

MACHINE LEARNING

With Heart Disease Dataset



ผู้จัดทำ

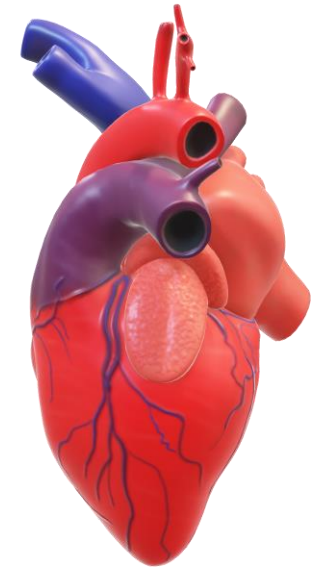


ชื่อ	นาย คีวกร ภาสวาง
เลขประจำตัว	6410451423
หมู่เรียนที่	200



Outline

- Heart Dataset
- Preprocessing
- Support Vector Machine Algorithm
- Performance
- Model Selection

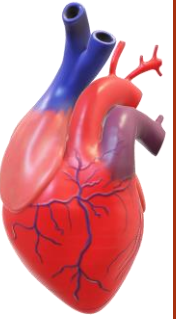




Heart Disease Dataset

- Age
- Sex
- Chest Pain Type
- Resting blood Pressure
- Serum cholesterol in mg/dl
- Fasting blood sugar > 120 mg/dl
- Resting Electrocardiographic results
- Maximum heart rate achieved
- Exercise induced angina
- Old peak = ST depression induced by exercise relative to rest
- The slope of the peak exercise ST segment
- Number of major vessel (0-3) colored by fluoroscopy
- Thalassemia : thal (0 = normal; 1 = fixed defect; 2=reversible defect)
- Target

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1025 non-null   int64
1   sex         1025 non-null   int64
2   cp          1025 non-null   int64
3   trestbps    1025 non-null   int64
4   chol        1025 non-null   int64
5   fbs         1025 non-null   int64
6   restecg     1025 non-null   int64
7   thalach     1025 non-null   int64
8   exang       1025 non-null   int64
9   oldpeak     1025 non-null   float64
10  slope       1025 non-null   int64
11  ca          1025 non-null   int64
12  thal        1025 non-null   int64
13  target      1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

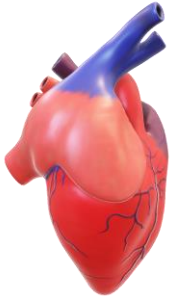


Preprocessing

Change value of target from 0 to -1

```
1 dataframe["target"] = dataframe["target"].replace(0, -1)
2 dataframe.sample(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
469	67	1	0	160	286	0	0	108	1	1.5	1	3	2	-1
37	59	1	0	138	271	0	0	182	0	0.0	2	0	2	1
202	52	1	3	152	298	1	1	178	0	1.2	1	0	3	1
809	54	0	2	110	214	0	1	158	0	1.6	1	0	2	1
656	57	0	1	130	236	0	0	174	0	0.0	1	1	2	-1



Preprocessing

Split to Training data and Test data by
no sklearn library

- Test_size = 0.25

(training = 75 %, test = 25 %)

- Random_state = 40

(default : np.random.randint(100))

```
1 def train_test_split(features, labels, test_size = 0.25, random_state = np.random.randint(100)):  
2     train_size = int(np.round((1 - test_size) * features.shape[0]))  
3     np.random.seed(random_state)  
4     new_features = np.random.permutation(features)  
5     np.random.seed(random_state)  
6     new_labels = np.random.permutation(labels)  
7  
8     features_train = new_features[:train_size]  
9     labels_train = new_labels[:train_size]  
10    features_test = new_features[train_size:]  
11    labels_test = new_labels[train_size:]  
12  
13    return features_train, features_test, labels_train, labels_test
```



Support Vector Machine Algorithm

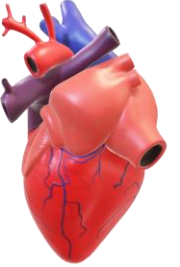
Setup

- **Features :** $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$

- **Weight :** $\vec{w} = \begin{bmatrix} \vec{w} \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

- **Formula of hyperplane :**

$$\vec{w}^T \vec{x} = \begin{bmatrix} \vec{w} \\ b \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix} = \vec{w}^T \vec{x} + b = y$$



Support Vector Machine Algorithm

- With Soft Margin
- Formula of update weight :

$$cost = \gamma w^T w + slack_i ; slack_i \begin{cases} 1 - y_i(w^T x_i) & \text{if } y_i(w^T x_i) < 1 \\ 0 & \text{otherwise} \end{cases}$$

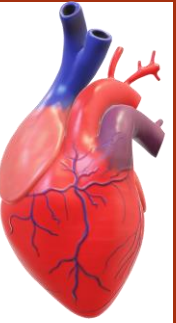
$$\frac{d}{dw} cost = \gamma w + slack_i ; slack_i \begin{cases} -y_i x_i & \text{if } y_i(w^T x_i) < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$step = -learningRate * \frac{d}{dw} cost$$
$$w_{t+1} = w_t + step$$



Source Code of update weight

```
1 def computeWeight(self, weight, feature_train, labels_train, n_record):
2     new_weight = weight
3     for i in range(n_record):
4         check = labels_train[i] * (np.inner(new_weight, feature_train[i]))
5         if check < 1:
6             cost = self.gamma * new_weight - np.inner(labels_train[i], feature_train[i])
7             step = (-1) * self.learning_rate * cost
8             new_weight += step
9         else:
10            cost = self.gamma * new_weight
11            step = (-1) * self.learning_rate * cost
12            new_weight += step
13
14     return new_weight
```



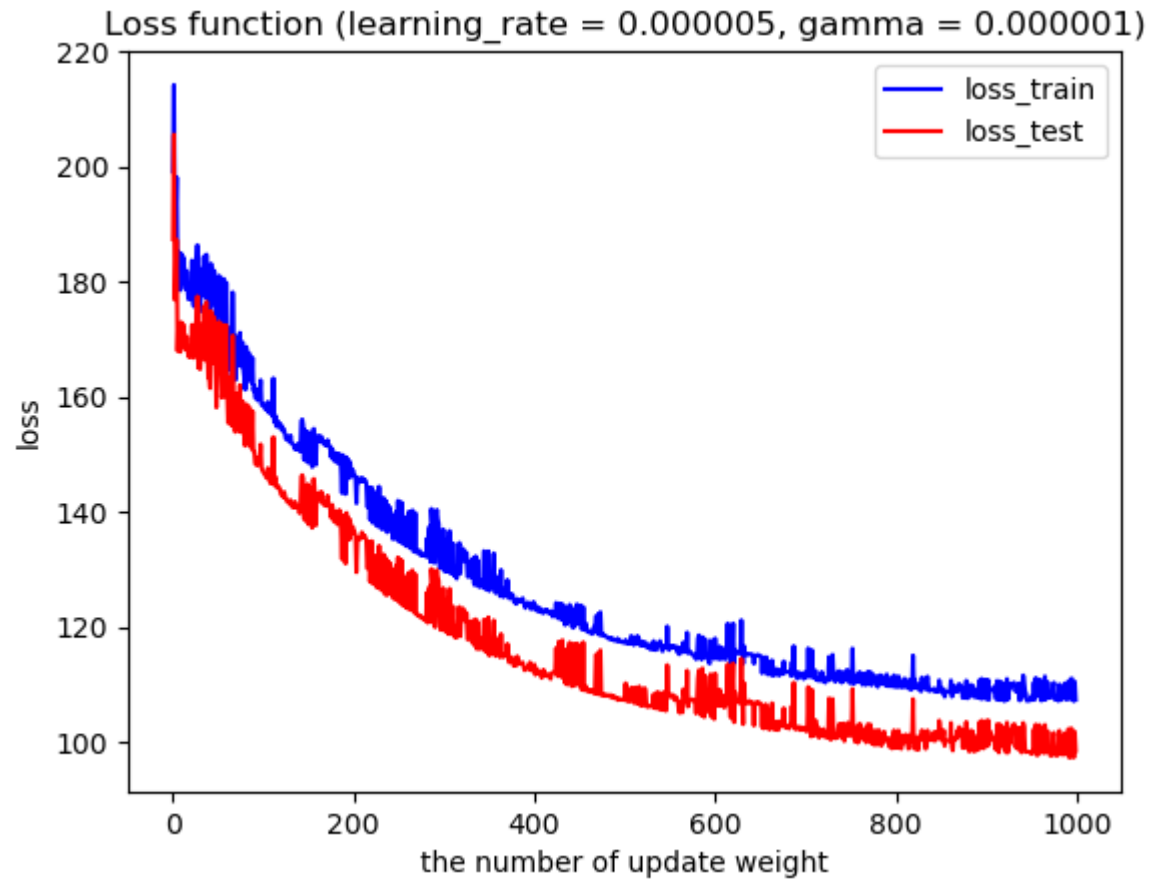
Early Stopping



```
1 for i in range(1000):  
2     self.weight = self.computeWeight(self.weight, self.features, self.labels,  
   self.n_record) # compute to get weight's value  
3  
4     self.array_weight = np.append(self.array_weight, [self.weight], axis=0)
```



Result After Learning



Performance	Training Dataset	Test Dataset
Accuracy	0.8362	0.8516
Sensitivity	0.8022	0.7881
Specificity	1.0	1.0

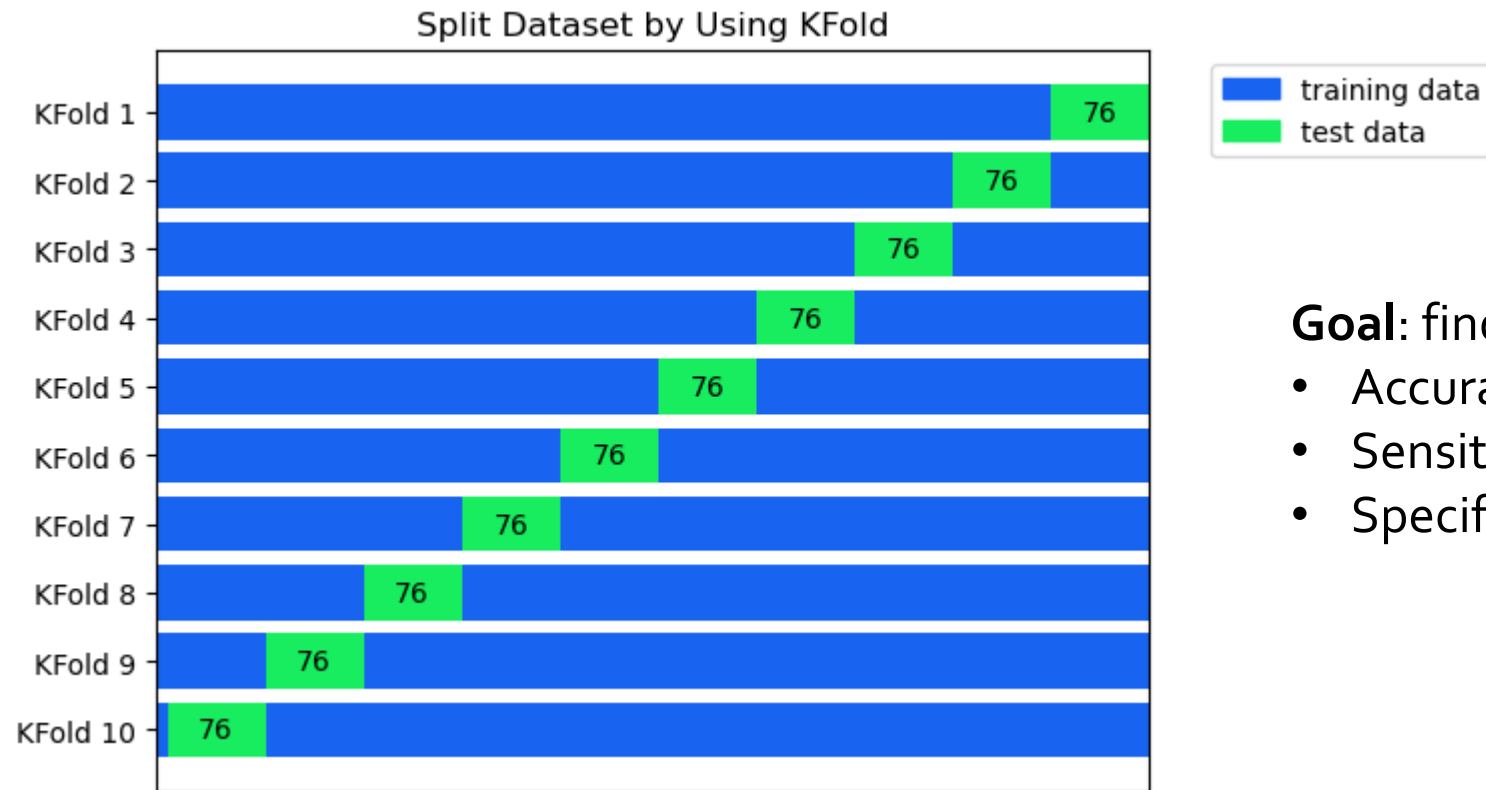
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Sensitivity = \frac{TP}{TP + FP}$$

$$Specificity = \frac{TN}{TN + FP}$$

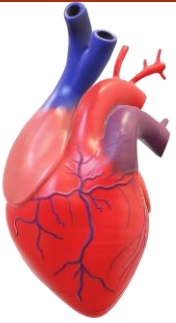


Model Selection by Using K-Fold



Goal: find mean and max value score

- Accuracy
- Sensitivity
- Specificity



Result by Using K-Fold

learning rate=0.000005, gamma=0.000001

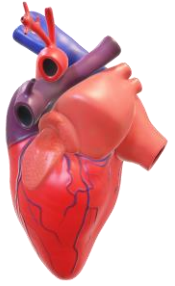
Performance	Training Dataset	Validation (Test) Dataset
Mean of Accuracy	0.8278	0.8342
Mean of Sensitivity	0.8056	0.8106
Mean of Specificity	1.0	1.0
Max of Accuracy	0.8427	0.9211
Max of Sensitivity	0.8490	0.9302
Max of Specificity	1.0	1.0



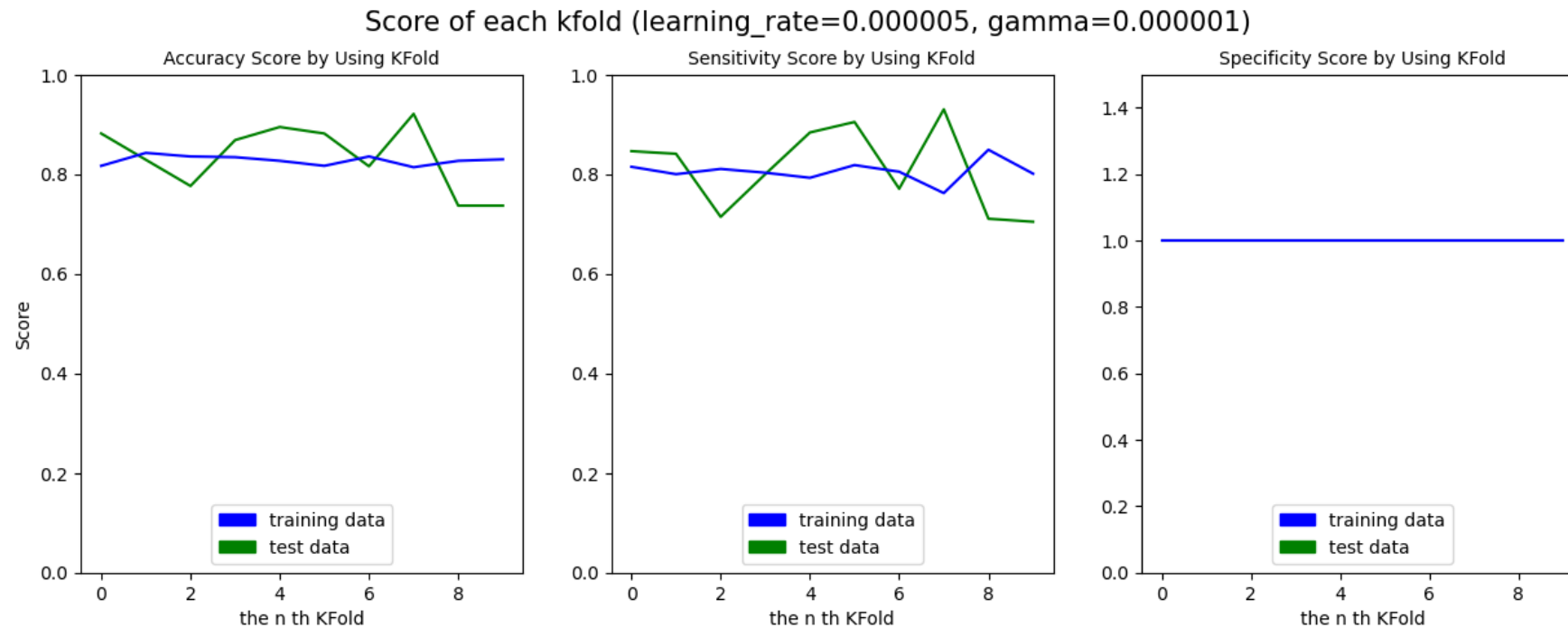
Result by Using K-Fold

learning rate=0.000005, gamma=0.001

Performance	Training Dataset	Validation (Test) Dataset
Mean of Accuracy	0.7441	0.7355
Mean of Sensitivity	0.7738	0.7688
Mean of Specificity	1.0	1.0
Max of Accuracy	0.7561	0.8026
Max of Sensitivity	0.8282	0.9
Max of Specificity	1.0	1.0



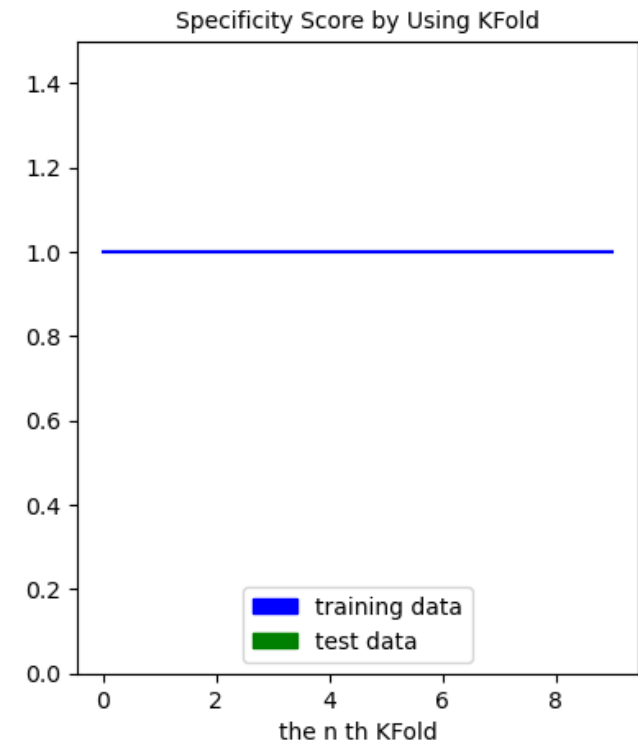
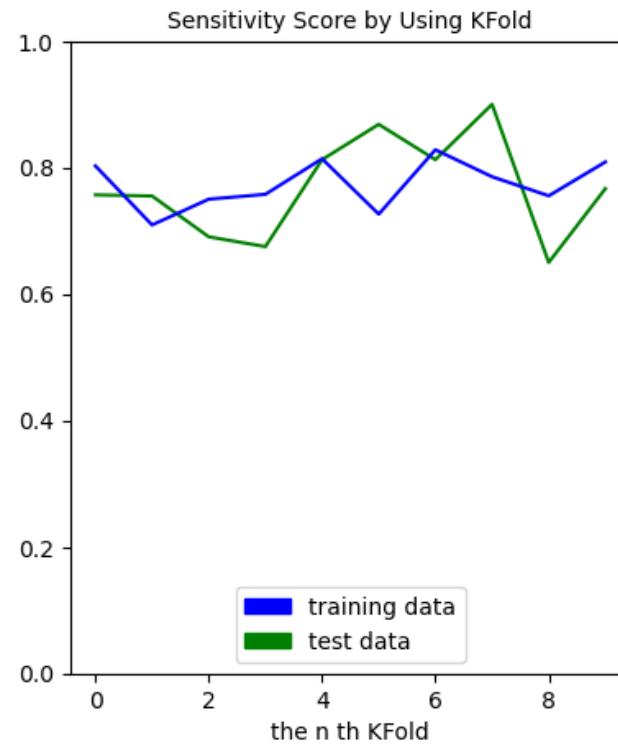
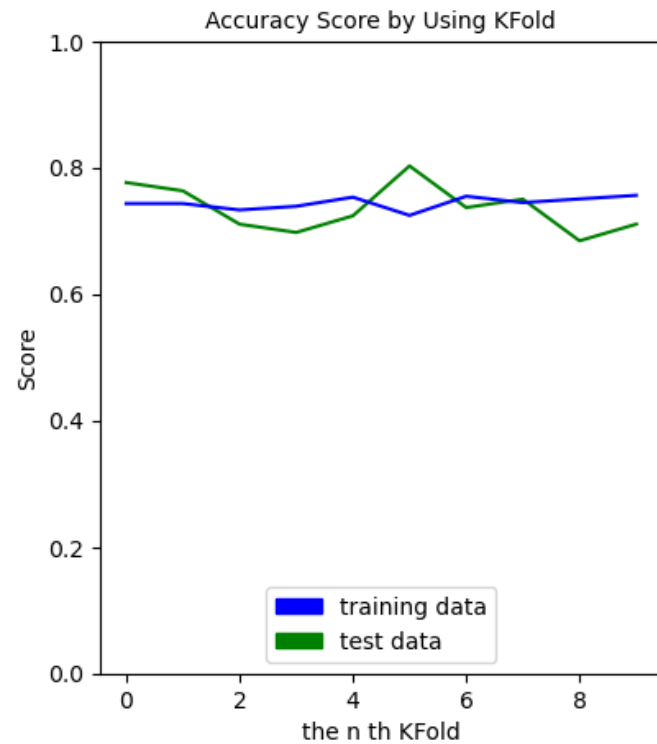
Result by Using K-Fold

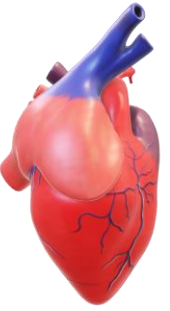




Result by Using K-Fold

Score of each kfold (learning_rate=0.000005, gamma=0.001)





Summary

การใช้ SVM with soft margin ที่มี learning rate = 0.000005 และ gamma = 0.000001 มีประสิทธิภาพที่ดีกว่าการใช้ SVM with soft margin ที่มี learning rate = 0.000005 และ gamma = 0.001 โดยการทดลองนี้ใช้ K-fold ในการเลือกโมเดล และวัดประสิทธิภาพโดยใช้ความถูกต้อง ความไวและความจำเพาะ

Source Code : <https://github.com/OatKID/MLProject>

Thank You

