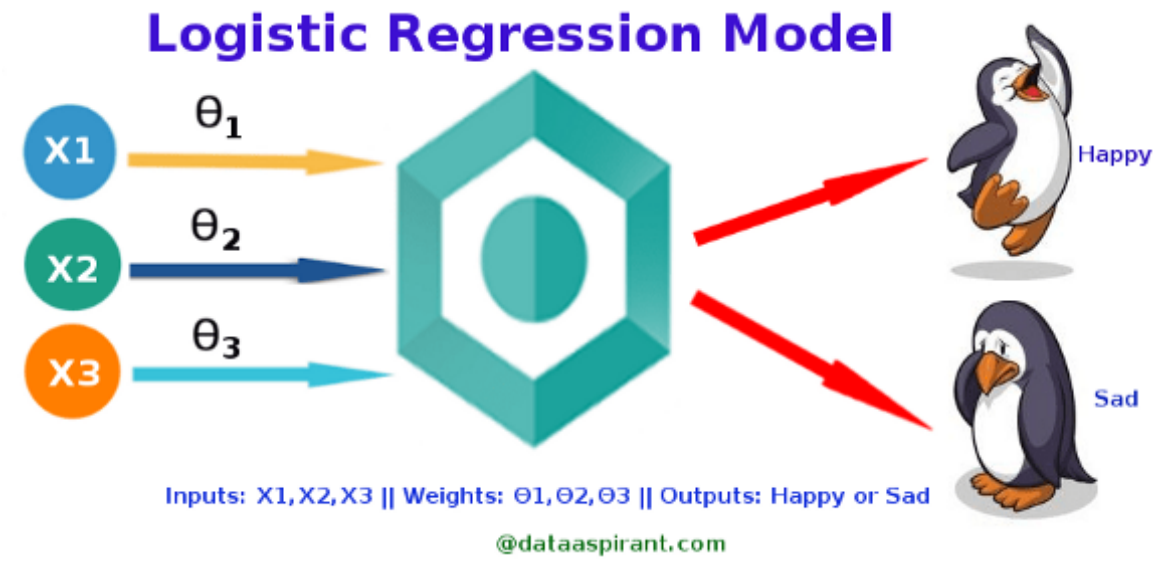
Logistic Regression ทำนาย
นิค : Machine Learning 101

Mr.P L Follow
Nov 10, 2018 · 5 min read

> อะไรที่ว่าใช่ ถ้าลองมองดีๆมันอาจไม่ใช่ก็ได้ ....



หลังจากที่ลองเล่น Machine Learning แบบ Regression มาจนเข้าใจแล้ว ทีนี้ก็กลับมาที่ Classification อีกรอบ แต่รอบนี้จะแปลกหน่อยเพราะชื่อ ML ตัวนี้คือ Logistic Regression แต่ทำงานเป็น Classification ..... เหมือนเจอดัก 5555

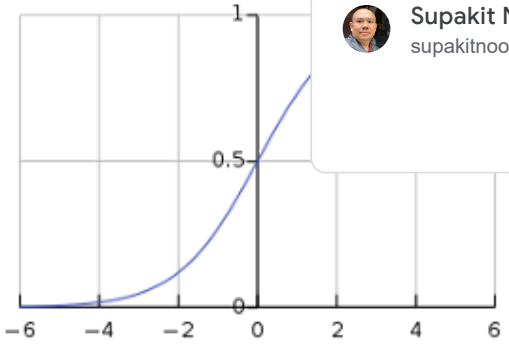
แต่มันมีเหตุผลอยู่ว่าทำไมถึงเป็น Regression ที่ได้ผลลัพธ์เป็น Classification โดยเจ้า LR เนี่ยมีคณิตศาสตร์เบื้องหลัง+output ที่แปลกที่สุดในบันดาล ML เลยก็ว่าได้ เพราะมันมี Sigmoid function และ Classification ที่สามารถทำนายได้แค่ 2 Class เท่านั้น ....

Logistic Regression

แปลไทยแบบตลกๆคือ การขนส่งแบบถดถอย... เออแปลกดีแต่ความจริงแล้วมันคือการทำ Classification โดย output จริงๆแล้วออกมาเป็น Regression (จำนวนตัวเลข) จากนั้นนำไป mapping กับ Sigmoid function (หลักคณิตศาสตร์) ก็จะได้ว่าข้อมูลอันนี้เป็นคลาสนี้ หรือ ไม่ได้เป็นคลาสนี้นั้นเอง (เป็นไปได้แค่ 2 คลาสเท่านั้น)

Regression ตรงไหน ?

Output ของ Logistic Regression ความจริงแล้วเป็นจำนวนตัวเลขนะครับแต่เมื่อนำมาใช้งานร่วมกับ Sigmoid หน้าตาแบบนี้



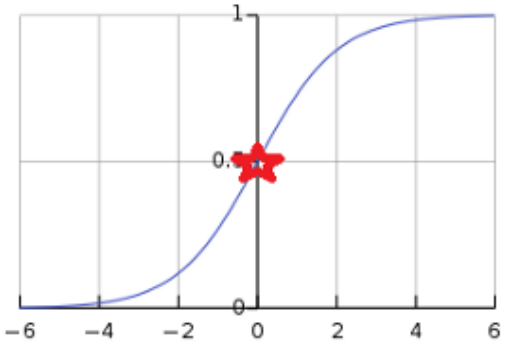
แกน x เป็น output ของ Regression แกน y เป็นความน่าจะเป็น

ทีนี้สมมุติ Output จริงๆของเราทายออกมาได้ 0

แกน Y คือ ความน่าจะเป็นที่จะเป็น Class นั้นๆ อยู่ในช่วง 0–1

แกน X คือ ค่า Regression ที่ทายออกมาได้

เราเอา Output มาเทียบกับกราฟในรูปก็จะได้แกน Y เป็น 0.5



ถ้าความน่าจะเป็นอยู่ในช่วง **0.0–0.49** ก็จะได้ว่าไม่เป็นคลาสนั้นๆ

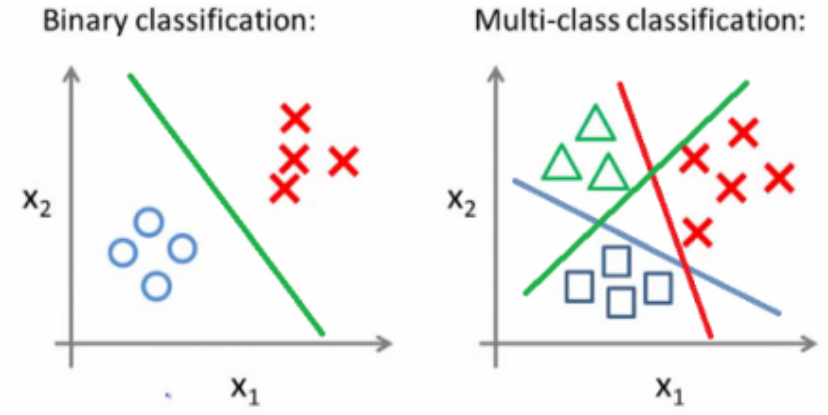
ถ้าความน่าจะเป็นอยู่ในช่วง **0.49–1.00** ก็จะได้ว่าเป็นคลาสนั้นๆ

โดยที่ กฎของความน่าจะเป็นค่าจะต้องไม่เกิน 1 โดยเจ้า Logistic Regression สามารถให้ output ออกมาได้ว่าเป็นคลาสนี้ หรือ ไม่เป็นคลาสนี้โดยการใช้ Sigmoid function และมันยังสามารถให้ output ออกเป็น “**ความน่าจะเป็น**” ของคลาสนั้นๆได้อีกด้วย

ทำไมถึงทำนายได้แค่ 2 คลาส ?

จากรูปจะเห็นได้ว่าแกน y คือ ความน่าจะเป็นของคลาสนั้น โดยจุดกลางของกราฟคือ 0.5 ทำให้แบ่งได้ 2 คลาสคือ เป็นคลาสนั้น และไม่เป็นคลาสนั้น

มีวิธีทำนายได้หลายๆคลาสไหม ?



ทำได้ครับ โดยใช้ Sigmoid Function จำนวน X อัน โดยที่ X คือจำนวนคลาสที่ต้องการมาบอกว่าเป็นคลาสนั้นๆหรือไม่ แต่วิธีนี้เราจะได้เห็นชัดๆใน Neural Network

**Multi layer Percepton : ใช้ Sigmoid สำหรับทำ Multi-Class Classification

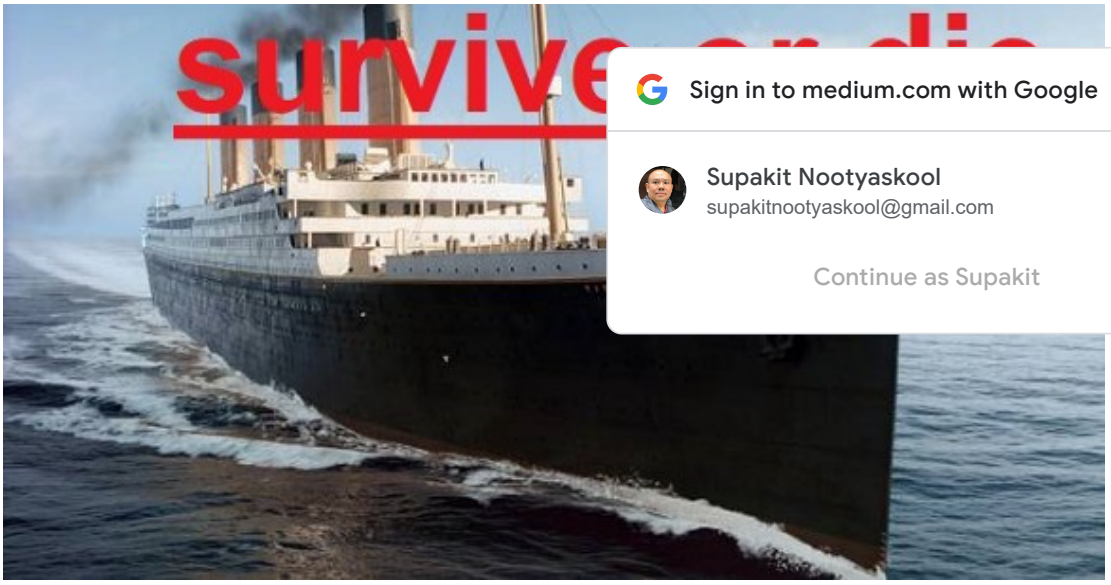
คณิตศาสตร์เบื้องหลัง

มาถึงตรงนี้ทุกคนคงจะเข้าใจหน้าที่ของ Sigmoid กันแล้วหน้าตาของมันก็ประมาณนี้

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$

คือการนำ output ที่เป็นตัวเลขมาแมพกับ Sigmoid ฟังก์ชั่นนั้นเอง

ลงมือทดลอง



วันนี้จะใช้ Titanic Dataset โดยจะทำนายว่าผู้โดยสารจะมีโอกาสรอดจากเหตุการณ์เรือ Titanic ล่มมากขนาดไหน

โหลด **dataset.csv** ให้เรียบร้อยก่อน

peeratpop/Machine_Learning_0-100

Contribute to peeratpop/Machine_Learning_0-100 development by creating an account on GitHub.

github.com

ทีนี้เปิด **Jupyter notebook / VSCode** แล้วลองสำรวจไฟล์ dataset กันก่อน

```
import pandas as pd
f=pd.read_csv('titanic_data.csv')
f.shape
```

891 คือจำนวนข้อมูลที่มี ส่วน 12 คือหัวข้อที่มี

```
import pandas as pd
f=pd.read_csv('titanic_data.csv')
f.shape
```

```
(891, 12)
```

```
f.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

แต่ในชีวิตการทำงานกับ data จริงๆเราจะต้องทำการตรวจสอบข้อมูลก่อน
ถ้าเราลองสังเกตุข้อมูลใน data ของเราจะพบว่ามันไ
เลยแม้แต่น้อย ถ้าเราลองสังเกตุค่า “ว่าง” (NULL)


```
f.isnull().sum()
```

```
f.isnull().sum()

Pclass      0
Fare        0
Survived    0
Sex         0
Age       177
dtype: int64
```

ช่องของอายุว่างถึง 177 แถว

```
f[f['Age'].isnull()]
```

826	3	56.4958	0	male	NaN
828	3	7.7500	1	male	NaN
832	3	7.2292	0	male	NaN
837	3	8.0500	0	male	NaN
839	1	29.7000	1	male	NaN
846	3	69.5500	0	male	NaN
849	1	89.1042	1	female	NaN
859	3	7.2292	0	male	NaN
863	3	69.5500	0	female	NaN
868	3	9.5000	0	male	NaN
878	3	7.8958	0	male	NaN
888	3	23.4500	0	female	NaN

177 rows × 5 columns

ค่าอายุเป็น NaN ไม่สามารถใช้งานได้

ทีนี้เราจะทำยังไงกับค่าว่าง ?

ถ้าวิธีที่ง่ายที่สุดคือการลบทิ้ง แต่เราต้องคำนึงก่อนว่าถ้าลบแล้วข้อมูลมันหายขนาดไหน ? ตอนนี้เรามีข้อมูล 891 อัน ถ้าลบค่าว่างออก จะเหลือแค่ 714 อัน ไม่แย่มาก แต่เรามีวิธีแก้ที่ดีกว่านี้

นั้นคือการนำค่ามาแทนที่นั้นเองโดยการใช้คำสั่ง imputer อ่านเพิ่มเติมได้จากบทความด้านล่างเลย

ขั้นแรกให้แปลงข้อมูลอายุของเราให้เป็น array ก่อน

```
import numpy as np
age = f['Age'].values
age = np.reshape(age,(-1,1))
```

ทีนี้ใช้ฟังก์ชั่น SimpleImputer เพื่อนำค่าอื่นมาแทนที่ "NaN" โดยค่าที่เราจะให้มาแทนคือค่าที่ปรากฏบ่อยที่สุด

```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values = np.nan ,
strategy='most_frequent')
imp.fit(age)
```

ทีนี้เราก็แปลงค่าว่างในคอลัมน์อายุของเราเป็นค่าอื่นได้เลย แล้วลองดูว่ายังมีค่าว่างอีกไหม ?

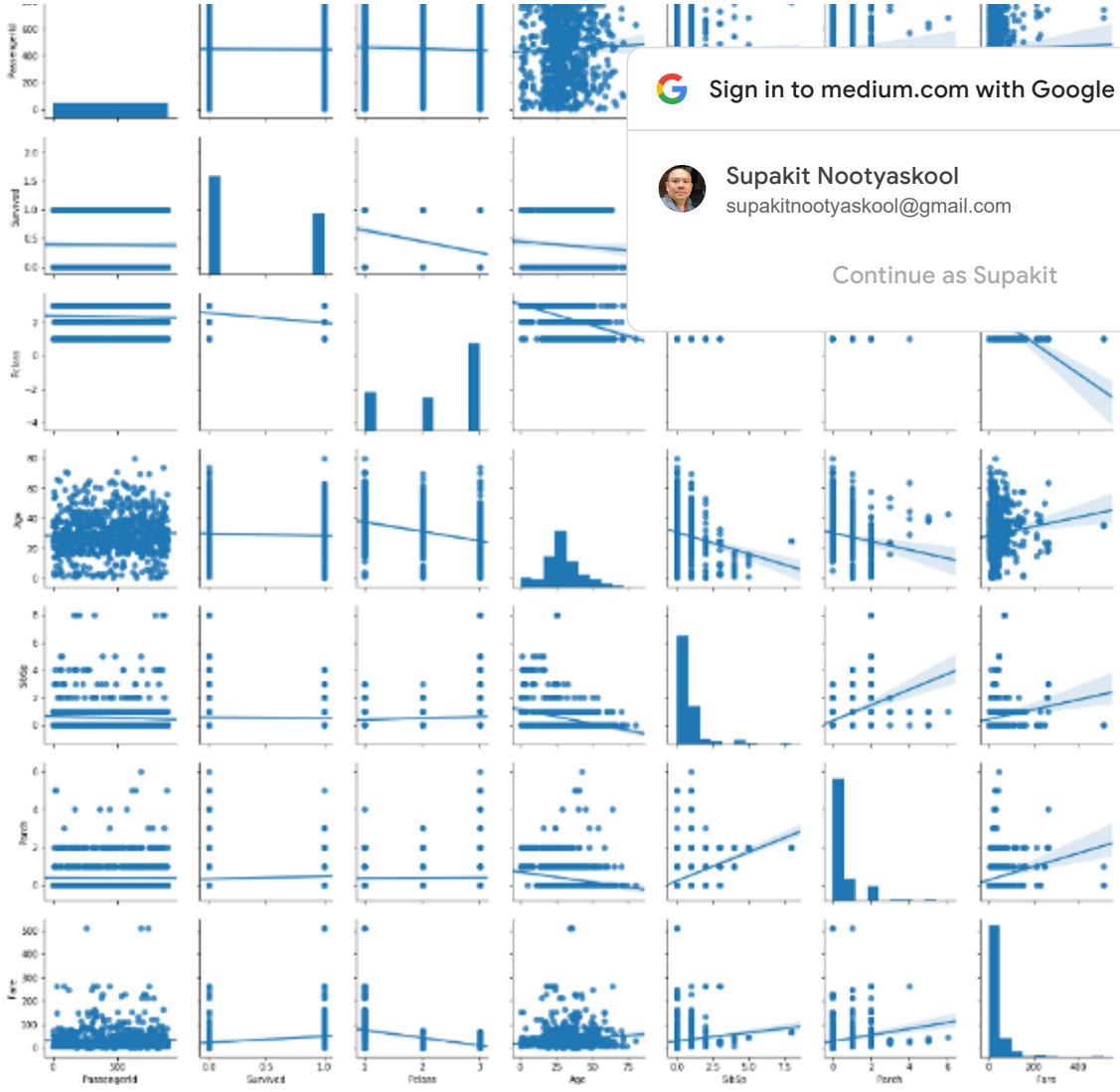
```
f['Age'] = imp.transform(age)
f[f['Age'].isnull()]
```

Pclass	Fare	Survived	Sex	Age

ค่าว่างหายหมดเลย

การเลือก Feature

ทำการเลือกเฉพาะข้อมูลที่น่าจะมีปัจจัย ตามหลักแล้วขั้นตอนนี้ข้อมูลที่เลือกจะต้องสอดคล้องกับเฉลยด้วย โดยผมจะเลือกจากหนัง Titanic ที่ฉายไปนั้นคือ คนที่รวยส่วนมากจะรอด คนที่จนก็รอดได้แต่ยากมาก และอีกวิธีที่ผมแนะนำคือใช้ lib ที่ชื่อว่า seaborn ด้วยคำสั่ง pairplot จะเป็นการบอกความสัมพันธ์ทั้งหมดของข้อมูลของเรานั้นเอง



แต่มันก็มีวิธีเลือกอีกแบบที่ง่ายๆคือการดูค่าความสัมพันธ์กัน Correlated โดยให้ดูแนวเดียวกัน ถ้าค่ายิ่งใกล้ 1 แสดงว่าค่านั้นสอดคล้องกัน

```
f.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

ให้ดูเหมือน confusion matrix

แต่เราจะเลือกวิธีบ้านๆด้วยโค้ดบ้านๆกัน ด้วยการเลือก Class ของคนที่ขึ้นเรือ (คลาส1คือหรูสุด) (Pclass) และราคาตั๋ว(Fare) และรอดชีวิตหรือไม่(Su
เฉลย(Survived)


```
X = f[['Pclass','Fare']]
X.head()
```

	Pclass	Fare
0	3	7.2500
1	1	71.2833
2	3	7.9250
3	1	53.1000
4	3	8.0500

ใช้แค่ระดับของตั๋วกับค่าตั๋วก็พอแล้ว

```
y = f['Survived']
y.head()
```

```
0  0
1  1
2  1
3  1
4  0
Name: Survived, dtype: int64
```

เฉลยของเรานั้นเอง

ทีนี้ให้ทำการแบ่งแยกข้อมูลชุดฝึกเพื่อไม่ให้เกิด over/under fitting

Under Fitting / Over Fitting ปัญหาที่มองไม่เห็นแต่สัมผัสได้ว่ามี.....

ทำโมเดลเก่งแค่ไหน แต่ถ้าเตรียมข้อมูลไม่เป็นก็ไร้ความหมาย

medium.com

โดยแบ่งข้อมูลจาก 100% เป็น ฝึก 70% ทดสอบ 30%

```
from sklearn.model_selection import tra
X_train, X_test, y_train, y_test = tra
test_size=0.3, random_state=42)
```



```
print(X_train.s
print(X_test.sh
```

```
(623, 2)
(268, 2)
```

ข้อมูลสำหรับฝึกมีจำนวน 623 ทดสอบมี 268

จากนั้นสร้างโมเดล Logistic Regression **ด้วย** sklearn **โดยโมเดลนี้เป็นโมเดลแบบเบสิคก่อน เพื่อความง่ายในการทำครั้งแรก**

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

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

ทีนี้ลองทดสอบโมเดลของเราด้วยชุดข้อมูล test **วัดความถูกต้องด้วย** accuracy

```
y_pred = lr.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
y_pred = lr.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.6940298507462687
```

ความแม่นยำยังไม่ถึง 70

ผลออกมา 69% ซึ่งถือว่าน้อยมาก แต่มันมีวิธีแก้ซึ่งถ้
ส่วนในการรอดด้วย แต่เอาไว้ก่อน ลองเล่นแบบนี้ดูก่อ

ลองทดสอบ ถ้าเกิดเราคือแจ็คในเรื่อง Titanic เป็นผู้
รอดของเราคือ .... 0.24 โอเค เหมือนในหนังเป๊ะ พี่แจ็ค

```
ans = lr.predict_proba([[3,0]])
ans[:]
```

```
array([[0.75760875, 0.24239125]])
```

ทีนี้ลองดูค่าเฉลี่ยของ dataset ชุดนี้ก่อน

```
f.describe()
```

	Pclass	Fare	Survived	Age
count	891.000000	891.000000	891.000000	891.000000
mean	2.308642	32.204208	0.383838	28.566970
std	0.836071	49.693429	0.486592	13.199572
min	1.000000	0.000000	0.000000	0.420000
25%	2.000000	7.910400	0.000000	22.000000
50%	3.000000	14.454200	0.000000	24.000000
75%	3.000000	31.000000	1.000000	35.000000
max	3.000000	512.329200	1.000000	80.000000

ค่า mean คือค่าเฉลี่ย

ทีนี้ลองไปทดสอบโมเดลกับค่า mean คือชั้น 2 และค่าตั๋วประมาณ 32 เหรียญนิดๆ ซึ่งถ้าเป็นไปตามหลักสถิติถ้าเราใช้ค่ากลางโอกาศรอดของเราจะต้องใกล้เคียงกับ 0.5 ทีนี้ทดสอบจริงๆค่าที่ได้คือ 0.412 ยังพอรับได้อยู่

```
ans = lr.predict_proba([[2,32.204208]])
ans[:]
```

```
array([[0.58803257, 0.41196743]])
```

โอกาศรอด 0.412

ทีนี้ลองเป็นคนรวยดูสิ อยากรู้ว่าจะรอดจากเรือล่มได้หรือไม่


```
ans = lr.predict_proba
ans[:]
```

```
array([[0.37982978, 0
```

```
ans = lr.predict_proba([[1,500]])
ans[:]
```

```
array([[0.14010399, 0.85989601]])
```

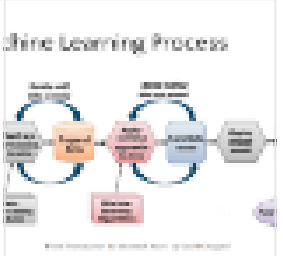
โอกาสรอดของคนรวยคือเยอะมาก

ลองปรับค่าไปที่คนรวยๆคือผู้โดยสารชั้น 1 กับค่าตั๋ว 75 เหรียญ โอกาสรอดคือ 0.62 ถ้าเรายอมจ่ายค่าตั๋วจาก 32 -> 75 เหรียญ โอกาสรอดก็จะเพิ่มขึ้นประมาณ 20%

แต่ลองรวยมหาศาลแบบในตัวร้ายเรื่อง Titanic ดูสิ ถ้าจ่ายค่าตั๋วไปถึง 300 เหรียญโอกาสรอดคือ **0.86** (ในเรื่องตอนเรือจะล่ม ตัวร้ายเอาเงินให้พนักงานเพื่อให้รอด คนรวยจึงมีโอกาสรอดมากกว่า)

แต่ถ้าเราอยากให้แม่นยำขึ้นละ ? ก็เพิ่มปัจจัยเข้าไปอีกแต่ต้องสัมพันธ์กัน ที่เห็นได้ชัดเจนเลยในหนังคือ **“ผู้หญิงขึ้นเรือก่อน”** ทำให้รู้ว่าเพศก็มีผล เลยทำการเพิ่มเพศเข้าไปในชุดฝึกด้วย แต่ sklearn ไม่สามารถนำตัวอักษรเข้าไปฝึกได้ เลยใช้ concat ใน pandas ช่วยแปลงตัวอักษรเป็นตัวเลขนั้นเอง [คล้ายๆ one-hot]

อ่านเพิ่มเติมได้ที่บทความนี้



```
f = pd.concat([f,pd.get_dummies(f['Sex'], prefix='Sex',
                                dummy_na=True)],axis=1).drop(['Sex'],axis=1)
f.head()
```

	Pclass	Fare	Survived	Sex_female	Sex_male	Sex_nan
0	3	7.2500	0	0	1	0
1	1	71.2833	1	1	0	0
2	3	7.9250	1	1	0	0
3	1	53.1000	1	1	0	0

4	3	8.0500	0	0	1	0

เมื่อเราเพิ่มข้อมูลเพศเข้ามาแล้ว ให้ทำการแยกข้อมูล

```
X = f[['Pclass','Fare','Sex_female','S
y = f['Survived']
X_train, X_test, y_train, y_test = tra
test_size=0.3, random_state=42)
```

ทีนี้ลองฝึกฝนโมเดลใหม่ โดยใช้โมเดลเดิมแค่เปลี่ยนข้อมูลโดยการเพิ่มเพศเข้ามา

```
lr.fit(X_train,y_train)

y_pred = lr.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
y_pred = lr.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

0.7873134328358209

ความแม่นยำเพิ่มขึ้นเกือบ 0.1

ความแม่นยำของเราก็เพิ่มขึ้นเป็น 0.787 มากกว่าเดิมเกือบ 0.1

เช็คเมทริกซ์แห่งความสับสนหน่อย ว่าเพราะอะไรทำไมถึงไม่ 100%

```
conf = sklm.confusion_matrix(y_test, y_pred)
print('                 Confusion matrix')
print('                          Survie            Die    ')
print('Actual Survie      %6d' % conf[0,0] + '            %5d' %
conf[0,1] )
print('Actual Die          %6d' % conf[1,0] + '            %5d' %
conf[1,1] )
```

จะได้ว่าจำนวนคนที่ไม่รอดมีน้อยกว่าคนรอด แถมยังทายผิดไปเยอะอีกเลยโดนหักคะแนนไปเยอะพอตัว

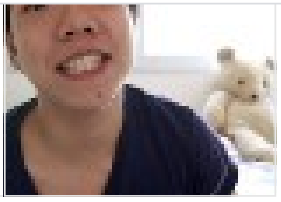
Confusion matrix

	Survie	Die
Actual Survie	101	25
Actual Die	25	64

ทายผิดไปนิดหน่อย ยั

ทีนี้ได้เวลาเพิ่มประสิทธิภาพของโมเดลแล้วด้วย
สำหรับคำอธิบายส่วนนี้ผมเคยเขียนไว้แล้วที่

K-NN กับ Sklearn : Machine Learning 101
มาหาเพื่อนบ้านใกล้เคียงของดอกไม้ชนิดต่างๆกัน
medium.com



โดยรอบนี้เราจะใช้อายุเข้ามาเกี่ยวด้วย

```
X = f[['Pclass','Fare','Sex_female','Sex_male','Age']]
y = f['Survived']

from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
X = scale.fit_transform(X)
```

แบ่งข้อมูลอีกครั้งและทำการทดสอบด้วยโมเดลเดิมแค่เปลี่ยนข้อมูล

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
from sklearn.metrics import accuracy_score
print('ACC ',accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

ACC 0.7985074626865671

	precision	recall	f1-score	support
0	0.81	0.86	0.83	157
1	0.78	0.71	0.75	111
micro avg	0.80	0.80	0.80	268
macro avg	0.80	0.79	0.79	268
weighted avg	0.80	0.80	0.80	268

ความแม่นยำเกือบๆ 80%

จะสังเกตุว่าความแม่นยำนั้นเพิ่มขึ้นแล้วเพราะข้อมูลของเรามันเข้าที่เข้าทางแล้ว ให้ทำการจัดการกับโมเดลของเราต่อเพราะโมเดลที่ใช้เป็นแค่โ
ทำการปรับปรุงโมเดลด้วยการใช้ GridSearhCV ได้เ


```
from sklearn.model_selection import Gr
parameters = {'C': np.arange(1,10,0.5)
lr_best = GridSearchCV(lr, parameters,
lr_best.fit(X_train,y_train)
lr_best.best_estimator_
```

```
LogisticRegression(C=3.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

ค่า C = 3 คือค่าที่ดีที่สุดไม่ใช่ 1

ทีนี้ลองเทรนใหม่ด้วยข้อมูลที่ scale แล้วและใช้ GridSearchCV ปรับปรุงโมเดลแล้ว

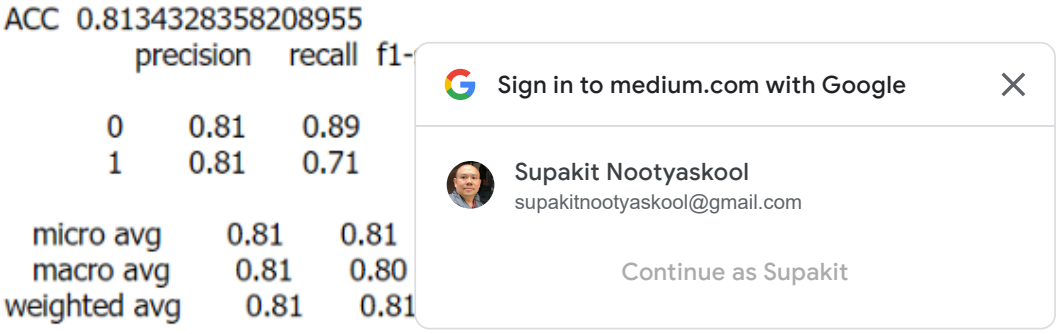
```
y_pred = lr_best.predict(X_test)
from sklearn.metrics import accuracy_score
print('ACC ',accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
ACC  0.7985074626865671
              precision    recall  f1-score   support

           0       0.81      0.86      0.83       157
           1       0.78      0.71      0.75       111

   micro avg       0.80      0.80      0.80       268
   macro avg       0.80      0.79      0.79       268
weighted avg       0.80      0.80      0.80       268
```

ไม่เปลี่ยนแปลงเลย

สาเหตุที่ไม่มีอะไรเปลี่ยนแปลงเลยคือ “โมเดลทำได้ดีที่สุดแล้ว” ต้องเข้าใจก่อนว่าไม่ใช่ว่าทุกโมเดลจะสามารถทำความแม่นยำได้ 100% หรือข้อมูลทุกอันสามารถสร้างโมเดลให้มีความแม่นย่า 100% บางครั้งโมเดลก็ทำได้สุดๆแค่นี้ ต่อให้ปรับปรุงอะไรก็ไม่สามารถเพิ่มได้มากกว่านี้แล้ว โดยสิ่งที่ผมแนะนำได้คือแนะนำให้โมเดลหลายๆอันแล้วมาเปรียบเทียบกันเพื่อหาว่าอันไหนดีที่สุด

แต่ที่ถ้าใครสนใจอยากจะ Challenge ตัวเองก็ลองทำตามบทความนี้แล้วให้ความแม่นมากกว่าที่ผมแอบไปทำก็ลองดูได้นะครับ (คำใบ้คือ เพิ่ม Feature) ใครทำได้แล้วลองคอมเม้นดูนะครับ



ลองเพิ่ม feature อะไรบางอย่างเข้าไปความแม่นเพิ่มขึ้น 2%

สรุปการทดลอง

ในการทดสอบ Logistic Regression กับ Titanic Dataset จะเห็นได้ว่าถ้าเราเพิ่มปัจจัยเข้าในชุดฝึก ความแม่นยำก็จะเพิ่มขึ้นนั้นเอง แต่ปัจจัยนั้นต้องเกี่ยวด้วยจริงๆ ถ้าเราเพิ่ม **“ชื่อผู้โดยสาร”** ความแม่นยำก็ไม่เพิ่มขึ้น เผลอๆจะ error มากด้วยซ้ำ แต่ถ้าเราเพิ่มอายุเข้าไปทำและทำการ Scale ข้อมูลความแม่นยำของเราก็ยังเพิ่มขึ้นได้อีกนิดหน่อยด้วย เพราะฉนั้นทุกๆครั้งที่ทำโมเดลไม่ควรข้ามขั้นตอนการทำ Scale และจูนโมเดลด้วยนั้นเอง !

บทความต่อไป :

SVM อดีตเคยหอมหวานปัจจุบันแอบเซง

Github : https://github.com/peeratpop/Machine_Learning_101

Medium : https://medium.com/@pingloaf

Linkedin : https://www.linkedin.com/in/peerat-limkonchotiwat/

บทความนี้เป็นส่วนหนึ่งของบทความ

เริ่มเรียน Machine/Deep Learning 0–100 (Introduction)

เริ่มเรียน Machine Learning 0–100 zero to Mr.incredible (Introduction)

Machine Learning ไม่ยากเหมือนที่คิด

medium.com

Machine Learning | Logistic Regression | Titanic | Thailand