การทดลองการจำแนกคลาสบนชุดข้อมูล fertility2.csv

1. ชุดข้อมูลที่ใช้ในการฝึกสอนและเรียนรู้ของเครื่อง

ชุดข้อมูลไฟล์ fertility2.csv เป็นชุดข้อมูลประชากรที่มีบุตรยากของเพศชาย (Diagnosis) ของกระทรวงสาธารณสุข ชุดข้อมูลที่ได้รับมีจำนวนทั้งหมด 139 ระเบียน (record) ซึ่งประกอบด้วยข้อมูลสุขภาพ ได้แก่ ภาวะโรคไต (Kidney disease) ระดับน้ำตาลในเลือด (Fasting blood sugar) การติดเชื้อในทางเดินปัสสาวะ (Uri infection) รวมถึงพฤติกรรมการใช้ชีวิต ได้แก่การออกกำลังกาย (exercise habit) การดื่มสุรา (Frequency of alcohol consumption) การสูบบุหรี่ (Smoking habit) จำนวนชั่วโมงที่นั่งอยู่กับที่ (#hours sit) อายุ และอาชีพ ดังภาพที่ 1.1

	Age	kidney diseases	Fasting Blood sugar	Uri infection	exercise habit	Frequency of alcohol consumption	Smoking habit	profession	#hours spent sitting per day	Diagnosis
33	28	yes	87	yes	more than 3 months ago	once a week	occasional	engineer	8	Normal
67	27	yes	88	no	no	hardly ever or never	never	engineer	8	Normal
21	32	yes	100	no	more than 3 months ago	several times a week	occasional	engineer	5	weak
77	31	yes	90	yes	no	once a week	never	engineer	3	Normal
48	30	yes	94	yes	more than 3 months ago	once a week	never	accountant	6	Normal

ภาพที่ 1.1 ตัวอย่างชุดข้อมูลจากไฟล์ fertility.csv

2. การเตรียมข้อมูล (Data Preprocessing)

2.1 การตรวจสอบค่า NaN Value

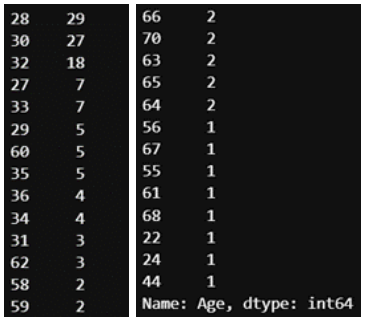
ในชุดข้อมูลนี้ไม่มีค่า NaN ปรากฏ ดังภาพที่ 2.1

```
Age                               0
kidney diseases                   0
Fasting Blood sugar               0
Uri infection                     0
exercise habit                    0
Frequency of alcohol consumption  0
Smoking habit                     0
profession                        0
#hours spent sitting per day      0
Diagnosis                         0
dtype: int64
```

ภาพที่ 2.1 นับค่า NaN ภายในชุดข้อมูล

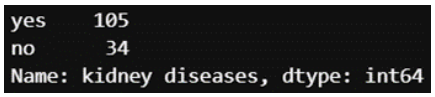
2.2 การทำข้อมูลให้เป็นมาตรฐาน (Normalization)

จากภาพที่ 2.6 สังเกตได้ว่า exercise habit มีค่าที่ซ้ำซ้อนกันของค่า no และ no exercise



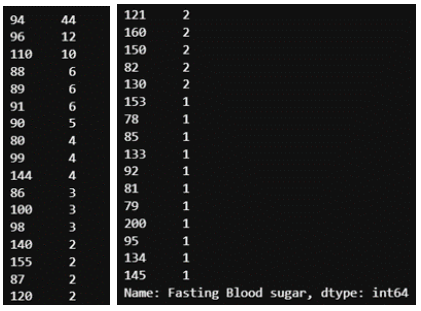
```
28    29
30    27
32    18
27     7
33     7
29     5
60     5
35     5
36     4
34     4
31     3
62     3
58     2
59     2

66    2
70    2
63    2
65    2
64    2
56    1
67    1
55    1
61    1
68    1
22    1
24    1
44    1
Name: Age, dtype: int64
```

ภาพที่ 2.2 ค่าอายุที่ปรากฏทั้งหมดใน Age



```
yes    105
no      34
Name: kidney diseases, dtype: int64
```

ภาพที่ 2.3 ค่าที่ปรากฏทั้งหมดใน kidney disease

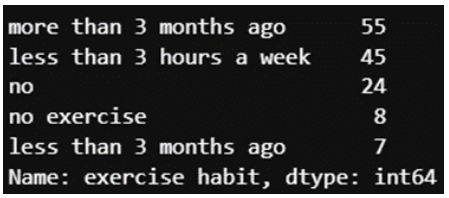


```
94    44
96    12
110   10
88     6
89     6
91     6
90     5
80     4
99     4
144    4
86     3
100    3
98     3
140    2
155    2
87     2
120    2

121    2
160    2
150    2
82     2
130    2
153    1
78     1
85     1
133    1
92     1
81     1
79     1
200    1
95     1
134    1
145    1
Name: Fasting Blood sugar, dtype: int64
```

ภาพที่ 2.4 ค่าที่ปรากฏทั้งหมดใน Fasting Blood sugar

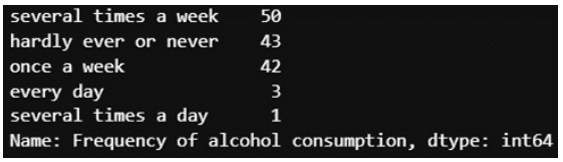
```
no     75
yes    64
Name: Uri infection, dtype: int64
```

ภาพที่ 2.5 ค่าที่ปรากฏทั้งหมดใน Uri infection



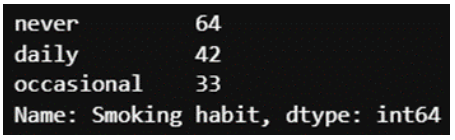
```
more than 3 months ago      55
less than 3 hours a week    45
no                          24
no exercise                  8
less than 3 months ago       7
Name: exercise habit, dtype: int64
```

ภาพที่ 2.6 ค่าที่ปรากฏทั้งหมดใน exercise habit



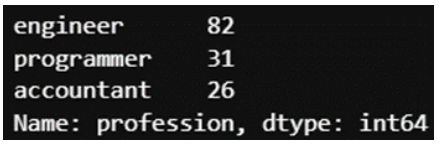
```
several times a week    50
hardly ever or never    43
once a week             42
every day                3
several times a day      1
Name: Frequency of alcohol consumption, dtype: int64
```

ภาพที่ 2.7 ค่าที่ปรากฏทั้งหมดใน Frequency of alcohol consumption



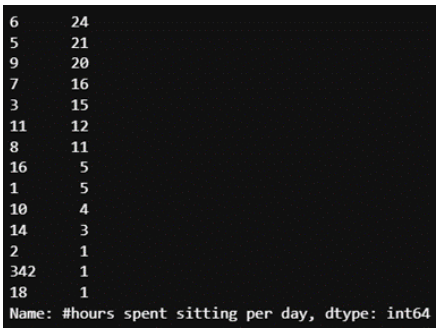
```
never          64
daily          42
occasional     33
Name: Smoking habit, dtype: int64
```

ภาพที่ 2.8 ค่าที่ปรากฏทั้งหมดใน profession



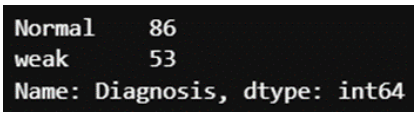
```
engineer      82
programmer    31
accountant    26
Name: profession, dtype: int64
```

ภาพที่ 2.9 ค่าที่ปรากฏทั้งหมดใน profession



```
6      24
5      21
9      20
7      16
3      15
11     12
8      11
16      5
1       5
10      4
14      3
2       1
342     1
18      1
Name: #hours spent sitting per day, dtype: int64
```

ภาพที่ 2.10 ค่าที่ปรากฏทั้งหมดใน #hours spent sitting per day



```
Normal    86
weak      53
Name: Diagnosis, dtype: int64
```

ภาพที่ 2.11 ค่าที่ปรากฏทั้งหมดใน Diagnosis

จึงต้องปรับให้ค่า no มีค่าเดียวกันกับ no exercise ดังภาพที่ 2.12

Covert text into numbers

```
# change exercise habit "no" to "no exercise" (same category)
df["exercise habit"] = df["exercise habit"].apply(lambda x: "no exercise" if x == "no" else x)

df.sample(5)
```

	Age	kidney diseases	Fasting Blood sugar	Uri infection	exercise habit	Frequency of alcohol consumption	Smoking habit	profession	#hours spent sitting per day	Diagnosis
43	28	yes	94	no	more than 3 months ago	hardly ever or never	never	accountant	8	Normal
44	28	yes	87	no	no exercise	hardly ever or never	occasional	accountant	6	Normal
83	33	yes	89	yes	no exercise	hardly ever or never	never	accountant	5	Normal
102	60	no	144	no	no exercise	once a week	occasional	engineer	16	weak
2	27	yes	99	no	more than 3 months ago	hardly ever or never	never	engineer	9	Normal

ภาพที่ 2.12 รูปแบบของการเขียน code เพื่อใช้แปลงค่าจาก no เป็น no exercise ใน exercise habit

2.3 การเข้ารหัสข้อมูลภายในชุดข้อมูล (Data Encoding)

กำหนดให้ค่า ‘weak’ ใน Diagnosis มีค่าเป็น 1 เพื่อระบุว่าเป็นโรค และ ‘Normal’ เป็น 0 ดังภาพที่ 2.13

Convert text to number using LabelEncoder

```
# encode weak to 1 and normal to 0
df["Diagnosis"] = df["Diagnosis"].apply(lambda x: 1 if x == "weak" else 0)

df.sample(5)
```

	Age	kidney diseases	Fasting Blood sugar	Uri infection	exercise habit	Frequency of alcohol consumption	Smoking habit	profession	#hours spent sitting per day	Diagnosis
45	28	yes	94	no	no exercise	hardly ever or never	occasional	accountant	8	1
56	28	no	81	no	more than 3 months ago	once a week	occasional	engineer	14	0
81	33	no	94	yes	no exercise	once a week	occasional	accountant	3	1
52	28	yes	94	no	more than 3 months ago	once a week	occasional	engineer	9	0
119	59	yes	82	no	less than 3 hours a week	several times a week	occasional	engineer	9	0

ภาพที่ 2.13 รูปแบบของการเขียน code ให้ค่า ‘weak’ ใน Diagnosis มีค่าเป็น 1 เพื่อระบุว่าเป็นโรค และ ‘Normal’ เป็น 0

ใช้ LabelEncoder Class จาก sklearn.preprocessing เพื่อช่วยแปลงค่าจากข้อความเป็นตัวเลข ดังภาพที่ 2.14

```
# select only string value columns
text_columns = [col for col in df.columns if df[col].dtype == 'O']
for col in text_columns:
    df[col] = LabelEncoder().fit_transform(df[col])

df.sample(5)
```

	Age	kidney diseases	Fasting Blood sugar	Uri infection	exercise habit	Frequency of alcohol consumption	Smoking habit	profession	#hours spent sitting per day	Diagnosis
25	30	1	121	0	3	1	1	3	5	0
113	62	0	155	1	3	4	2	0	9	1
9	29	1	98	0	2	1	1	3	5	0
8	30	0	95	1	3	2	1	3	5	0
123	63	0	110	0	3	1	0	2	1	0

ภาพที่ 2.14 รูปแบบของการเขียน code เพื่อใช้แปลงค่าจากข้อความเป็นตัวเลขโดยใช้ LabelEncoder

3. การออกแบบการทดลอง (Experiment Design)

3.1 การคัดเลือก Feature และกำหนด Class ผลเฉลย

- Feature ประกอบไปด้วย ภาวะโรคไต (Kidney disease) ระดับน้ำตาลในเลือด (Fasting blood sugar) การติดเชื้อในทางเดินปัสสาวะ (Uri infection) รวมถึงพฤติกรรมการใช้ชีวิต ได้แก่การออกกำลังกาย (exercise habit) การดื่มสุรา (Frequency of alcohol consumption) การสูบบุหรี่ (Smoking habit) จำนวนชั่วโมงที่นั่งอยู่กับที่ (#hours sit) อายุ และอาชีพ
- Class ผลเฉลยจะเป็นภาวะการมีบุตรยากของเพศชาย (Diagnosis)

3.2 ขั้นตอนการทดลอง

- การทดลองจะใช้อัลกอริทึมในการจำแนกคลาส ได้แก่ ต้นไม้ตัดสินใจ (Decision Tree), ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine), เบย์อย่างง่าย (Naive Bayes), เพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor), และวิธีถดถอยแบบโลจิสติก (Logistic Regression)
- ตัวแปรควบคุมในแต่ละการทดลอง ประกอบไปด้วย
  1) ใช้ Train test split โดยแบ่งชุดข้อมูลที่มีออกเป็นชุดข้อมูลที่ใช้ในการฝึกสอน (Training) ร้อยละ 80 ของชุดข้อมูลทั้งหมด และชุดข้อมูลที่ใช้ในการทดสอบ (Testing) ร้อยละ 20 ของชุดข้อมูลทั้งหมด (test_size = 0.2)
  2) กำหนดรูปแบบการสุ่มค่าโดยใช้ random_state = 13
  3) ในแต่ละการทดลองจะใช้ Cross Validation Grid Search (GridSearchCV) เพื่อคัดเลือกโมเดลที่มีประสิทธิภาพมากที่สุดในแต่ละอัลกอริทึม เพื่อเปรียบเทียบ

ประสิทธิภาพของแต่ละอัลกอริทึมในการจำแนกคลาสในชุดข้อมูลชุดเดียวกันนี้ต่อไป และแบ่งการทดลองเป็นชุดย่อย ๆ 5 ครั้ง (cv = 5)

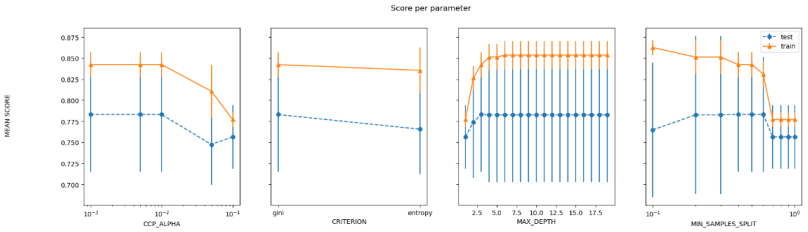
4. ผลการทดลอง (Results)

4.1 ต้นไม้ตัดสินใจ (Decision Tree)

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนดวิธีการคัดเลือก Feature มาสร้างเป็นโนด (Node) ในต้นไม้ (criterion) ได้แก่ gini, และ entropy
2) กำหนดระดับความลึกของต้นไม้ (max_depth) ได้แก่ 1, 2, 3, ... , 19, 20
3) กำหนดจำนวนตัวอย่างขั้นต่ำที่ยอมให้ขั้นตอนวิธีเลือก Feature มาสร้างเป็นโนด (Node) ในต้นไม้ (min_samples_split) ได้แก่ 0.1, 0.2, 0.3, ... , 0.9, 1.0
4) กำหนดให้มีการตัดเล็มต้นไม้แบบ Post-pruning ด้วย Cost complexity หรือ ccp_alpha ได้แก่ 0.1, 0.01, 0.05, 0.005, และ 0.001

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.1

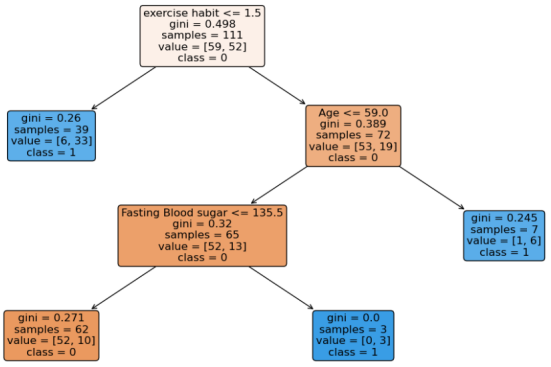


ภาพที่ 4.1 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของต้นไม้ตัดสินใจ (Decision Tree)

ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

1) criterion ได้แก่ gini
2) max_depth ได้แก่ 3
3) min_samples_split ได้แก่ 0.4
4) ccp_alpha ได้แก่ 0.01

สามารถแสดงแผนภาพต้นไม้ตัดสินใจได้ดังภาพที่ 4.2

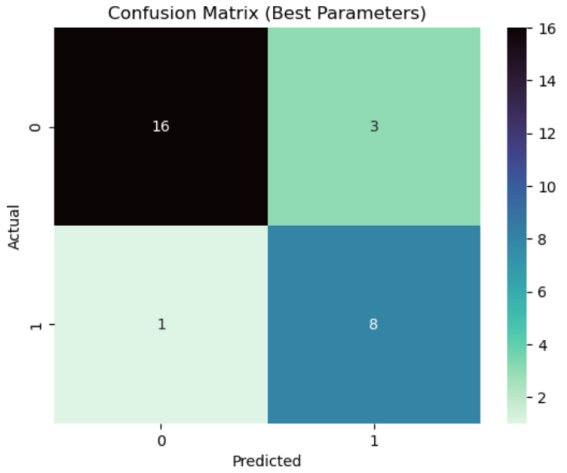


ภาพที่ 4.2 แผนภาพต้นไม้ตัดสินใจ (Decision Tree)
จากไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุด

ประสิทธิภาพของต้นไม้ตัดสินใจ (Decision Tree) ที่ดีที่สุด ดังภาพที่ 4.3 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.4

```
Best Parameters: {'ccp_alpha': 0.01, 'criterion': 'gini', 'max_depth': 3, 'min_samples_split': 0.4}
Accuracy: 0.8571428571428571
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.84      0.89        19
           1       0.73      0.89      0.80         9

    accuracy                           0.86        28
   macro avg       0.83      0.87      0.84        28
weighted avg       0.87      0.86      0.86        28
```

ภาพที่ 4.3 แสดงค่าประสิทธิภาพจาก Classification Report
ของต้นไม้ตัดสินใจ (Decision Tree) ที่ดีที่สุด



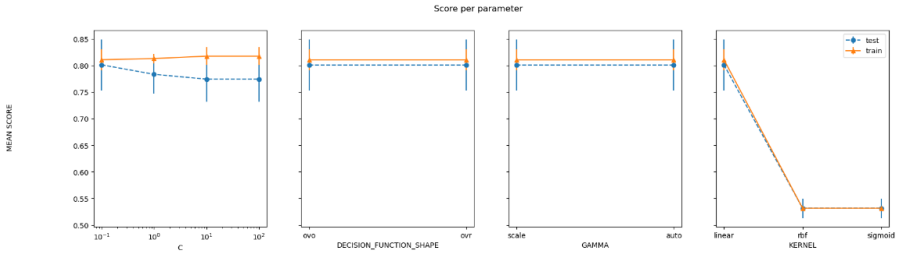
ภาพที่ 4.4 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบ
กับต้นไม้ตัดสินใจ (Decision Tree) ที่ดีที่สุด

4.2 ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine)

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนดฟังก์ชันที่ใช้กำหนดมิติของข้อมูลฝึก (kernel) ได้แก่ linear, rbf, และ sigmoid
2) กำหนดค่า regularization เพื่อเพิ่มระยะมาร์จิน (C) ได้แก่ 0.1, 1, 10, และ 100
3) กำหนดความผันแปรของ kernel (gamma) ได้แก่ scale, และ auto
4) กำหนดวิธีการจัดการกับการจำแนกหมวดหมู่ (decision_function_shape) ได้แก่ ovo, และ ovr

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.5

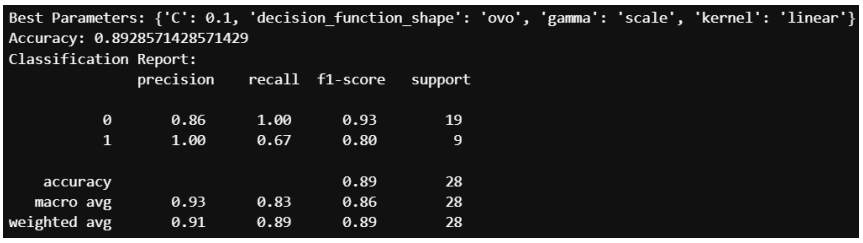


ภาพที่ 4.5 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine)

ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

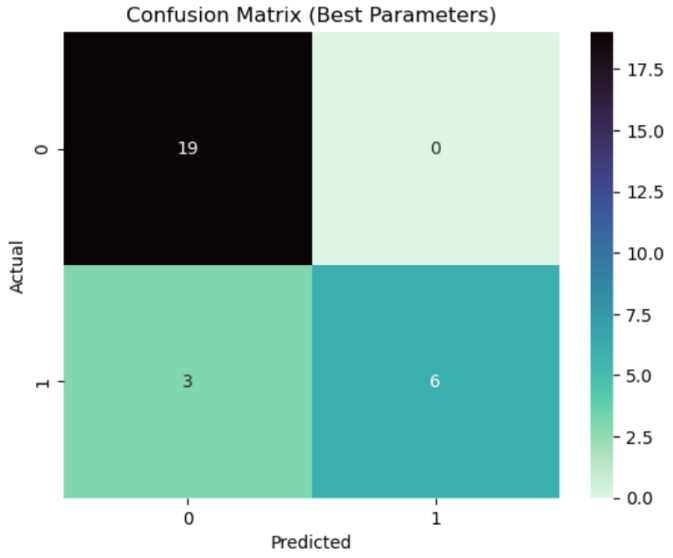
1) kernel ได้แก่ linear
2) C ได้แก่ 0.1
3) gamma ได้แก่ scale
4) decision_function_shape ได้แก่ ovo

ประสิทธิภาพของซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine) ที่ดีที่สุด ดังภาพที่ 4.6 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.7



```
Best Parameters: {'C': 0.1, 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'linear'}
Accuracy: 0.8928571428571429
Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.93        19
           1       1.00      0.67      0.80         9

    accuracy                           0.89        28
   macro avg       0.93      0.83      0.86        28
weighted avg       0.91      0.89      0.89        28
```

ภาพที่ 4.6 แสดงค่าประสิทธิภาพจาก Classification Report ของซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine) ที่ดีที่สุด



ภาพที่ 4.7 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine) ที่ดีที่สุด

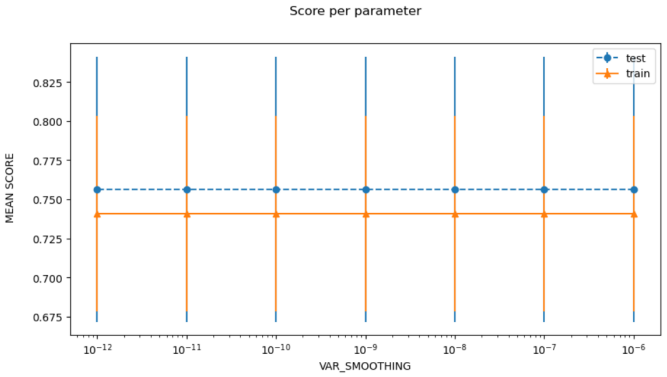
4.3 เบย์อย่างง่าย (Naive Bayes)

4.3.1 Gaussian Naive Bayes

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนดปรับค่าความสำคัญของความแตกต่างในความน่าจะเป็นในขั้นตอนการคำนวณค่าเข้าใกล้ศูนย์ (var_smoothing) ได้แก่ 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11, และ 1e-12

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.8



ภาพที่ 4.8 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของ Gaussian Naive Bayes

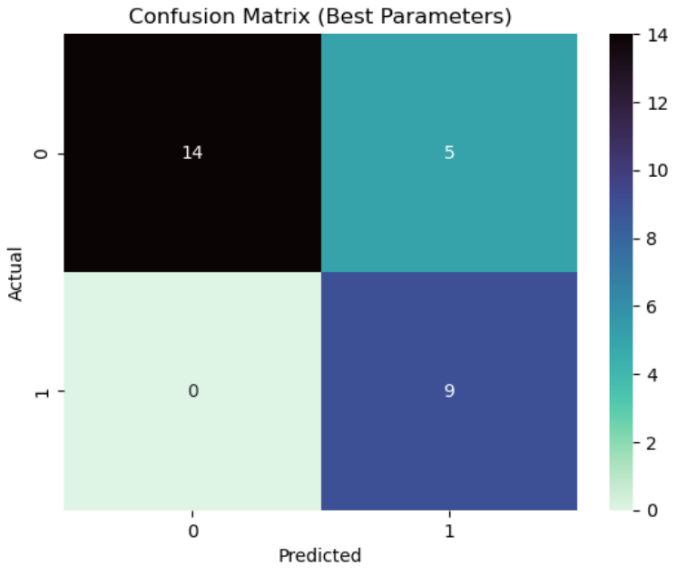
ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

1) var_smoothing ได้แก่ 1e-06

ประสิทธิภาพของ Gaussian Naive Bayes ที่ดีที่สุด ดังภาพที่ 4.9 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.10

```
Best Parameters: {'var_smoothing': 1e-06}
Accuracy: 0.8214285714285714
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.74      0.85        19
           1       0.64      1.00      0.78         9

    accuracy                           0.82        28
   macro avg       0.82      0.87      0.82        28
weighted avg       0.89      0.82      0.83        28
```

ภาพที่ 4.9 แสดงค่าประสิทธิภาพจาก Classification Report ของ Gaussian Naive Bayes ที่ดีที่สุด



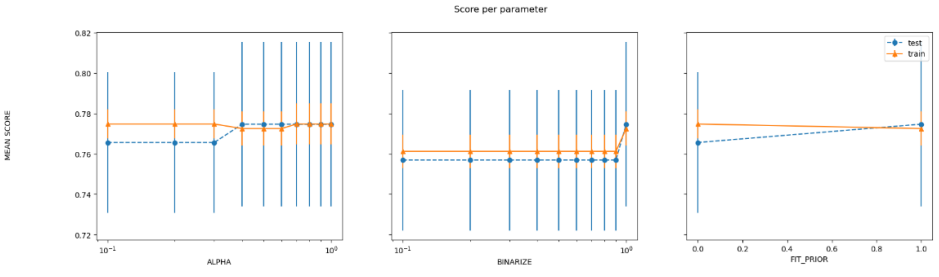
ภาพที่ 4.10 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับ Gaussian Naive Bayes ที่ดีที่สุด

4.3.2 Bernoulli Naive Bayes

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนด regularization (alpha) ได้แก่ 0.1, 0.2, 0.3, ... , 0.9, 1.0
2) กำหนดค่าที่ใช้ในการแบ่งข้อมูลเป็นสองกลุ่ม (binarize) ได้แก่ 0.1, 0.2, 0.3, ... , 0.9, 1.0
3) กำหนดว่าโมเดล Naive Bayes ควรฝึกค่า prior probabilities จากข้อมูลที่ให้มาหรือไม่ (fit_prior) ได้แก่ True, False

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.11



ภาพที่ 4.11 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของ Bernoulli Naive Bayes

ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid

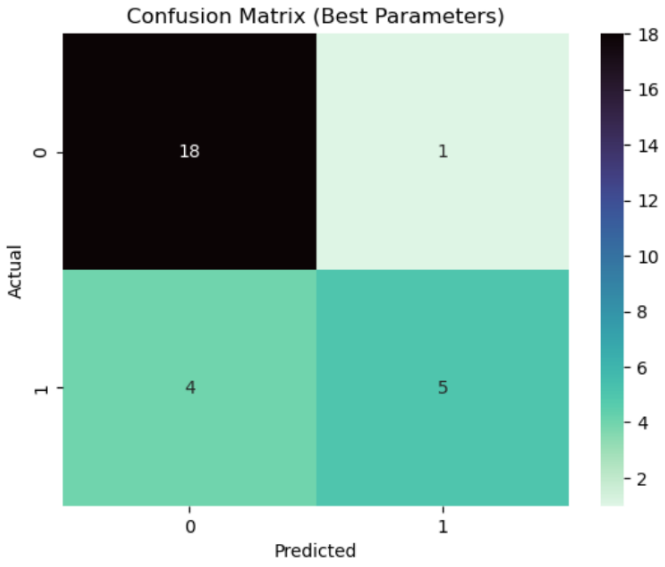
Search (GridSearchCV) ประกอบด้วย

1) alpha ได้แก่ 0.4
2) binarize ได้แก่ 1.0
3) fit_prior ได้แก่ True

ประสิทธิภาพของ Bernoulli Naive Bayes ที่ดีที่สุด ดังภาพที่ 4.12 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.13

```
Best Parameters: {'alpha': 0.4, 'binarize': 1.0, 'fit_prior': True}
Accuracy: 0.8214285714285714
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.95      0.88        19
           1       0.83      0.56      0.67         9

    accuracy                           0.82        28
   macro avg       0.83      0.75      0.77        28
weighted avg       0.82      0.82      0.81        28
```

ภาพที่ 4.12 แสดงค่าประสิทธิภาพจาก Classification Report ของ Bernoulli Naive Bayes ที่ดีที่สุด



ภาพที่ 4.13 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับ Bernoulli Naive Bayes ที่ดีที่สุด

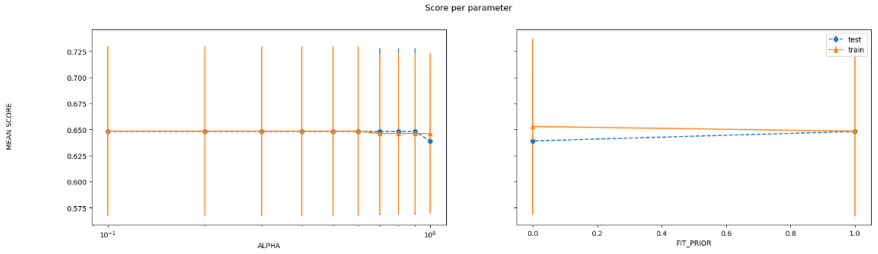
4.3.3 Multinomial Naive Bayes

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนด regularization (alpha) ได้แก่ 0.1, 0.2, 0.3, ... , 0.9, 1.0

2) กำหนดว่าโมเดล Naive Bayes ควรฝึกค่า prior probabilities จากข้อมูลที่ให้มาหรือไม่ (fit_prior) ได้แก่ True, False

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.14



ภาพที่ 4.14 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของ Multinomial Naive Bayes

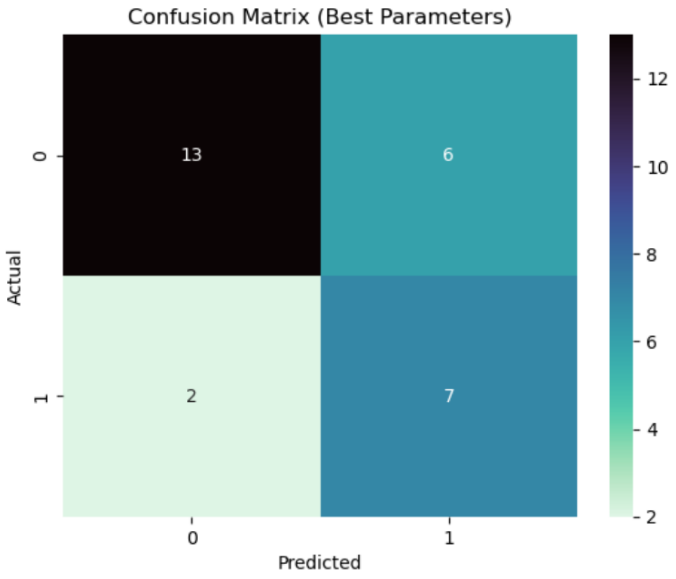
ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

1) alpha ได้แก่ 0.1
2) fit_prior ได้แก่ True

ประสิทธิภาพของ Bernoulli Naive Bayes ที่ดีที่สุด ดังภาพที่ 4.15 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.16

```
Best Parameters: {'alpha': 0.1, 'fit_prior': True}
Accuracy: 0.7142857142857143
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.68      0.76        19
           1       0.54      0.78      0.64         9

    accuracy                           0.71        28
   macro avg       0.70      0.73      0.70        28
weighted avg       0.76      0.71      0.72        28
```

ภาพที่ 4.15 แสดงค่าประสิทธิภาพจาก Classification Report ของ Multinomial Naive Bayes ที่ดีที่สุด



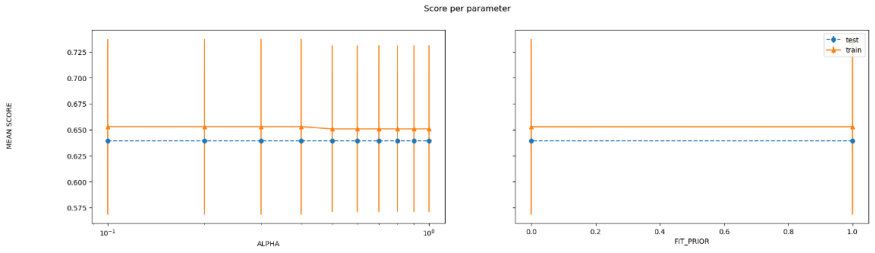
ภาพที่ 4.16 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับ Multinomial Naive Bayes ที่ดีที่สุด

4.3.4 Complement Naive Bayes

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนด regularization (alpha) ได้แก่ 0.1, 0.2, 0.3, ... , 0.9, 1.0
2) กำหนดว่าโมเดล Naive Bayes ควรฝึกค่า prior probabilities จากข้อมูลที่ให้มาหรือไม่ (fit_prior) ได้แก่ True, False

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.17



ภาพที่ 4.17 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของ Complement Naive Bayes

ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

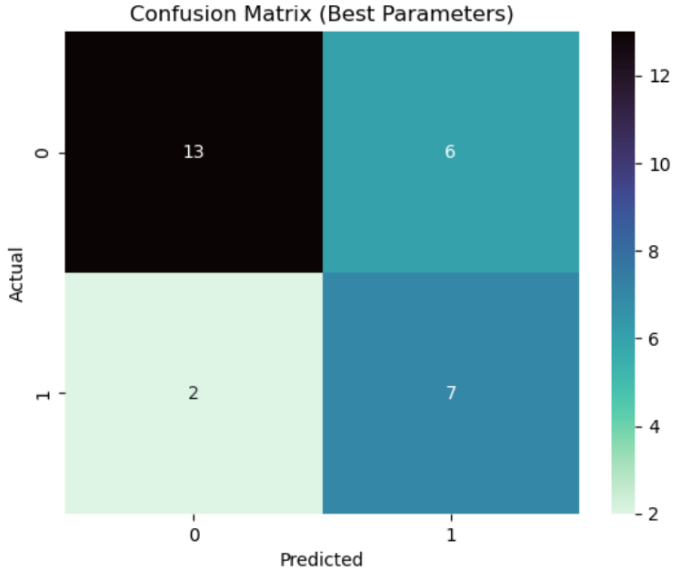
1) alpha ได้แก่ 0.1

2) fit_prior ได้แก่ True

ประสิทธิภาพของ Bernoulli Naive Bayes ที่ดีที่สุด ดังภาพที่ 4.18 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.19

```
Best Parameters: {'alpha': 0.1, 'fit_prior': True}
Accuracy: 0.7142857142857143
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.68      0.76        19
           1       0.54      0.78      0.64         9

    accuracy                           0.71        28
   macro avg       0.70      0.73      0.70        28
weighted avg       0.76      0.71      0.72        28
```

ภาพที่ 4.18 แสดงค่าประสิทธิภาพจาก Classification Report ของ Complement Naive Bayes ที่ดีที่สุด



ภาพที่ 4.19 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับ Complement Naive Bayes ที่ดีที่สุด

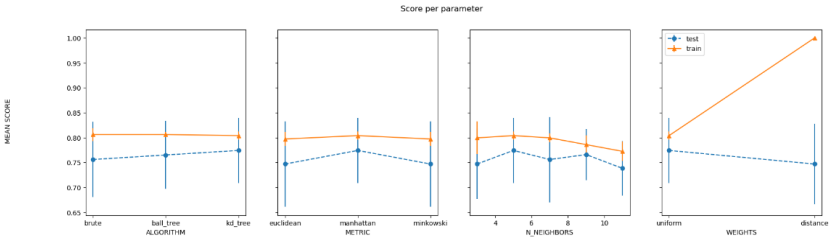
4.4 เพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor)

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) กำหนดจำนวนเพื่อนบ้าน (n_neighbors) ได้แก่ 3, 5, 7, 9, และ 11
2) กำหนดอัลกอริทึมที่ใช้คำนวณหาเพื่อนบ้านที่ใกล้ที่สุด (algorithm) ได้แก่ brute, ball_tree, และ kd_tree

3) กำหนดฟังก์ชันที่ใช้ในการกำหนดน้ำหนักที่ใช้แทนระยะทาง (weights) ได้แก่ uniform, และ distance
4) กำหนดวิธีการคำนวณระยะทาง (metric) ได้แก่ euclidean, manhattan, และ minkowski

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.20



ภาพที่ 4.20 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของเพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor)

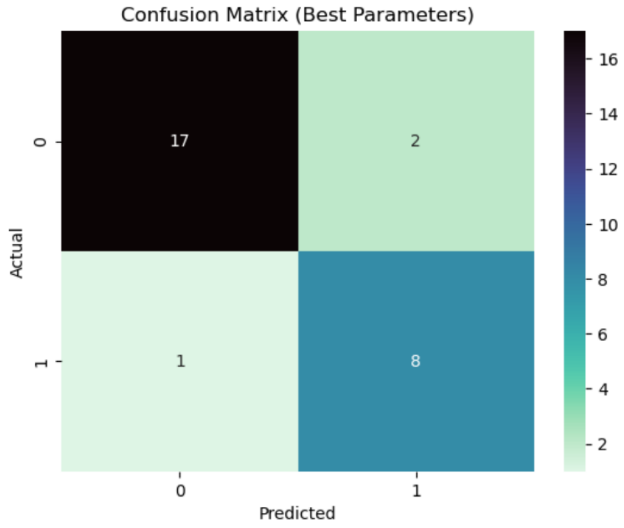
ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

1) n_neighbors ได้แก่
2) algorithm ได้แก่
3) weights ได้แก่
4) metric ได้แก่

ประสิทธิภาพของเพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor) ที่ดีที่สุด ดังภาพที่ 4.21 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.22

```
Best Parameters: {'algorithm': 'kd_tree', 'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'uniform'}
Accuracy: 0.8928571428571429
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.89      0.92        19
           1       0.80      0.89      0.84         9

    accuracy                           0.89        28
   macro avg       0.87      0.89      0.88        28
weighted avg       0.90      0.89      0.89        28
```

ภาพที่ 4.21 แสดงค่าประสิทธิภาพจาก Classification Report ของเพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor) ที่ดีที่สุด



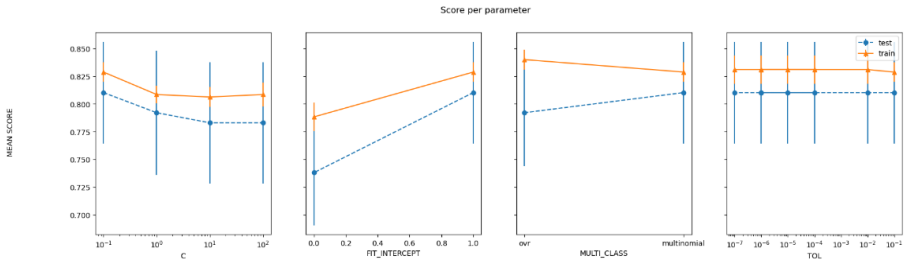
ภาพที่ 4.22 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับเพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor) ที่ดีที่สุด

4.5 วิธีถดถอยแบบโลจิสติก (Logistic Regression)

มีการกำหนดตัวแปรอิสระ หรือไฮเปอร์พารามิเตอร์ (hyperparameter) ดังนี้

1) การกำหนดค่า treshold สำหรับการหยุดการฝึก หากค่าความเปลี่ยนแปลงในค่าความคลาดเคลื่อนระหว่างการฝึกต่ำกว่าค่าที่กำหนด (tol) ได้แก่ 1e-1, 1e-2, 1e-6, 1e-4, 1e-5, 1e-6, และ 1e-7
2) กำหนดค่า regularization เพื่อเพิ่มระยะมาร์จิน (C) ได้แก่ 0.1, 1, 10, และ 100
3) กำหนดว่าโมเดลจะฝึกพารามิเตอร์ intercept หรือไม่ (fit_intercept) ได้แก่ True, และ False
4) กำหนดวิธีการจัดการกับการจำแนกหมวดหมู่ (multi_class) ได้แก่ ovr, และmultinomial

จากการทดลองเพื่อหาไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ได้ผลลัพธ์ดังภาพที่ 4.23



ภาพที่ 4.23 ผลลัพธ์จากการทำ Cross Validation Grid Search (GridSearchCV) โดยใช้ไฮเปอร์พารามิเตอร์ (hyperparameter) ของวิธีถดถอยแบบโลจิสติก (Logistic Regression)

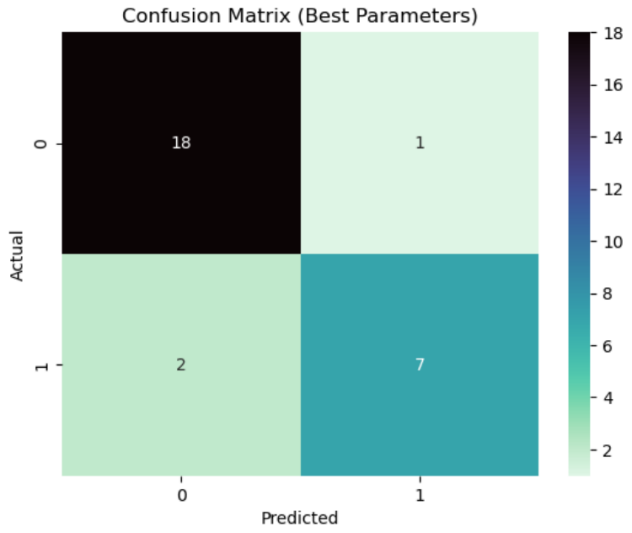
ซึ่งไฮเปอร์พารามิเตอร์ (hyperparameter) ที่ดีที่สุดผ่านการทำ Cross Validation Grid Search (GridSearchCV) ประกอบด้วย

1) tol ได้แก่ 0.1
2) C ได้แก่ 0.1
3) fit_intercept ได้แก่ True
4) multi_class ได้แก่ multinomial

ประสิทธิภาพของวิธีถดถอยแบบโลจิสติก (Logistic Regression) ที่ดีที่สุด ดังภาพที่ 4.24 และรายงานการจำแนกคลาสบนชุดข้อมูลทดลอง (Test Data) ดังภาพที่ 4.25

```
Best Parameters: {'C': 0.1, 'fit_intercept': True, 'multi_class': 'multinomial', 'tol': 0.1}
Accuracy: 0.8928571428571429
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.95      0.92        19
           1       0.88      0.78      0.82         9

    accuracy                           0.89        28
   macro avg       0.89      0.86      0.87        28
weighted avg       0.89      0.89      0.89        28
```

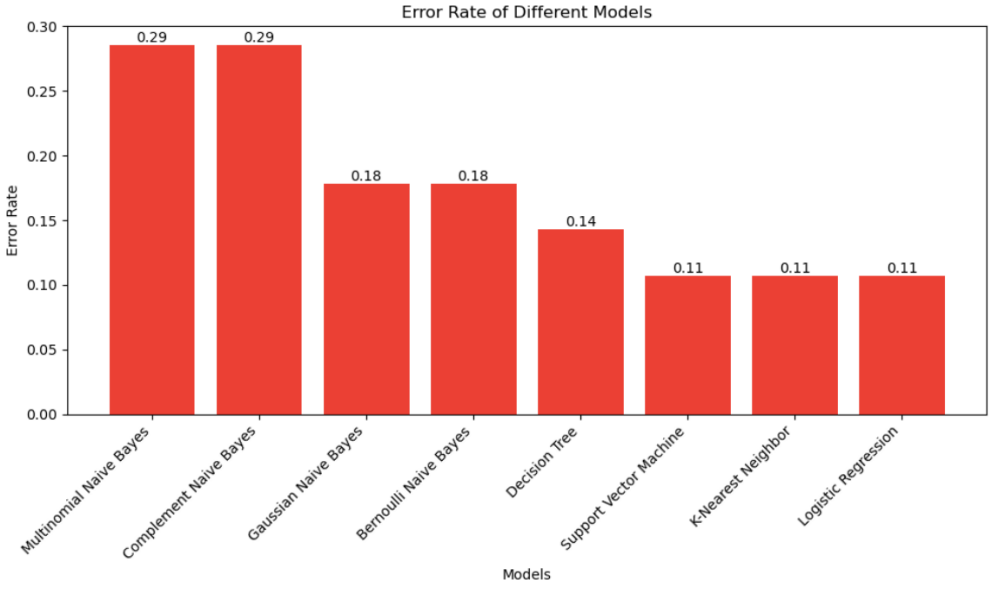
ภาพที่ 4.24 แสดงค่าประสิทธิภาพจาก Classification Report ของวิธีถดถอยแบบโลจิสติก (Logistic Regression) ที่ดีที่สุด



ภาพที่ 4.25 Confusion Matrix ที่เกิดจากการนำข้อมูลทดสอบมาใช้ทดสอบกับวิธีถดถอยแบบโลจิสติก (Logistic Regression) ที่ดีที่สุด

5. สรุปผลการทดลอง (Conclusion)

จากการนำโมเดลของแต่ละอัลกอริทึมมาใช้ในการจำแนกคลาสบนชุดข้อมูล fertility2.csv ซึ่งเป็นชุดข้อมูลเดียวกันนั้น ประสิทธิภาพของแต่ละอัลกอริทึมสามารถแสดงเป็นกราฟแท่งได้ดังภาพที่ 5.1



ภาพที่ 5.1 แสดงค่าความผิดพลาด (Error) ของอัลกอริทึมที่ใช้ในแต่ละโมเดล โดยเรียงจากมากไปน้อย

โมเดลที่ใช้อัลกอริทึมเบย์อย่างง่าย (Naive Bayes) โดยรวมมีค่าความผิดพลาดในการทำนายสูงที่สุดบนข้อมูลชุดนี้และตัวแปรควบคุมเหล่านี้ ส่วนซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine), เพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor), และวิธีถดถอยแบบโลจิสติก (Logistic Regression) ต่างทำงานได้ดี มีค่าความผิดพลาดในการทำนายต่ำที่สุดบนข้อมูลชุดนี้และตัวแปรควบคุมเหล่านี้

แต่เนื่องจากข้อมูลที่ใช้ในการทำการทดลองเป็นข้อมูลทางการแพทย์ การแยกแยะการเป็นโรคการมีบุตรยากของเพศชายจำเป็นที่จะต้องคำนวณค่า Sensitivity และ Specificity ในการคัดเลือกโมเดลที่มีประสิทธิภาพที่สุดจากเหตุการณ์ที่โมเดลมีค่า Accuracy เท่ากัน และค่าประสิทธิภาพอื่น ๆ ใกล้เคียงกัน ซึ่งค่า Sensitivity และ Specificity สามารถหาได้จาก

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

จาก Confusion Matrix ของซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine), เพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor), และวิธีถดถอยแบบโลจิสติก (Logistic Regression) สามารถสร้างตารางการคำนวณค่า Sensitivity และ Specificity บนชุดข้อมูลทดสอบ ได้ดังตารางที่ 5.1

อัลกอริทึมของโมเดล	**Sensitivity (Recall)**	**Specificity**
ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine)	$\frac{6}{6+3} = 0.67$	$\frac{19}{19+0} = 1.00$
เพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor)	$\frac{8}{8+1} = 0.89$	$\frac{17}{17+2} = 0.89$
วิธีถดถอยแบบโลจิสติก (Logistic Regression)	$\frac{7}{7+2} = 0.78$	$\frac{18}{18+1} = 0.95$

ตารางที่ 5.1 แสดงค่า Sensitivity และ Specificity ของซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine), เพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor), และวิธีถดถอยแบบโลจิสติก (Logistic Regression) บนชุดข้อมูลทดสอบ

จากข้อมูลข้างต้นสามารถสรุปได้ว่าโมเดลที่ใช้อัลกอริทึมเพื่อนบ้านใกล้สุด k ตัว (K-Nearest Neighbor) สามารถจำแนกคลาสได้ดีที่สุดบนข้อมูล fertility2.csv และมีตัวแปรควบคุม ดังนี้

- ในการทำ Train test split โดยแบ่งชุดข้อมูลที่มีออกเป็นชุดข้อมูลที่ใช้ในการฝึกสอน (Training) ร้อยละ 80 ของชุดข้อมูลทั้งหมด และชุดข้อมูลที่ใช้ในการทดสอบ (Testing) ร้อยละ 20 ของชุดข้อมูลทั้งหมด (test_size = 0.2)
- กำหนดรูปแบบการสุ่มค่าโดยใช้ random_state = 13
- ในแต่ละการทดลองจะใช้ Cross Validation Grid Search (GridSearchCV) เพื่อคัดเลือกโมเดลที่มีประสิทธิภาพมากที่สุดในแต่ละอัลกอริทึม เพื่อเปรียบเทียบประสิทธิภาพของแต่ละอัลกอริทึมในการจำแนกคลาสในชุดข้อมูลชุดเดียวกันนี้ต่อไป และแบ่งการทดลองเป็นชุดย่อย ๆ 5 ครั้ง (cv = 5)

เนื่องจากมีค่า Sensitivity ที่สูง ซึ่งเหมาะกับชุดข้อมูลที่ใช้ทำนายการเป็นโรค และมีค่า Accuraccy ที่สูงที่สุดจากอัลกอริทึมที่ใช้ทั้งหมด