

SVM

ชุดข้อมูล: Psychology.xls

Source Code: 6610450951_SVM.ipynb

Import library ที่สำคัญและอ่านข้อมูลจากชุดข้อมูล

Import libs

```
[127]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score
```

Read data

```
[105]: df = pd.read_excel("Psychology.xls")
df.head()
```

```
[105]:
```

	เพศ	อายุ	เคยมีแฟนมาแล้ว (คน)	จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์	ระยะเวลาการนอน	นิสัยการกินอาหาร	เคยคิดฆ่าตัวตาย	จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์	ระดับความเครียดด้านการเงิน	เคยกินยานอนหลับ	ภาวะซึมเศร้า
0	Male	28	5	3	5-6 ชั่วโมง	อาหารสุขภาพ	Yes	8	3	Yes	Yes
1	Male	23	5	2	มากกว่า 8 ชั่วโมง	อาหารทั่วไป	No	10	4	No	Yes
2	Female	23	1	3	น้อยกว่า 5 ชั่วโมง	อาหารสุขภาพ	Yes	0	3	No	No
3	Female	20	5	5	มากกว่า 8 ชั่วโมง	Junkfood	Yes	2	5	No	Yes
4	Male	29	4	3	มากกว่า 8 ชั่วโมง	Junkfood	Yes	1	3	No	Yes

ตรวจสอบคอลัมน์ว่ามีคอลัมน์ใดบ้าง และแต่ละคอลัมน์ประกอบไปด้วยค่าอะไรบ้าง

ด้วยคำสั่ง .unique สำหรับทุกคอลัมน์ที่ปรากฏ

```
เพศ ['Male' 'Female']
อายุ [28 23 20 29 31 24 33 25 19 34 21 30 32 26 22 27 18]
เคยมีแฟนมาแล้ว (คน) [5 1 4 2 3 '?']
จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์ [3 2 5 4 '?']
ระยะเวลาการนอน ['5-6 ชั่วโมง' 'มากกว่า 8 ชั่วโมง' 'น้อยกว่า 5 ชั่วโมง' '7-8 ชั่วโมง']
นิสัยการกินอาหาร ['อาหารสุขภาพ' 'อาหารทั่วไป' 'Junkfood']
เคยคิดฆ่าตัวตาย ['Yes' 'No']
จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์ [8 10 0 2 1 3 11 12 9 7 4 6 '?']
ระดับความเครียดด้านการเงิน [3 4 5 '?']
เคยกินยานอนหลับ ['Yes' 'No']
ภาวะซึมเศร้า ['Yes' 'No']
```

พบว่ามีความบางส่วนที่หายไป

ข้อมูลที่ได้จากชุดข้อมูลนี้

- เพศ
 - ชาย (Male)
 - หญิง (Female)
- อายุ
 - ตัวเลขจำนวนเต็มที่อยู่ในช่วง 18 – 34 ปี
- เคยมีแฟนมาแล้ว (คน)
 - ตัวเลขจำนวนเต็มที่อยู่ในช่วง 1 – 5
 - พบค่าที่หายไป “?”
- จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์
 - ตัวเลขจำนวนเต็มที่อยู่ในช่วง 1 – 5
 - พบค่าที่หายไป “?”
- ระยะเวลาการนอน
 - น้อยกว่า 5 ชั่วโมง
 - 5-6 ชั่วโมง
 - 7-8 ชั่วโมง
 - มากกว่า 8 ชั่วโมง
- นิัยการกินอาหาร
 - อาหารสุขภาพ
 - อาหารทั่วไป
 - Junkfood
- เคยคิดฆ่าตัวตาย
 - Yes
 - No
- จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์
 - จำนวนเต็มที่อยู่ในช่วง 0 – 12
 - พบค่าที่หายไป “?”
- ระดับความเครียดด้านการเงิน
 - จำนวนเต็มที่อยู่ในช่วง 1 - 5
 - พบค่าที่หายไป “?”
- เคยกินยานอนหลับ
 - Yes
 - No

และผลลัพธ์ของการเป็นโรคซึมเศร้า

- Yes และ No

ทดลองตัดชุดข้อมูลตัวอย่างทุกแถวที่มีสิ่งที่ไม่ทราบค่าปรากฏ

Ignore all rows that has '?' in it

```

: #Before
print(f"Data size before cut is {df.shape[0]}")

#cut
df = df[~df.isin(['?']).any(axis=1)]

#After
print(f"Data size after cut is {df.shape[0]}")

Data size before cut is 502
Data size after cut is 467

```

พบว่ามิชุดข้อมูลที่ถูกตัดออกไปทั้งหมด 35 ตัวอย่างจาก 502 ตัวอย่าง

Map ชุดข้อมูลทั้งตารางให้อยู่ในรูปตัวเลข เพื่อสะดวกต่อการนำไปคิดต่อ

Data Mapping

```

: df['เพศ'] = df['เพศ'].replace({'Male': 1, 'Female': 0})

df['ระยะเวลาการนอน'] = df['ระยะเวลาการนอน'].replace({
    '5-6 ชั่วโมง': 5.5,
    'มากกว่า 8 ชั่วโมง': 8.5,
    'น้อยกว่า 5 ชั่วโมง': 4.5,
    '7-8 ชั่วโมง': 7.5
})

df['นิสัยการกินอาหาร'] = df['นิสัยการกินอาหาร'].replace({
    'อาหารสุขภาพ': 0,
    'อาหารทั่วไป': 1,
    'Junkfood': 2
})

df['เคยคิดฆ่าตัวตาย'] = df['เคยคิดฆ่าตัวตาย'].replace({'Yes': 1, 'No': 0})

df['เคยกินยานอนหลับ'] = df['เคยกินยานอนหลับ'].replace({'Yes': 1, 'No': 0})
df['ภาวะซึมเศร้า'] = df['ภาวะซึมเศร้า'].replace({'Yes': 1, 'No': 0})

df.head()

```

กำหนดให้การ Map ข้อมูลเป็นดังนี้

- เพศ
 - ชาย -> 1
 - หญิง -> 0
- ระยะเวลาการนอน จะใช้การประมาณค่า
 - น้อยกว่า 5 ชั่วโมง -> 4.5
 - 5-6 ชั่วโมง -> 5.5
 - 7-8 ชั่วโมง -> 7.5
 - มากกว่า 8 ชั่วโมง -> 8.5
- นิสัยการกินอาหาร
 - อาหารสุขภาพ -> 0
 - อาหารทั่วไป -> 1

○ Junkfood -> 2

- เคยคิดฆ่าตัวตาย

○ Yes -> 1

○ No -> 0

- เคยกินยานอนหลับ

○ Yes -> 1

○ No -> 0

และ Map ข้อมูลผลลัพธ์เป็น

ภาวะซึมเศร้า

- Yes -> 1

- No -> 0

เปลี่ยนชื่อคอลัมน์เป็นภาษาอังกฤษ เพื่อไม่ให้เกิดปัญหาด้าน Unicode

Rename Columns

```
c1 = {
    'เพศ': 'Sex',
    'อายุ': 'Age',
    'เคยมีแฟนมาแล้ว (คน)': 'GirlfriendCount',
    'จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์': 'ExercisePerWeek',
    'ระยะเวลาการนอน': 'SleepingTime',
    'นิสัยการกินอาหาร': 'FoodEatingBehavior',
    'เคยคิดฆ่าตัวตาย': 'SuicidalThoughts',
    'จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์': 'LibraryTimePerWeek',
    'ระดับความเครียดด้านการเงิน': 'MoneyStress',
    'เคยกินยานอนหลับ': 'SleepingPills',
    'ภาวะซึมเศร้า': 'Sadness'
}
df = df.rename(columns=c1)
df.head()
```

	Sex	Age	GirlfriendCount	ExercisePerWeek	SleepingTime	FoodEatingBehavior	SuicidalThoughts	LibraryTimePerWeek	MoneyStress	SleepingPills	Sadness
0	1	28	5	3	5.5	0	1	8	3	1	1
1	1	23	5	2	8.5	1	0	10	4	0	1
2	0	23	1	3	4.5	0	1	0	3	0	0
3	0	20	5	5	8.5	2	1	2	5	0	1
4	1	29	4	3	8.5	2	1	1	3	0	1

เตรียม Split ข้อมูล Features และ Target (Training Data)

Prepare Training data

```
[114]: X = df.drop('Sadness', axis=1).values

        scaler = StandardScaler()
        X = scaler.fit_transform(X.astype(float))
        y = df['Sadness'].values

[115]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)
```

โดยเตรียมฟีเจอร์ X สำหรับเรียนรู้ โดยตัดคอลัมน์สุดท้าย (เฉลย) ออก

และแปลงข้อมูล (Preprocess) ด้วย Standard Scaler รวมไปถึงกำหนดให้ y คือเฉลย

จากนั้น Split ข้อมูล Train: Test ด้วยสัดส่วน 80:20 และกำหนดให้ random_state (seed) คือ 12

ทำ SVM

SVM

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 5: Train the SVM model
svm = SVC(kernel='linear', C=1.0)
svm.fit(X_train, y_train)
```

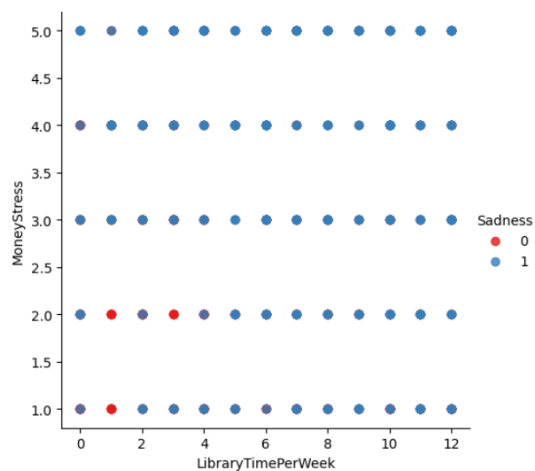
SVC

SVC(kernel='linear')

Plot กราฟตัวอย่างของ 2 features คือ LibraryTimePerWeek และ MoneyStress

```
sns.lmplot(x='LibraryTimePerWeek', y='MoneyStress', data=df, hue='Sadness', palette='Set1', fit_reg=False)
```

<seaborn.axisgrid.FacetGrid at 0x197d187d590>



ทดลอง Predict จากชุดทดสอบ

Predict

```
] y_pred = svm.predict(X_test)

comparison_df = pd.DataFrame({
    'Actual Sadness': y_test,
    'Predicted Sadness': y_pred
})

comparison_df = comparison_df.reset_index(drop=True)
comparison_df
```

	Actual Sadness	Predicted Sadness
0	1	1
1	0	0
2	0	0
3	0	0
4	1	1
...
89	1	1
90	1	1
91	1	1
92	0	0
93	0	0

94 rows × 2 columns

Accuracy

```
: accuracy = accuracy_score(y_test, y_pred)
accuracy
```

```
: 0.9787234042553191
```