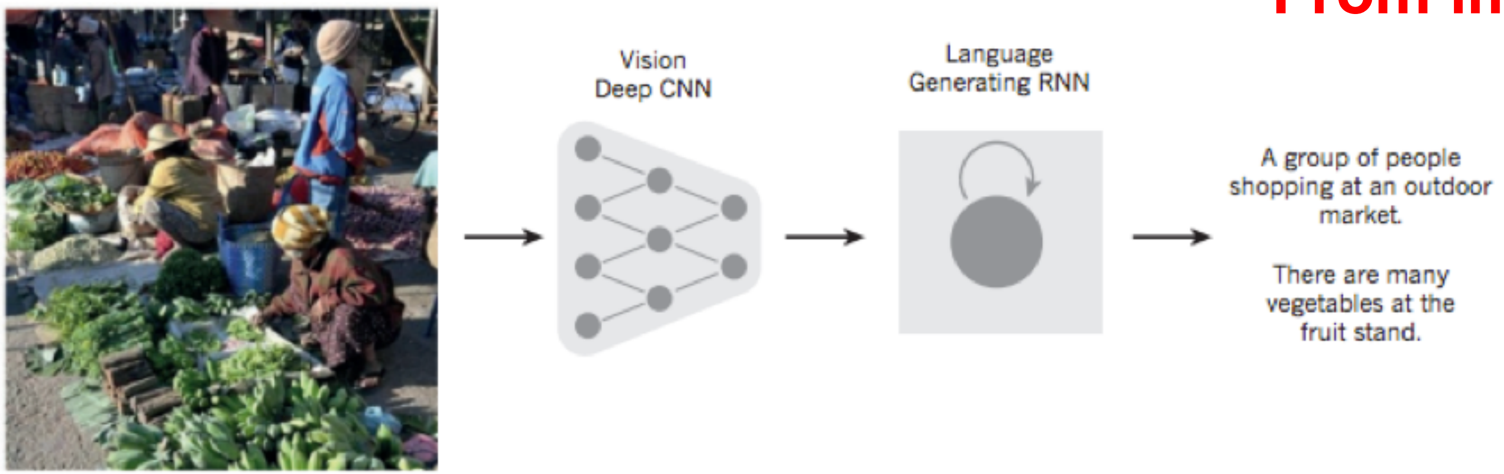
Support Vector Machines (SVM)



Cortes, Corinna, and Vladimir Vapnik. “Support-vector networks.” Machine learning 20.3 (1995): 273-297.

Source: http://www.asimovinstitute.org/neural-network-zoo/

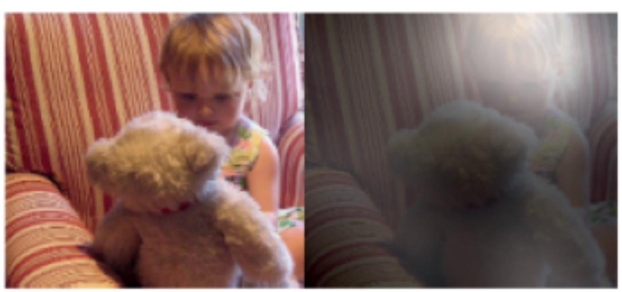
From image to text



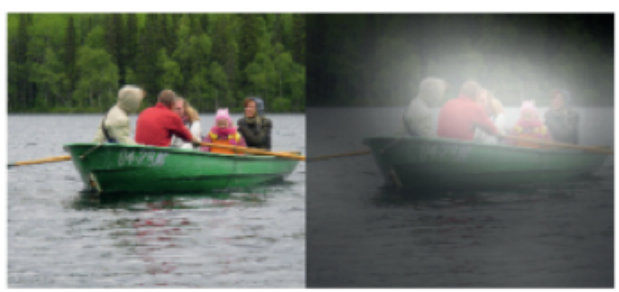
A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.



A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

From image to text

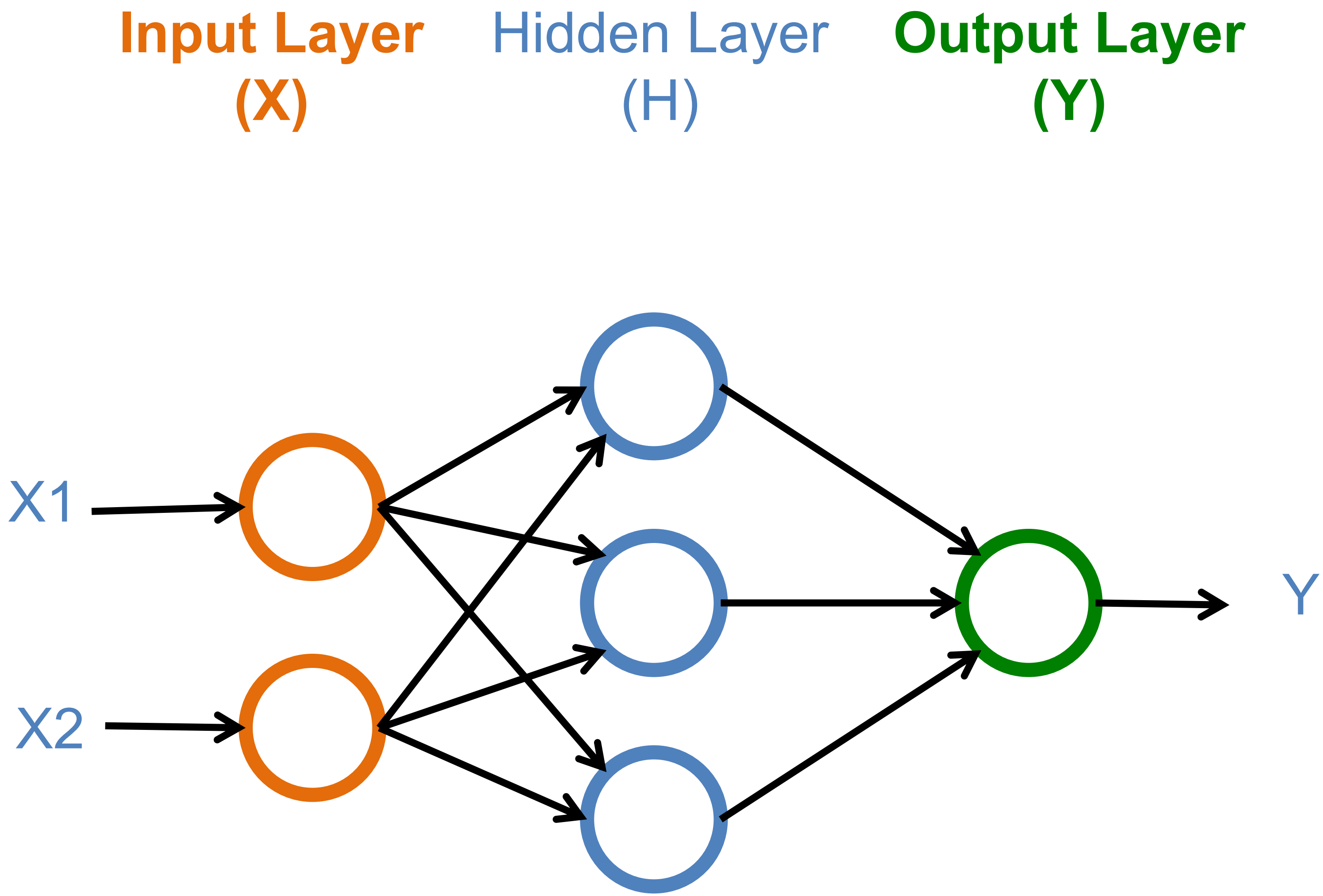
Image: deep convolution neural network (CNN)

Text: recurrent neural network (RNN)

A group of **people** sitting on a boat in the water.

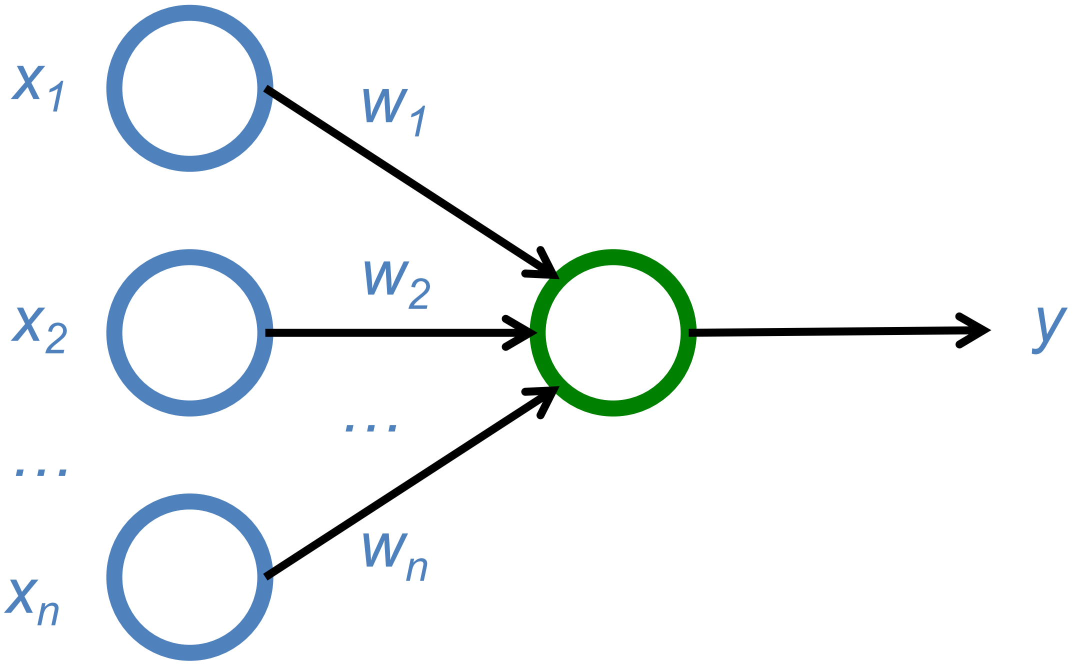
Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

Neural Networks



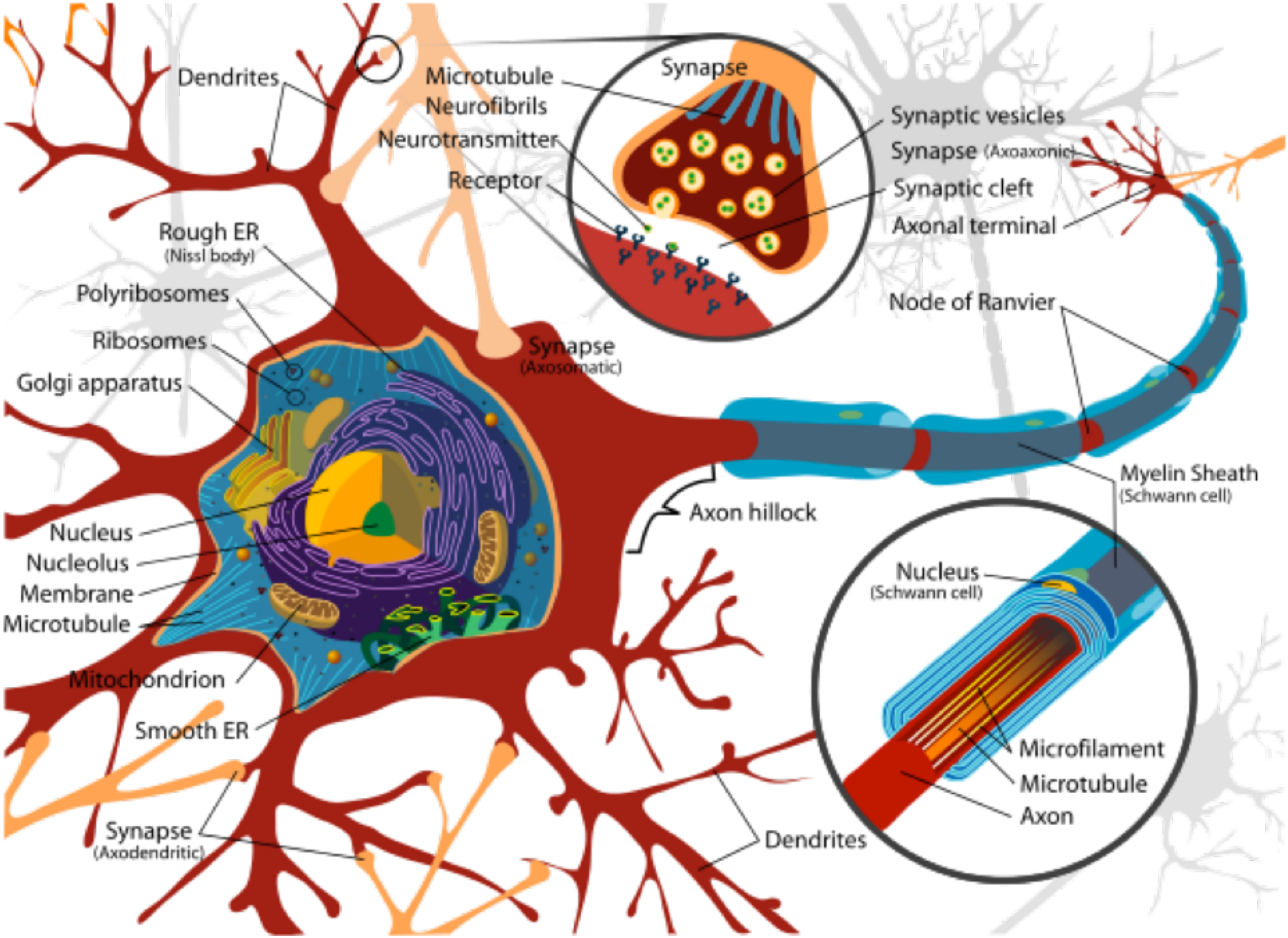
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

The Neuron



Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

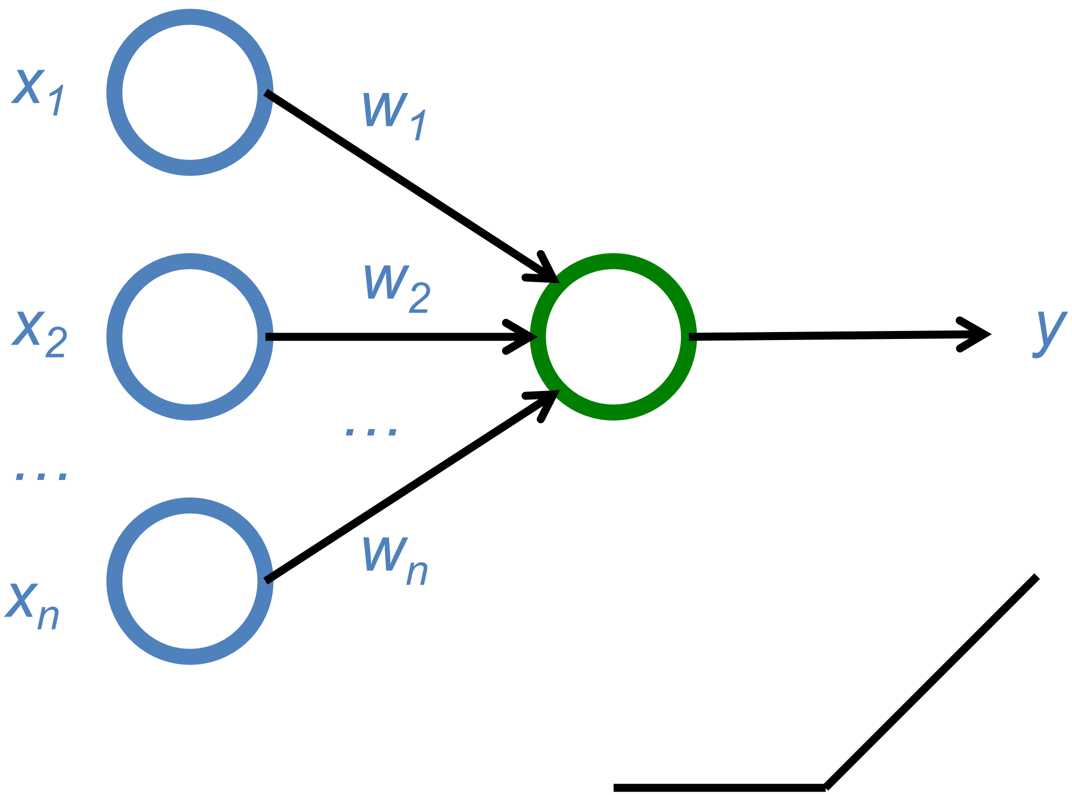
Neuron and Synapse



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

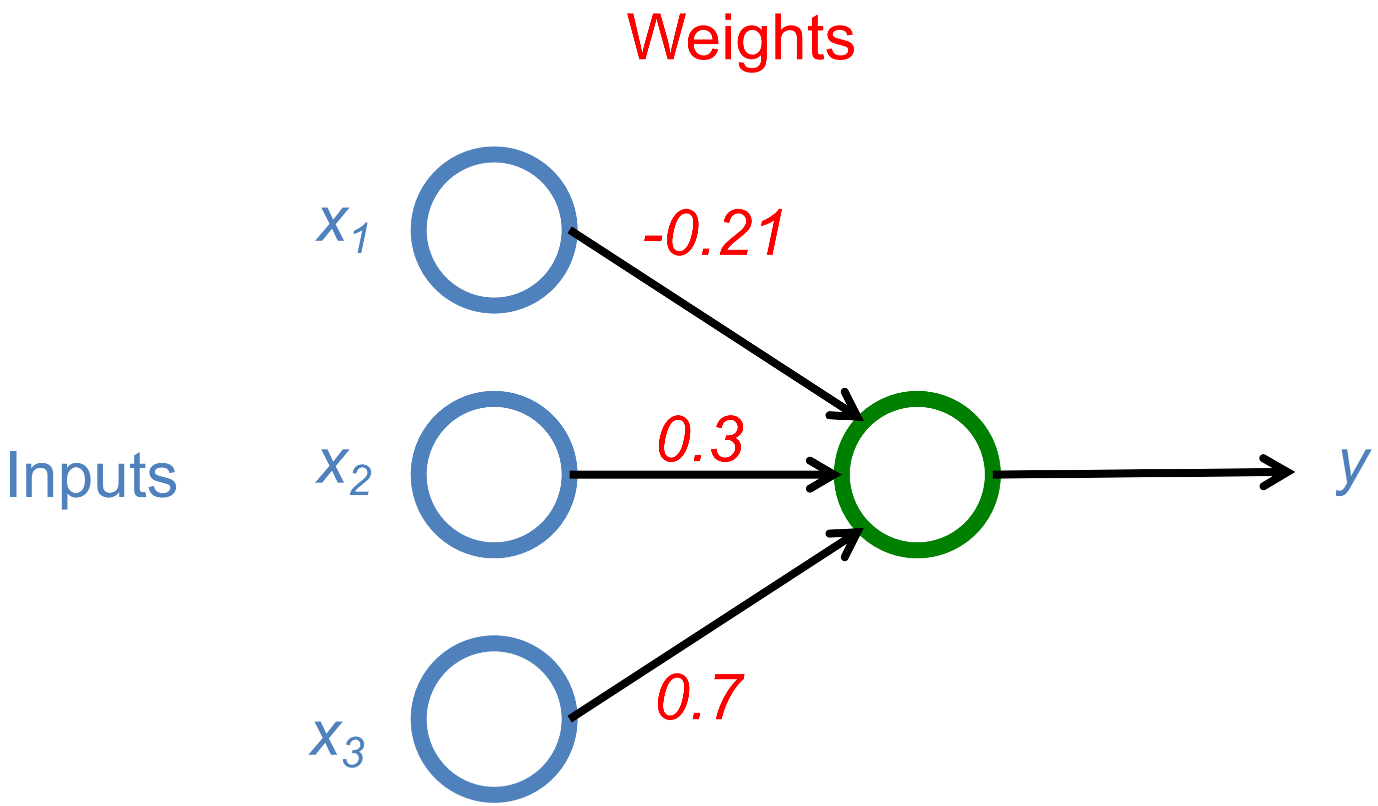
The Neuron

$$y = F\left(\sum_i w_i x_i\right)$$



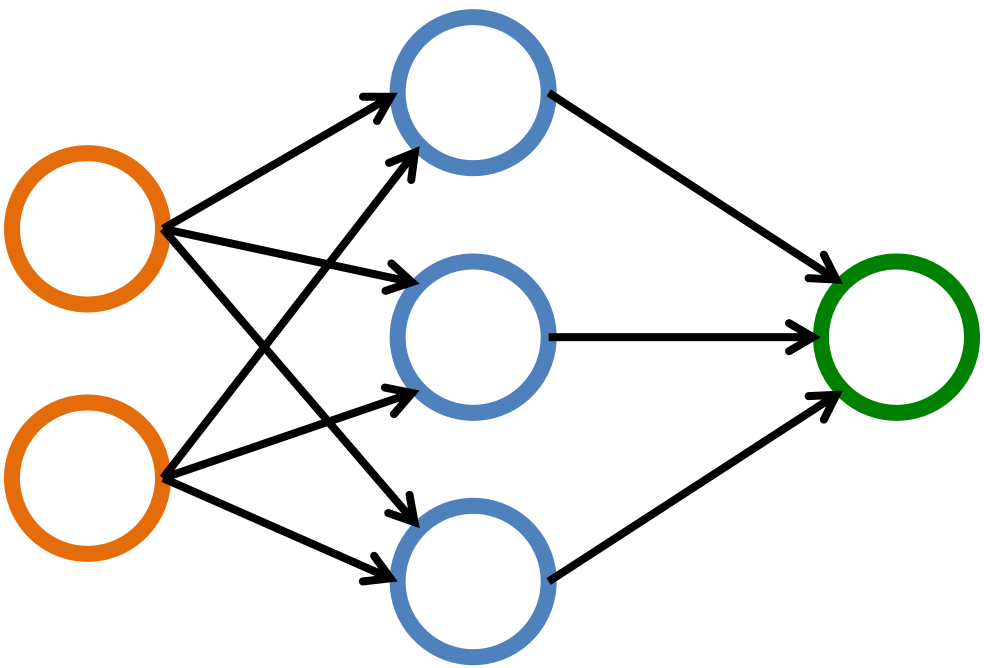
$$F(x) = \max(0, x)$$

$$y = max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$



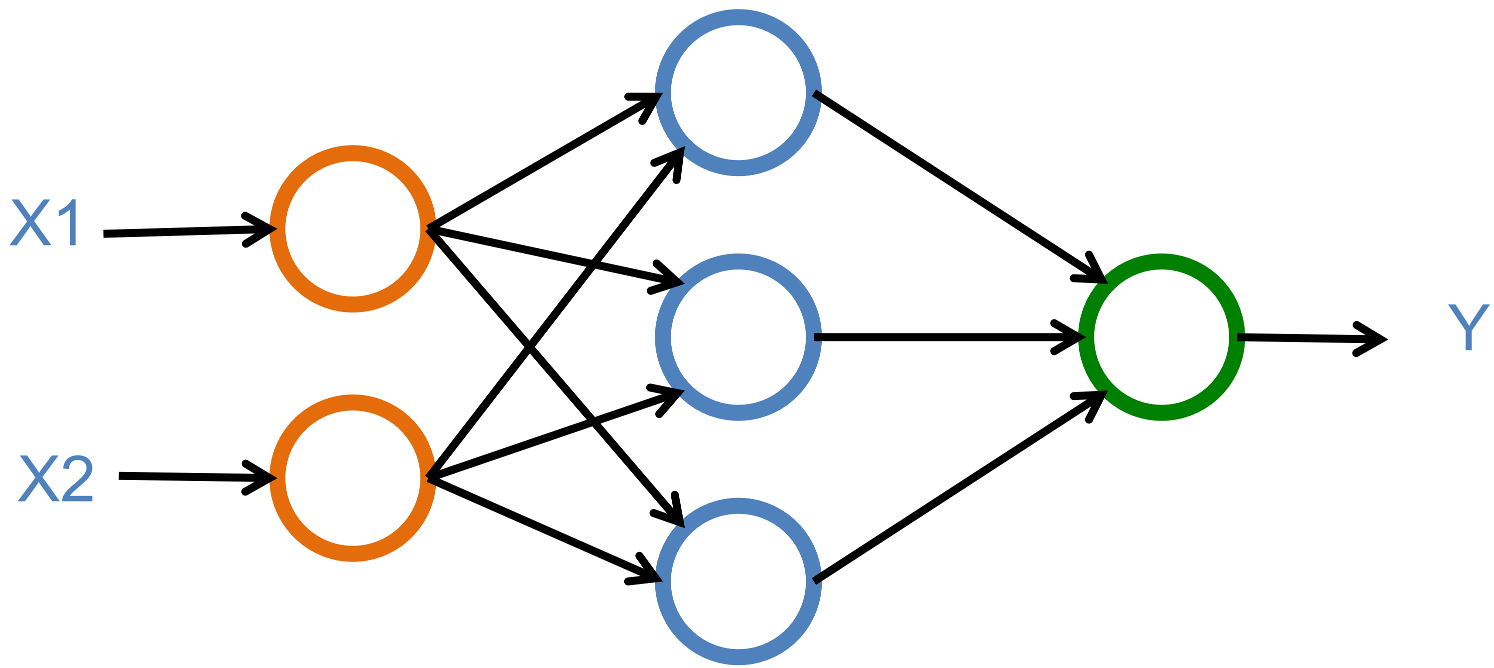
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Neural Networks



Neural Networks

Input Layer (X) | Hidden Layer (H) | **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

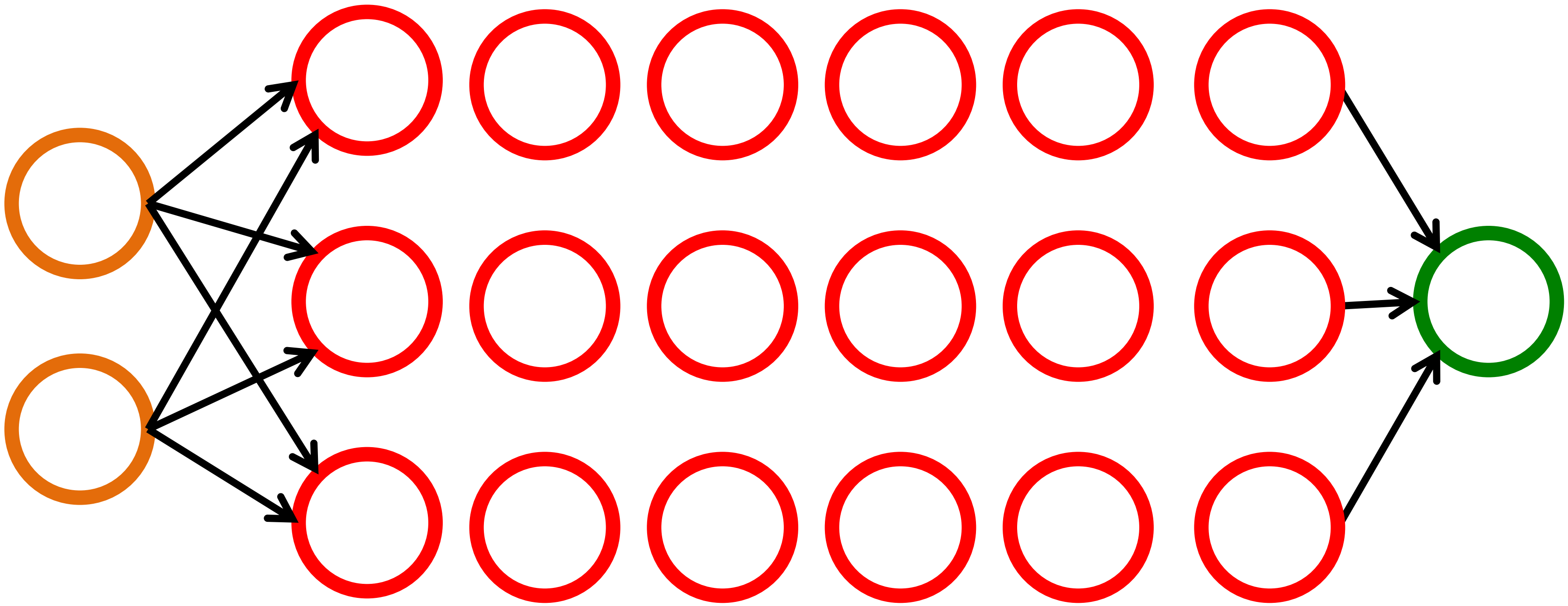
Neural Networks

Input Layer (X)

Hidden Layers (H)

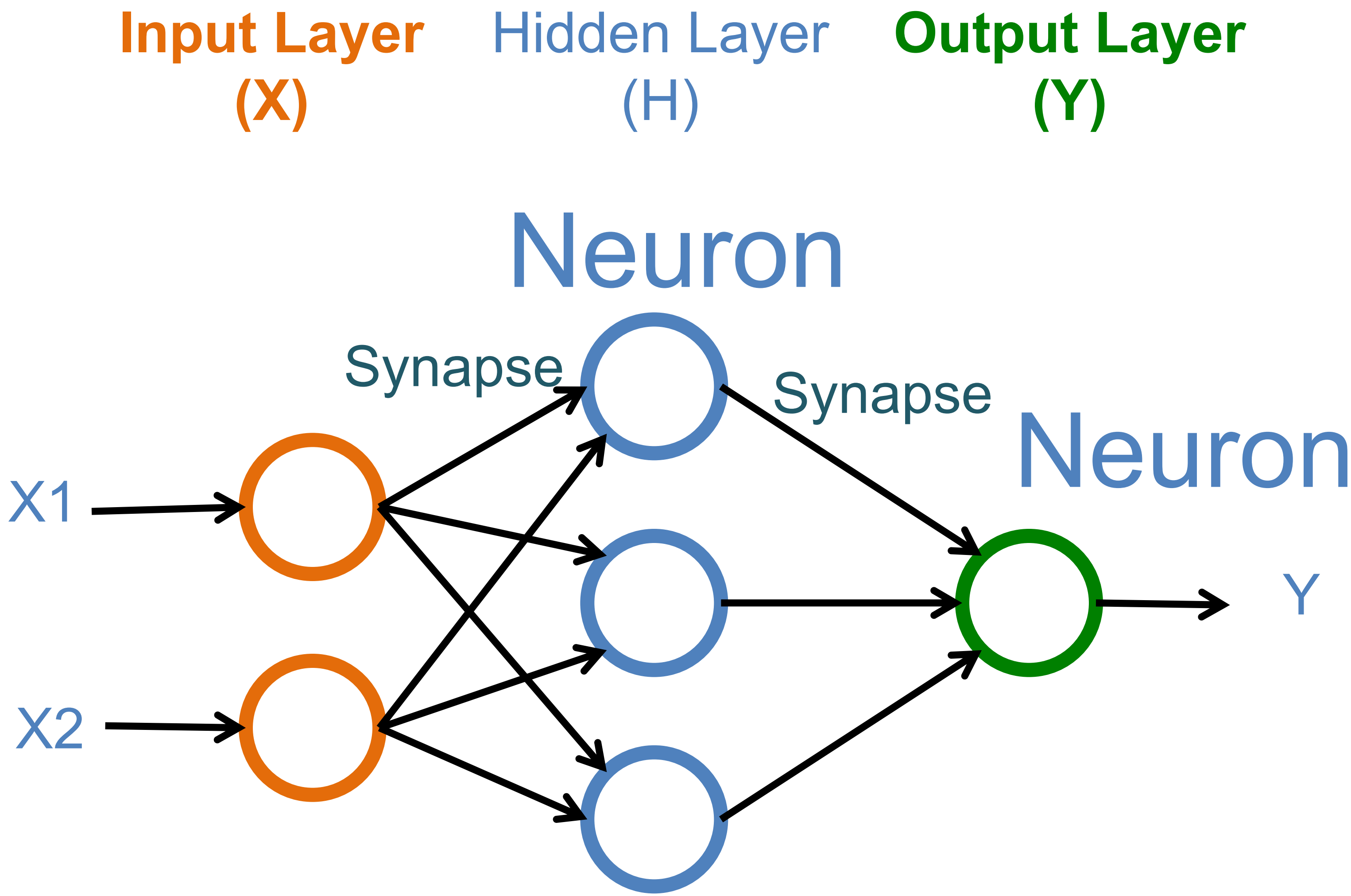
Output Layer (Y)

Deep Neural Networks
Deep Learning



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

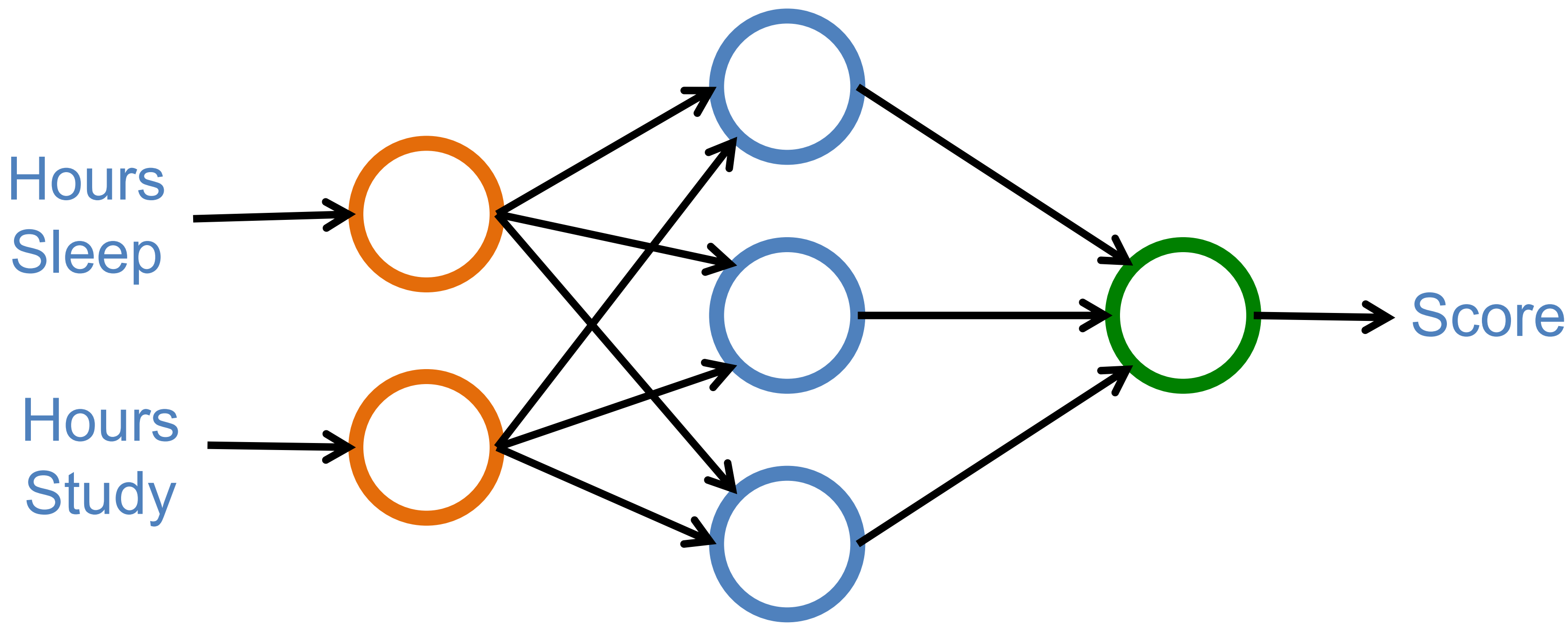
Neural Networks



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X) | Hidden Layer (H) | **Output Layer (Y)**



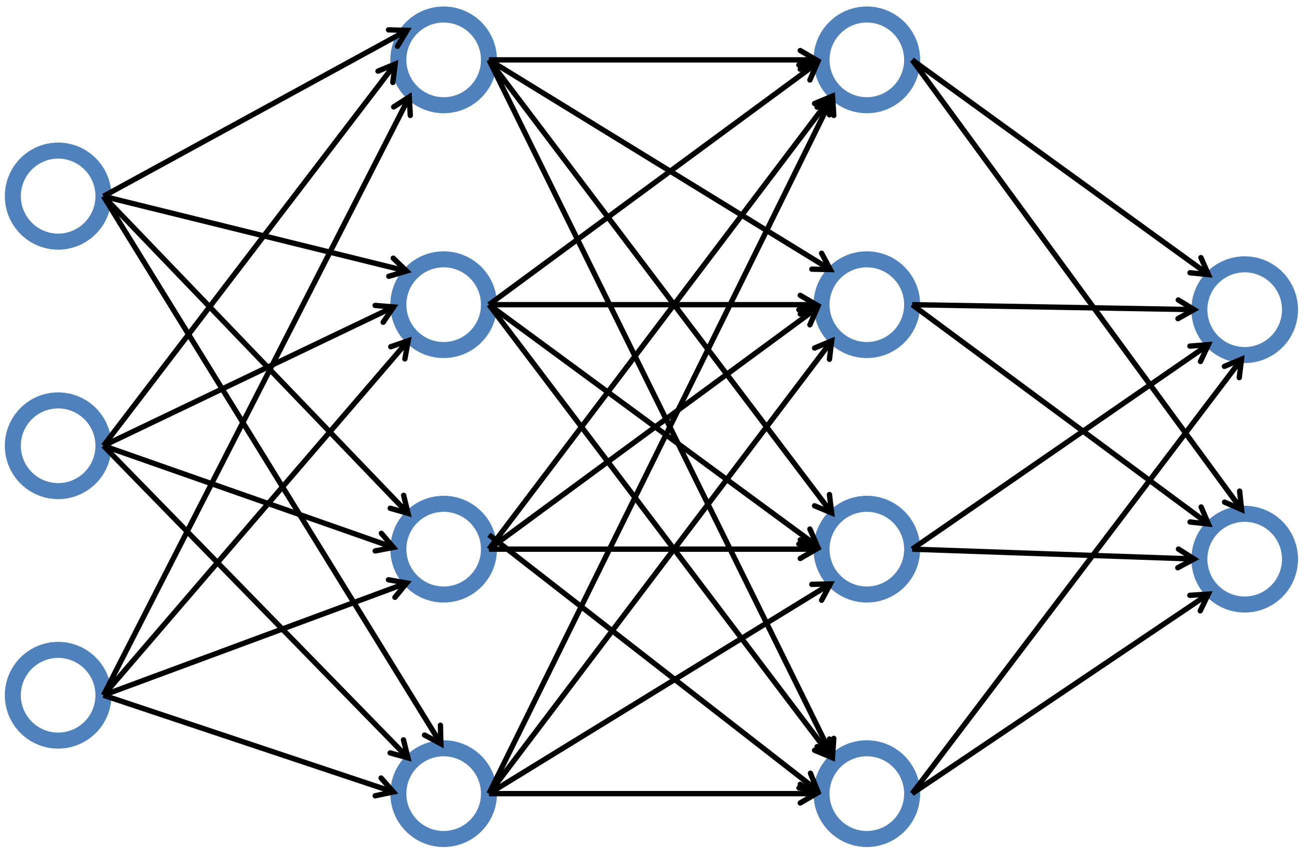
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X)

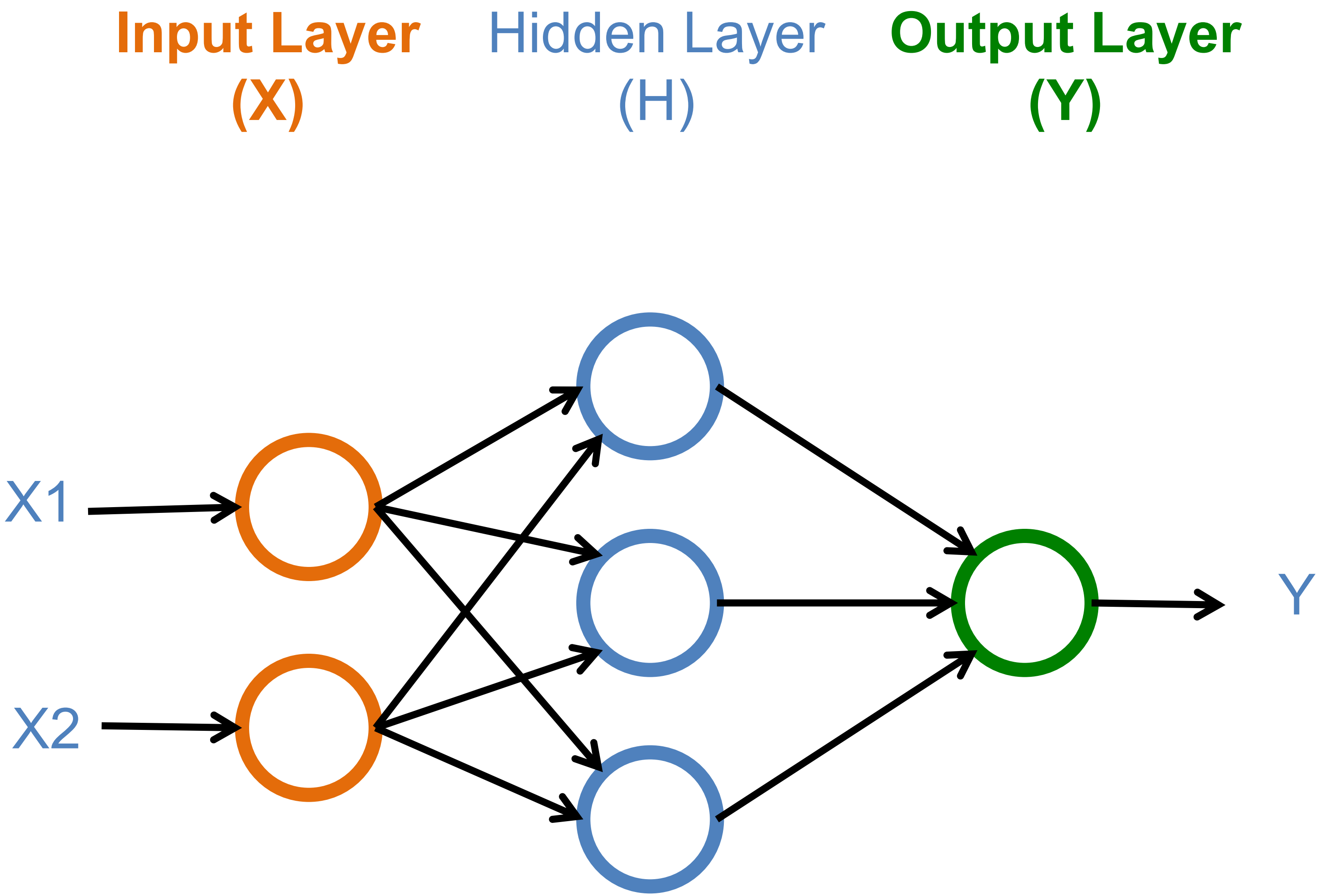
Hidden Layer (H)

Output Layer (Y)



Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2

Neural Networks



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

X		Y
Hours Sleep	Hours Study	Score
3	5	75
5	1	82
10	2	93
8	3	?

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

	X		Y
	Hours Sleep	Hours Study	Score
Training	3	5	75
	5	1	82
	10	2	93
Testing	8	3	?

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

$$Y = W X + b$$

Output

input

$$Y = W\,X + b$$

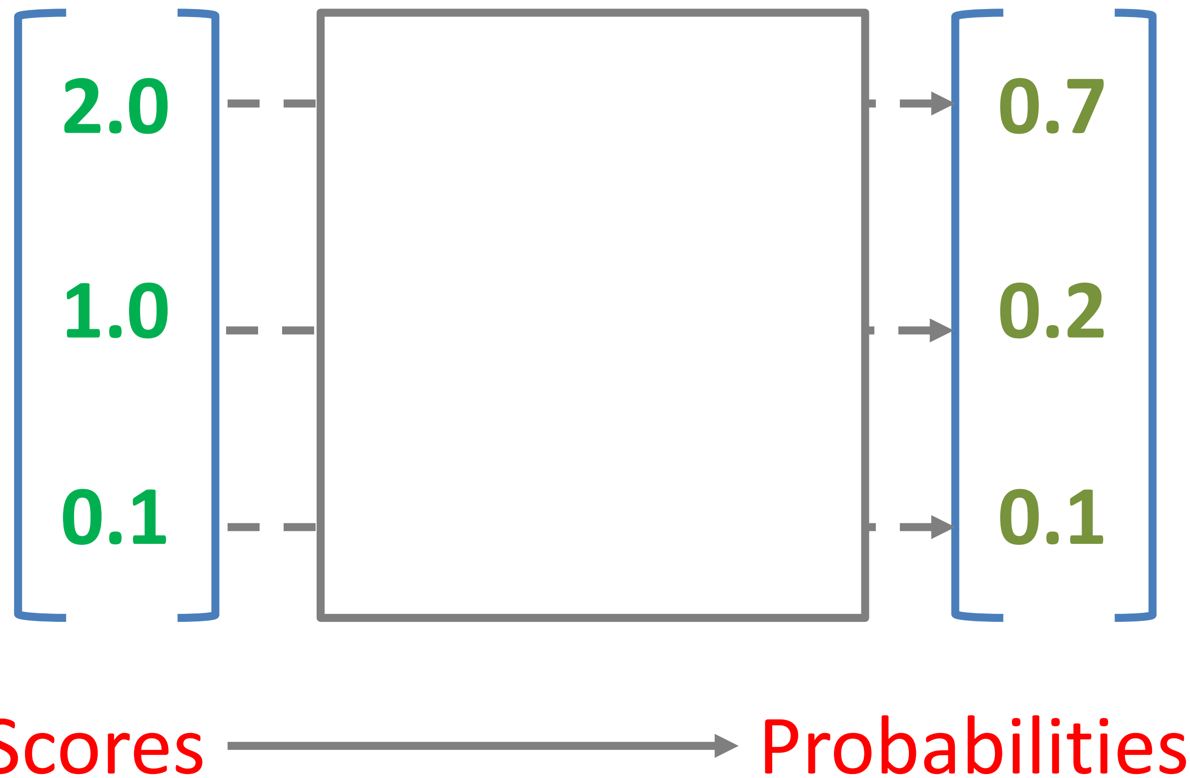
Weights

bias

Trained

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

$$WX + b = Y$$



Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

SoftMAX

$$\mathbf{W}\ \mathbf{X} + \mathbf{b} = \mathbf{Y} \qquad \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \rightarrow S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \rightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Logits Scores → Probabilities

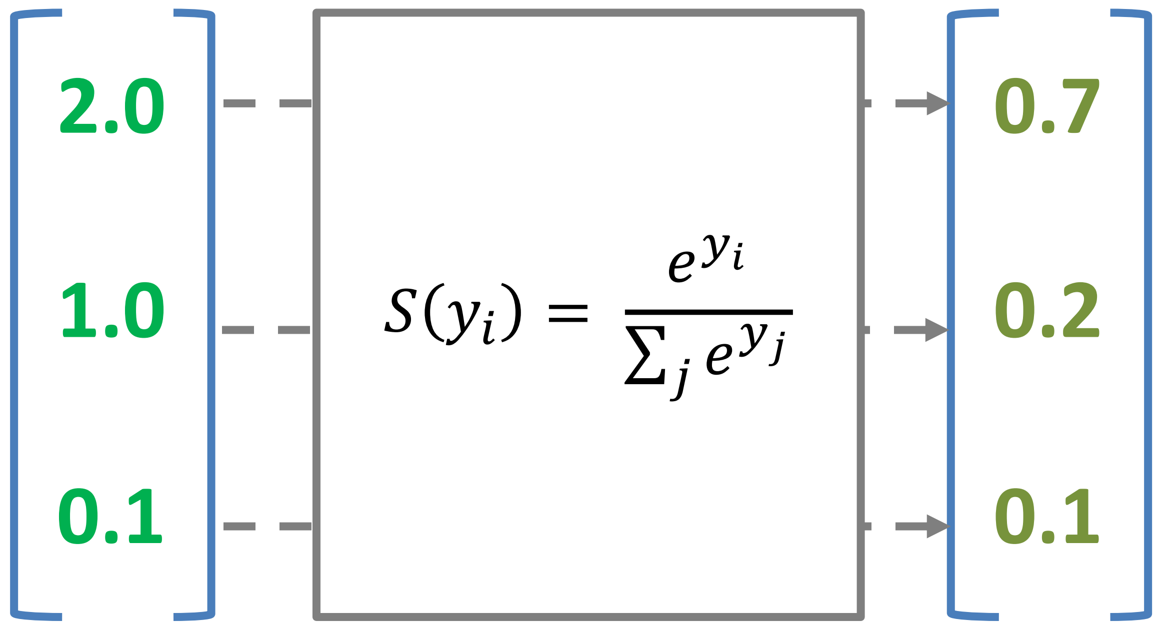
Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}} = \frac{e^{2.0}}{e^{2.0}+e^{1.0}+e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0}+2.7182^{1.0}+2.7182^{0.1}} = 0.7$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{1.0}}{e^{2.0}+e^{1.0}+e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0}+2.7182^{1.0}+2.7182^{0.1}} = 0.2$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{0.1}}{e^{2.0}+e^{1.0}+e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0}+2.7182^{1.0}+2.7182^{0.1}} = 0.1$$

W X + b = Y



Logits

Scores → Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

Training a Network

=

Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Training a Network

=

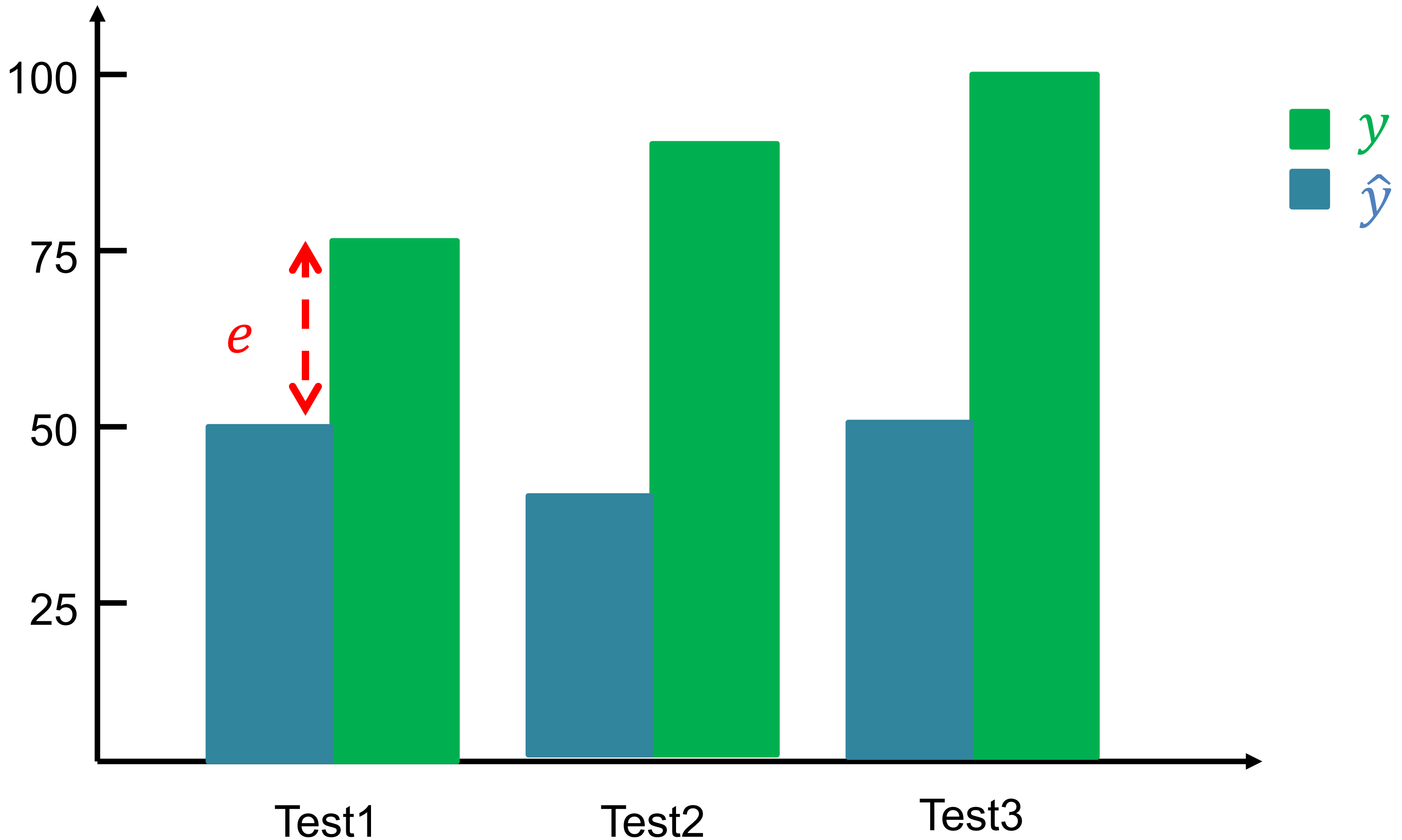
Minimize the Cost Function

Minimize the Loss Function

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Error = Predict Y - Actual Y

Error : Cost : Loss



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

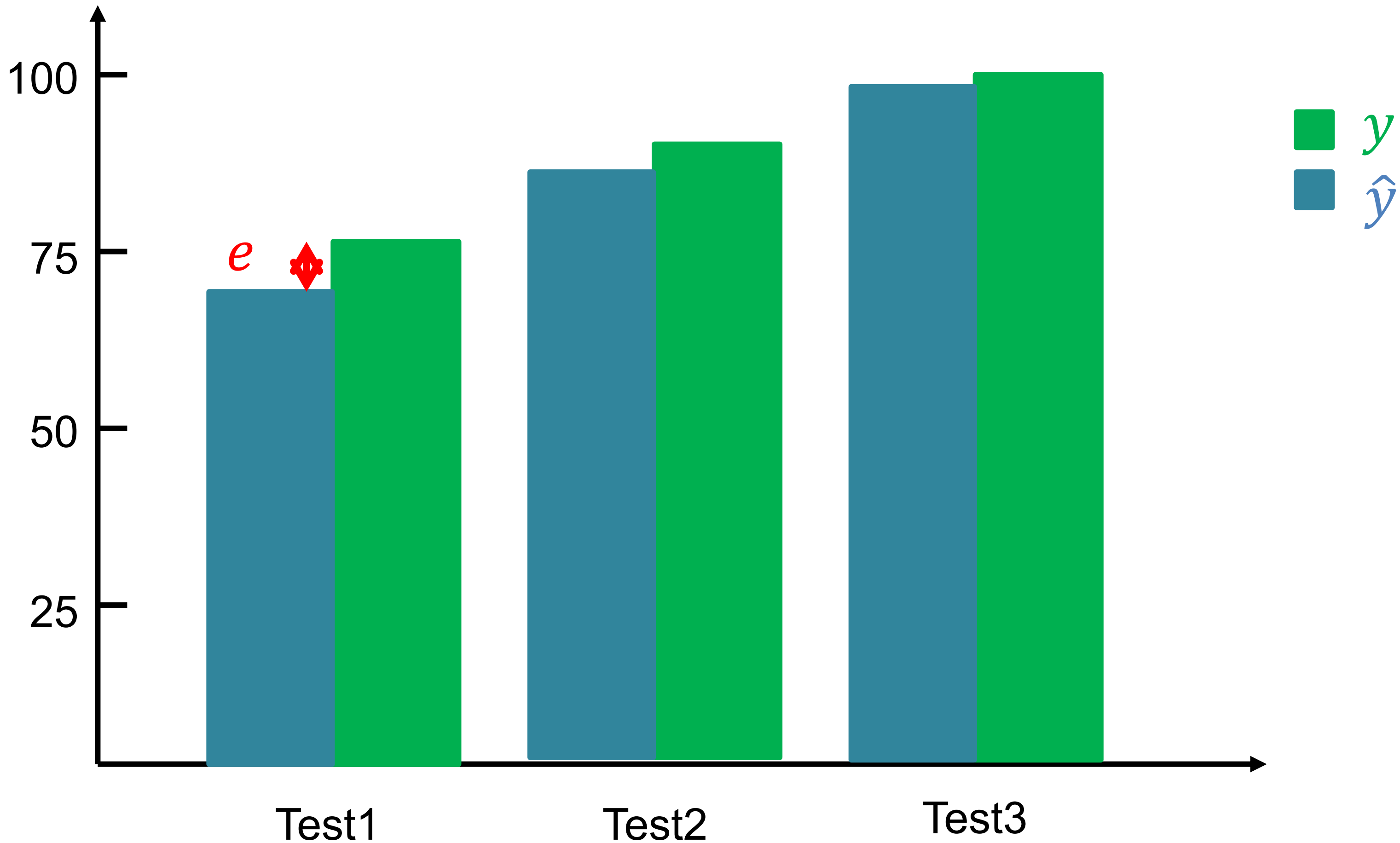
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Error = Predict Y - Actual Y

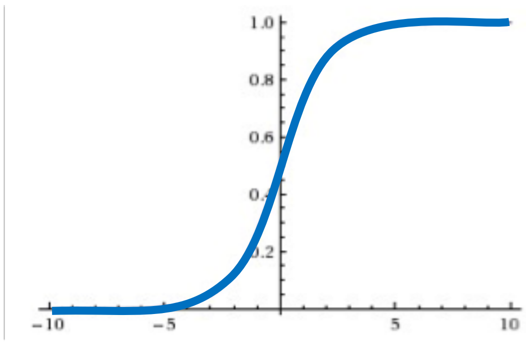
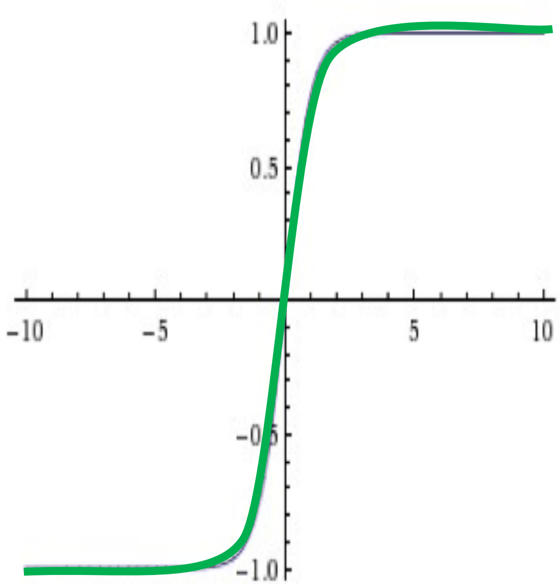
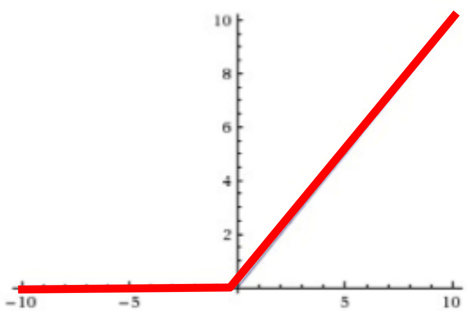
Error : Cost : Loss



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

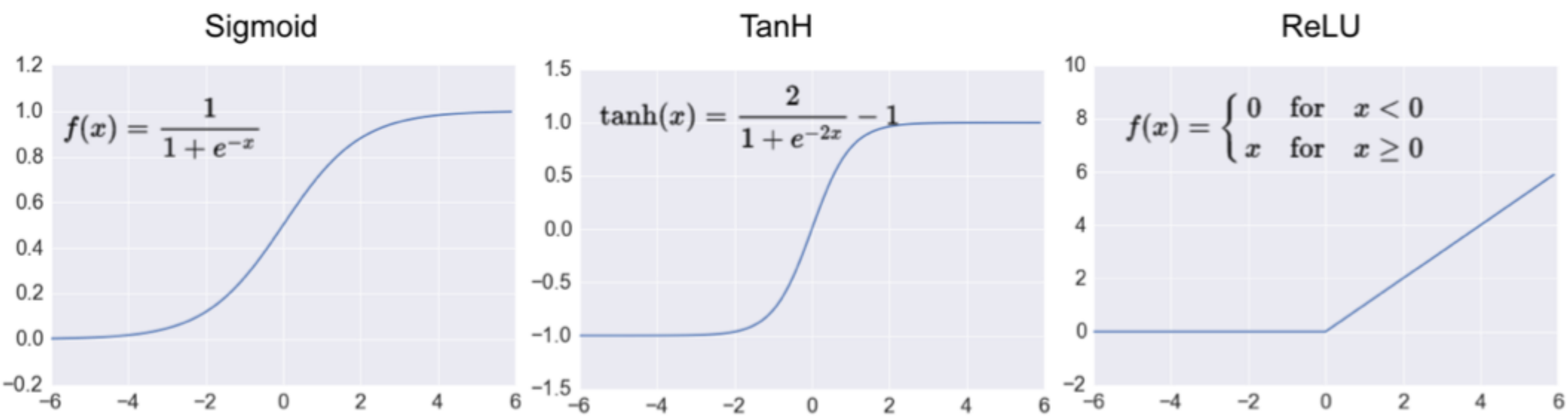
Activation Functions

Activation Functions

Sigmoid	TanH	ReLU (Rectified Linear Unit)
		
[0, 1]	[-1, 1]	$f(x) = max(0, x)$

Source: http://cs231n.github.io/neural-networks-1/

Activation Functions



Source: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Loss Function

Binary Classification: 2 Class

Activation Function:
Sigmoid

Loss Function:
Binary Cross-Entropy

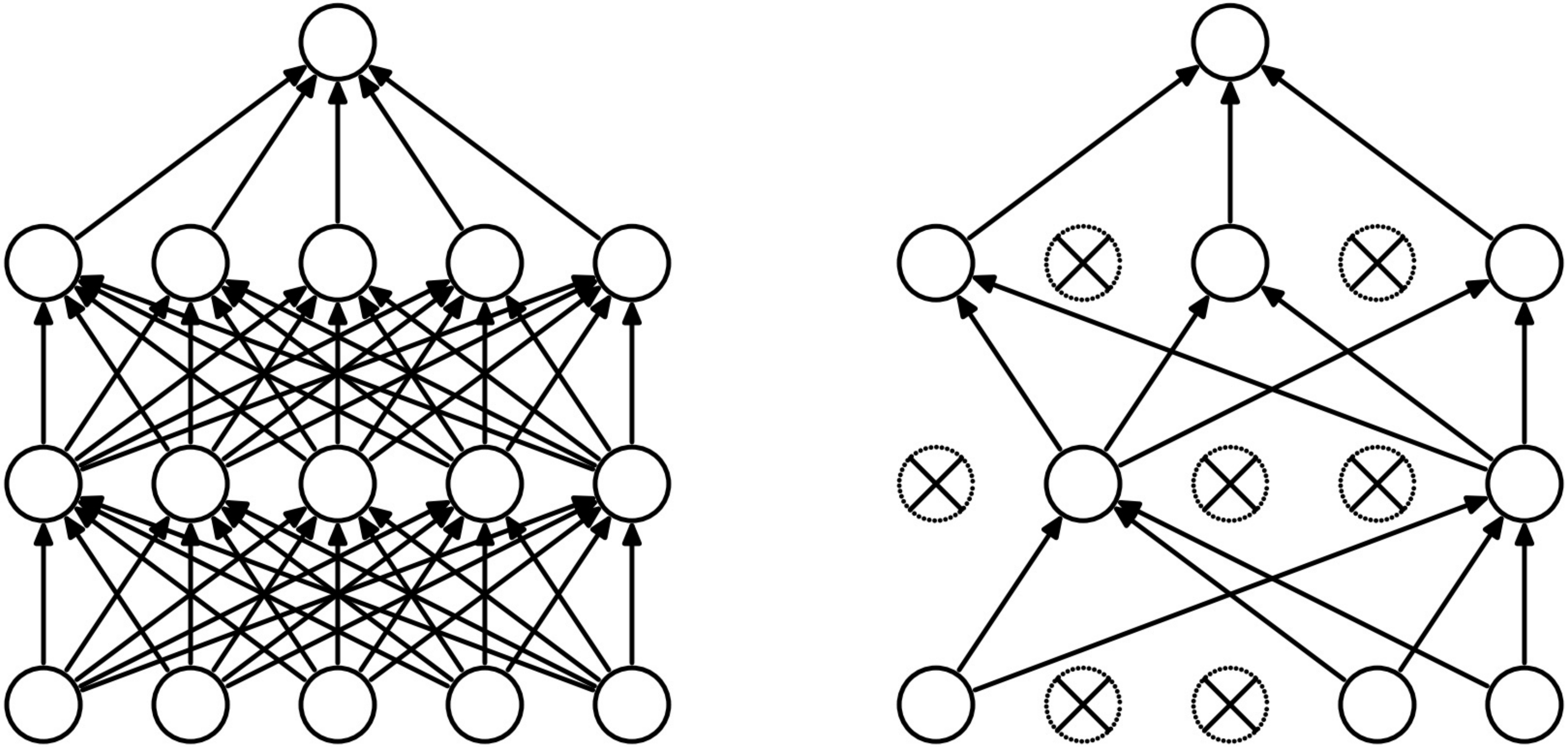
Multiple Classification: 10 Class

Activation Function:
SoftMAX

Loss Function:
Categorical Cross-Entropy

Dropout

Dropout: a simple way to prevent neural networks from overfitting



(a) Standard Neural Net (b) After applying dropout.

Source: Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* 15, no. 1 (2014): 1929-1958.

Learning Algorithm

While not done:

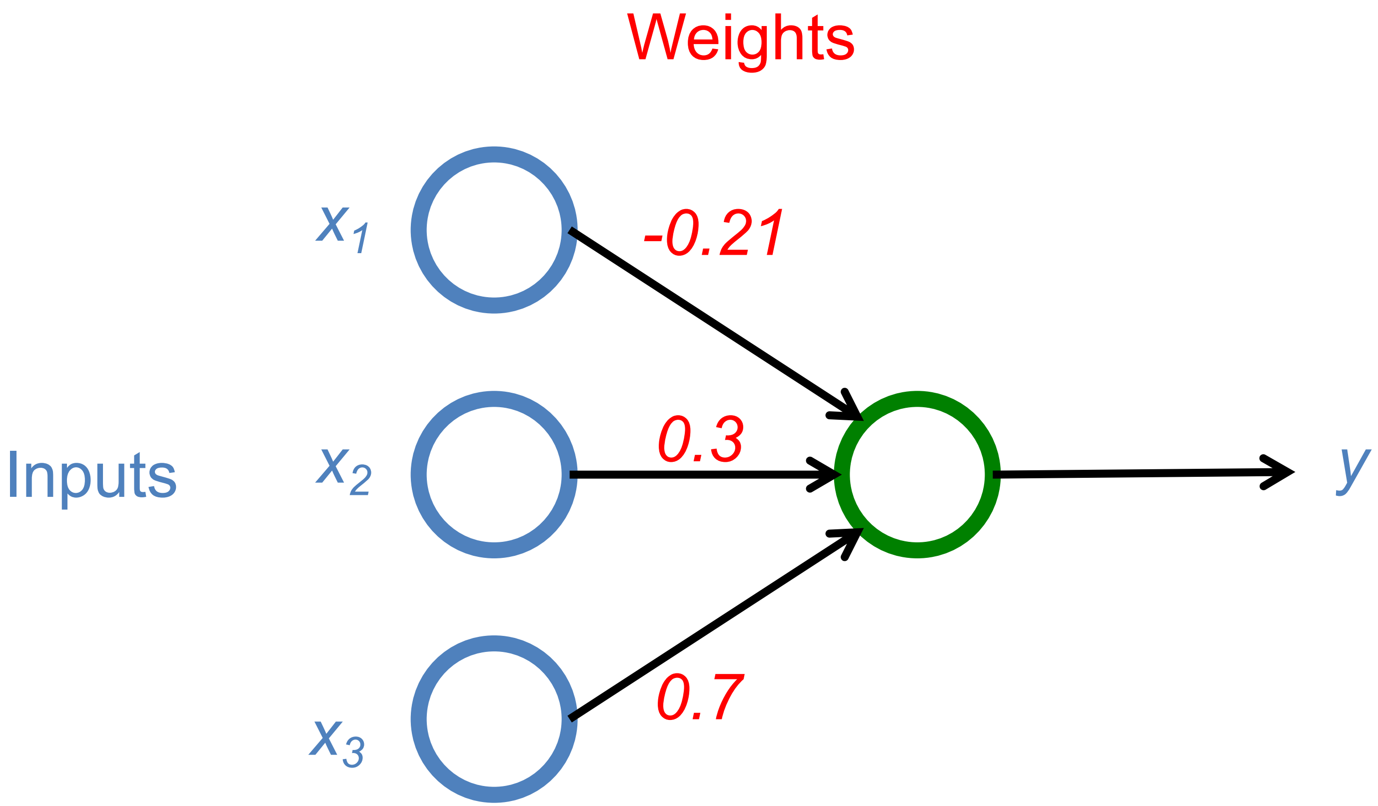
Pick a random training example "(input, label)"

Run neural network on "input"

Adjust weights on edges to make output closer to "label"

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

$$y = \max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$

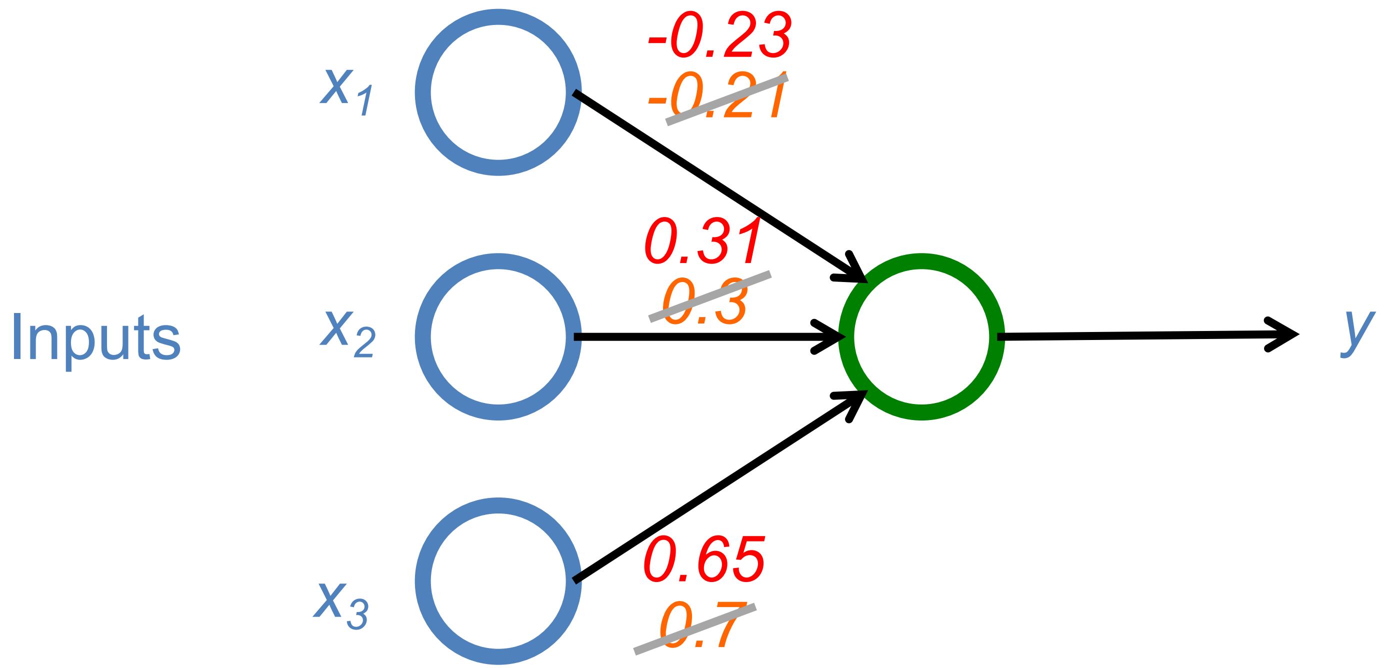


Next time:

$$y = max\,(\,0, -0.23 * x_1 + 0.31 * x_2 + 0.65 * x_3\,)$$

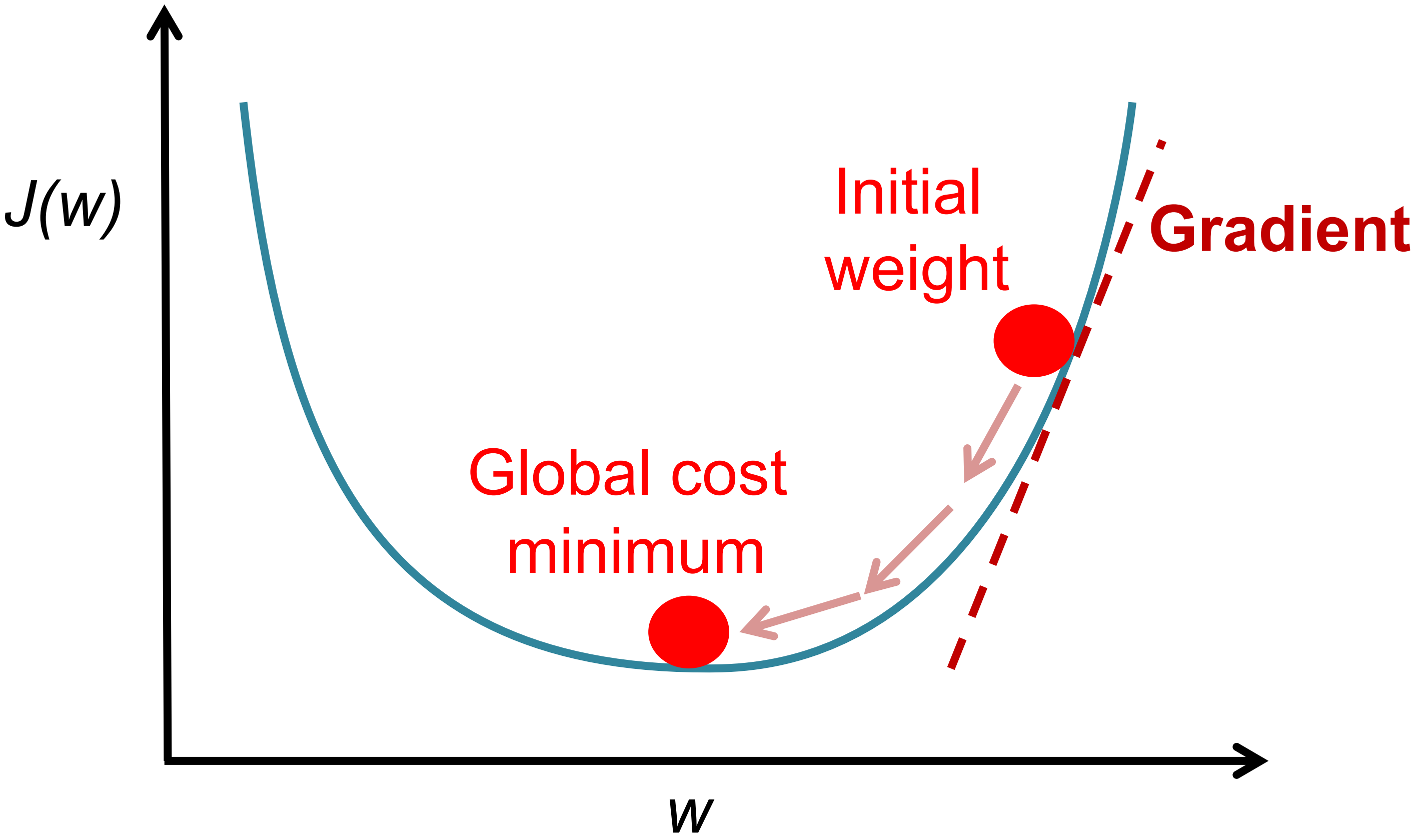
$$\cancel{y = max\,(\,0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3\,)}$$

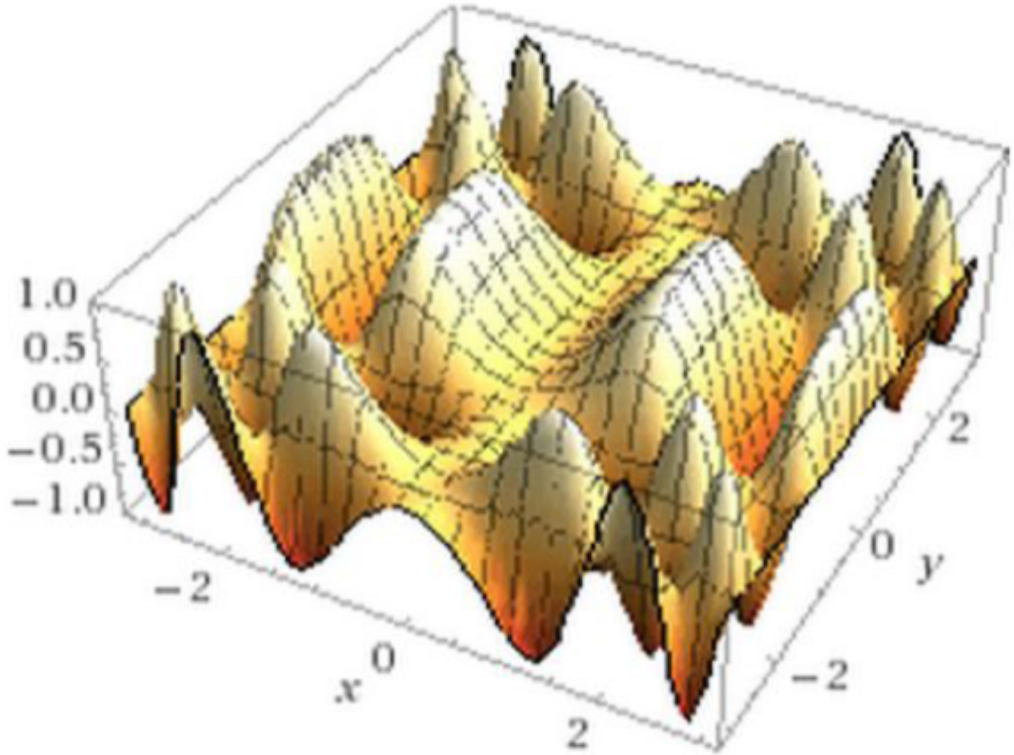
Weights



Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Optimizer:
Stochastic Gradient Descent (SGD)

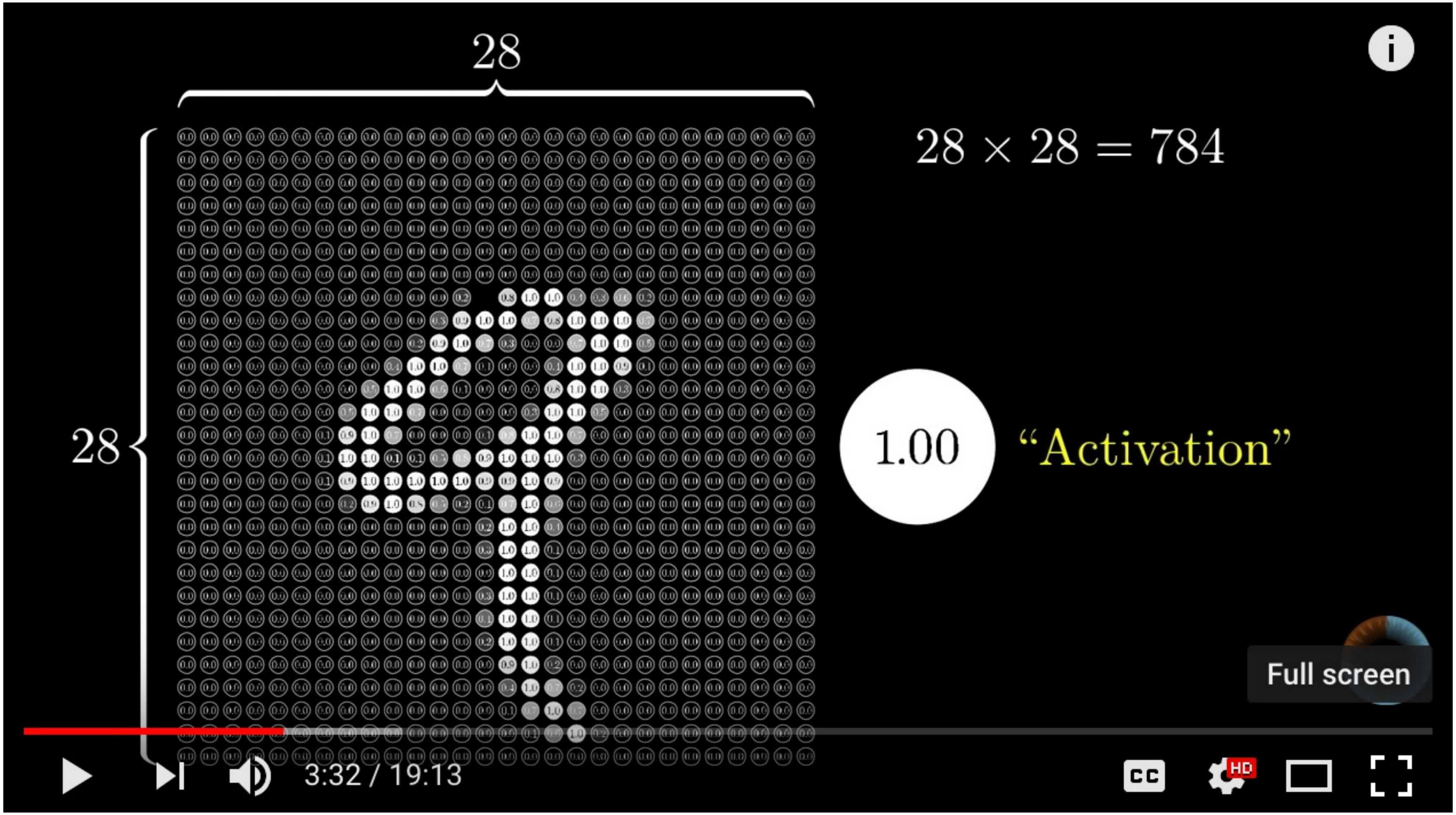




This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

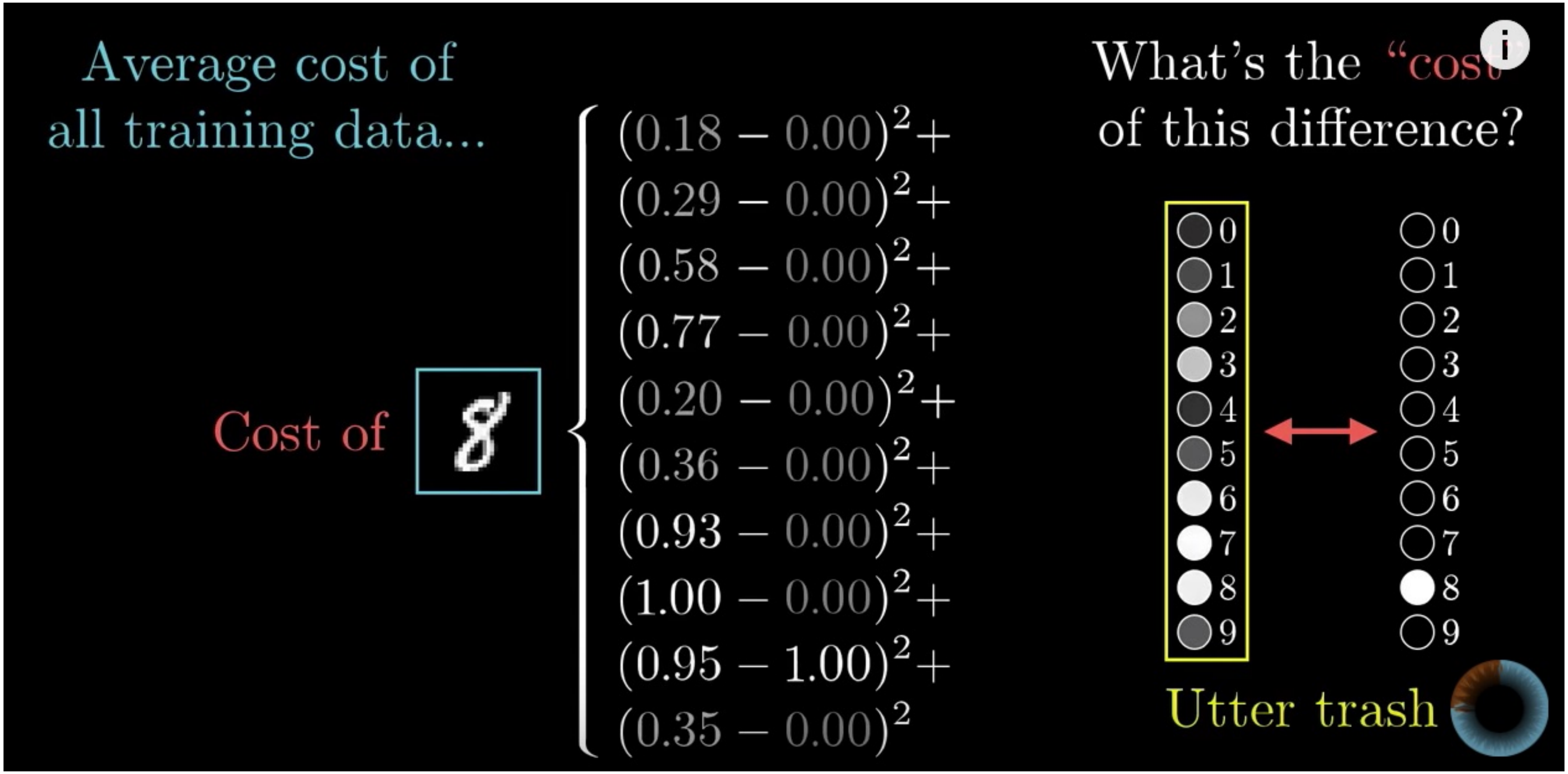
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Neural Network and Deep Learning



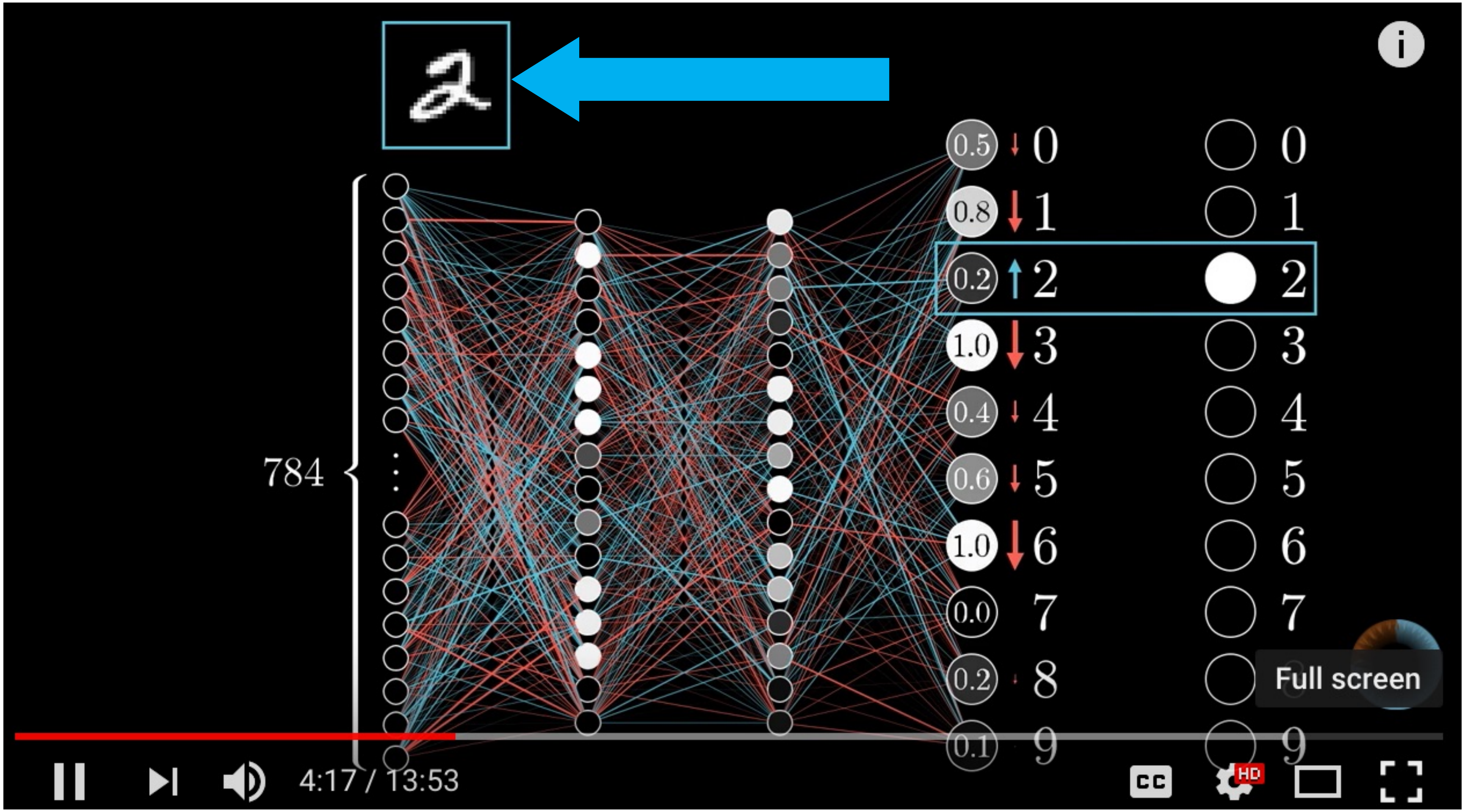
Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk

Gradient Descent
how neural networks learn



Source: 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w

Backpropagation



Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U

Learning Algorithm

While not done:

Pick a random training example “(input, label)”

Run neural network on “input”

Adjust weights on edges to make output closer to “label”

Deep Learning for Financial Time Series Forecasting

Deep Learning
for
Financial Market Prediction
Stock Market Prediction
Stock Price Prediction
Time Series Prediction

Time Series Data

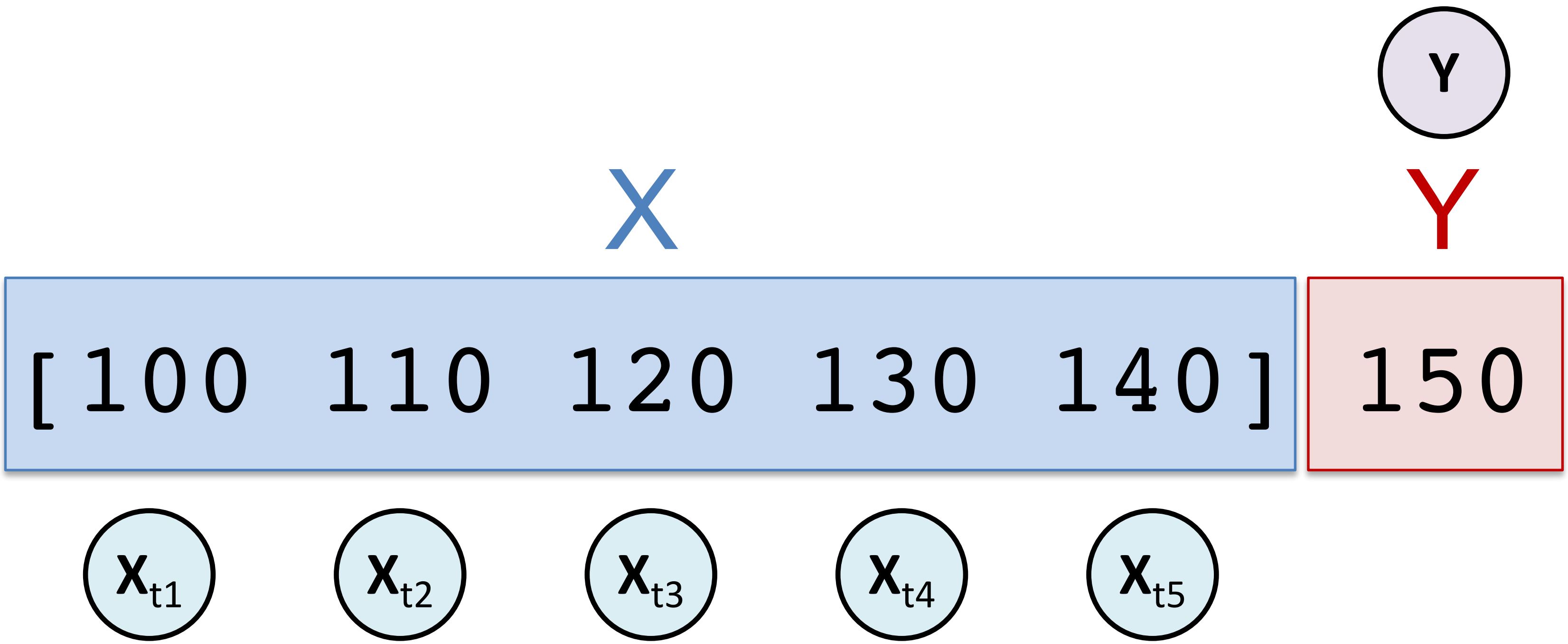
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```

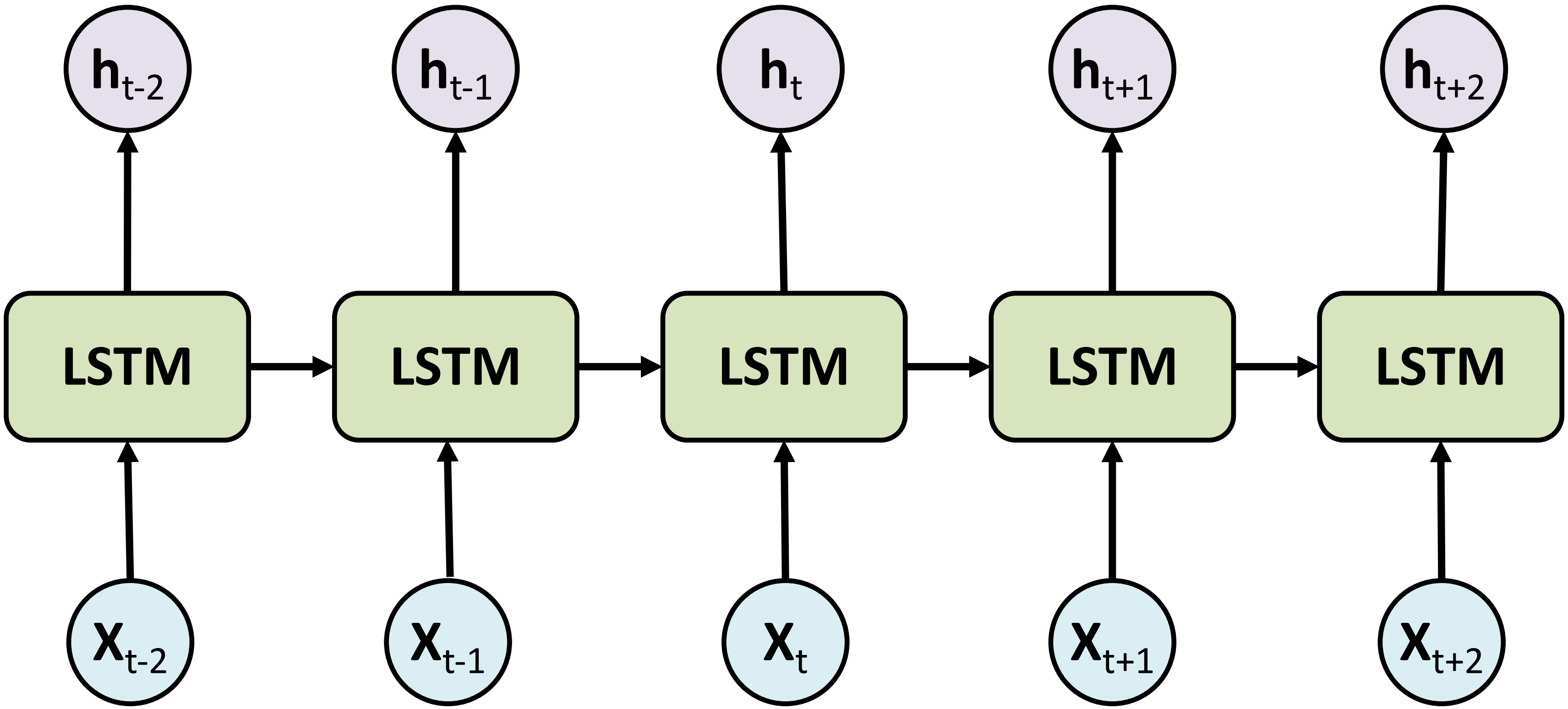
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Time Series Data

[100, 110, 120, 130, 140, 150]



Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

X	Y
[10 20 30]	40
[20 30 40]	50
[30 40 50]	60
[40 50 60]	70
[50 60 70]	80
[60 70 80]	90

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

```python
# univariate data preparation
from numpy import array
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# summarize the data
for i in range(len(X)):
    print(X[i], y[i])
```

```
[10 20 30] 40
[20 30 40] 50
[30 40 50] 60
[40 50 60] 70
[50 60 70] 80
[60 70 80] 90
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

CO Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECTED EDITING

LSTM for Time Series Forecasting

```python
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
%matplotlib inline

# define dataset
X = array([[100, 110, 120], [110, 120, 130], [120, 130, 140], [130, 140, 150], [140, 150, 160]])
y = array([130, 140, 150, 160, 170])
# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape((X.shape[0], X.shape[1], 1))
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(3, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
history = model.fit(X, y, epochs=2000, verbose=0)
# demonstrate prediction
x_input = array([150, 160, 170])
x_input = x_input.reshape((1, 3, 1))
yhat = model.predict(x_input, verbose=0)
print('yhat', yhat)
print(model.summary())
# list all data in history
print(history.history.keys())
# summarize history for loss
print('loss:', '%f'%history.history['loss'][-1])
print('loss:', history.history['loss'][-1])
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

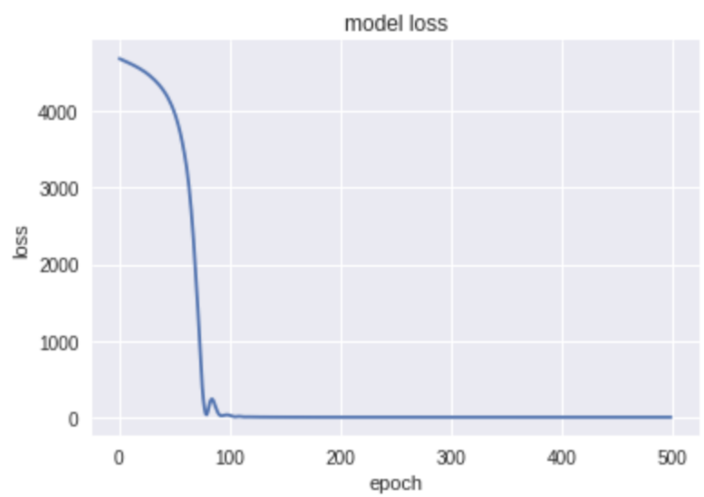
```
yhat [[181.34615]]
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

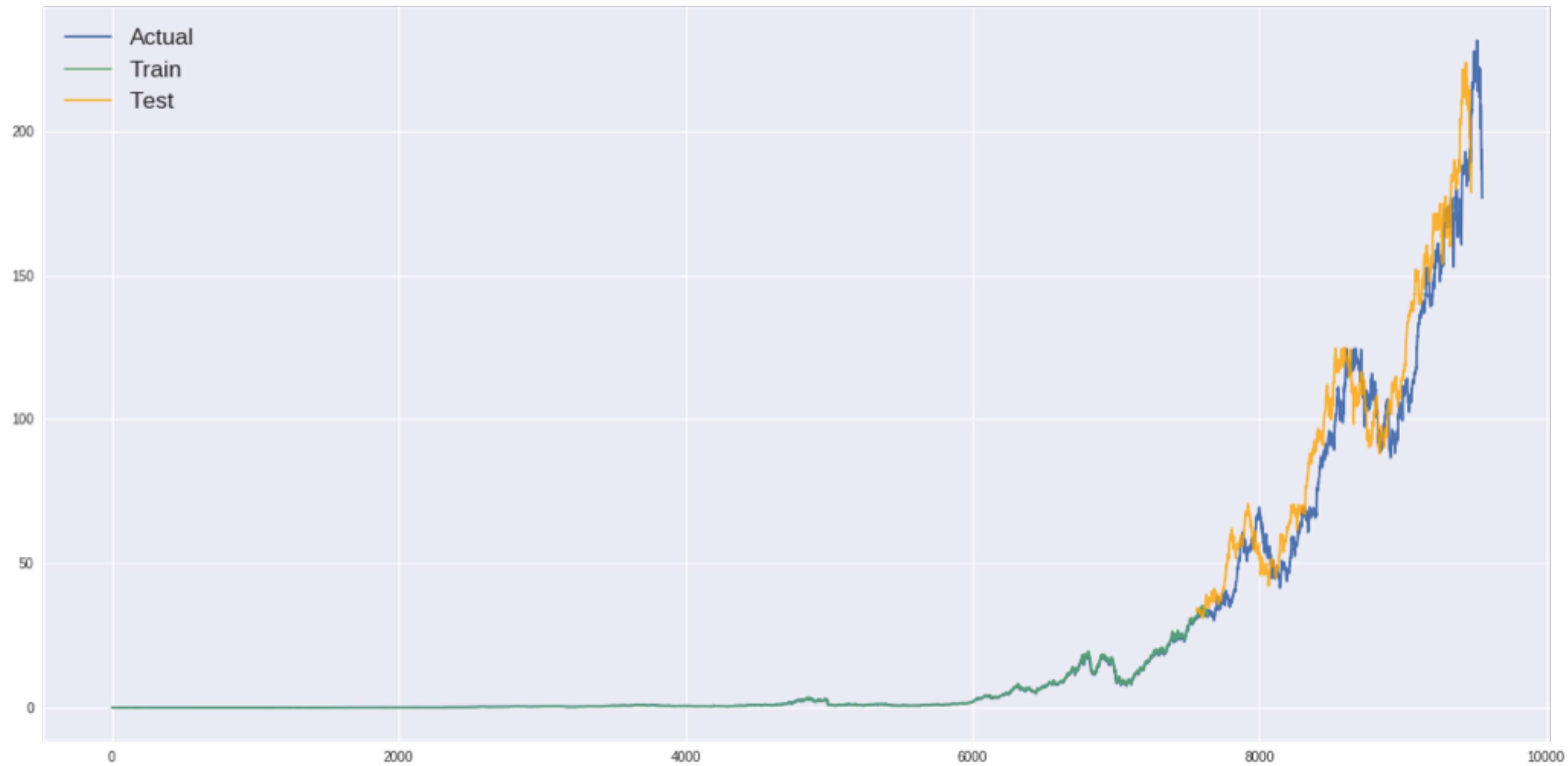
```
Using TensorFlow backend.
[[102.31296]]
yhat [[102.31296]]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50)                10400
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
None
dict_keys(['loss'])
loss: 0.000000
loss: 1.2578432517784677e-07
```



Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM



Source: https://github.com/yash-1337/AAPL_LSTM_Stock_Predictor/blob/master/AAPL_daily_LSTM_stock_predictor.ipynb

Basic Classification

Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjcutlzNHjRSLbeVl4kd5z

tf01_basic_classification.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Table of contents | Code snippets | Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ *2 cells hidden*

Train your first neural network: basic classification

View on TensorFlow.org | Run in Google Colab | View source on GitHub

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

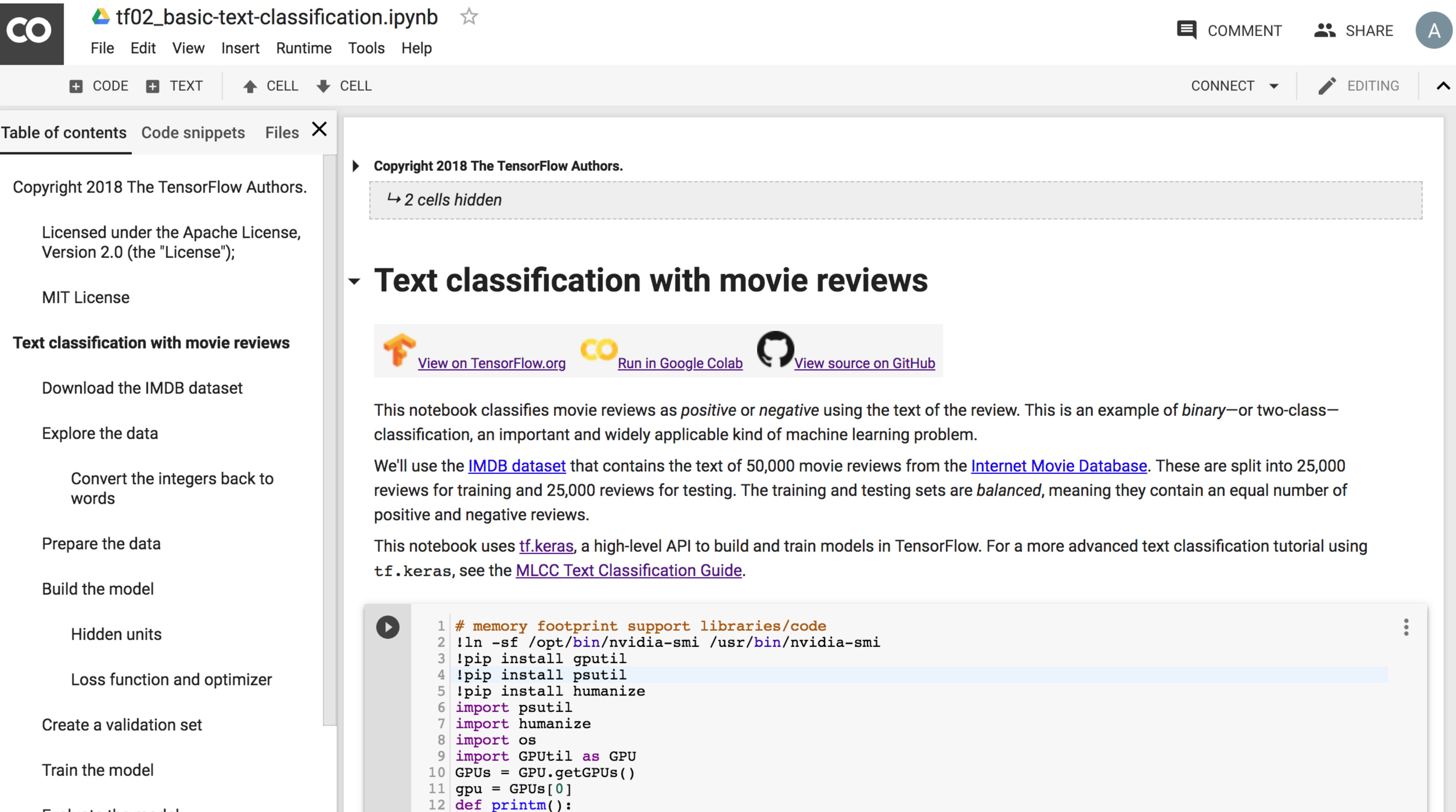
```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Pro
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format
printm()
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb

Text Classification

IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLIrLYtPCvCHaoO1W-i_gror



Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtgd2_25K_AURjR6SCVBRdlj

tf03_basic-regression.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECT EDITING

Table of contents Code snippets Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ *2 cells hidden*

Predict house prices: regression

View on TensorFlow.org Run in Google Colab View source on GitHub

In a *regression* problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a *classification* problem, where we aim to predict a discrete label (for example, where a picture contains an apple or an orange).

This notebook builds a model to predict the median price of homes in a Boston suburb during the mid-1970s. To do this, we'll provide the model with some data points about the suburb, such as the crime rate and the local property tax rate.

This example uses the `tf.keras` API, see this guide for details.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: "
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format(gpu.memo
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

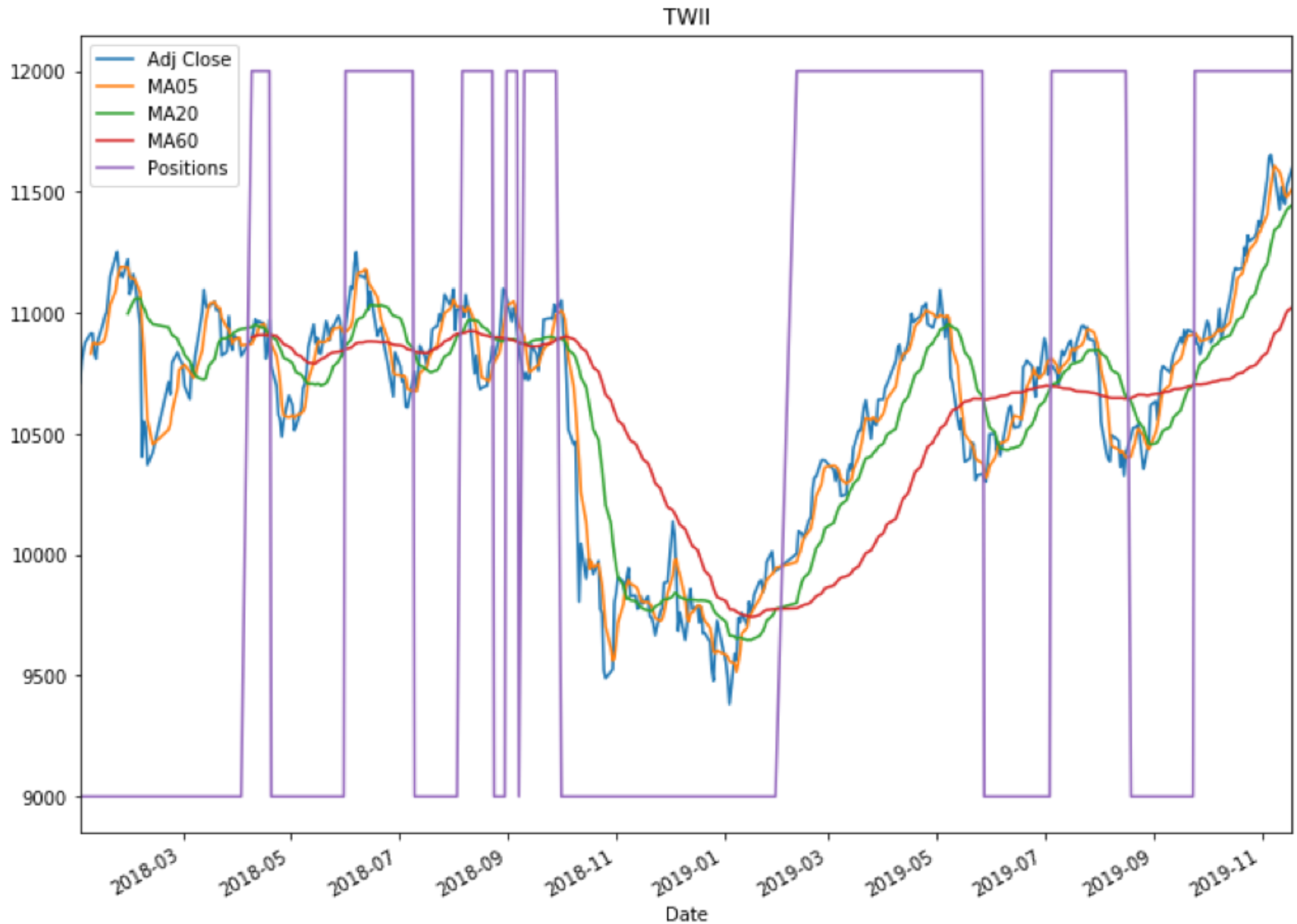
+ Code + Text

RAM Disk Editing

```python
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(2017, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
print(df.tail())
df2 = pd.read_csv('AAPL.csv') #df.from_csv('AAPL.csv')
print(df2.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)
```

https://tinyurl.com/imtkupython101

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



np.where (df['MA20'] > df['MA60'], 12000, 9000)

```
# 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
df['Positions'] = np.where(df['MA20'] > df['MA60'], 12000,  9000)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05
'],'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Posit
ions']})
```

```
np.where
(df['MA20'] > df['MA60’],
1,
0)
```

```
# 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
df['Positions'] = np.where(df['MA20'] > df['MA60’], 1,  0)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05
'],'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Posit
ions’]})
```

Summary

- Machine Learning
- Deep Learning

References

- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media, 2019, https://github.com/ageron/handson-ml2
- Yves Hilpisch (2018), "Python for Finance: Mastering Data-Driven Finance", 2nd Edition, O'Reilly Media. https://github.com/yhilpisch/py4fi2nd
- 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
- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Avinash Jain (2017), Introduction To Python Programming, Udemy, https://www.udemy.com/pythonforbeginnersintro/
- 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/
- Jason Brownlee (2018), How to Develop LSTM Models for Time Series Forecasting, https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.
- Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/
- Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow