Microsoft

Using Machine Learning Models

数据科学 – 机器学习模型入门

线性回归与分类 Linear Regression & Classification

Nov 2020

Microsoft Reactor | Ryan Chung

Microsoft

Ryan Chung

Instructor / DevelopIntelligence
Founder / MobileDev.TW

@ryanchung403 on WeChat
Ryan@MobileDev.TW

Microsoft

Reactor

developer.microsoft.com/reactor/

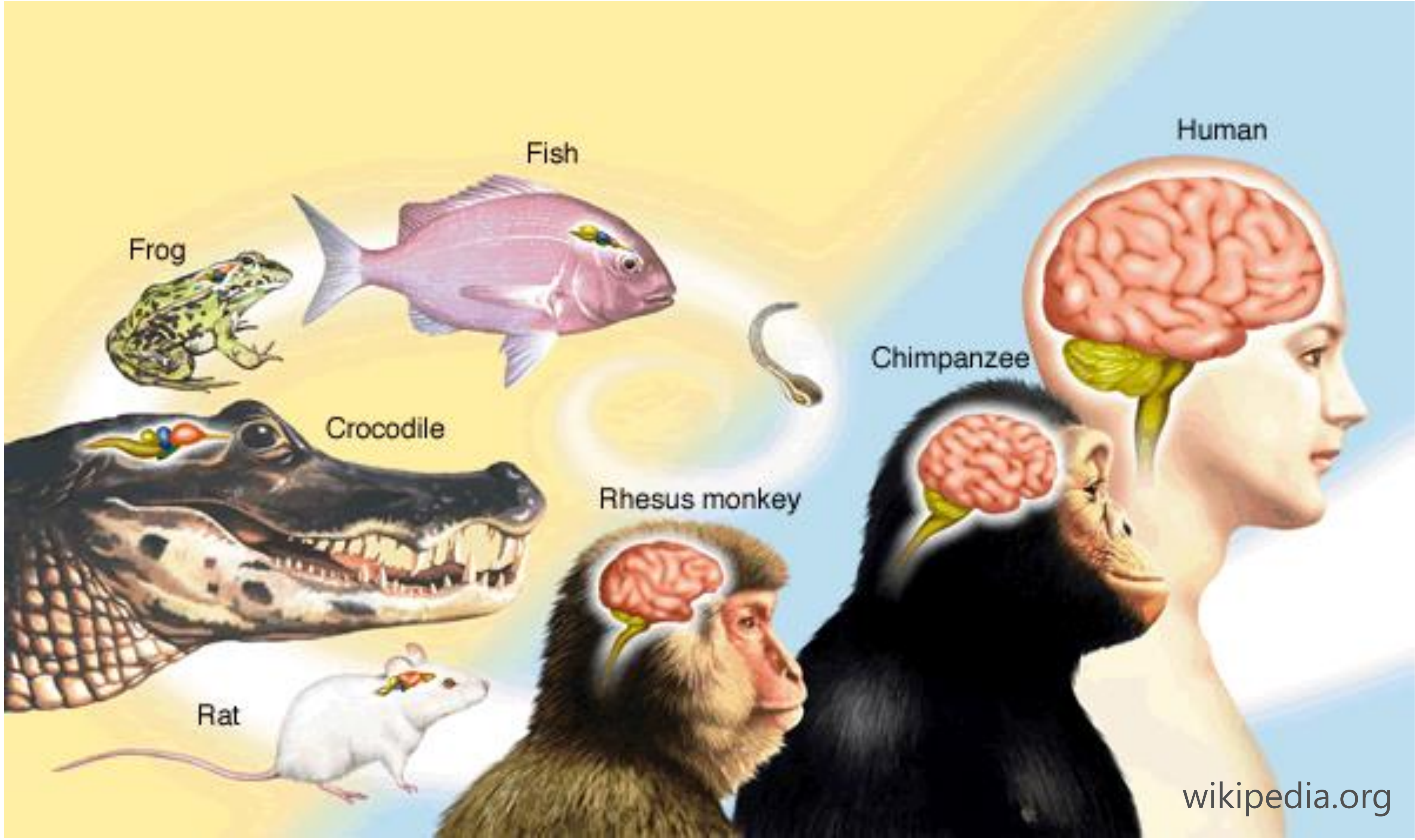
@MSFTReactor on Twitter

前置作业

- 先把待会需要用到的模组都汇入

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from sklearn import metrics
from sklearn.metrics import r2_score

from sklearn.datasets import load_iris
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
```

项目一：大脑 VS. 身体


Fish
Human
Frog
Chimpanzee
Crocodile
Rhesus monkey
Rat
wikipedia.org

载入资料

```python
mammals = pd.read_csv('Data/mammals.csv')
mammals.head()
```

	Mammal	body	brain
0	Arctic fox	3.385	44.5
1	Owl monkey	0.480	15.5
2	Mountain beaver	1.350	8.1
3	Cow	465.000	423.0
4	Grey wolf	36.330	119.5

kg g

北极狐

猫头鹰猴

山河狸

牛

灰狼

载入资料

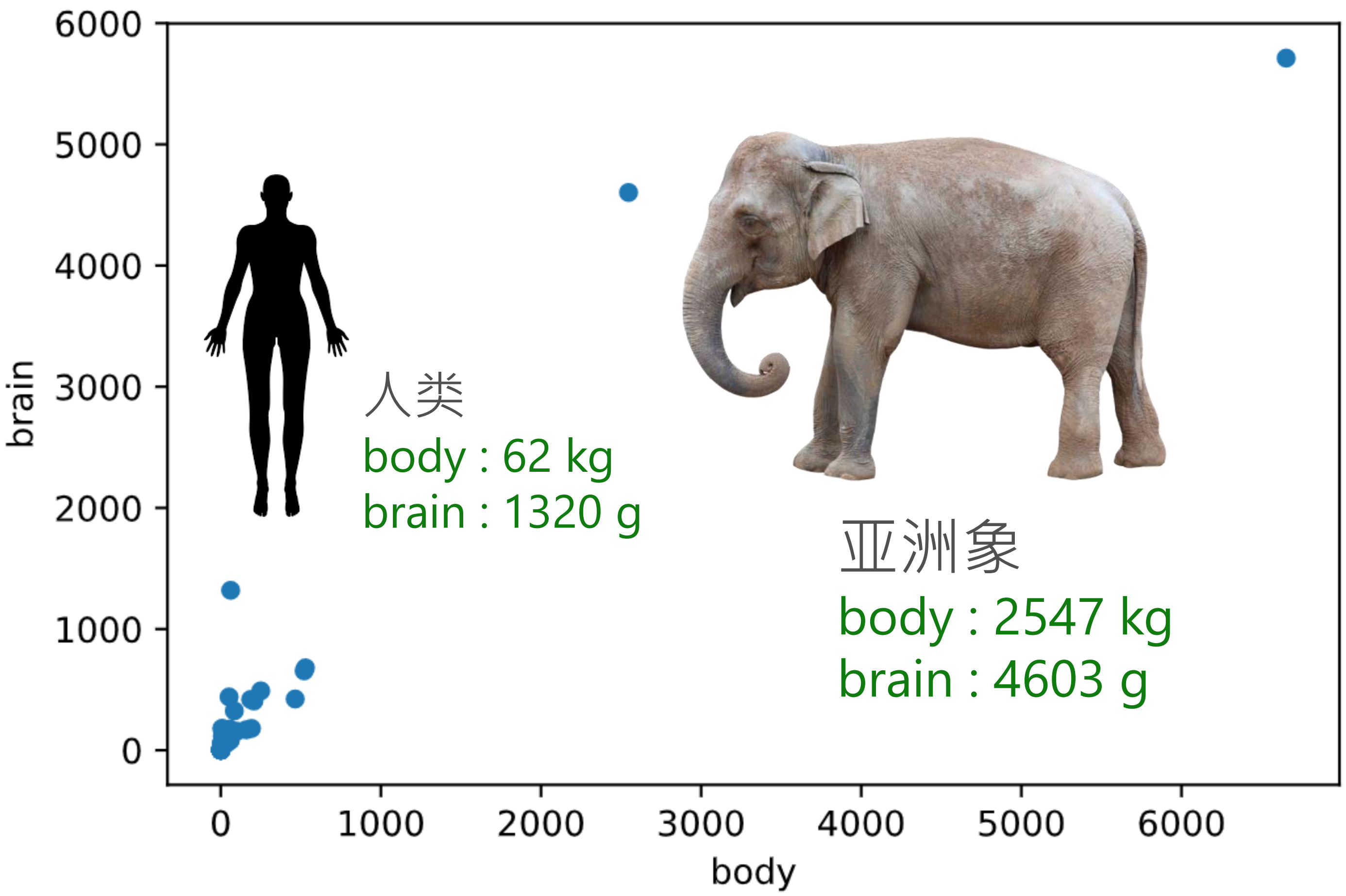
```
mammals.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Mammal  62 non-null     object
 1   body    62 non-null     float64
 2   brain   62 non-null     float64
dtypes: float64(2), object(1)
memory usage: 1.6+ KB
```

用图表来观察

- body单位是kg、braing单位是g

```
mammals.plot.scatter(x='body',y='brain')
```

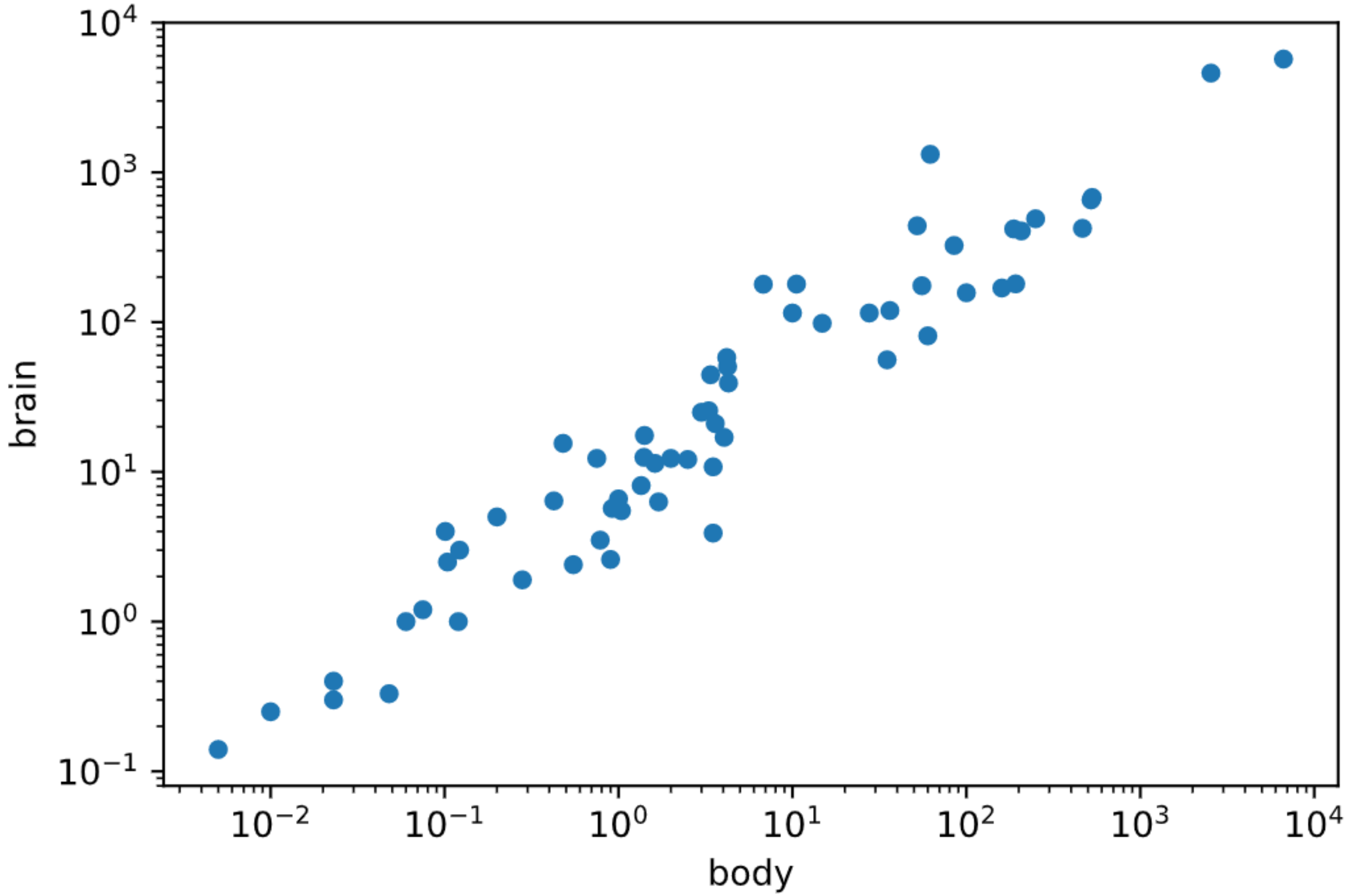


非洲象
body : 6654 kg
brain : 5712 g

用图表来观察

- 各别取log来看看效果

```
mammals.plot.scatter(x='body',y='brain',loglog=True)
```



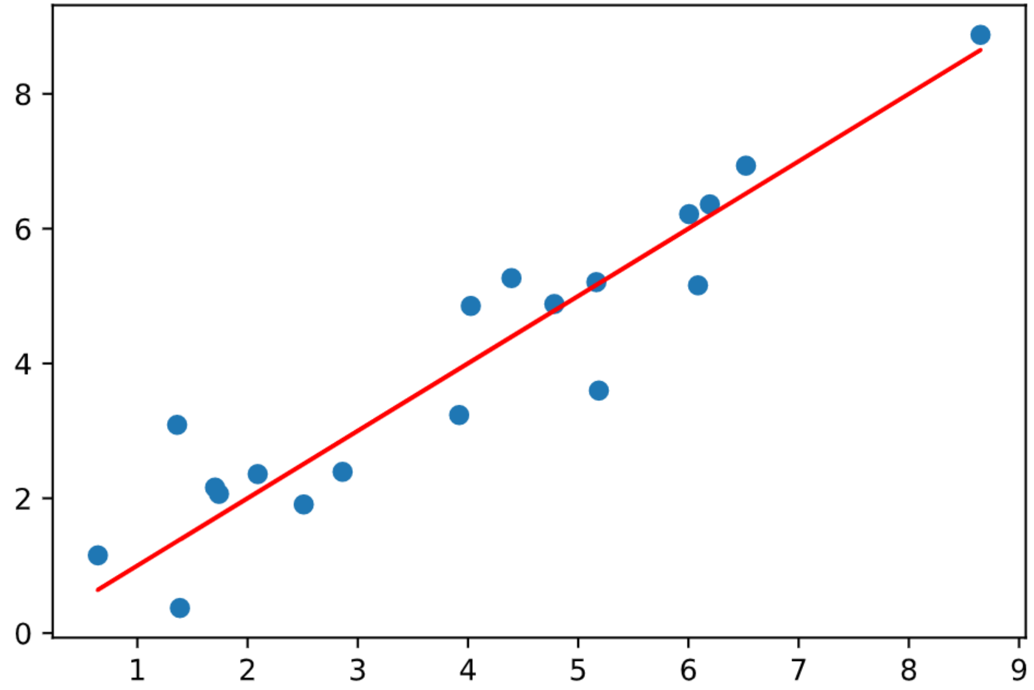
loglog

x 跟 y 都取log

准备资料集，开始进行模型训练与评估

```
mammals['body_log'] = np.log(mammals['body'])
mammals['brain_log'] = np.log(mammals['brain'])

X = mammals[['body_log']]
y = mammals['brain_log']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 0)

reg = LinearRegression()

reg.fit(X_train, y_train)

r2_score(y_test, reg.predict(X_test))

plt.scatter(y_test, reg.predict(X_test))
plt.plot(y_test, y_test, color='red')
```

```
0.8757444044097347
```



项目二：
猞猁属年度捕获数量

- Lynx (猞猁属)
- 数据期间：1821 ~ 1934
- 每年被捕获的数量
- 地点：加拿大马更些河域

基本观察

```python
lynx = pd.read_csv('Data/lynx.csv')
lynx.head()
lynx.info()
lynx.describe()
```

```
[25]  lynx.head()
```

	Year	Lynx
0	1821	269
1	1822	321
2	1823	585
3	1824	871
4	1825	1475

```
[26]  lynx.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114 entries, 0 to 113
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Year    114 non-null    int64
 1   Lynx    114 non-null    int64
dtypes: int64(2)
memory usage: 1.9 KB
```

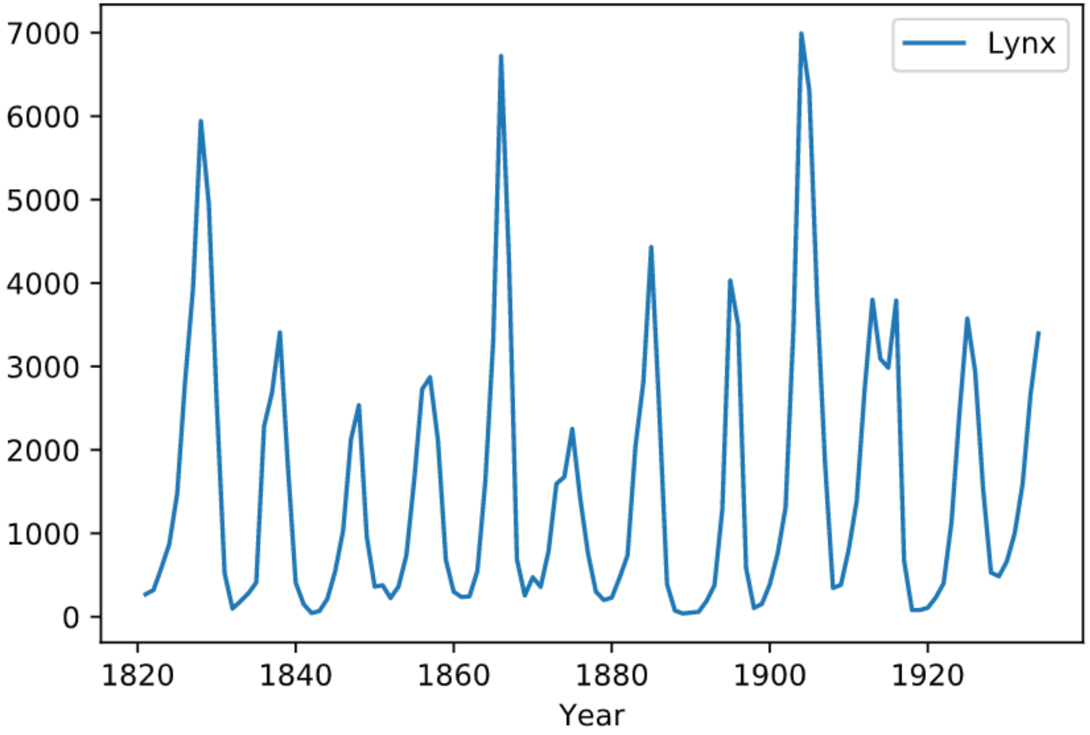
```
[27]  lynx.describe()
```

	Year	Lynx
count	114.000000	114.000000
mean	1877.500000	1538.017544
std	33.052988	1585.843914
min	1821.000000	39.000000
25%	1849.250000	348.250000
50%	1877.500000	771.000000
75%	1905.750000	2566.750000
max	1934.000000	6991.000000

年份与捕获数量

- 高高低低，来回震荡
- 食物充足 -> 食物短缺

```
lynx.plot(x='Year',y='Lynx')
```



尝试使用Linear Regression来预测

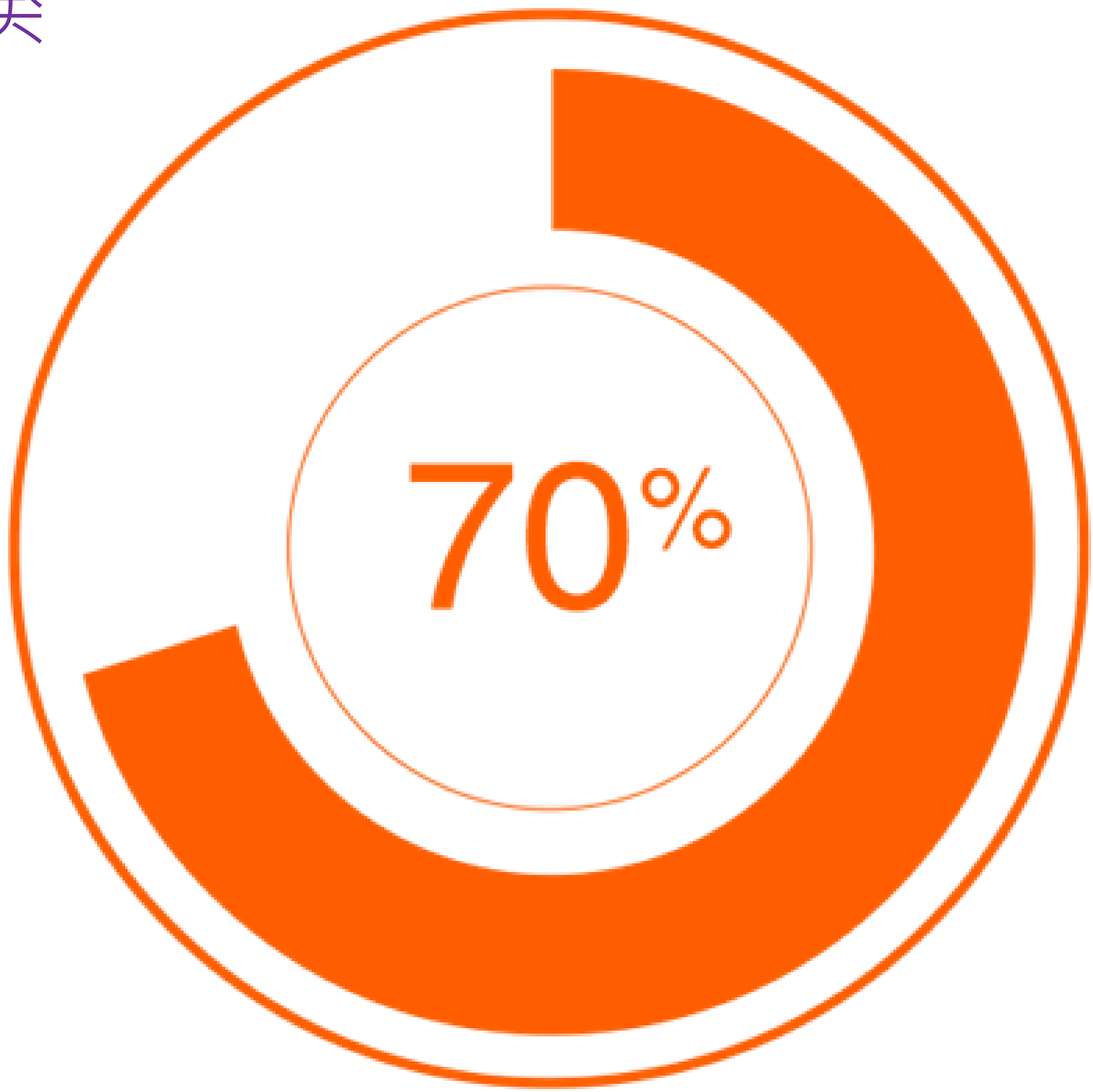
- 结果惨不忍睹…

```
X_lynx = lynx[['Year']]
y_lynx = lynx['Lynx']

X_lynx_train, X_lynx_test, y_lynx_train, y_lynx_test
 = train_test_split(X_lynx, y_lynx, test_size = 0.3,
 random_state=0)

lr = LinearRegression()
lr.fit(X_lynx_train, y_lynx_train)

r2_score(y_lynx_test, lr.predict(X_lynx_test))
```

```
[42] r2_score(y_lynx_test, lr.predict(X_lynx_test))

-0.008784749085493981
```

项目三：花卉分类

- 高达70%的数据科学专案与分类有关



Anderson's Iris data set / Iris flower data set
安德森鸢尾花卉数据集

样本数：150

类别：0-Setosa 山鸢尾、1-Versicolour 变色鸢尾、2-Virginica 维吉尼亚鸢尾

	花萼长度	花萼宽度	花瓣长度	花瓣宽度	类别
index ▲	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0

DataFrame 资讯探索

```
iris = load_iris()
iris
iris_df = pd.DataFrame(data = np.c_[iris['data'],iris['target']], columns = iris['feature_names']+['class'])
```

```
iris_df.head()
```

head() 取最前面几笔(预设值5笔)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

```
iris_df.tail()
```

tail() 取最后面几笔(预设值5笔)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
147	6.5	3.0	5.2	2.0	2.0
148	6.2	3.4	5.4	2.3	2.0
149	5.9	3.0	5.1	1.8	2.0

```
iris_df.info()
```

info() 资料集摘要资讯

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
class                150 non-null float64
dtypes: float64(5)
memory usage: 5.9 KB
```

```
iris_df.shape
```

shape 维度
(150笔, 5个栏位)

```
(150, 5)
```

```
iris_df['sepal length (cm)'].mean()
```

mean() 计算平均值

```
5.843333333333335
```

数据切割

```
X_iris = iris_df[['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']]
y_iris = iris_df['class']

X_iris_train, X_iris_test, y_iris_train, y_iris_test = train_test_split(X_iris, y_iris, test_size = 0.3,
 random_state=0)
```

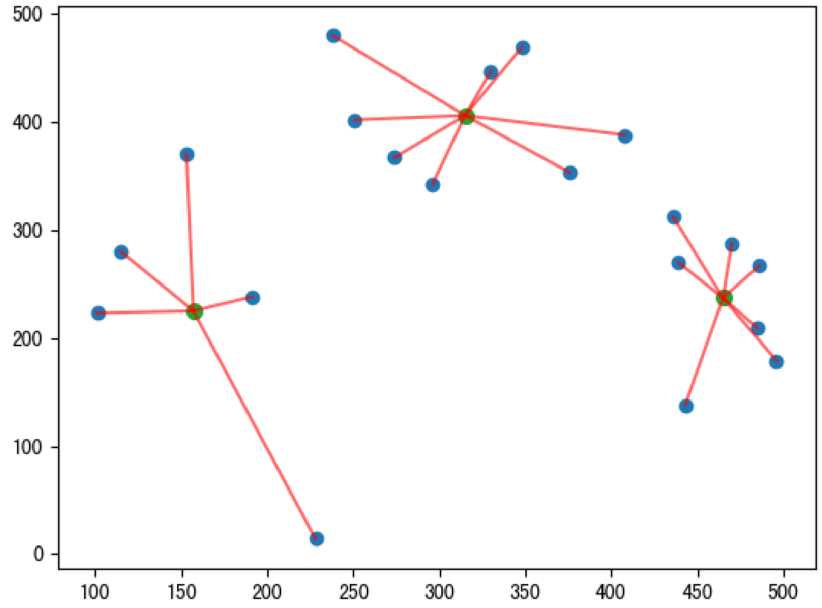
K-means Clustering

- 结果不是很漂亮

n_clusters
要分成几群

```python
this_KMeans = KMeans(n_clusters=3, random_state=0)
this_km = this_KMeans.fit(X_iris_train)
y_pred = this_km.predict(X_iris_test)

metrics.accuracy_score(y_iris_test,y_pred)
```

```
[91]  metrics.accuracy_score(y_iris_test,y_pred)

0.08888888888888889
```



K-means Clustering

- 尝试调整random_state，突然变得很厉害!

n_clusters
要分成几群

```
this_KMeans = KMeans(n_clusters=3, random_state=1)
this_km = this_KMeans.fit(X_iris_train)
y_pred = this_km.predict(X_iris_test)

metrics.accuracy_score(y_iris_test,y_pred)
```

```
[95] metrics.accuracy_score(y_iris_test,y_pred)

     0.9111111111111111
```

K-Nearest Neighbors

用你的邻居来看你是哪一类的人

- 换一个方法试试，结果非常亮眼!

```
this_KNC = KNeighborsClassifier(n_neighbors=5)
this_KNC_model = this_KNC.fit(X_iris_train, y_iris_train)
y_knc_pred = this_KNC_model.predict(X_iris_test)

metrics.accuracy_score(y_iris_test,y_knc_pred)
```

```
[99]  metrics.accuracy_score(y_iris_test,y_knc_pred)

      0.9777777777777777
```

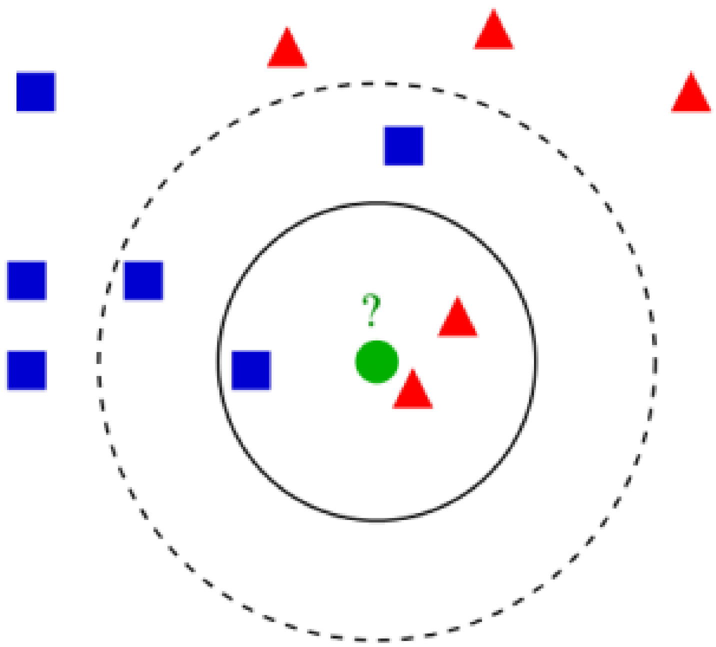


Photo: Antti Ajanki AnAj
via Wikimedia Commons, CC

K-Nearest Neighbors (KNN)

- 变更切割中的random_state会不会有影响?

```
X_iris_train, X_iris_test, y_iris_train, y_iris_test = train_test_split
(X_iris, y_iris, test_size = 0.3, random_state=2)

this_KNC = KNeighborsClassifier(n_neighbors=5)
this_KNC_model = this_KNC.fit(X_iris_train, y_iris_train)
y_knc_pred = this_KNC_model.predict(X_iris_test)

metrics.accuracy_score(y_iris_test,y_knc_pred)
```

```
[104]  metrics.accuracy_score(y_iris_test,y_knc_pred)

       1.0
```

大量测试数据切割造成的影响性

- 利用cross_val_score方法来进行交叉验证

```
scores = cross_val_score(this_KNC_model,X_iris, y_iris, cv=10, scoring='accuracy')
scores
scores.mean()
```

```
[113]  scores

array([1.        , 0.93333333, 1.        , 1.        , 0.86666667,
       0.93333333, 0.93333333, 1.        , 1.        , 1.        ])

[114]  scores.mean()

0.9666666666666668
```

CV
将资料随机平均分成n个集合，一个集合当作测试资料，剩下的都作为训练资料

KNN的n_neighbors该设定为什么值?

- 抓个范围，全部跑一跑测试看看!

```
k_range = list(range(1,26))
k_dict = {}
for k in k_range:
    this_KNC = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(this_KNC,X_iris, y_iris, cv=10, scoring='accuracy')
    k_dict[k] = scores.mean()

scores_max = max(k_dict, key=k_dict.get)
print(scores_max, k_dict[scores_max])
```

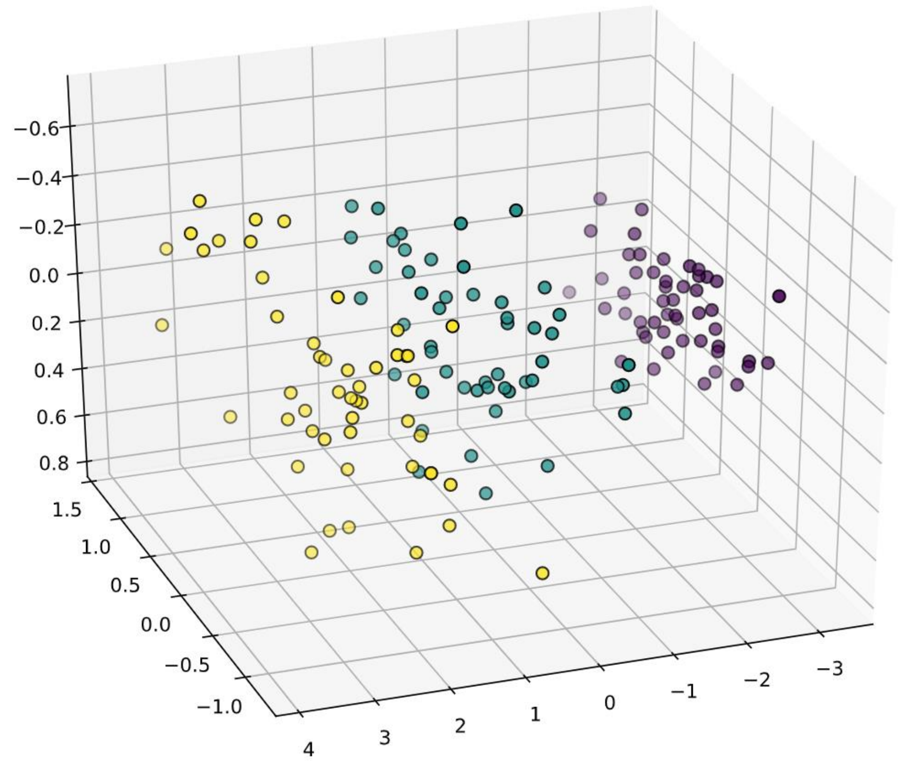
key = k_dict.get
依据dictionary的值来排序

```
[117] print(scores_max, k_dict[scores_max])

13 0.9800000000000001
```

PCA 转换

```
X_iris_reduced = PCA(n_components=3).fit_transform(X_iris)

fig = plt.figure(1, figsize=(8,6))
ax = Axes3D(fig, elev=-150, azim=110)
ax.scatter(X_iris_reduced[:, 0],X_iris_reduced[:,1],X_iris_reduced[:,2],
c=y_iris, cmap='viridis',edgecolors='k',s=40)
```

iris数据集有4个特征值
利用PCA降至3维来呈现



c
决定不同的着色

cmap
Colormap 色调与数值变化

edgecolors
点的边框颜色 (black)

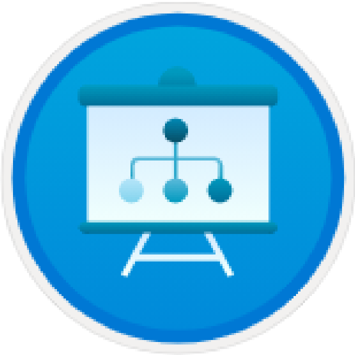
s
点的大小

elev
海拔视角(默认值30)

azim
方位视角(默认值-60)

小结

- 数据的初步观察很重要，有些数据集的特性并不适合用线性回归
- 数据集的训练与测试切割，也可能会影响预测准确度
- 模型的参数、模型的选择、数据的特性，都牵动着最终结果的产出

Microsoft

Reactor

developer.microsoft.com/reactor/

@MSFTReactor on Twitter

Microsoft

议程结束
感谢聆听

请记得填写课程回馈问卷 (Event ID : 11919)

https://aka.ms/Reactor/Survey