深度學習Deep Learning (2)

112-1

朱學亭老師

課程大綱

- W1-課程介紹/Introduction
- W2-AICUP/Python/Colab and TensorFlow
- W3-Numpy/Pandas and PyTorch
- W4-Sklearn and 機器學習
- W5-神經網路, TensorFlow, PyTorch
- W6-載客熱點預測
- W7-自動光學檢查(AOI)-1
- W8-自動光學檢查(AOI)-2
- W10-RNN
- W11-GAN
- W12-Yolo
- W13-NLP1-Word2Vec
- W14-NLP2-Seq2Seq,Atention
- W15-NLP3-Transformer,BERT
- W16-AICUP 1
- W17-AICUP 2
- W18-Final presentation

大綱

- AICUP
- Python & Colab
- TensorFlow

(1) AICUP

秋季賽 2023.9~12

https://www.aicup.tw/copy-of-ai-cup-2023-1

2023 AI CUP

2023/9/1-2023/11/20

讓AI成為棋靈王

—— 圍棋棋力模仿與棋風辨識競賽 ——

競賽介紹

競賽說明

由於下圍棋是人類的智慧行為，模仿各種階段的圍棋棋力可以說就是**模仿各種階段的人類的智慧行為**，未來元宇宙的發展中，將會需要設計許多非人類角色(NPC)，增加元宇宙的豐富

需運用深度學習如CNN、ResNet、Transformer，或採用meta learning、few-shot learning、強化學習如MCTS、Alpha Zero等進階技術，實現AI方法廣度與深度

報名規範

年齡

年滿18歲皆可報名參加
未滿18歲則需經監護人同意

人數

參賽隊伍人數1~4人

報名方式

STEP1 每位隊員於『T-Brain AI實戰吧』註冊會員

STEP2 至『AI CUP報名系統』登入報名

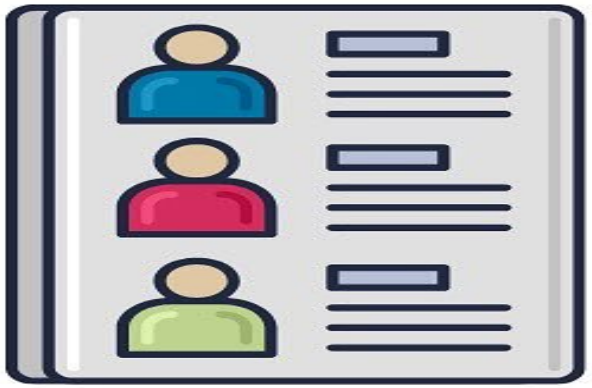
STEP3 可於隔日在『T-Brain AI實戰吧』參與競賽

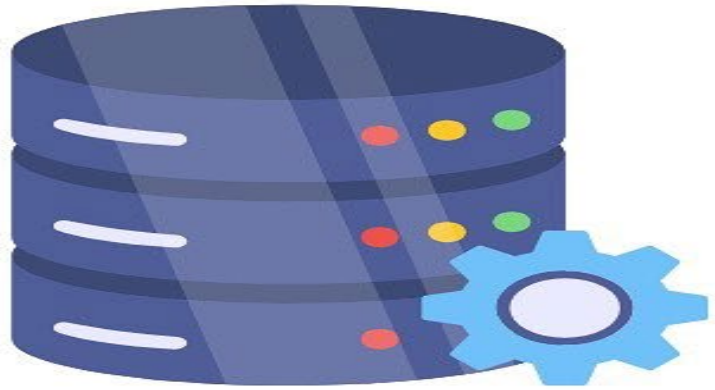
注意事項

『T-Brain AI實戰吧』與報名頁面填寫之
Emai需相符，否則將會報名失敗

電子健康紀錄
(electronic health record, EHR)

😇EHR好處😇

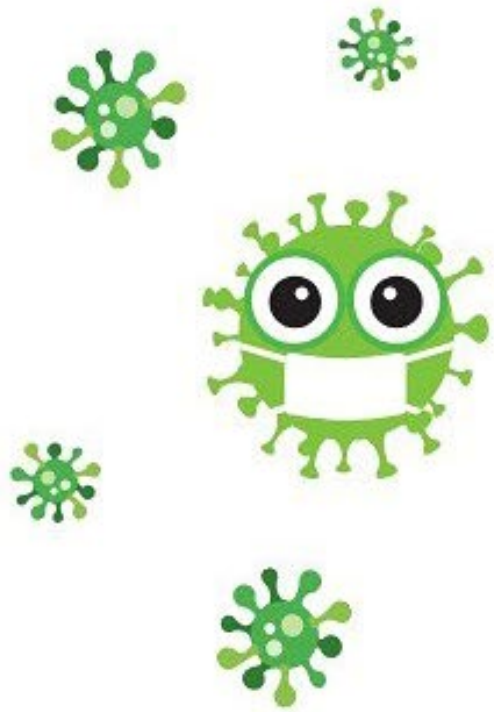




紙本的醫療訊息轉成數位化儲存

多靠抗体，

一些疾病死亡率降低了不少

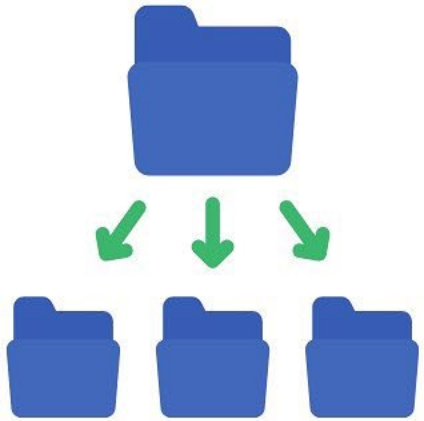


EHR壞處

EHR介面複雜、不同醫療服務無法銜接

患者病情無法準確預測

結論



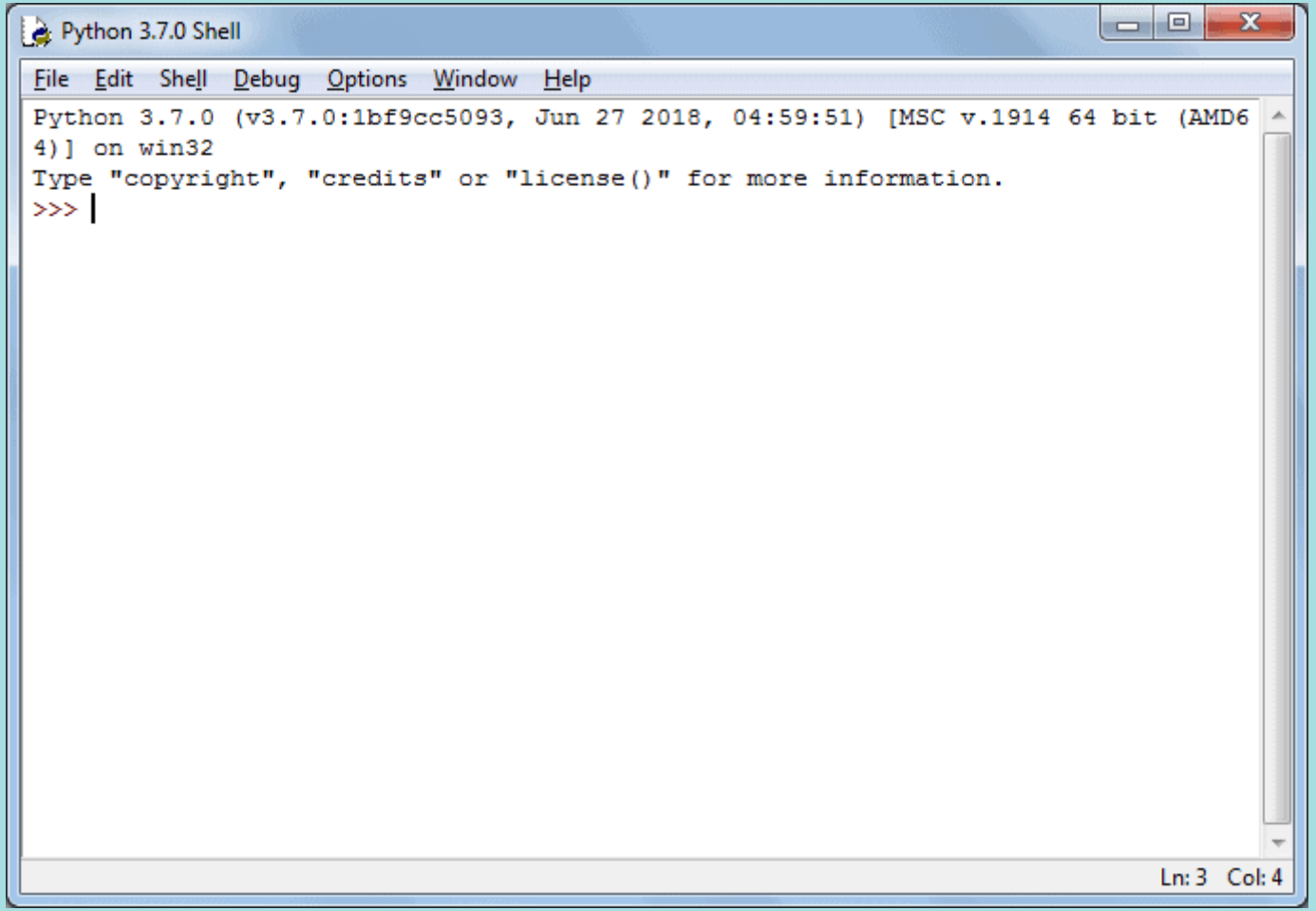
使用 EHR 資料訓練演算法前
要先將資料分類整理

(2) PYTHON & COLAB

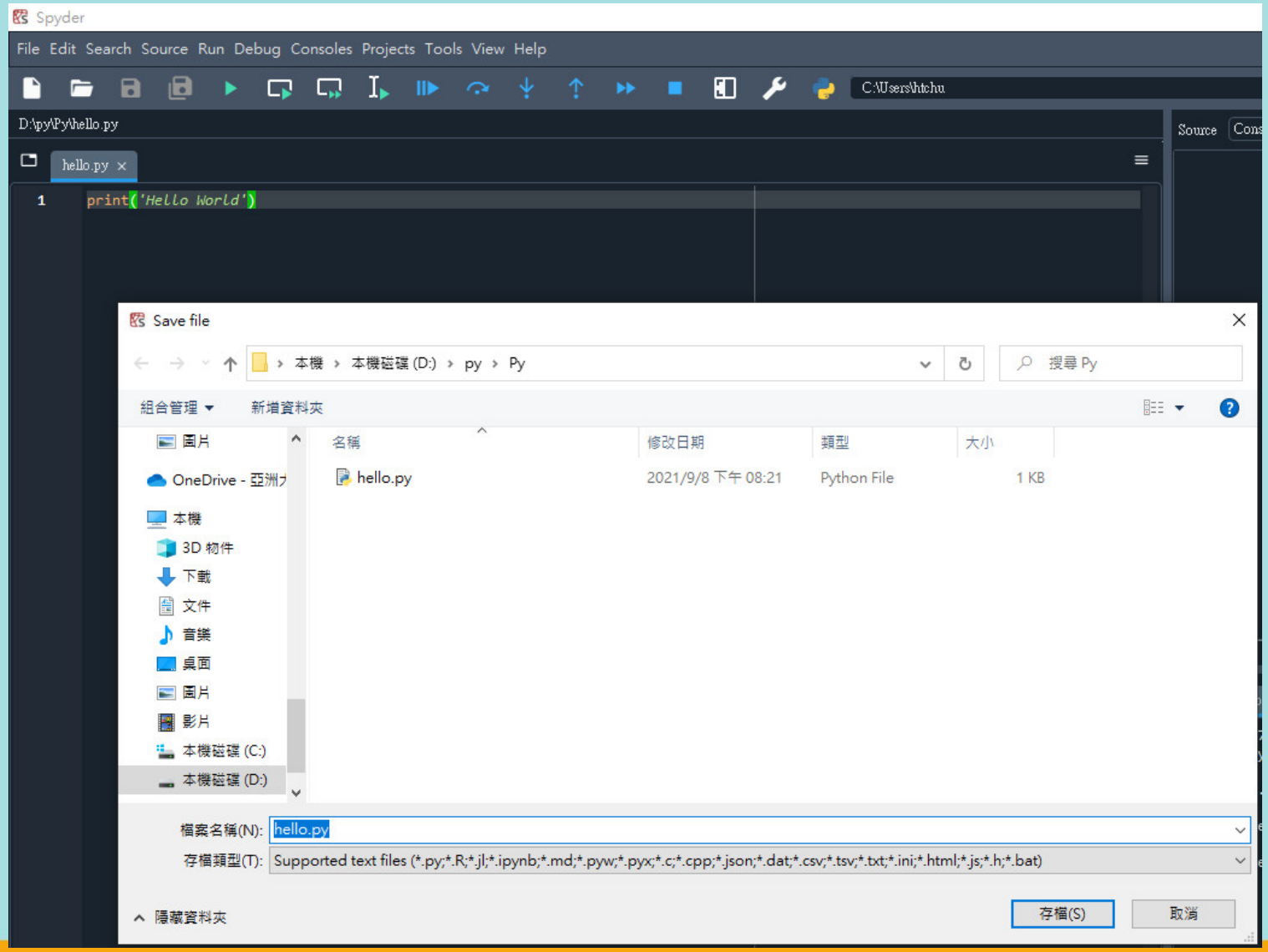
Python程式的編輯工具

- Python IDLE工具
- Spyder
- Visual Studio Code
- PyCharm
- Colab

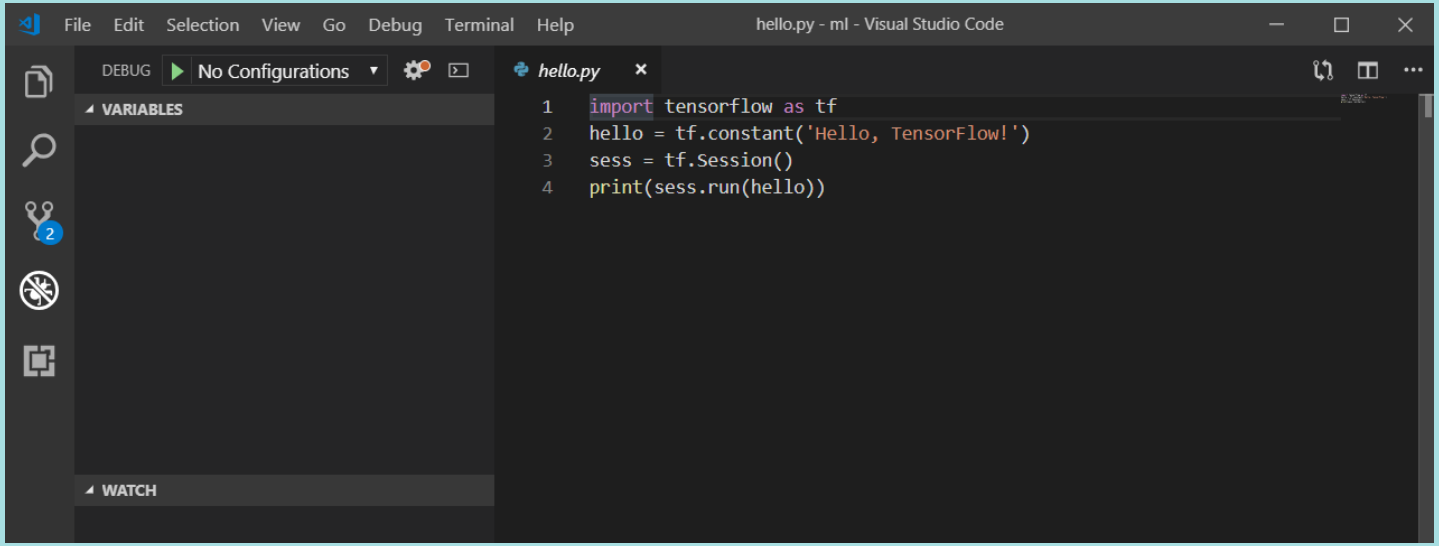
Python IDLE


Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
Ln: 3 Col: 4

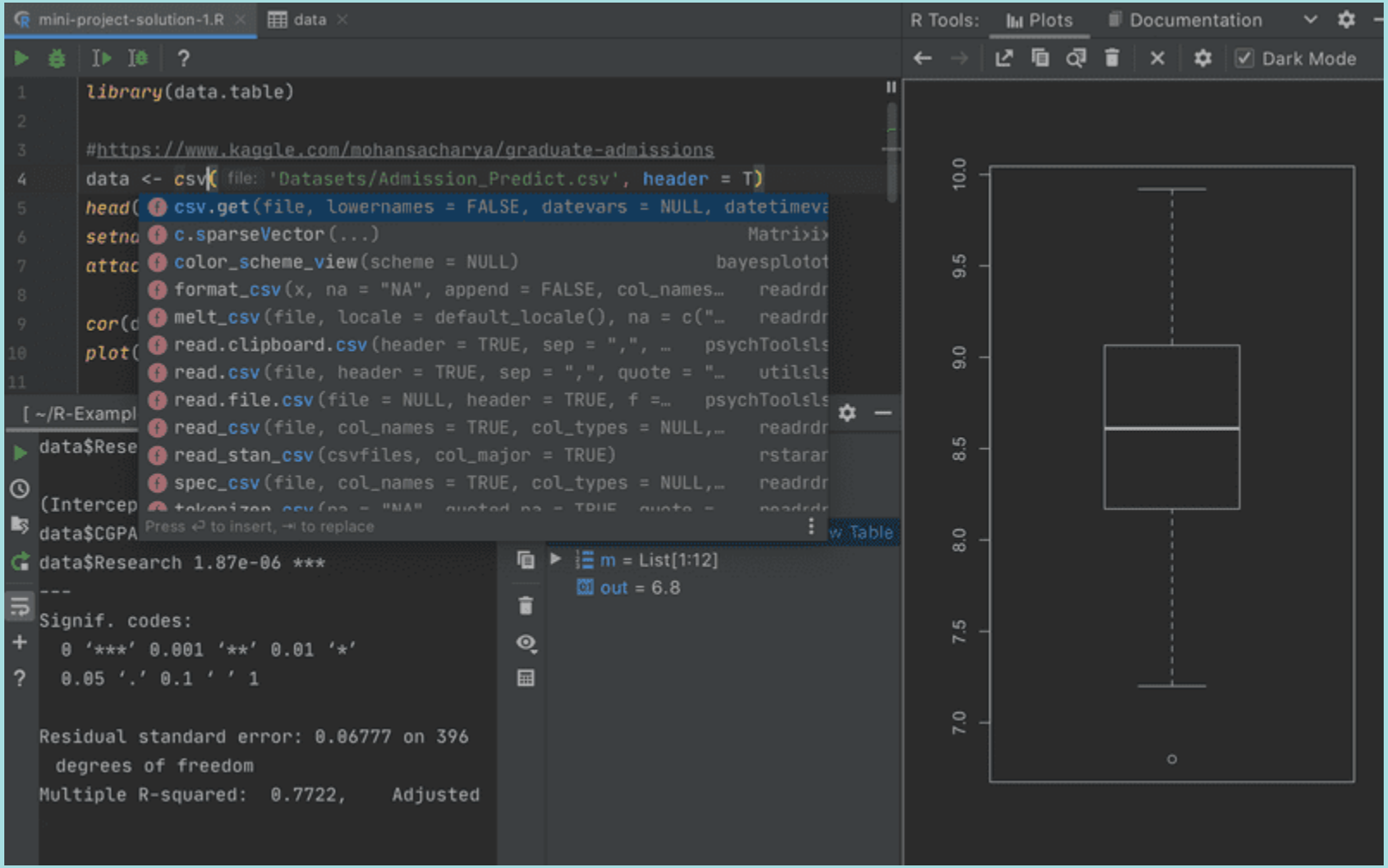
Spyder


Spyder
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\htchu
D:\py\Py\hello.py
hello.py
1 print('Hello World')
Source
Save file
本機 › 本機磁碟 (D:) › py › Py
搜尋 Py
組合管理
新增資料夾
名稱
修改日期
類型
大小
hello.py
2021/9/8 下午 08:21
Python File
1 KB
OneDrive - 亞洲
本機
3D 物件
下載
文件
音樂
桌面
圖片
影片
本機磁碟 (C:)
本機磁碟 (D:)
檔案名稱(N): hello.py
存檔類型(T): Supported text files (*.py;*.R;*.jl;*.ipynb;*.md;*.pyw;*.pyx;*.c;*.cpp;*.json;*.dat;*.csv;*.tsv;*.txt;*.ini;*.html;*.js;*.h;*.bat)
隱藏資料夾
存檔(S)
取消

Visual Studio Code

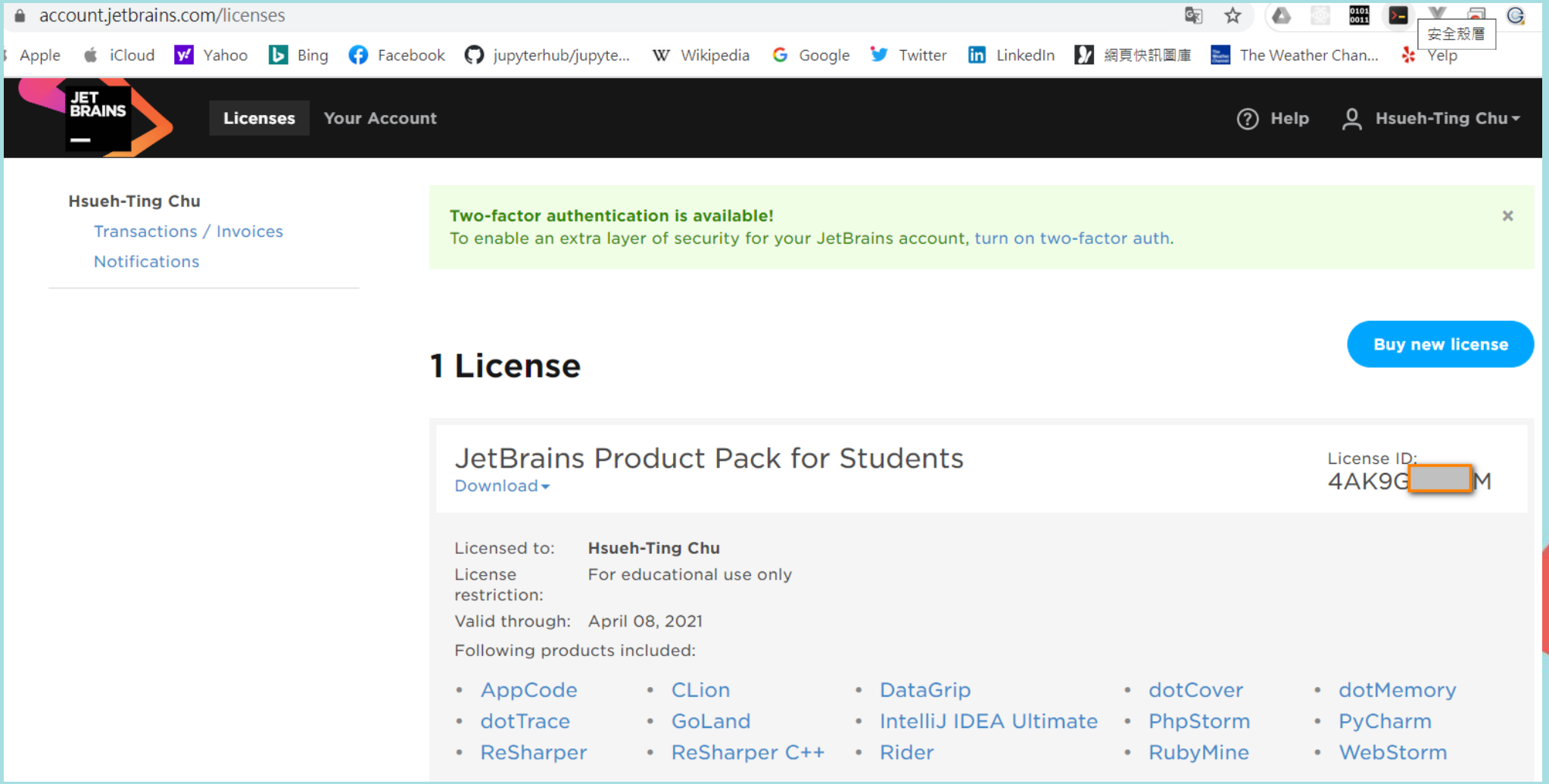


PyCharm


mini-project-solution-1.R
data
library(data.table)
#https://www.kaggle.com/mohansacharya/graduate-admissions
data <- csv(file: 'Datasets/Admission_Predict.csv', header = T)
head(
setna
attac
cor(
plot(
csv.get(file, lowernames = FALSE, datevars = NULL, datetimeva
c.sparseVector(...)
color_scheme_view(scheme = NULL)
format_csv(x, na = "NA", append = FALSE, col_names…
melt_csv(file, locale = default_locale(), na = c("…
read.clipboard.csv(header = TRUE, sep = ",", …
read.csv(file, header = TRUE, sep = ",", quote = "…
read.file.csv(file = NULL, header = TRUE, f =…
read_csv(file, col_names = TRUE, col_types = NULL,…
read_stan_csv(csvfiles, col_major = TRUE)
spec_csv(file, col_names = TRUE, col_types = NULL,…
Press ⏎ to insert, ⇥ to replace
[~/R-Exampl
data$Rese
(Intercep
data$CGPA
data$Research 1.87e-06 ***

Signif. codes:
0 '***' 0.001 '**' 0.01 '*'
0.05 '.' 0.1 ' ' 1
Residual standard error: 0.06777 on 396
degrees of freedom
Multiple R-squared: 0.7722, Adjusted
m = List[1:12]
out = 6.8
R Tools:
Plots
Documentation
Dark Mode
10.0
9.5
9.0
8.5
8.0
7.5
7.0

JetBrains Product Pack for Students

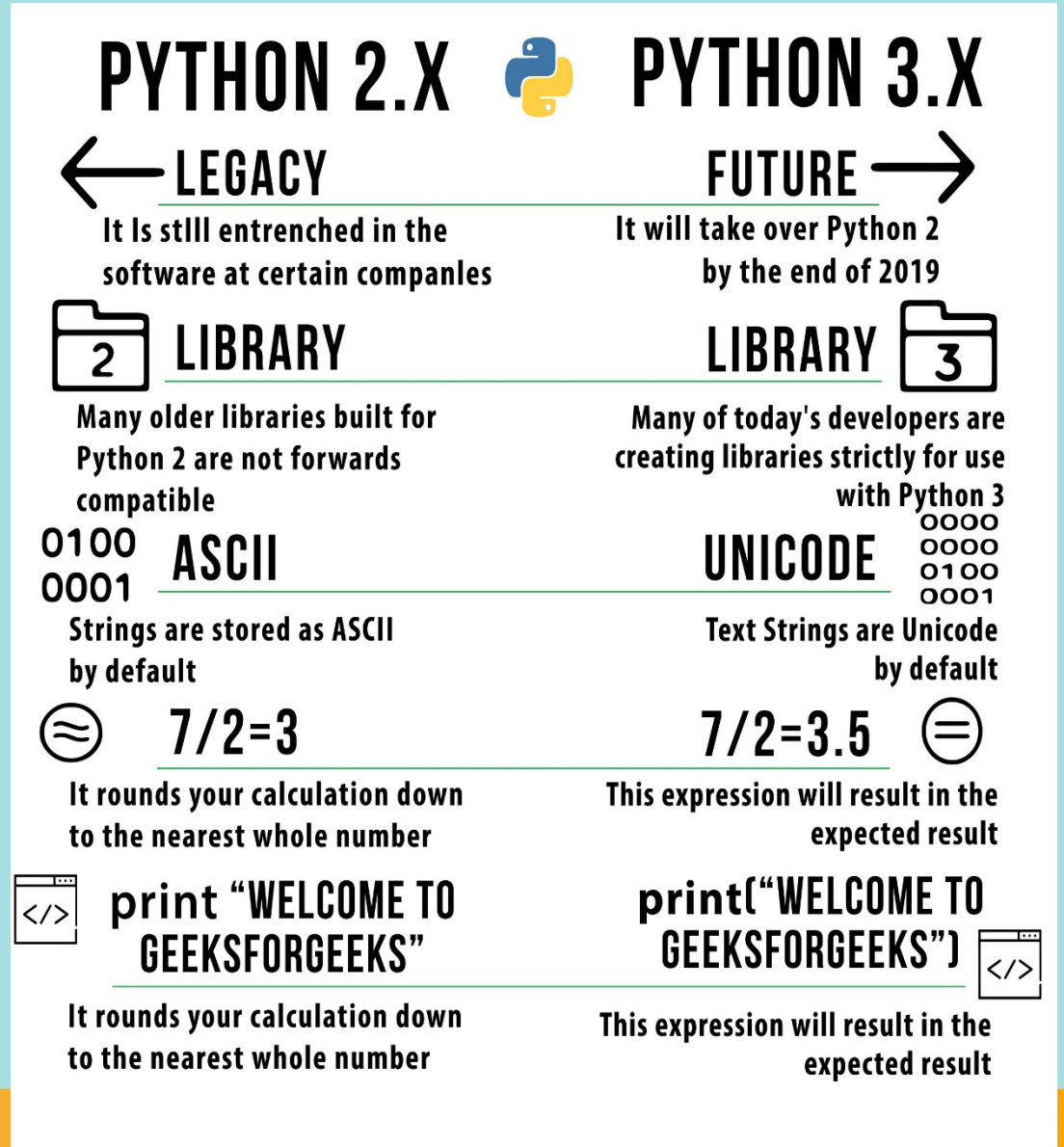


https://account.jetbrains.com/licenses

Zen of Python

1. Beautiful is better than ugly.
2. Explicit is better than implicit.
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. Readability counts.
8. Special cases aren't special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. There should be one– and preferably only one –obvious way to do it.[a]
14. Although that way may not be obvious at first unless you're Dutch.
15. Now is better than never.
16. Although never is often better than *right* now.
17. If the implementation is hard to explain, it's a bad idea.
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea – let's do more of those!

Python Versions



Python new features:

- Python 3.10: Structural Pattern Matching
- Python 3.6 : f-Strings
- Python 3.3 : Virtual Environments
- Python 3.2: Argparse

Python powerful features:

- Iterators
- Generators
- Decorators
- Context Managers

Python cheat sheet (1)

Beginner's Python Cheat Sheet

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

Lists (cont.)

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

Tuples

Tuples are similar to lists, but the items in a tuple can't be modified.

Making a tuple

```
dimensions = (1920, 1080)
```

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

```
equals            x == 42
not equal         x != 42
greater than      x > 42
  or equal to     x >= 42
less than         x < 42
  or equal to     x <= 42
```

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

```
if age >= 18:
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

Dictionaries

Dictionaries store connections between pieces of information. Each item in a dictionary is a key-value pair.

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print("The alien's color is " + alien['color'])
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(name + ' loves ' + str(number))
```

Looping through all keys

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

User input

Your programs can prompt the user for input. All input is stored as a string.

Prompting for a value

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```

Python cheat sheet (2)

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

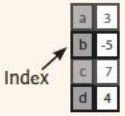
Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

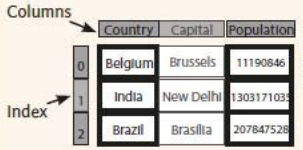
Series

A **one-dimensional** labeled array capable of holding any data type




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame



A **two-dimensional** labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
 -5

>>> df[1:]
   Country    Capital  Population
1   India  New Delhi  1303171035
2  Brazil   Brasilia   207847528
```

Get one element

Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at([0], ['Country'])
'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population 207847528
```

Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasília
```

Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```

Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```

Series s where value is not >1
s where value is <-1 or >2
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```

Set index a of Series s to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```

Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis
Sort by the values along an axis
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```

(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idxmin()/df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b     NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

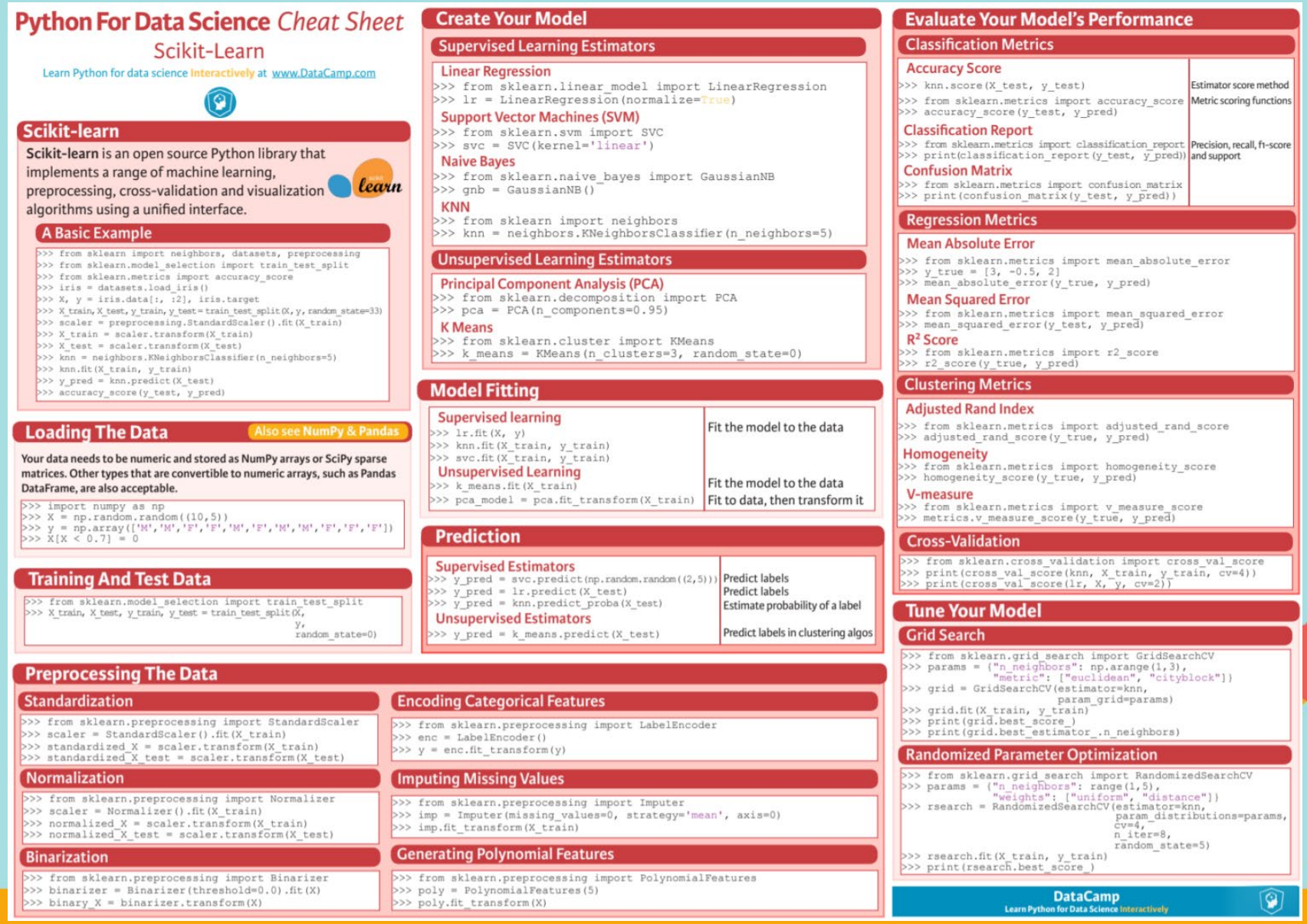
```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> df.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

Python cheat sheet (3)



Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science Interactively at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning

```
>>> lr.fit(X, y)
>>> knn.fit(X_train, y_train)
>>> svc.fit(X_train, y_train)
```

Fit the model to the data

Unsupervised Learning

```
>>> k_means.fit(X_train)
>>> pca_model = pca.fit_transform(X_train)
```

Fit the model to the data
Fit to data, then transform it

Prediction

Supervised Estimators

```
>>> y_pred = svc.predict(np.random.random((2,5)))
>>> y_pred = lr.predict(X_test)
>>> y_pred = knn.predict_proba(X_test)
```

Predict labels
Predict labels
Estimate probability of a label

Unsupervised Estimators

```
>>> y_pred = k_means.predict(X_test)
```

Predict labels in clustering algos

Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test)
```

Estimator score method

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Metric scoring functions

Classification Report

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Precision, recall, f1-score and support

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

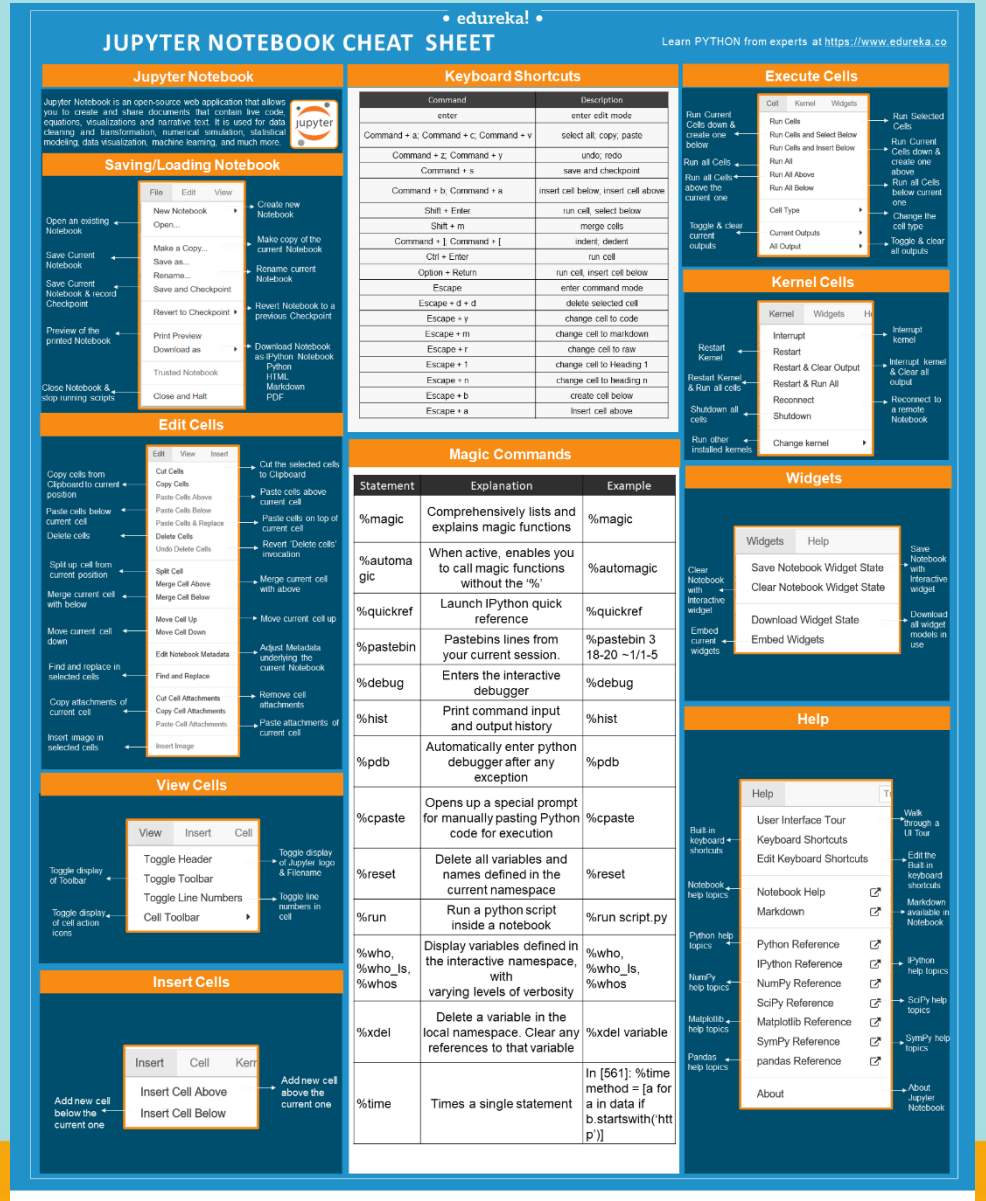
```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

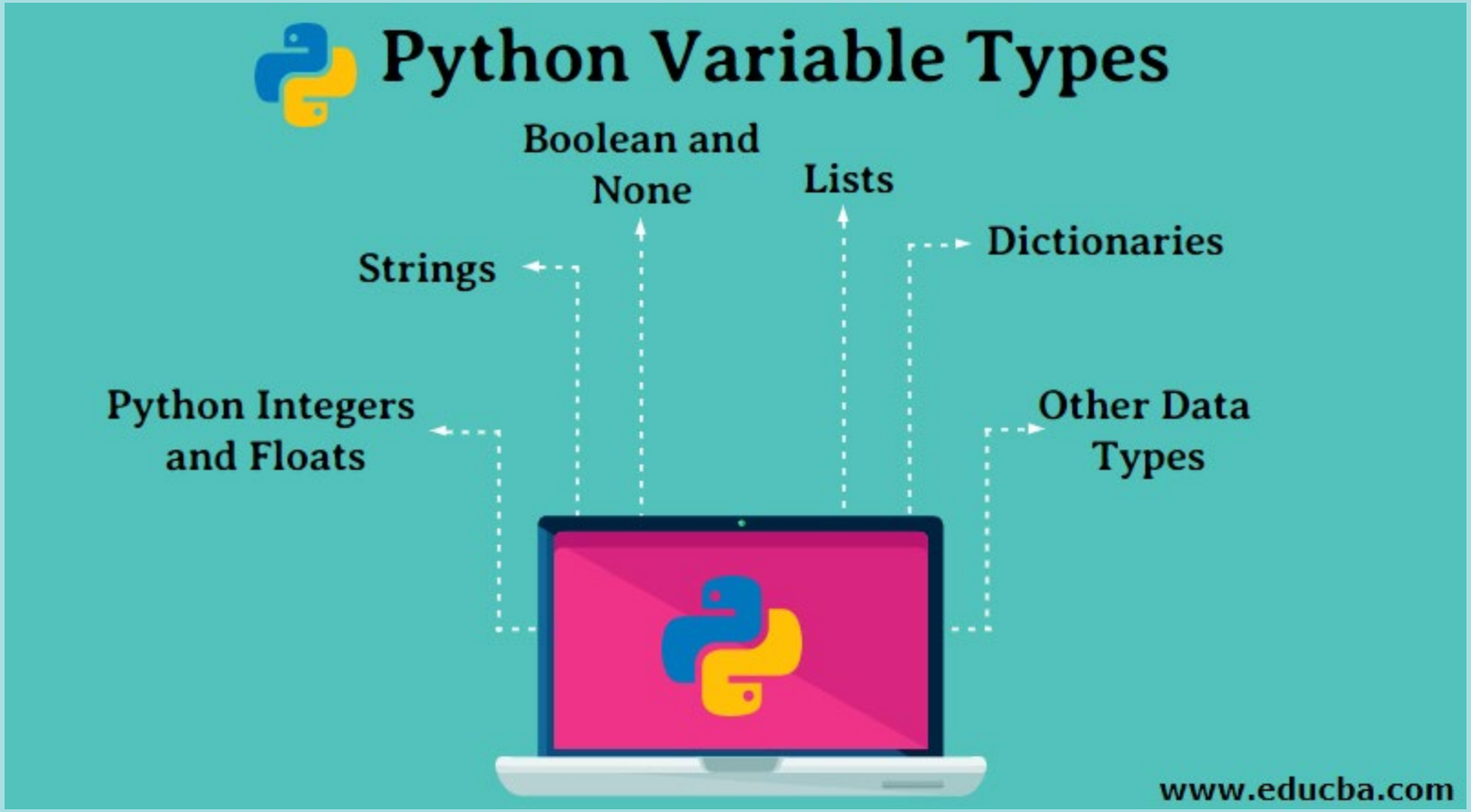
DataCamp
Learn Python for Data Science Interactively

Jupyter Notebook Cheat Sheet

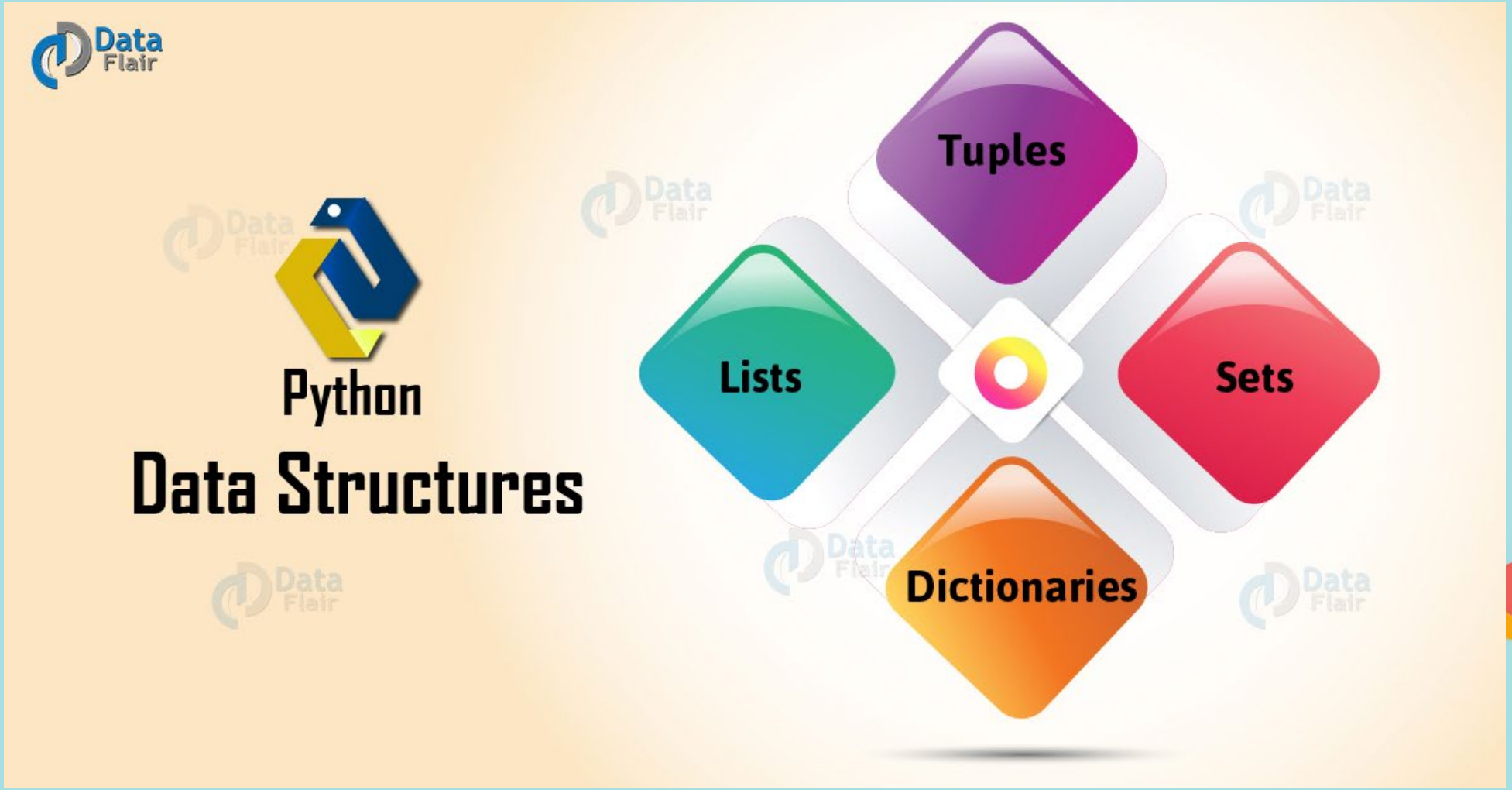


Command	Description
enter	enter edit mode
Command + a; Command + c; Command + v	select all; copy; paste
Command + z; Command + y	undo; redo
Command + s	save and checkpoint
Command + b; Command + a	insert cell below; insert cell above
Shift + Enter	run cell, select below
Shift + m	merge cells
Command +]; Command + [	indent; dedent
Ctrl + Enter	run cell
Option + Return	run cell, insert cell below
Escape	enter command mode
Escape + d + d	delete selected cell
Escape + y	change cell to code
Escape + m	change cell to markdown
Escape + r	change cell to raw
Escape + 1	change cell to Heading 1
Escape + n	change cell to heading n
Escape + b	create cell below
Escape + a	Insert cell above

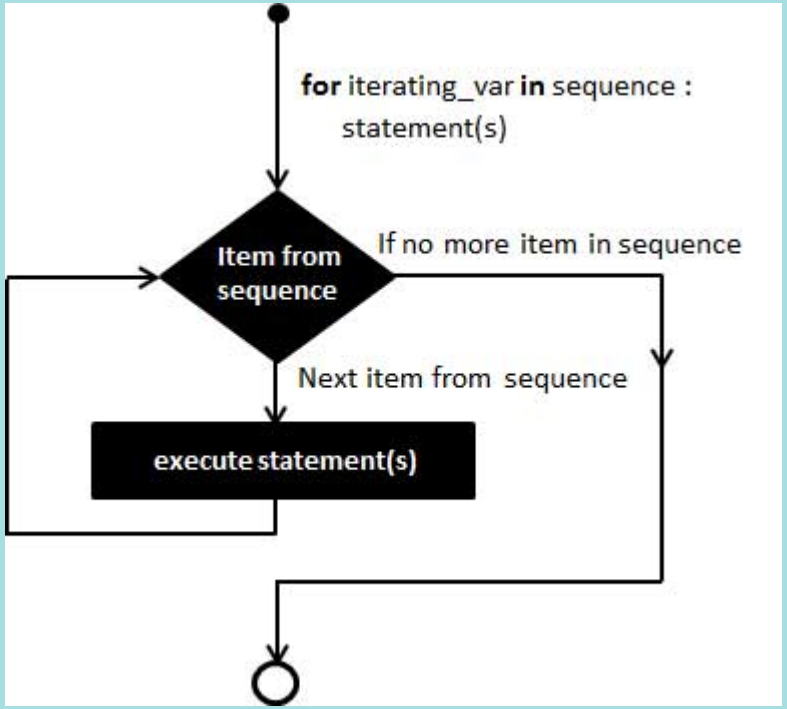
Variables

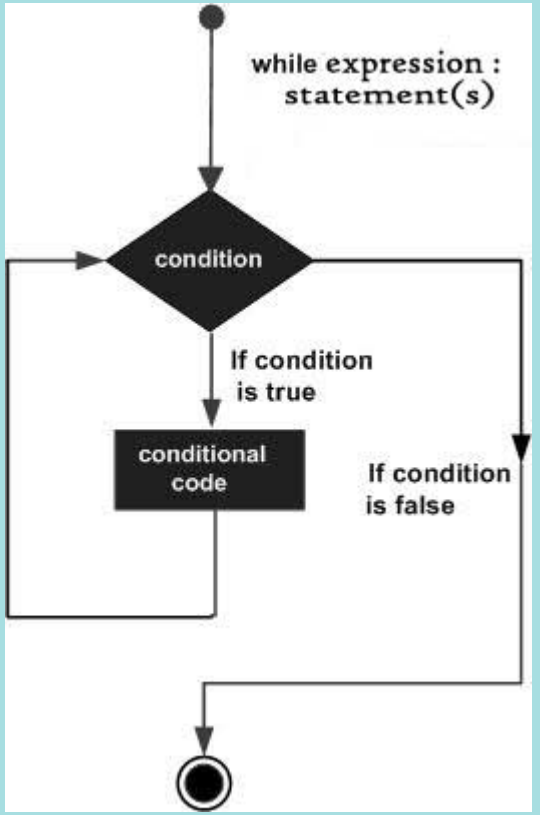

Python Variable Types
Boolean and None
Lists
Strings
Dictionaries
Python Integers and Floats
Other Data Types
www.educba.com

Variable containers

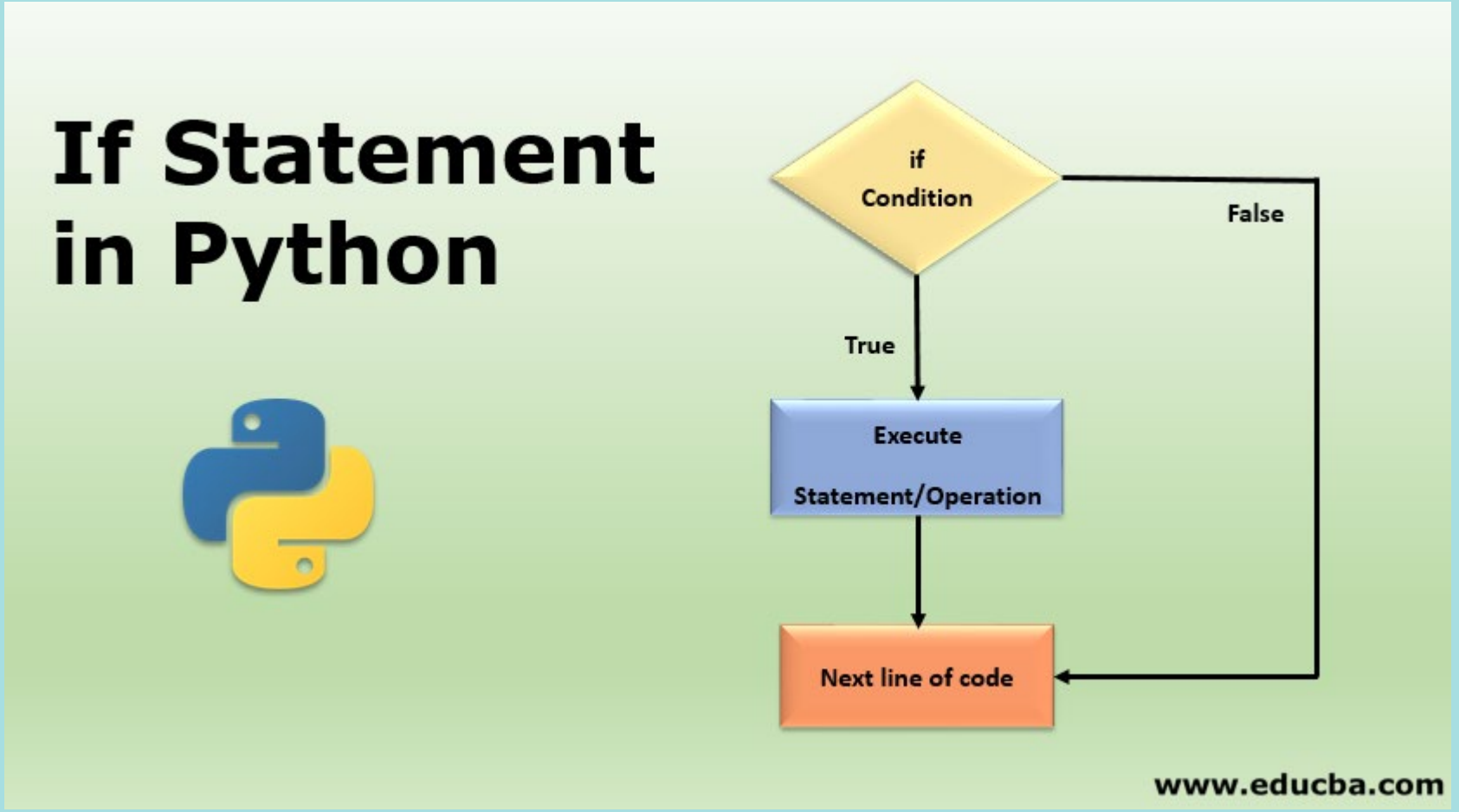


while-loop vs for-loop

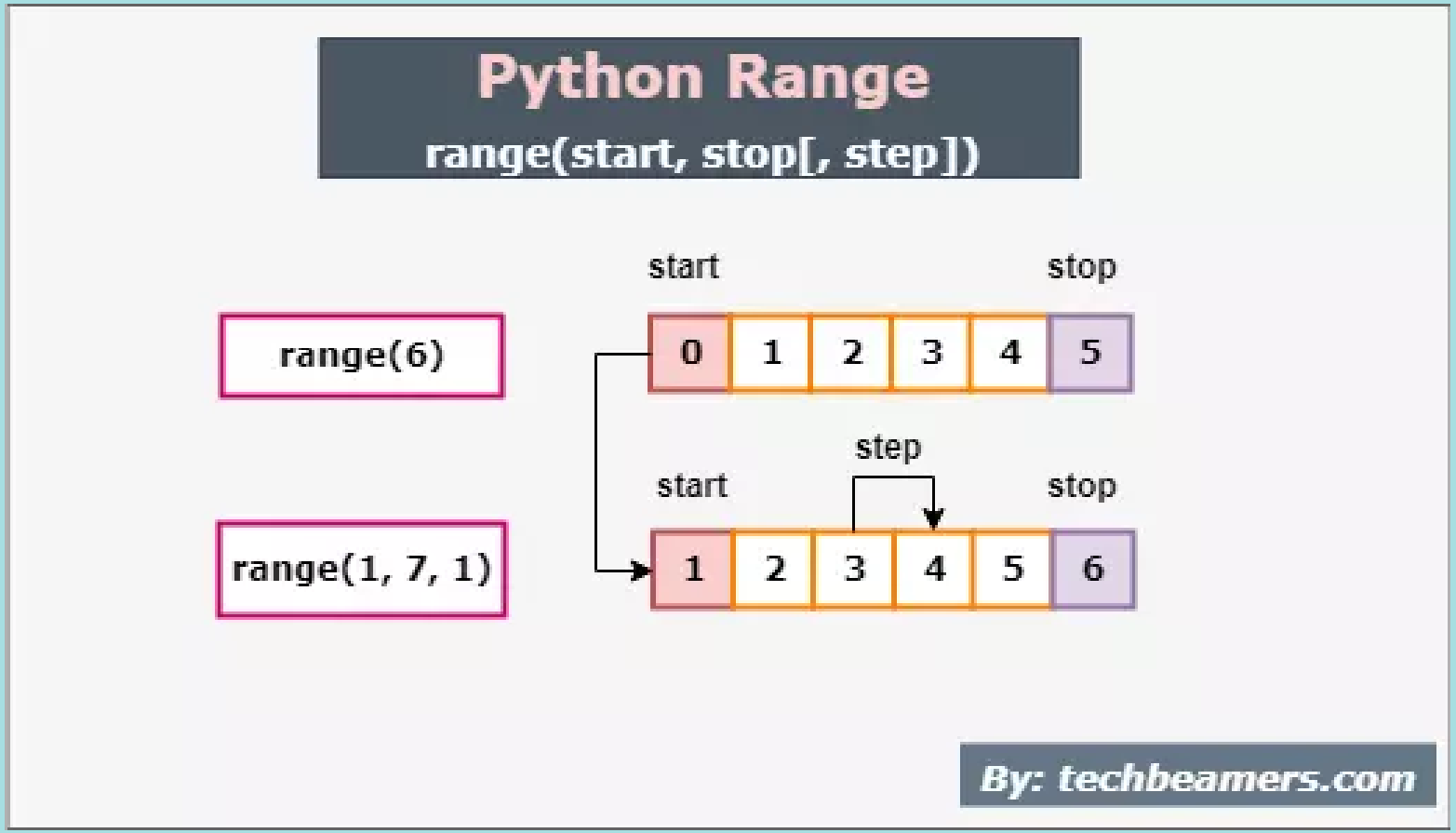




If condition



range() function



Kissipo

Kissipo = KISS principle + IPO model

KISS principle

https://en.wikipedia.org/wiki/KISS_principle

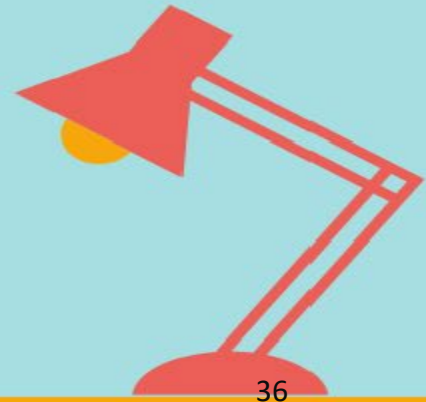
IPO model

https://en.wikipedia.org/wiki/IPO_model

35

Python程式練習一

- 列印99乘法表(要排列整齊)
- print() 函數
- range()函數
- for 迴圈

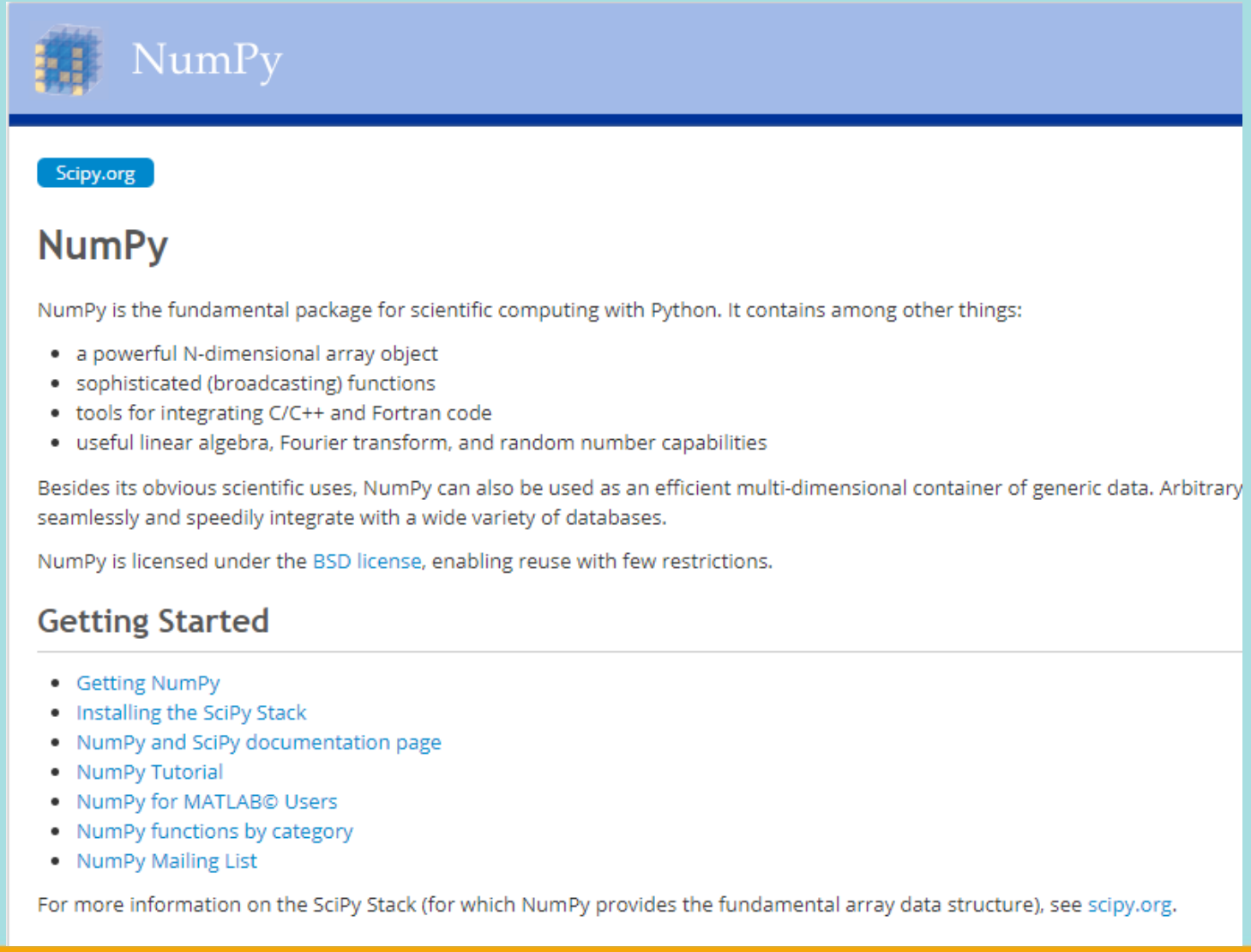


(3) TENSORFLOW

Deep Learning Programming framework

1. Python: https://www.python.org/
2. NumPy: https://numpy.org/
3. pandas: https://pandas.pydata.org/
4. scikit-learn: https://scikit-learn.org/
5. TensorFlow: https://www.tensorflow.org/
6. Keras: https://keras.io/
7. PyTorch: https://pytorch.org/

NumPy


NumPy
Scipy.org
NumPy
NumPy is the fundamental package for scientific computing with Python. It contains among other things:
a powerful N-dimensional array object
sophisticated (broadcasting) functions
tools for integrating C/C++ and Fortran code
useful linear algebra, Fourier transform, and random number capabilities
Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary
seamlessly and speedily integrate with a wide variety of databases.
NumPy is licensed under the BSD license, enabling reuse with few restrictions.
Getting Started
Getting NumPy
Installing the SciPy Stack
NumPy and SciPy documentation page
NumPy Tutorial
NumPy for MATLAB© Users
NumPy functions by category
NumPy Mailing List
For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see scipy.org.

matplotlib

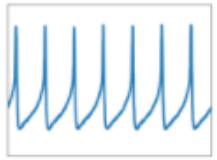
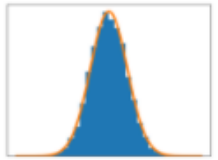
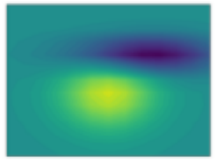
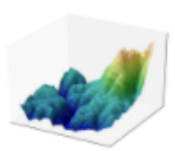
matplotlib

Version 2.1.2

home | examples | tutorials | pyplot | docs »

Introduction

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

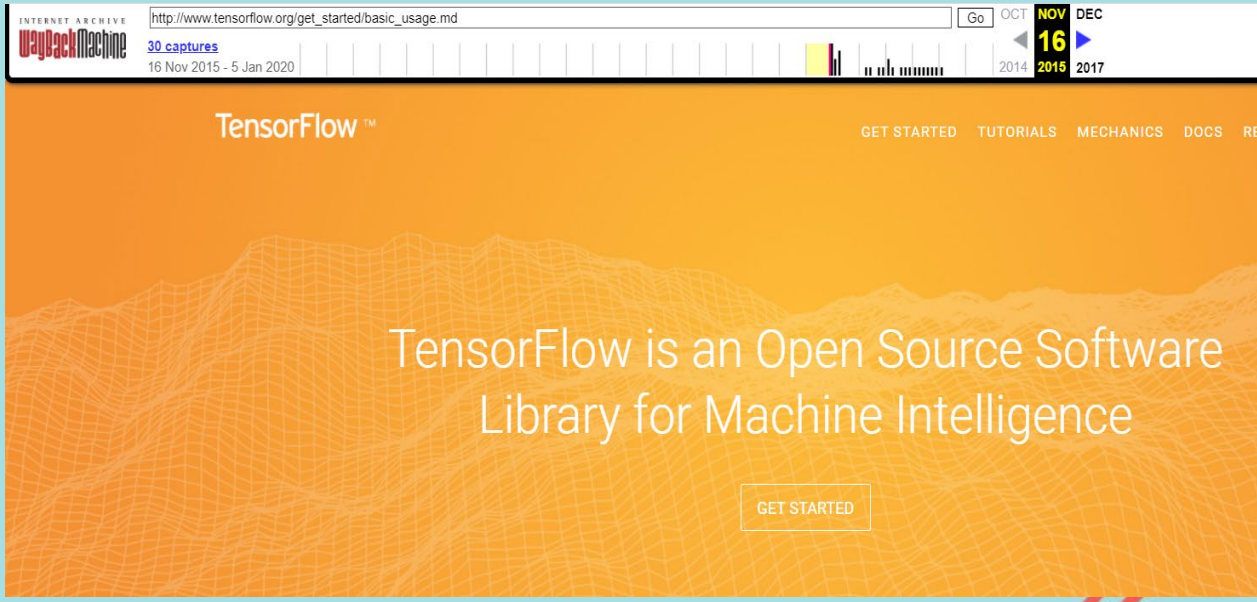
For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with `IPython`. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Installation

Visit the Matplotlib installation instructions.

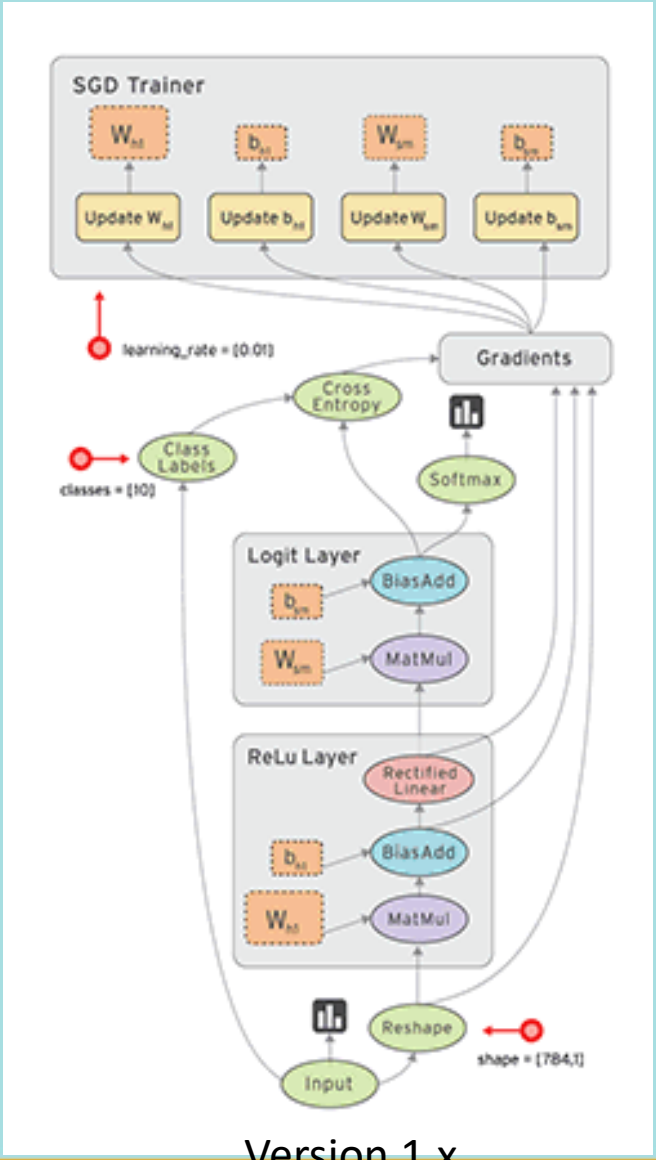
Tensorflow

TensorFlow	
Developer(s)	Google Brain Team[1]
Initial release	November 9, 2015;
Stable release	2.4.0[2] / Dec. 14, 2020
Repository	github.com/tensorflow/tensorflow
Written in	Python, C++, CUDA
Platform	Linux, macOS, Windows, Android



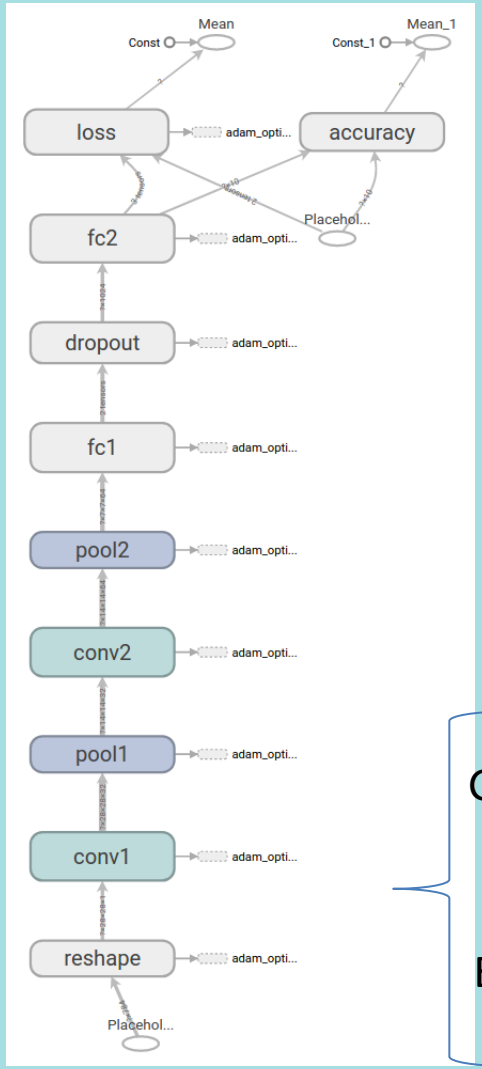
https://web.archive.org/web/20151109150056/https://www.tensorflow.org/

Tensorflow Version 1.x and 2.x



Version 1.x

tf.Graph
tf.Session
tf.Variable
tf.Tensor
tf.Constant
tf.Placeholder



Version 2.x: Eager + Keras API

tf.keras
tf.data
tf.GradientTape

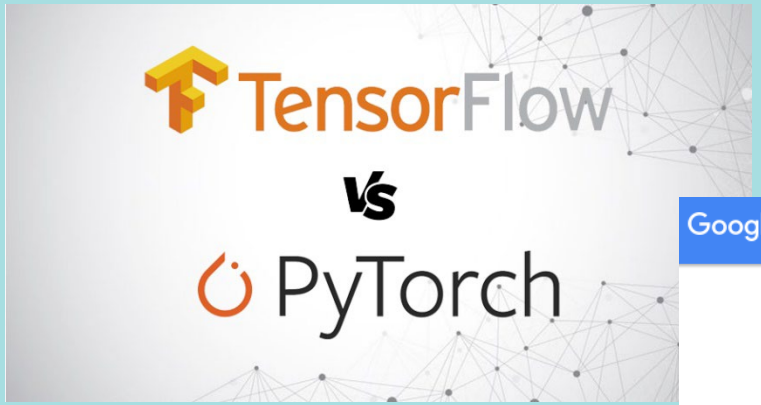
Graph mode

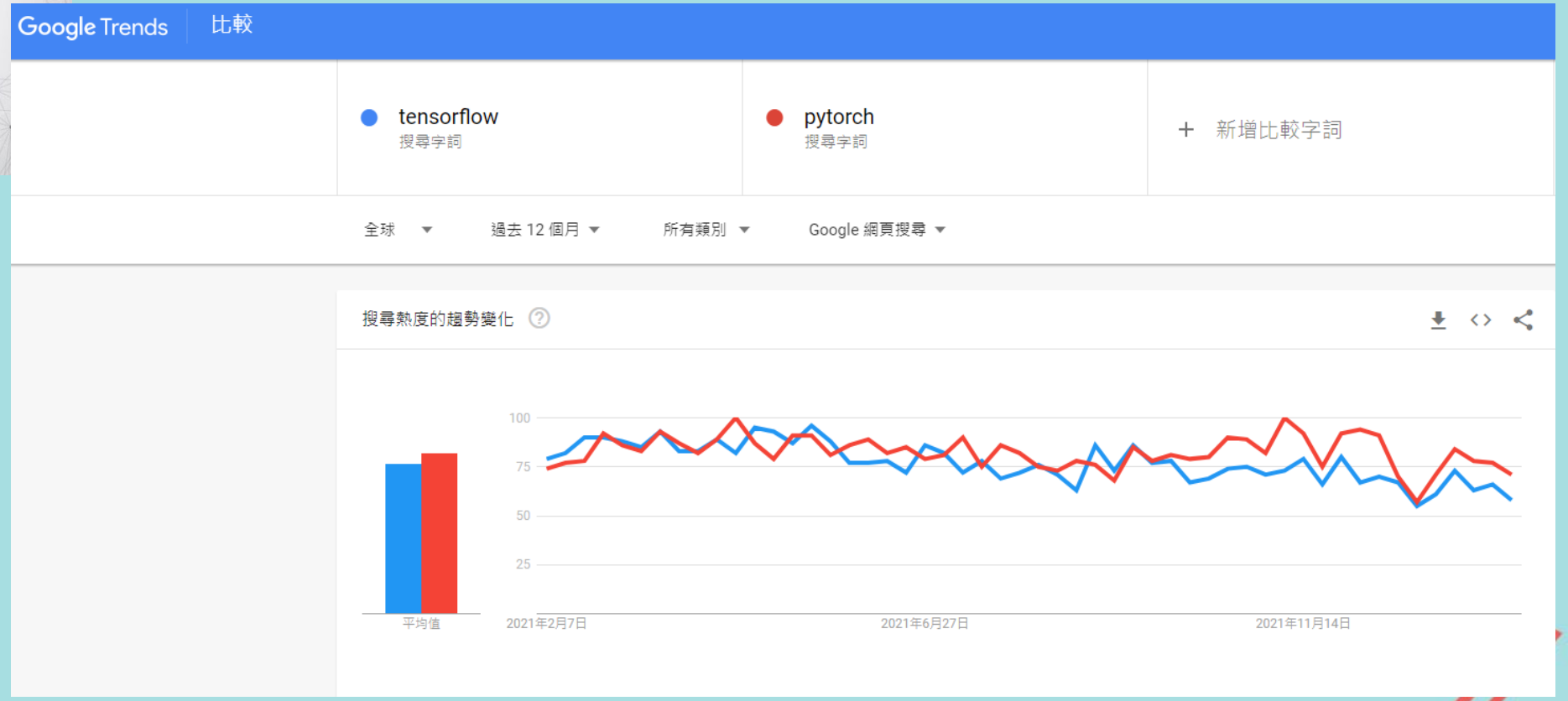
tf.Graph

Eager mode

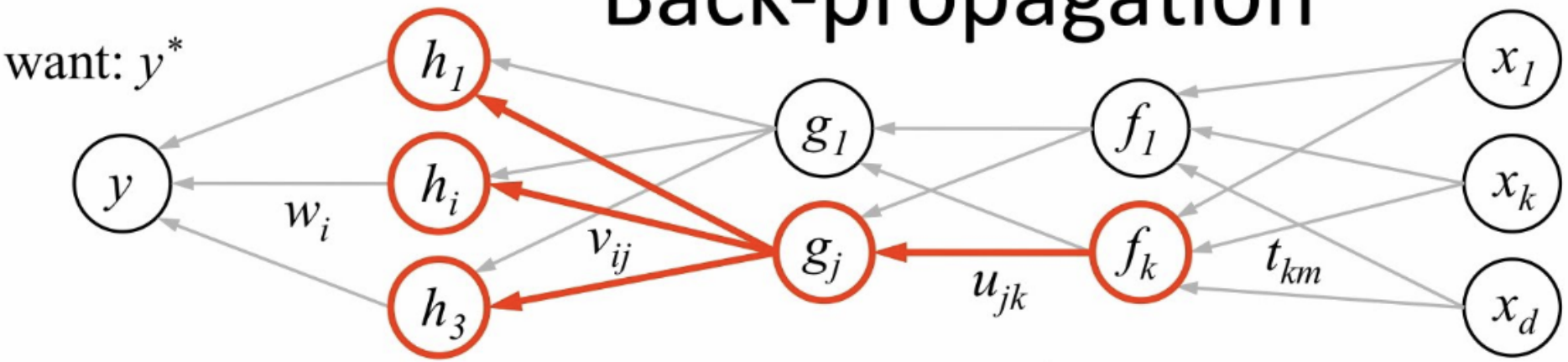
tf.Eager

PyTorch vs. TensorFlow





Back-propagation



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i) v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

(i) compute error on u_{jk}

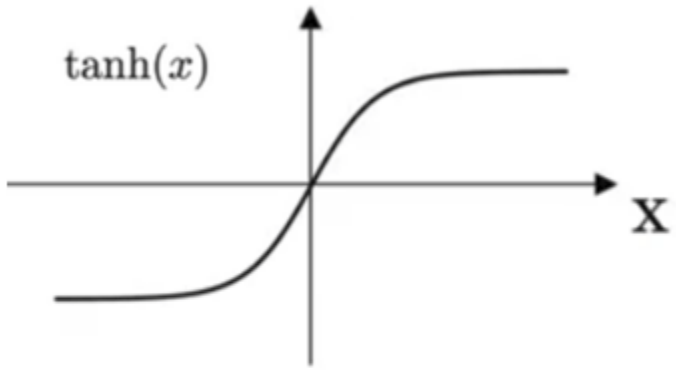
$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

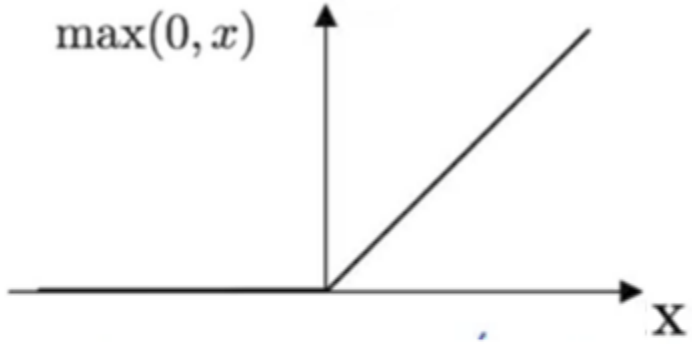
$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

Activation Functions

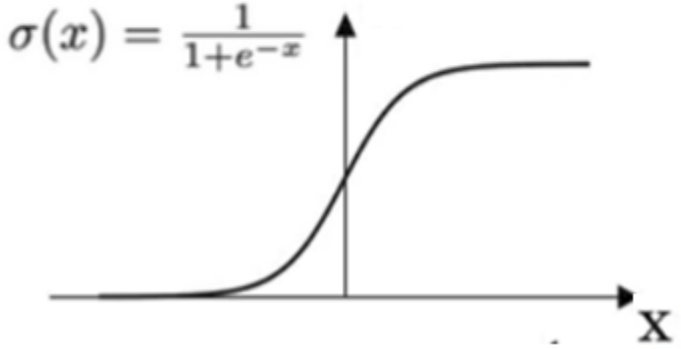
Hyper Tangent Function



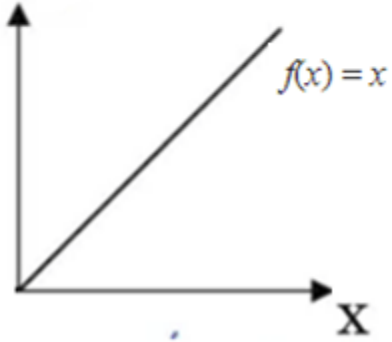
ReLU Function



Sigmoid Function

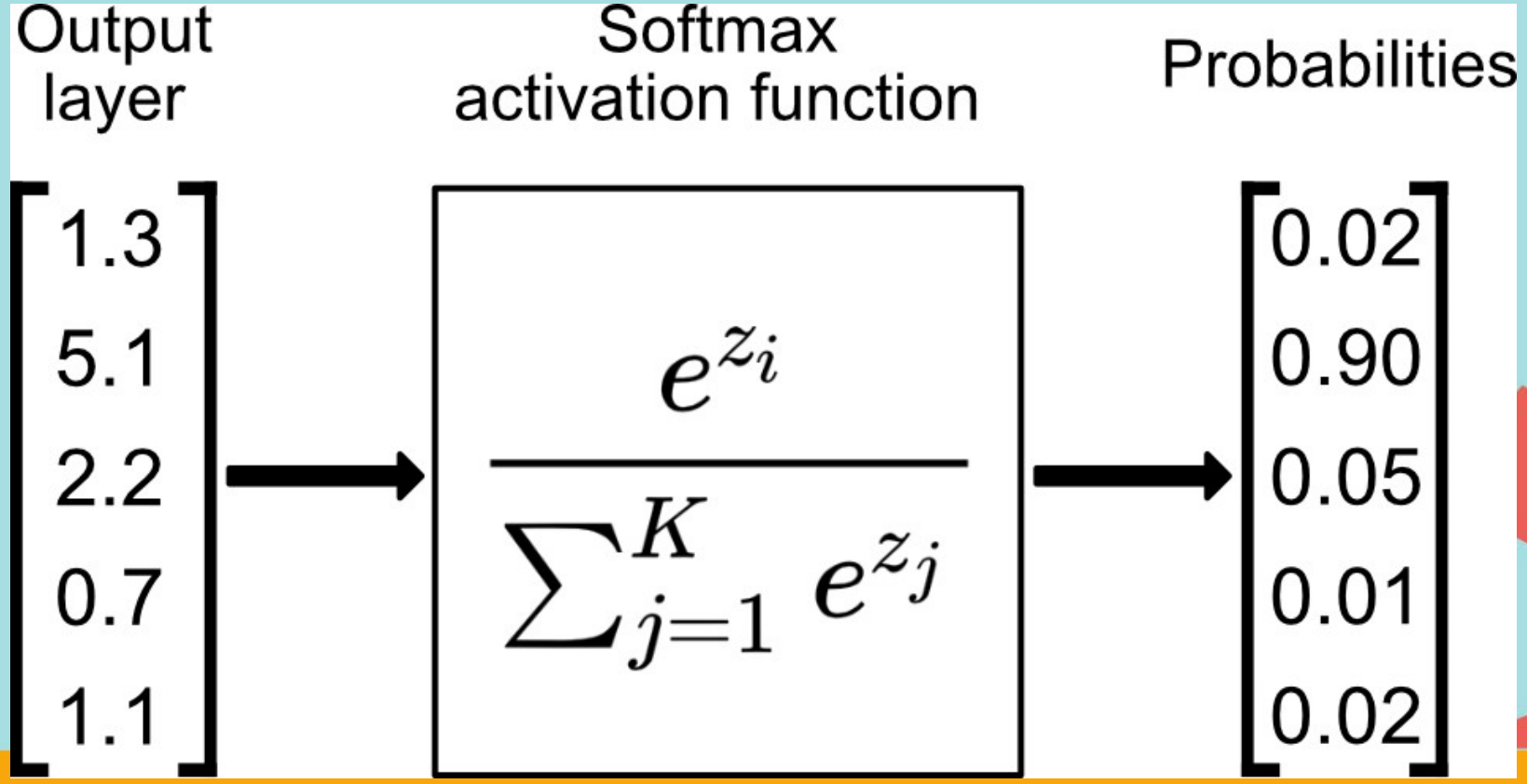


Identity Function

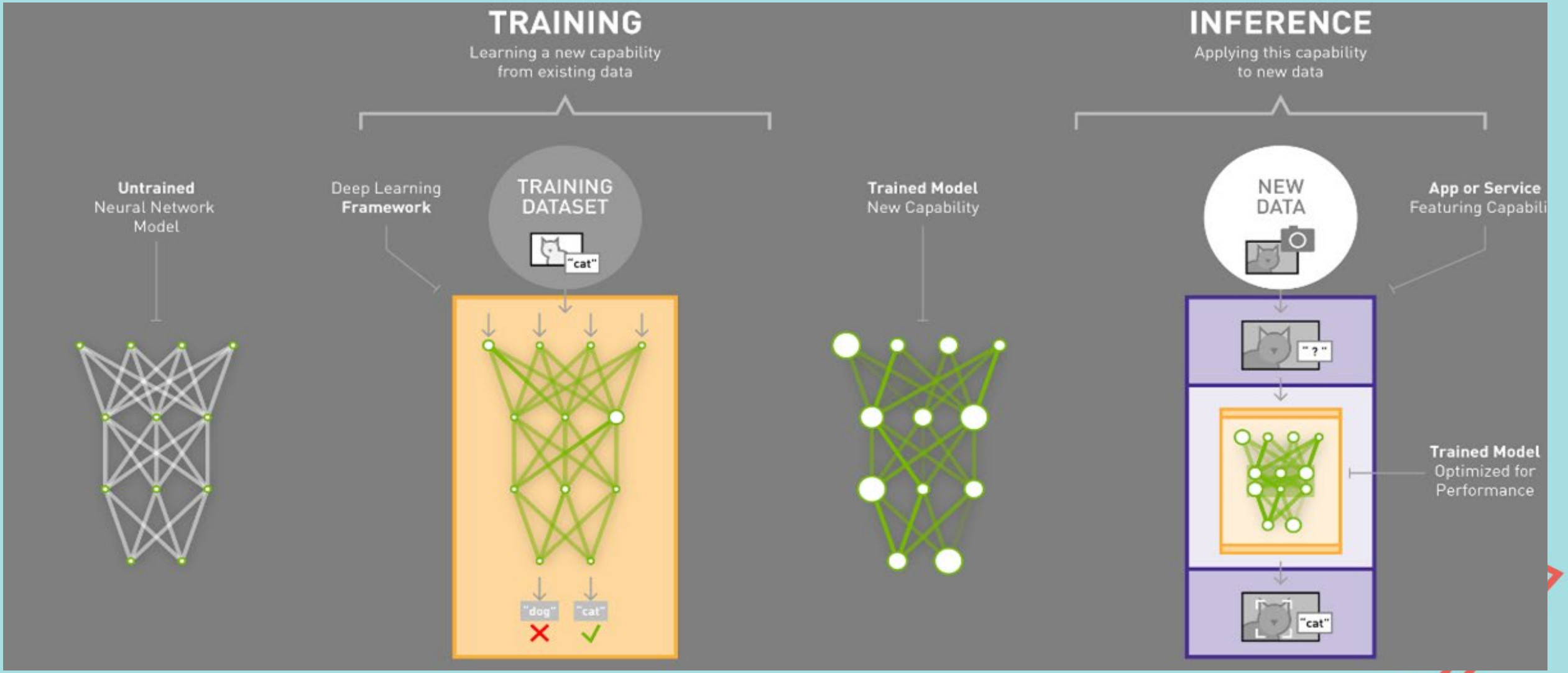


Softmax function

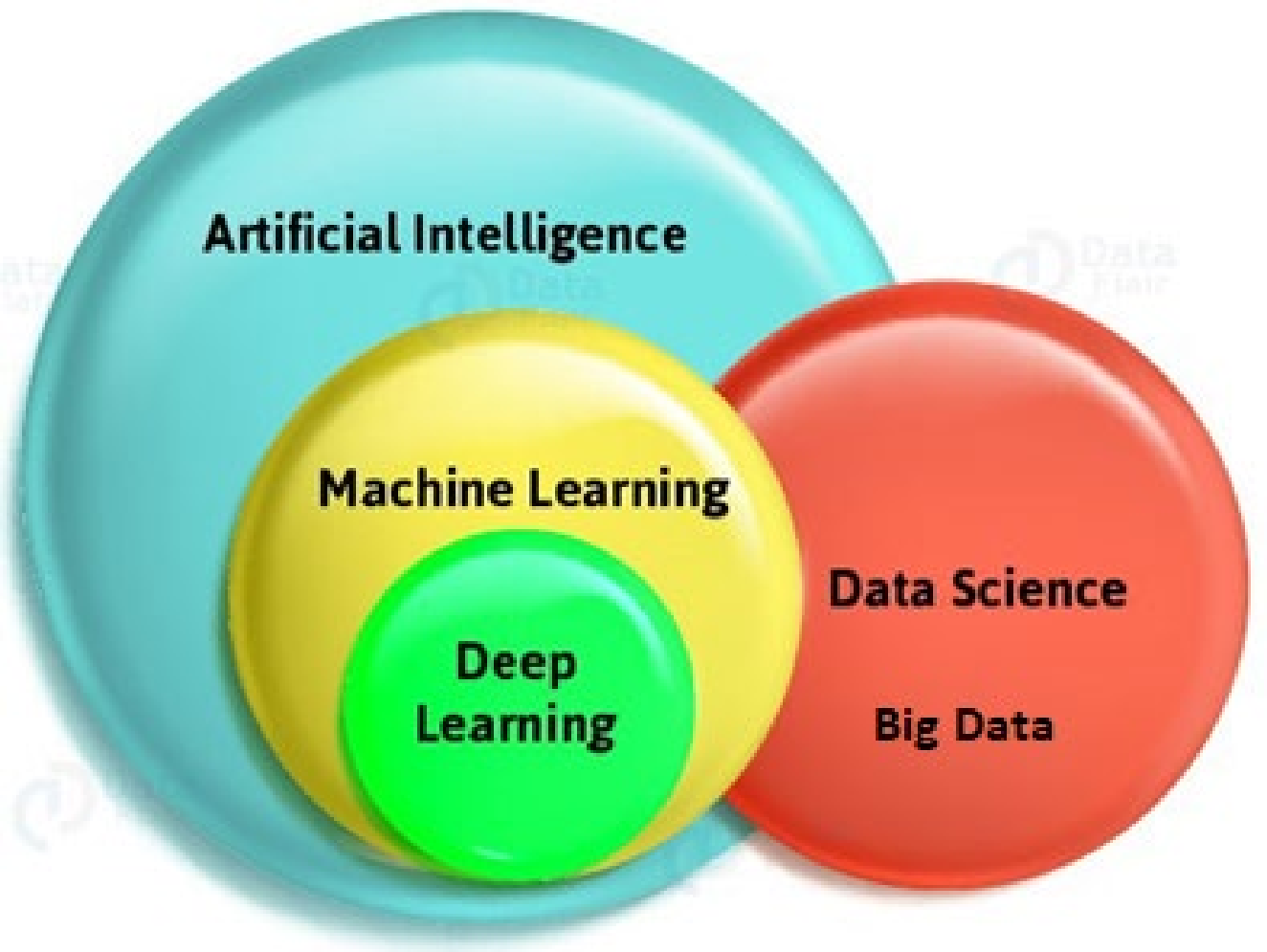
The softmax function, also known as softargmax or normalized exponential function, is a generalization of the logistic function to multiple dimensions.



How DL works?


TRAINING
Learning a new capability
from existing data
Untrained
Neural Network
Model
Deep Learning
Framework
TRAINING
DATASET
"cat"
"dog"
"cat"
Trained Model
New Capability
INFERENCE
Applying this capability
to new data
NEW
DATA
App or Service
Featuring Capabili
" ? "
"cat"
Trained Model
Optimized for
Performance

Modern AI = Big Data + Deep Learning



Mnist dataset

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

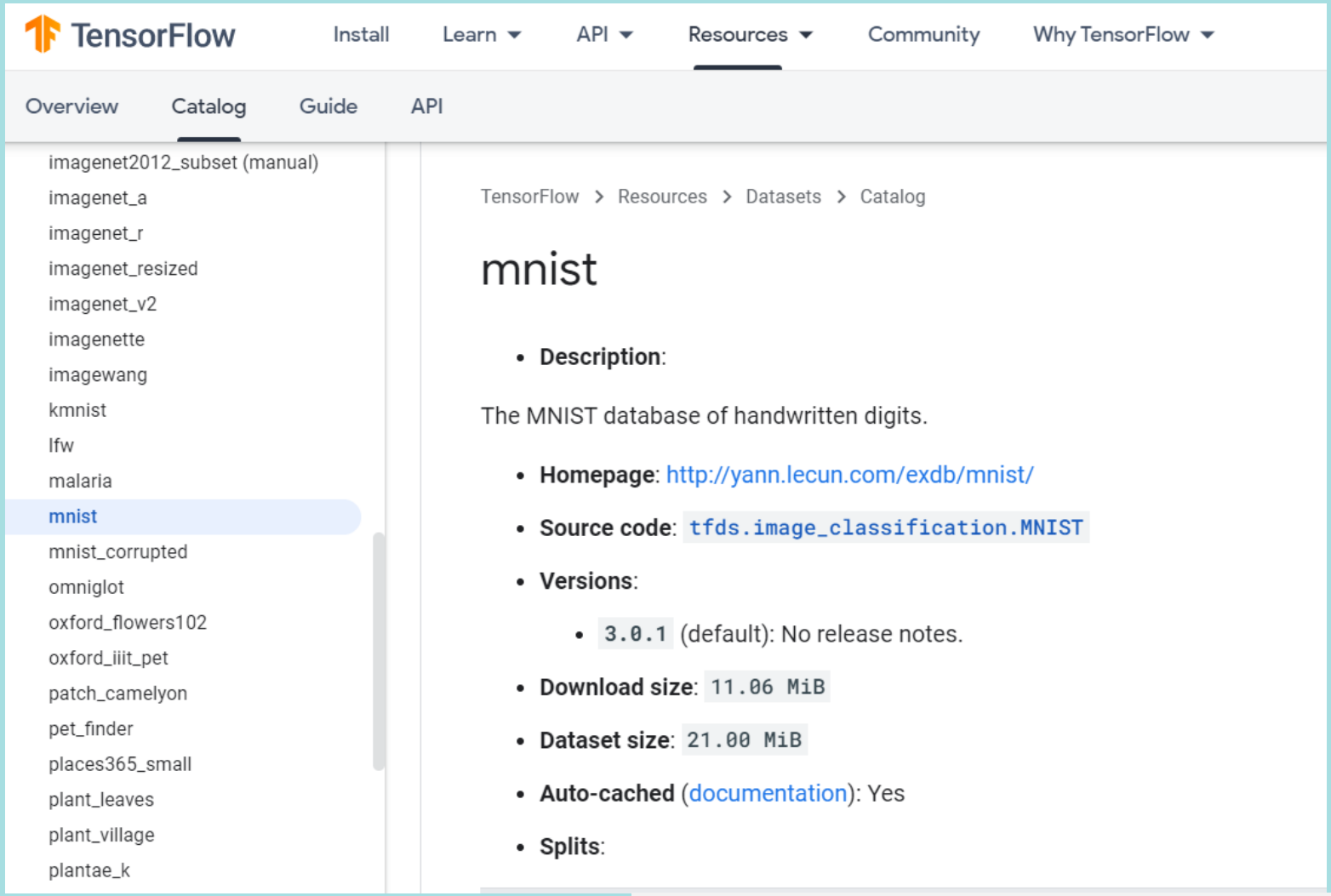
The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

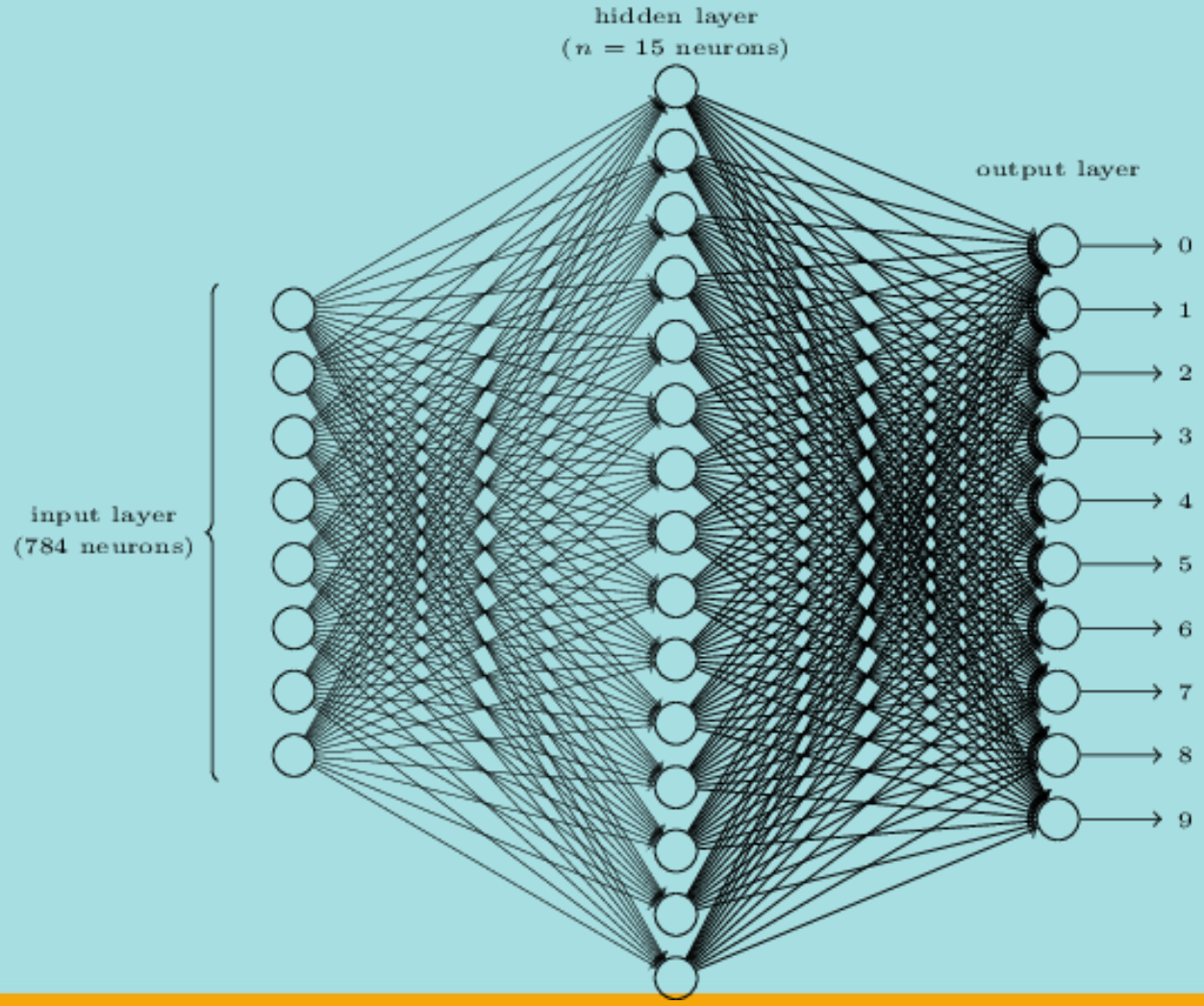
```
train-images-idx3-ubyte.gz:  training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:  training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:   test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:   test set labels (4542 bytes)
```

TF MNIST dataset



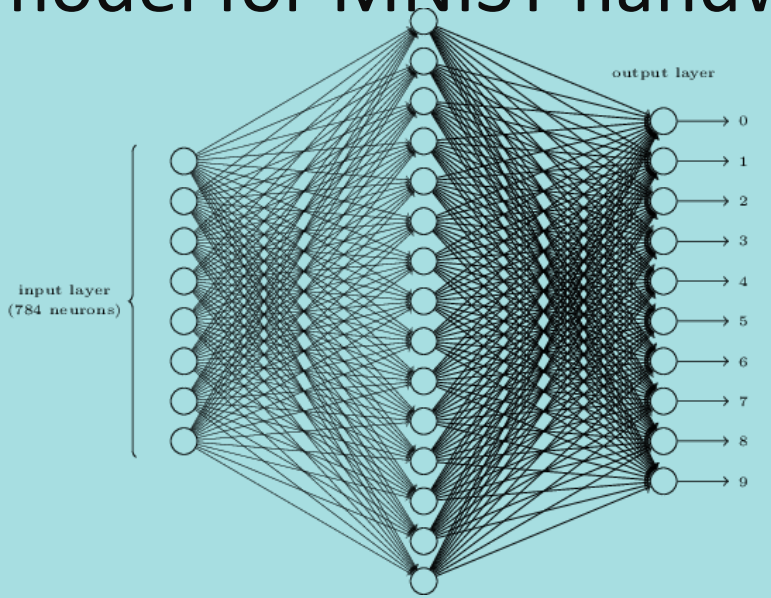
https://www.tensorflow.org/datasets/catalog/mnist

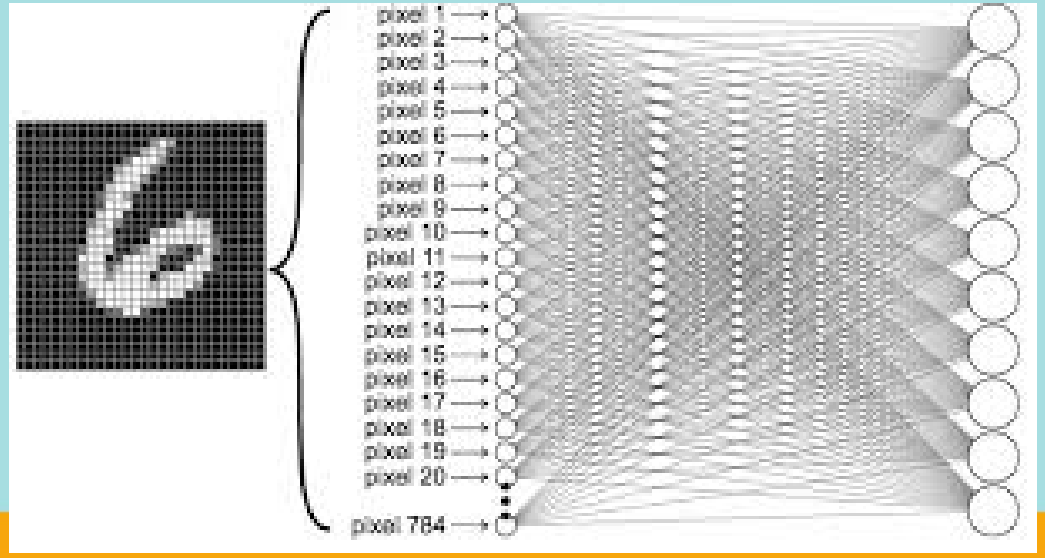
Building a neural network for handwritten digits



Deep Learning model for MNIST handwritten digits

Training





Inference

6

Thanks!

Q&A