การนำเข้าข้อมูลและการเตรียมข้อมูล

- โดยในชุดข้อมูลประกอบไปด้วย feature ดังนี้

features	Description
เพศ	ระบุเพศ
อายุ	ระบุอายุ
เคยมีแฟนมาแล้ว (คน)	จำนวนแฟนที่เคยมีมาแล้ว
จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์	จำนวนครั้งความถี่ในการออกกาลังกายในหนึ่งสัปดาห์
ระยะเวลาการนอน	ช่วงเวลาการนอนหลับ
นิสัยการกินอาหาร	ประเภทอาหารที่ชอบรับประทาน
เคยคิดฆ่าตัวตาย	ระบุว่าเคยมีความคิดฆ่าตัวตายหรือไม่
จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์	ระบุเวลาที่ใช้ในห้องสมุดต่อสัปดาห์
ระดับความเครียดด้านการเงิน	ระดับความเครียดเกี่ยวกับการเงิน
เคยกินยานอนหลับ	ระบุว่าเคยใช้ยานอนหลับหรือไม่
ภาวะซึมเศร้า	สถานะภาวะซึมเศร้า

- แสดงชุดข้อมูลที่นำเข้ามา

	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

- การแสดงรายละเอียดของชุดข้อมูล

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 502 entries, 0 to 501
Data columns (total 11 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   เพศ                                       502 non-null    object
 1   อายุ                                      502 non-null    int64
 2   เคยมีแฟนมาแล้ว  (คน)                      502 non-null    object
 3   จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์      502 non-null    object
 4   ระยะเวลาการนอน                            502 non-null    object
 5   นิสัยการกินอาหาร                          502 non-null    object
 6   เคยคิดฆ่าตัวตาย                           502 non-null    object
 7   จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์  502 non-null    object
 8   ระดับความเครียดด้านการเงิน                502 non-null    object
 9   เคยกินยานอนหลับ                           502 non-null    object
 10  ภาวะซึมเศร้า                              502 non-null    object
dtypes: int64(1), object(10)
memory usage: 43.3+ KB
```

- แสดงรายละเอียดของ features target

```
ภาวะซึมเศร้า
Yes    252
No     250
Name: count, dtype: int64
```

- แสดงชุดข้อมูลอีกครั้งหลังจากทำการแปลงข้อมูล

	เพศ	อายุ	เคยมีแฟนมาแล้ว (คน)	จำนวนครั้งที่ไปออกกำลังกายต่อสัปดาห์	ระยะเวลาการนอน	นิสัยการกินอาหาร	เคยคิดฆ่าตัวตาย	จำนวนชั่วโมงที่เข้าห้องสมุดในหนึ่งสัปดาห์	ระดับความเครียดด้านการเงิน	เคยกินยานอนหลับ	ภาวะซึมเศร้า
0	1	28	5.0	3.0	0	2	1	8.0	3.0	1	1
1	1	23	5.0	2.0	3	1	0	10.0	4.0	0	1
2	0	23	1.0	3.0	2	2	1	0.0	3.0	0	0
3	0	20	5.0	5.0	3	0	1	2.0	5.0	0	1
4	1	29	4.0	3.0	3	0	1	1.0	3.0	0	1

การทำนายผล

- การทำนายครั้งแรก

```
#แยก features และ target
col_names = patient.columns.tolist()
feature_cols = col_names[:-1]
X = patient[feature_cols]
y = patient['ภาวะซึมเศร้า']

# แบ่งข้อมูลออกเป็นชุดฝึก (train) และชุดทดสอบ (test)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.3, random_state=1)


# ปรับมาตรฐาน
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# สร้างโมเดล KNN
knn = KNeighborsClassifier(n_neighbors=5)

# ฝึกโมเดล
knn.fit(X_train, y_train)

# ทำนายผล
y_pred = knn.predict(X_test)

# ประเมินผล
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

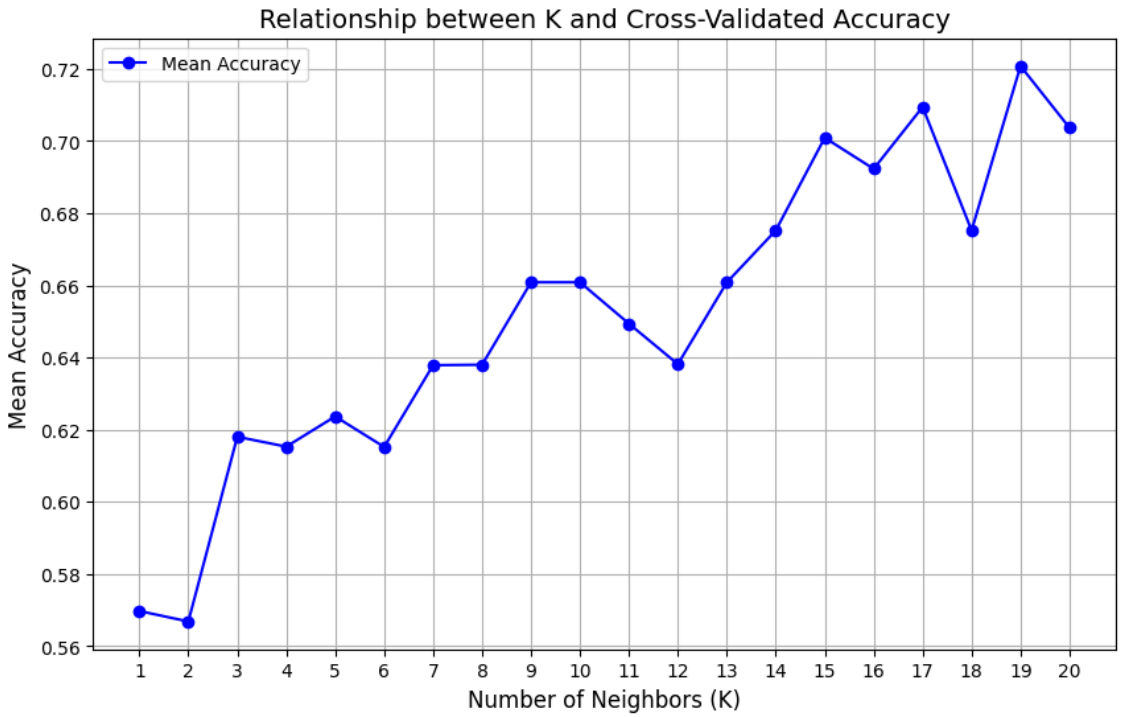
```
Accuracy: 0.6291390728476821
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.60      0.64        84
           1       0.57      0.67      0.62        67

    accuracy                           0.63       151
   macro avg       0.63      0.63      0.63       151
weighted avg       0.64      0.63      0.63       151
```

- ทำการหา GridSearchCV เพื่อหาค่า K ที่เหมาะสมที่สุดในการทำนายผล

```python
#กำหนดพารามิเตอร์ที่ต้องการค้นหา
param_grid = {
    'n_neighbors': list(range(1, 21)),  # จำนวนเพื่อนบ้าน K ที่ต้องการทดสอบ
    'weights': ['uniform', 'distance'],  # น้ำหนักของเพื่อนบ้าน
    'metric': ['euclidean', 'manhattan']  # วิธีวัดระยะทาง
}

#สร้างโมเดล KNN และ GridSearchCV
knn = KNeighborsClassifier()
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy', verbose=1, n_jobs=-1)

#ฝึก GridSearchCV กับชุดข้อมูล
grid_search.fit(X_train, y_train)

#แสดงผลลัพธ์ที่ดีที่สุด
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", grid_search.best_score_)

#ใช้โมเดลที่ดีที่สุดทำนายผลบนชุดทดสอบ
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# ประเมินผล
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
Best Parameters: {'metric': 'manhattan', 'n_neighbors': 18, 'weights': 'uniform'}
Best Cross-Validation Accuracy: 0.7406036217303823
Test Accuracy: 0.847682119205298
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.86        84
           1       0.82      0.84      0.83        67

    accuracy                           0.85       151
   macro avg       0.85      0.85      0.85       151
weighted avg       0.85      0.85      0.85       151
```

- กราฟแสดงความสัมพันธ์ระหว่างค่า K กับค่าเฉลี่ยของ accuracy scores



แกน X แสดงจำนวนเพื่อนบ้าน (Number of Neighbors, K) ที่ใช้ในโมเดล KNN โดยเริ่มจาก 1-20
แกน Y แสดงค่าเฉลี่ยความแม่นยำ (Mean Accuracy) ของโมเดลในแต่ละ K โดยจะมีค่าอยู่ในช่วง 0-1
จุดข้อมูล แต่ละจุดแทนค่าความแม่นยำเฉลี่ยที่ได้จากการทำ Cross-Validation สำหรับ K แต่ละค่า

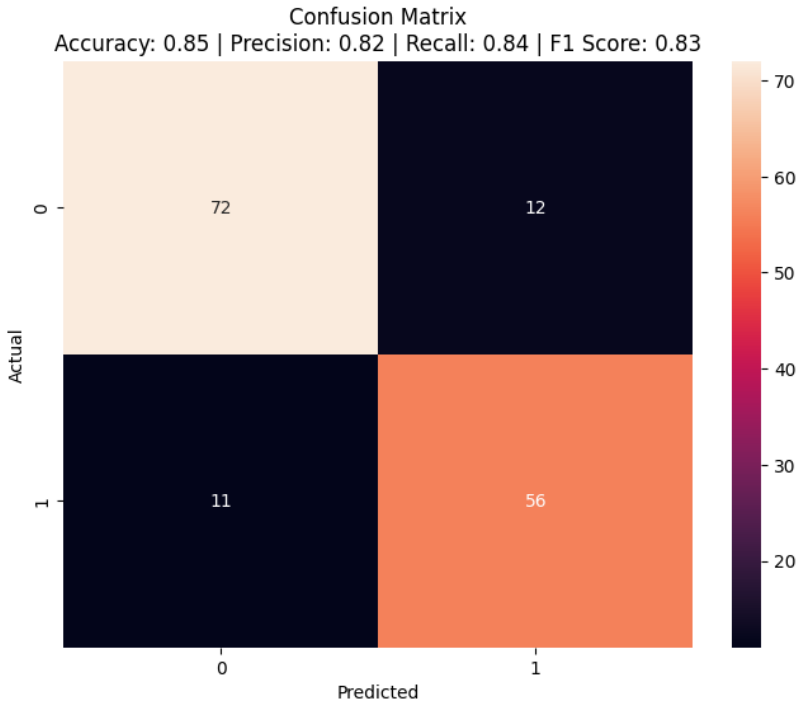
<u>การวิเคราะห์</u>

1. ช่วงเริ่มต้น (K=1K = 1K=1 ถึง K=4K = 4K=4):
- ค่า Mean Accuracy มีค่าต่ำ (ประมาณ 0.56) ที่ K=1K = 1K=1.
- ความแม่นยำเพิ่มขึ้นอย่างรวดเร็วเมื่อเพิ่ม KKK เป็น 3 หรือ 4.

2. ช่วงค่ากลาง (K=5K = 5K=5 ถึง K=14K = 14K=14):
- มีการขึ้นและลงของความแม่นยำเล็กน้อย.
- ความแม่นยำค่อนข้างนิ่งในช่วง K=9K = 9K=9 ถึง K=13K = 13K=13.

3. ช่วงปลาย (K=15K = 15K=15 ถึง K=20K = 20K=20):
- ค่า Mean Accuracy มีแนวโน้มเพิ่มสูงขึ้นอย่างชัดเจน.
- จุดสูงสุดของความแม่นยำอยู่ที่ประมาณ K=19K = 19K=19, มีค่าเฉลี่ยประมาณ 0.72.

- แสดงผลการทำนายในรูปแบบ Confusion Matrix



แกน X (Predicted) แสดงค่าผลลัพธ์ที่โมเดลทำนาย (0 หรือ 1)
แกน Y (Actual) แสดงค่าจริงที่เกิดขึ้นในชุดข้อมูล (0 หรือ 1)
โดยค่าตัวเลขในแต่ละช่องแสดงจำนวนตัวอย่างที่โมเดลจัดให้อยู่ในกลุ่มนั้น
ช่อง (0, 0): 72
ตัวอย่างที่โมเดลทำนายว่าเป็นคลาส 0 และค่าจริงคือคลาส 0 (True Negatives - TN)
ช่อง (0, 1): 12
ตัวอย่างที่โมเดลทำนายว่าเป็นคลาส 1 แต่ค่าจริงคือคลาส 0 (False Positives - FP)
ช่อง (1, 0): 11
ตัวอย่างที่โมเดลทำนายว่าเป็นคลาส 0 แต่ค่าจริงคือคลาส 1 (False Negatives - FN)
ช่อง (1, 1): 56
ตัวอย่างที่โมเดลทำนายว่าเป็นคลาส 1 และค่าจริงคือคลาส 1 (True Positives - TP)
โดยที่มีค่า Accuracy (ความแม่นยำ): 0.85 , **Precision** (ความถูกต้องของการทำนายคลาส 1): 0.82, **Recall (ความครอบคลุมของการทำนายคลาส 1):** 0.84, **F1 Score** (คะแนนเฉลี่ยถ่วงน้ำหนักของ Precision และ Recall): 0.83