

# Introduction to Python and Machine Learning

31<sup>st</sup> July 2019

Koh Wyhow

# Agenda

- Introduction to Python
- Data visualization using seaborn
  - Creating statistical plots easily with seaborn*
- Hypothesis testing: z-test
  - Getting started with statistical hypothesis testing – a simple z-test*
- Correlation: Contingency table & chi-square test
  - Estimating the correlation between two variables with a contingency table and a chi-square test*
- Predictive analytics: Logistic regression
  - Logistic regression using Python*
- Predictive analytics: Natural Language Processing
  - Learning from text from Naïve Bayes for NLP*
- Predictive analytics: GCP AutoML
  - Introduction to Google Cloud Platform's AutoML*
- Recommendations

# About me:

- [Koh Wyhow](#) (not SIR – I have a name =P)
- Manager, Data Science @ Star Media Group
- Formerly:
  - *Data Scientist @ INVOKE Malaysia*
  - *Consultant @ EY Data and Analytics*
  - *Further Mathematics lecturer @ Taylor's College*
- Majored in Mathematics @ National University of Singapore, 2013

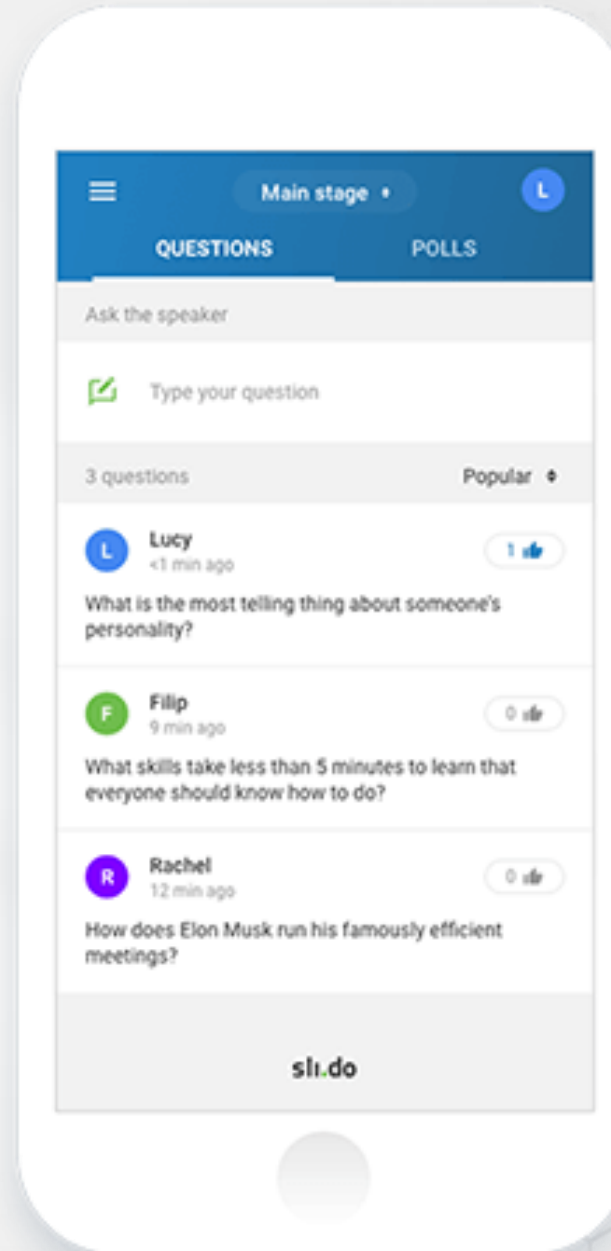
# About you:

- <https://padlet.com/kohwyhow/python20190731>
  - *What's your name?*
  - *What do you do?*
  - *What's your email?*
  - *How do you plan to use Python?*



# Questions:

- [www.sli.do](https://www.sli.do) code: #V817





# Introduction to Python



# Why Python?

- Free!
- It's easy to learn (my first language was Java)
- Large community to support, and extensive libraries and [documentation](#)
- [scikit-learn](#)
- Giants are using it (e.g. NASA, Disney, Netflix, Electronic Arts, Google etc.)
- Reasons:
  - *Machine learning and AI*
  - *Compatible with major platforms and systems*
  - *As a first language, easy to branch out to other languages*
- Sources: [PYPL](#), [TIOBE](#)

## PYPL PopularitY of Programming Language

Worldwide, Jun 2019 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.08 %	+4.7 %
2		Java	20.51 %	-1.8 %
3		Javascript	8.29 %	-0.2 %
4	↑	C#	7.41 %	-0.5 %
5	↓	PHP	6.96 %	-1.2 %
6		C/C++	5.76 %	-0.4 %

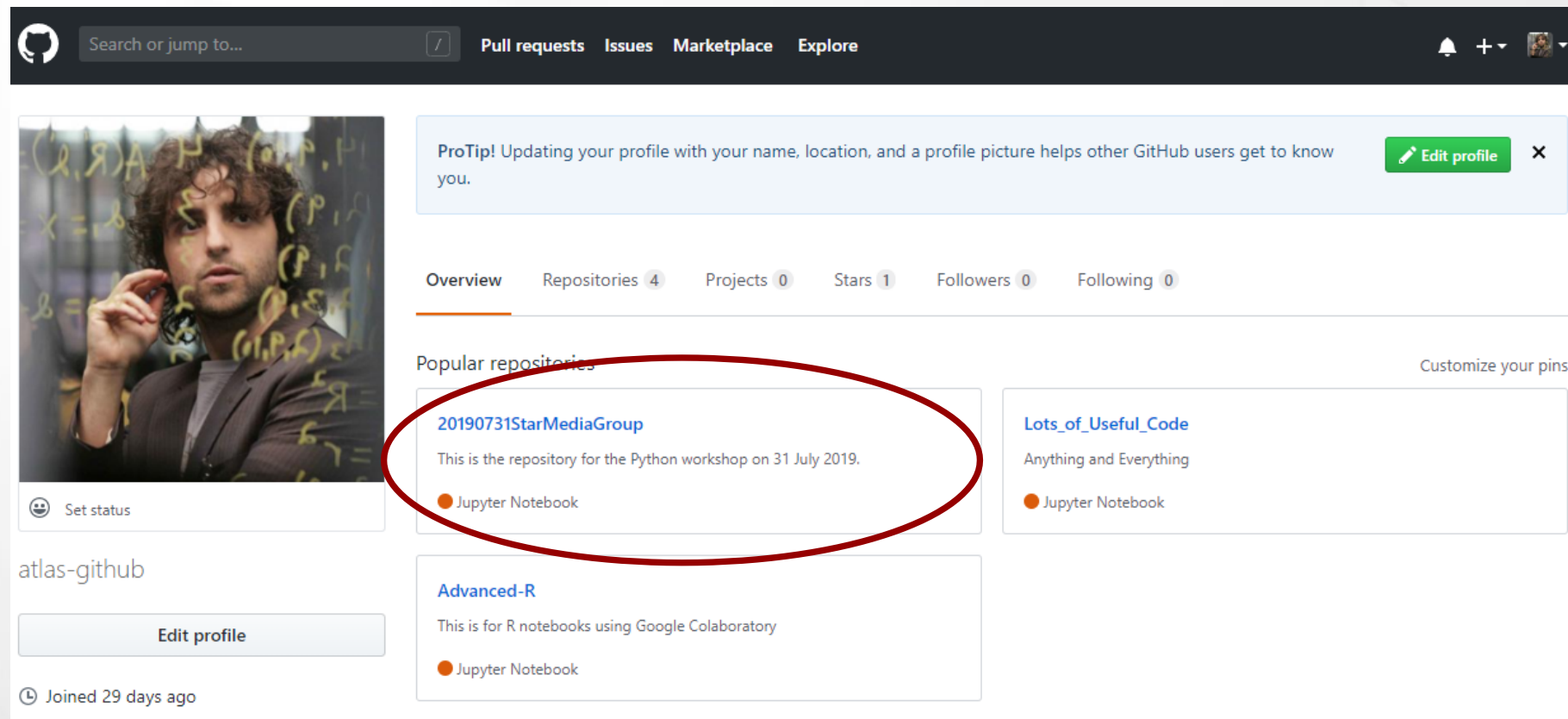
## TIOBE Index for June 2019

June Headline: Python continues to soar in the TIOBE index

Jun 2019	Jun 2018	Change	Programming Language	Ratings
1	1		Java	15.004%
2	2		C	13.300%
3	4	↑	Python	8.530%
4	3	↓	C++	7.384%
5	6	↑	Visual Basic .NET	4.624%
6	5	↓	C#	4.483%
7	8	↑	Javascript	2.716%

# GitHub:

■ [www.github.com/atlas-github](https://www.github.com/atlas-github)



The screenshot shows the GitHub profile page for a user named 'atlas-github'. The profile picture is a man with curly hair, and the bio area contains mathematical formulas. The 'Popular repositories' section is highlighted with a red oval, showing three repositories: '20190731StarMediaGroup', 'Advanced-R', and 'Lots\_of\_Useful\_Code'. The '20190731StarMediaGroup' repository is the first one listed and is circled in red. The 'Lots\_of\_Useful\_Code' repository is the second one listed. The 'Advanced-R' repository is the third one listed. The '20190731StarMediaGroup' repository is described as 'This is the repository for the Python workshop on 31 July 2019.' and is a Jupyter Notebook. The 'Advanced-R' repository is described as 'This is for R notebooks using Google Colaboratory' and is a Jupyter Notebook. The 'Lots\_of\_Useful\_Code' repository is described as 'Anything and Everything' and is a Jupyter Notebook. The user's profile shows 4 repositories, 0 projects, 1 star, 0 followers, and 0 following. The user joined 29 days ago.

Search or jump to... Pull requests Issues Marketplace Explore

**ProTip!** Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you. [Edit profile](#)

Overview Repositories 4 Projects 0 Stars 1 Followers 0 Following 0

Popular repositories Customize your pins

**20190731StarMediaGroup**  
This is the repository for the Python workshop on 31 July 2019.  
Jupyter Notebook

**Lots\_of\_Useful\_Code**  
Anything and Everything  
Jupyter Notebook

**Advanced-R**  
This is for R notebooks using Google Colaboratory  
Jupyter Notebook

atlas-github  
[Edit profile](#)  
Joined 29 days ago



# Data visualization using seaborn



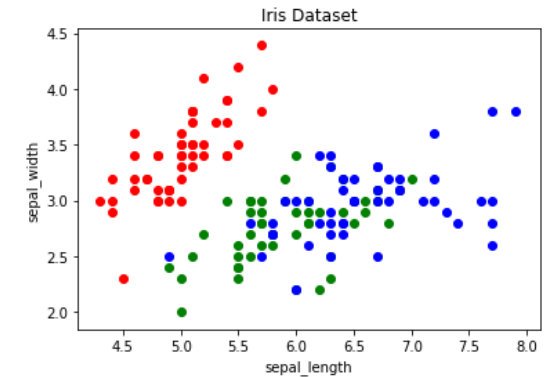
# What is seaborn?

- Numerous other data visualization packages
  - *Matplotlib, pandas visualization, ggplot, plotly*
- Reasons to use seaborn
  - *Can create graphs in 1 line of code which takes 10 in matplotlib*
  - *Awesome standard designs and colour palettes*
  - *Easy to learn [here](#)*
- Source: [Medium](#)

```
1 # create color dictionary
2 colors = {'Iris-setosa':'r', 'Iris-versicolor':'g', 'Iris-virginica':'b'}
3 # create a figure and axis
4 fig, ax = plt.subplots()
5 # plot each data-point
6 for i in range(len(iris['sepal_length'])):
7     ax.scatter(iris['sepal_length'][i], iris['sepal_width'][i], color=colors[iris['class'][i]])
8 # set a title and labels
9 ax.set_title('Iris Dataset')
10 ax.set_xlabel('sepal_length')
11 ax.set_ylabel('sepal_width')
```

matplotlib\_simple\_scatterplot\_with\_colors.py hosted with ❤ by GitHub

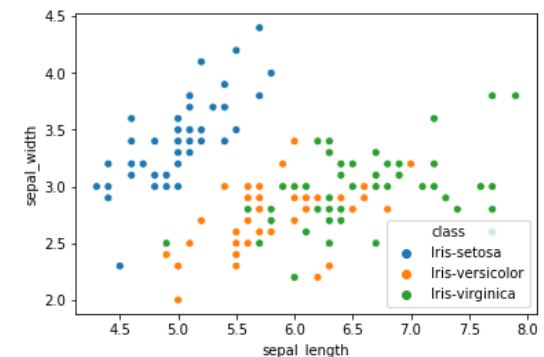
[view raw](#)



```
1 sns.scatterplot(x='sepal_length', y='sepal_width', hue='class', data=iris)
```

seaborn\_simple\_scatterplot\_colored.py hosted with ❤ by GitHub

[view raw](#)



# How to use seaborn?

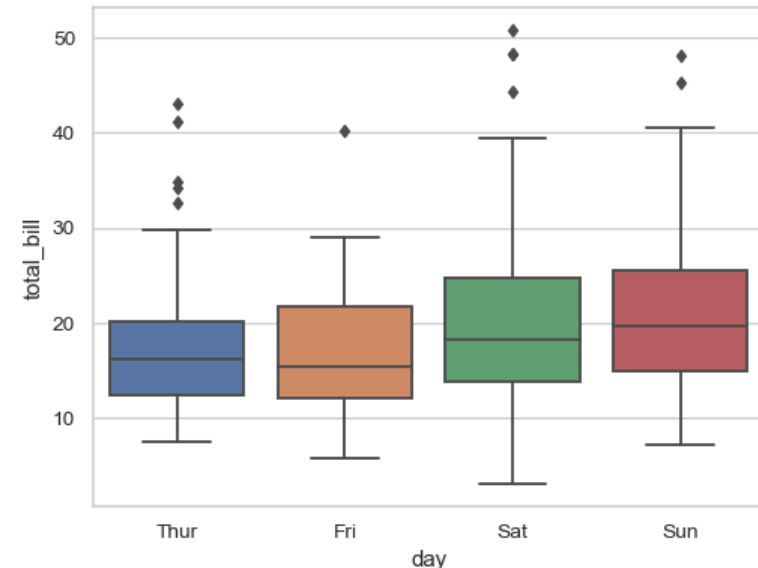
```
seaborn.boxplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None,
orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5,
linewidth=None, whis=1.5, notch=False, ax=None, **kwargs)
```

Parameters:

- x, y, hue** : names of variables in data or vector data, optional  
Inputs for plotting long-form data. See examples for interpretation.
- data** : DataFrame, array, or list of arrays, optional  
Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.
- order, hue\_order** : lists of strings, optional  
Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.
- orient** : "v" | "h", optional  
Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables, but can be used to specify when the "categorical" variable is a numeric or when plotting wide-form data.
- color** : matplotlib color, optional  
Color for all of the elements, or seed for a gradient palette.

Draw a vertical boxplot grouped by a categorical variable:

```
>>> ax = sns.boxplot(x="day", y="total_bill", data=tips)
```



# Hypothesis testing – z-test

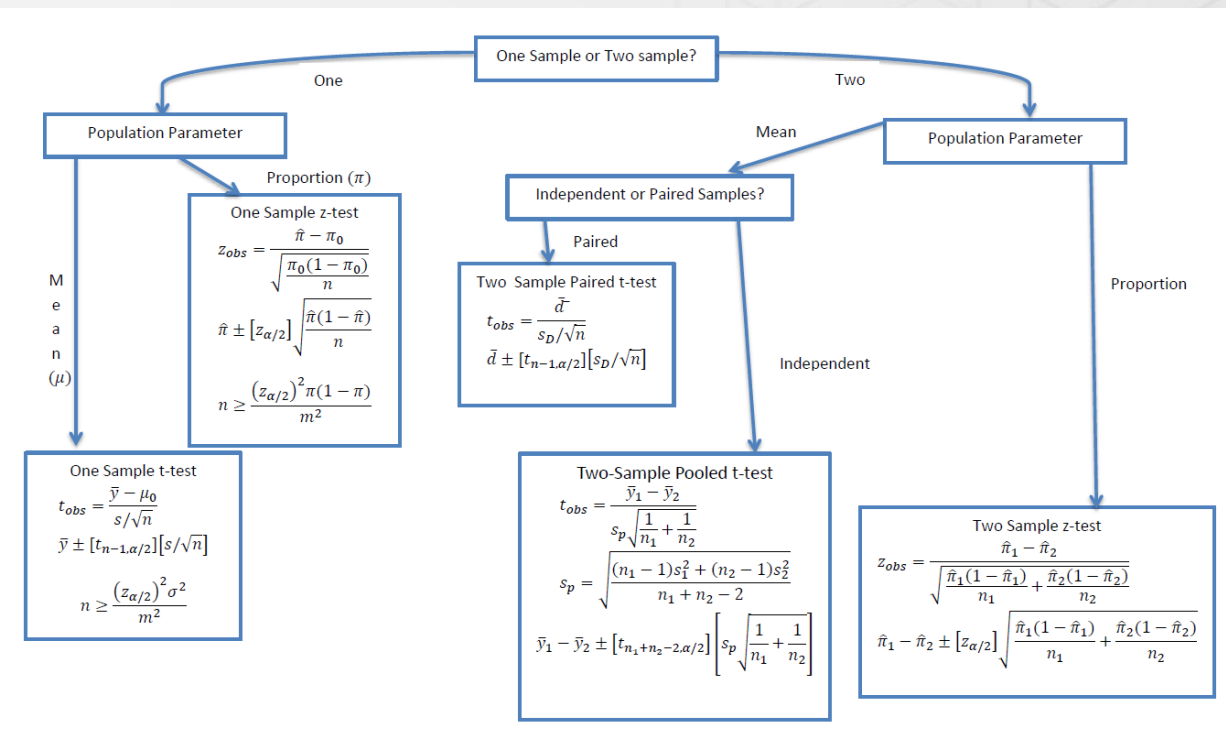


# Hypothesis-testing

## ■ A few types:

- *Normality*  
Sample size > 30, population variance known
- *T-test*  
Sample size < 30, population variance unknown
- *Chi-square test*  
To test whether variables are independent
- *ANOVA*  
To analyse the differences among group means in a sample

■ Source: [Towards Data Science](#)





# How to use statsmodels.stats?

```
statsmodels.stats.proportion.proportions_ztest(count, nobs, value=None, alternative='two-sided',  
prop_var=False)
```

## Parameters:

**count** : *integer or array\_like*

the number of successes in nobs trials. If this is array\_like, then the assumption is that this represents the number of successes for each independent sample

**nobs** : *integer or array-like*

the number of trials or observations, with the same length as count.

**value** : *float, array\_like or None, optional*

This is the value of the null hypothesis equal to the proportion in the case of a one sample test. In the case of a two-sample test, the null hypothesis is that  $\text{prop}[0] - \text{prop}[1] = \text{value}$ , where prop is the proportion in the two samples. If not provided value = 0 and the null is  $\text{prop}[0] = \text{prop}[1]$

**alternative** : *string in ['two-sided', 'smaller', 'larger']*

The alternative hypothesis can be either two-sided or one of the one-sided tests, smaller means that the alternative hypothesis is  $\text{prop} < \text{value}$  and larger means  $\text{prop} > \text{value}$ . In the two sample test, smaller means that the alternative hypothesis is  $p_1 < p_2$  and larger means  $p_1 > p_2$  where  $p_1$  is the proportion of the first sample and  $p_2$  of the second one.

## Examples

```
>>> count = 5  
>>> nobs = 83  
>>> value = .05  
>>> stat, pval = proportions_ztest(count, nobs, value)  
>>> print('{0:0.3f}'.format(pval))  
0.695
```

```
>>> import numpy as np  
>>> from statsmodels.stats.proportion import proportions_ztest  
>>> count = np.array([5, 12])  
>>> nobs = np.array([83, 99])  
>>> stat, pval = proportions_ztest(count, nobs)  
>>> print('{0:0.3f}'.format(pval))  
0.159
```

# Contingency table & chi-square test

# Contingency table and chi-square test

[www.sli.do](http://www.sli.do): #V817

## ■ Contingency table

- *Aka. cross tabulation*
- *A table in a matrix format to display frequency distribution of variables*

## ■ Chi-square test

- *Aka. Pearson's chi-squared test*
- *Used to determine whether there is significant difference between expected and observed frequencies in one or more categories*

## ■ Applications:

- *Data exploration*

## ■ Source: [Statistics How To](#)

**AIDS \* SEXPREF Crosstabulation**

Count		SEXPREF			Total
		Males	Females	Both	
AIDS	Yes	4	2	3	9
	No	3	16	2	21
Total		7	18	5	30

**Chi-Square Tests**

	Value	df	Asymp. Sig. (2-tailed)
Pearson Chi-Square	7.657 <sup>a</sup>	2	.022
Likelihood Ratio	7.803	2	.020
Linear-by-Linear Association	.062	1	.803
N of Valid Cases	30		

a. 4 cells (66.7%) have expected count less than 5. The minimum expected count is 1.50.

# How to use scipy.stats?

```
scipy.stats.chi2_contingency(observed, correction=True, lambda_=None)
```

Parameters:

**observed** : *array\_like*

The contingency table. The table contains the observed frequencies (i.e. number of occurrences) in each category. In the two-dimensional case, the table is often described as an “R x C table”.

**correction** : *bool, optional*

If True, and the degrees of freedom is 1, apply Yates’ correction for continuity. The effect of the correction is to adjust each observed value by 0.5 towards the corresponding expected value.

**lambda** : *\_float or str, optional.*

By default, the statistic computed in this test is Pearson’s chi-squared statistic. *lambda\_* allows a statistic from the Cressie-Read power divergence family to be used instead. See [power\\_divergence](#) for details.

	Pizza Rolls	Chips & Dip	Cookies	Total
Poker	10	3	12	25
Trivial Pursuit	8	14	7	29
Monopoly	14	17	7	38
Wii Bowling	12	7	4	23
<b>Total</b>	<b>44</b>	<b>41</b>	<b>30</b>	<b>115</b>

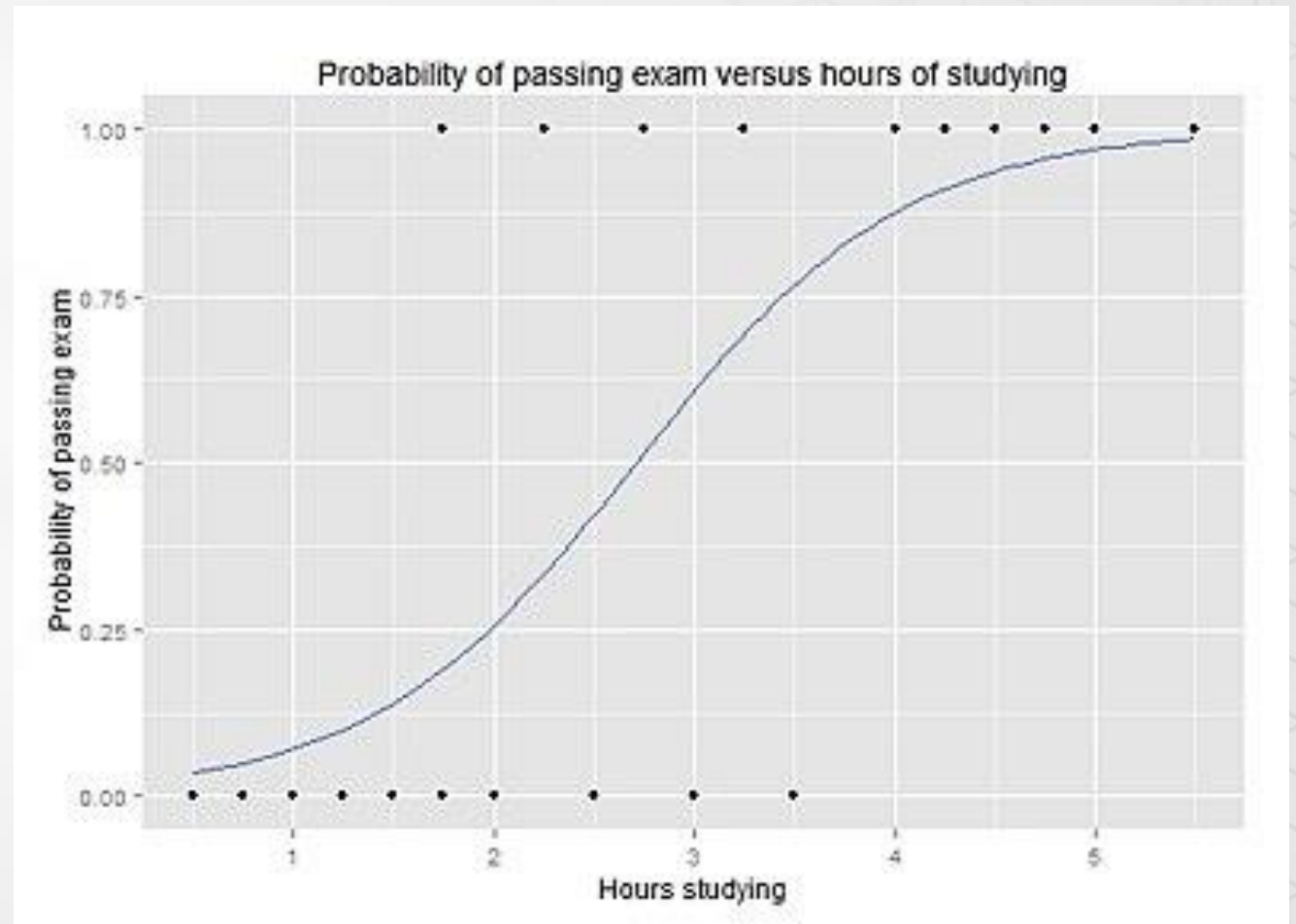
# Logistic regression





# Logistic regression

- Output is a probability a given input belongs to a certain class
- A type of classification algorithm
  - Binary labels: logistic regression
  - More than 2 labels: multinomial logistic regression
- Applications:
  - Profile behaviour
  - Which factors contribute to a certain health condition (yes vs. no)?



# How to use sklearn.linear\_model?

```
sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='warn', max_iter=100, multi_class='warn', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)
```

## Parameters:

**penalty** : str, 'l1', 'l2', 'elasticnet' or 'none', optional (default='l2')  
Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.  
New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

**dual** : bool, optional (default=False)  
Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual=False when n\_samples > n\_features.

**tol** : float, optional (default=1e-4)  
Tolerance for stopping criteria.

**C** : float, optional (default=1.0)  
Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

## Examples

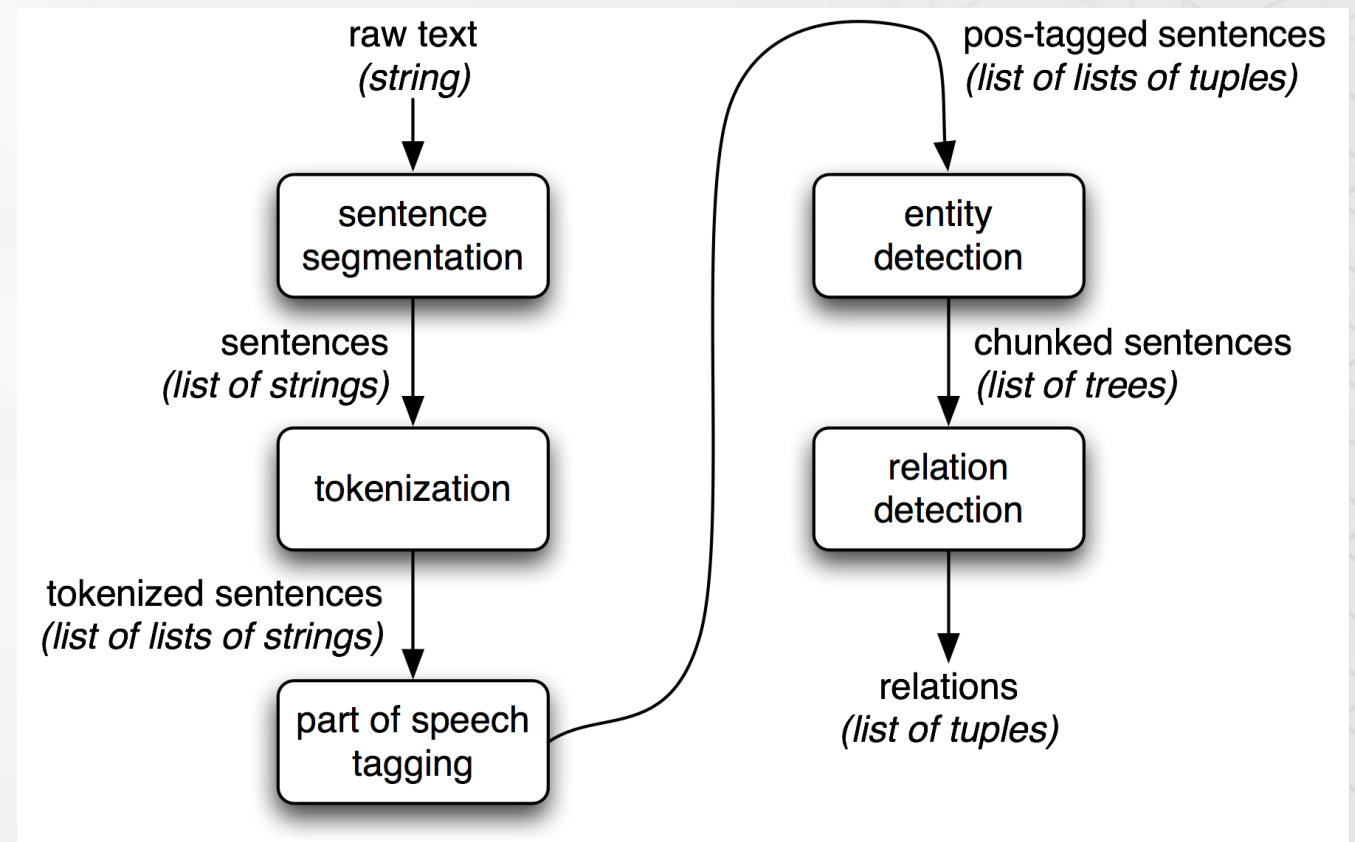
```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0, solver='lbfgs',
...                          multi_class='multinomial').fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

# Natural Language Processing



# Natural Language Processing (NLP)

- How to programme computers to process and analyse large amounts of unstructured text
- Applications:
  - Text classification
  - Behaviour profile of users
  - Taxonomy construction
  - [Named Entity Recognition](#)
  - Natural Language Generation (WIP)
- Advanced applications:
  - [TalkToTransformer](#)
  - [BusUncle](#)
  - [OpenAI GPT-2](#)



# How to use sklearn.naive\_bayes?

sklearn.naive\_bayes. **MultinomialNB(alpha=1.0, fit\_prior=True, class\_prior=None)**

## Parameters:

**alpha** : float, optional (default=1.0)

Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

**fit\_prior** : boolean, optional (default=True)

Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

**class\_prior** : array-like, size (n\_classes,), optional (default=None)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

## Examples

```
>>> import numpy as np
>>> X = np.random.randint(5, size=(6, 100))
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> from sklearn.naive_bayes import MultinomialNB
>>> clf = MultinomialNB()
>>> clf.fit(X, y)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
>>> print(clf.predict(X[2:3]))
[3]
```



# Google Cloud Platform's AutoML



# Recommendations and Feedback

- Need to continuously learn independently
- Meet up with others to look for ideas and inspiration
- Facebook:
  - [TensorFlow & Deep Learning Malaysia](#)
  - [Malaysia R User Group \(MYRUG\)](#)
- Meetup:
  - [Kuala Lumpur Artificial Intelligence Meetup](#)
- Workshop feedback:
  - *QR code on your tables*😊

**Thank you!**

***Star***  
MEDIA GROUP