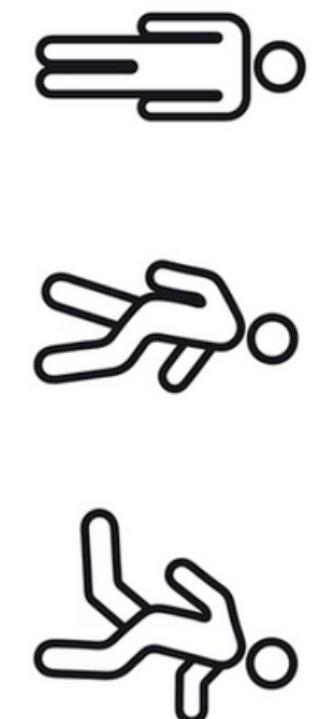
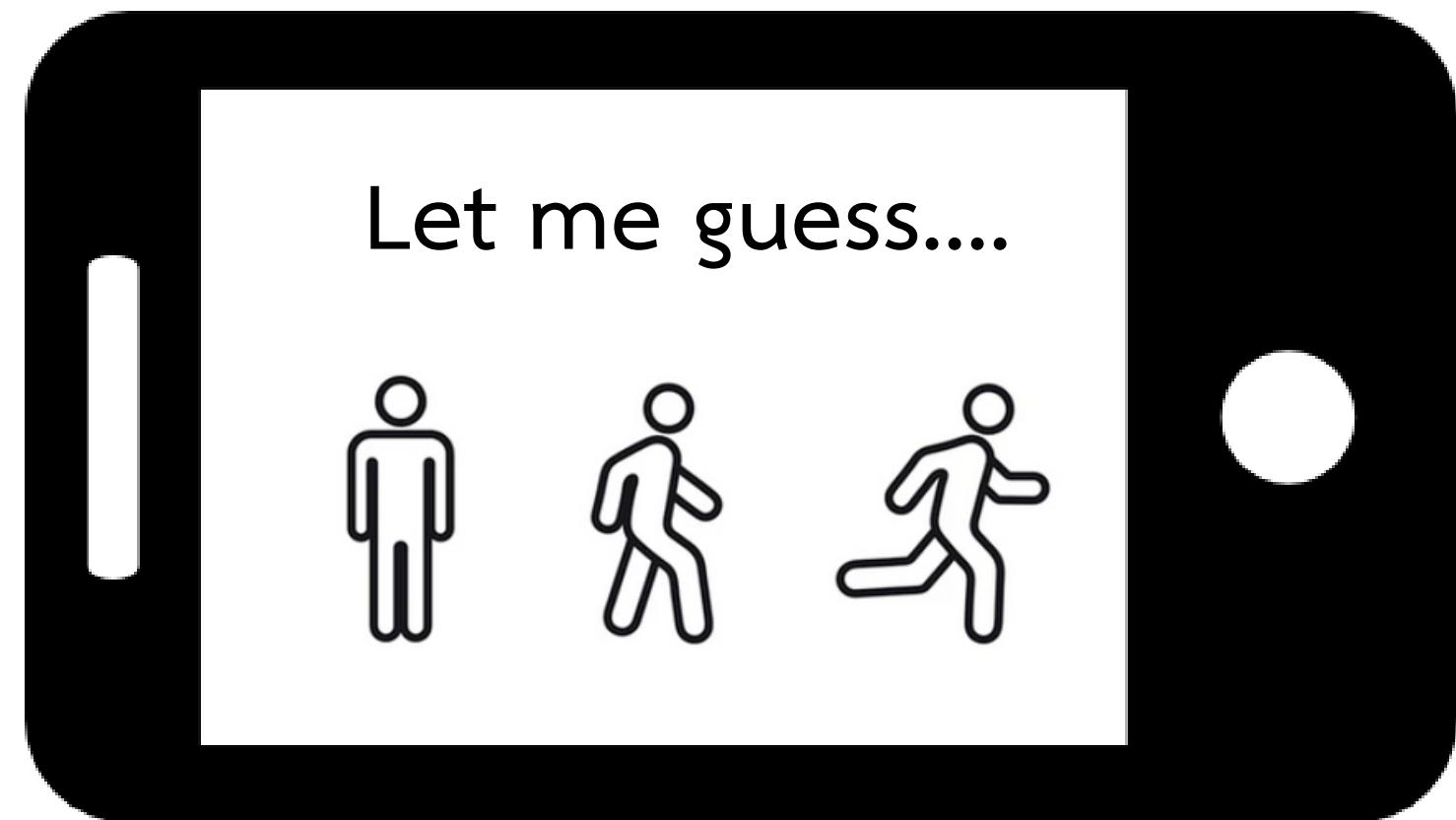


# Human Activity Recognition with Smartphones



การจดจำและวิเคราะห์พฤติกรรมของมนุษย์ด้วยสมาร์ทโฟน

# Human Activity Recognition with Smartphones

คืออะไร?

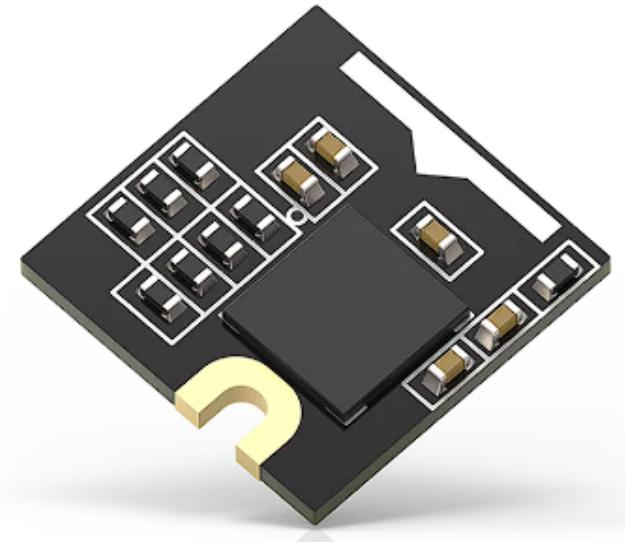
คือการจดจำและวิเคราะห์พฤติกรรมของมนุษย์

โดยการใช้เซนเซอร์ของสมาร์ทโฟน ในการเก็บข้อมูล

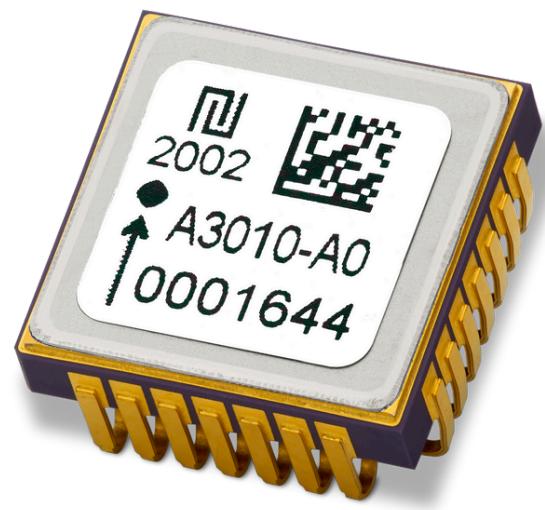
เซนเซอร์ที่ใช้



Accelerometer

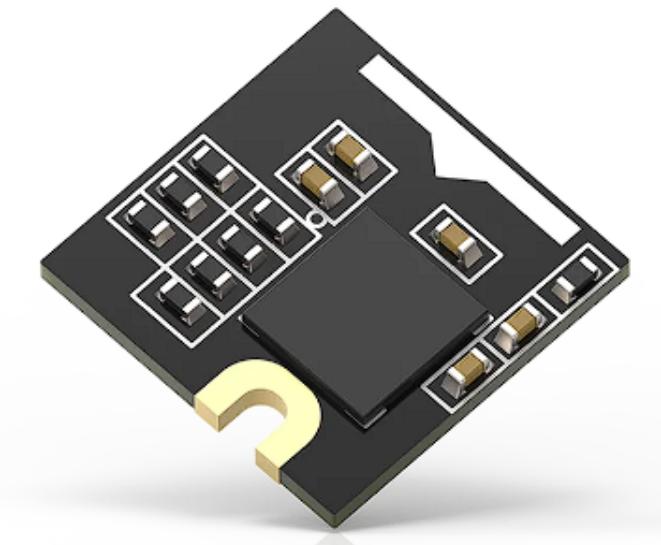
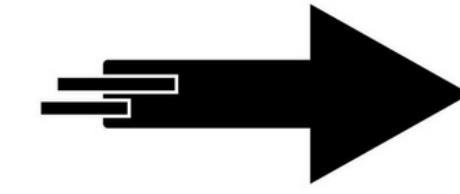


Gyroscope



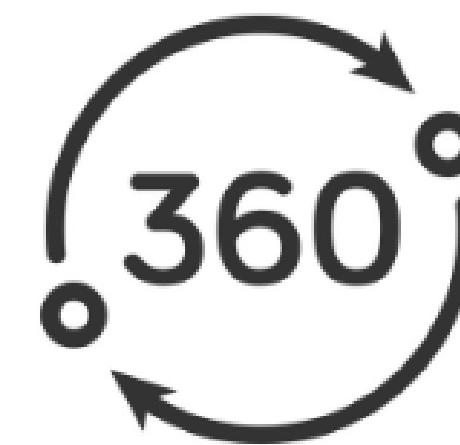
## Accelerometer

Accelerometer ใช้วัดความเร่งเสถียร  
และการเคลื่อนที่ของวัตถุ เช่นในโทรศัพท์  
มือถือหรืออุปกรณ์เล่นเกม เพื่อทำให้เกิด<sup>↑</sup>  
การตอบสนองตามการเคลื่อนที่ของผู้ใช้

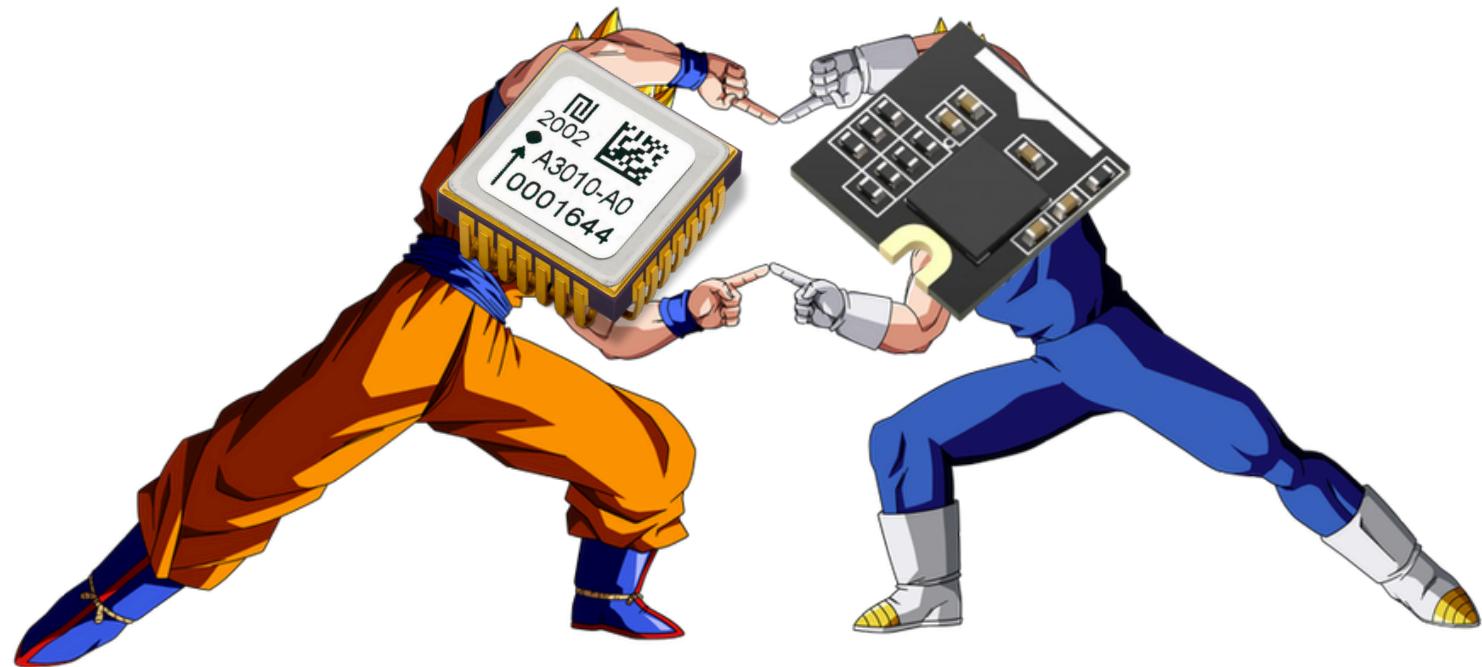


## Gyroscope

Gyroscope ใช้วัดความเร็วหมุนของวัตถุ  
หรือการเปลี่ยนทิศทางของการหมุน ช่วย  
ให้อุปกรณ์สามารถตรวจจับการหมุนหรือ<sup>↑</sup>  
เอียงของตัวเองได้



# เตรียมข้อมูล



Enhanced Dataset

+ ความแม่นยำของข้อมูล

ความเร่ง + ทิศทาง

# สิ่งที่มีในข้อมูล

ข้อมูลประกอบไปด้วย 561 พีเจอร์ ที่ได้มาจากการวัดค่าของเซนเซอร์ accelerometer กับ gyroscope รวมกันที่อ่านค่าได้จากสมาร์ทโฟน ประกอบไปด้วย 6 ท่าทางดังนี้



เดิน

เดินขึ้นบันได

เดินลงบันได

นั่ง

ยืน

เออนตัวลง

ข้อมูลนี้ประกอบไปด้วยข้อมูลสำหรับการเทรนทั้งหมด 7,352 ตัวอย่าง

และข้อมูลสำหรับการทดสอบทั้งหมด 2,947 ตัวอย่าง

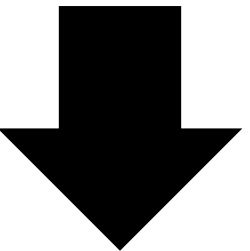
# <Code and Process>

```
if (Ready = true) { sout("Let's Start!");  
} else sout("F");
```

# 1. การนำเข้าข้อมูล

```
import pandas as pd
```

นำเข้าไลบรารีที่ใช้ในการจัดการกับข้อมูล



```
data = pd.read_csv(r"PATH/train.csv")
```

นำเข้าข้อมูลที่จะใช้ในโปรเจค

```
data_test = pd.read_csv(r"PATH/test.csv")
```

# ตัวอย่างข้อมูลที่เก็บมาผ่านเซนเซอร์ทั้งสอง

	<b>tBodyAcc-mean()-X</b>	<b>tBodyAcc-mean()-Y</b>	<b>tBodyAcc-mean()-Z</b>	<b>tBodyAcc-std()-X</b>	<b>Activity</b>
7347	0.299665	-0.057193	-0.181233	-0.195387	WALKING_UPSTAIRS
7348	0.273853	-0.007749	-0.147468	-0.235309	WALKING_UPSTAIRS
7349	0.273387	-0.017011	-0.045022	-0.218218	WALKING_UPSTAIRS
7350	0.289654	-0.018843	-0.158281	-0.219139	WALKING_UPSTAIRS
7351	0.351503	-0.012423	-0.203867	-0.269270	WALKING_UPSTAIRS

## Features

(Independent Variables)

## Target

(dependent Variables)

## 2. เช็คข้อมูล

```
data.head()
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z
0	0.288585	-0.020294	-0.132905
1	0.278419	-0.016411	-0.123520
2	0.279653	-0.019467	-0.113462
3	0.279174	-0.026201	-0.123283
4	0.276629	-0.016570	-0.115362

```
data.tail()
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z
7347	0.299665	-0.057193	-0.181233
7348	0.273853	-0.007749	-0.147468
7349	0.273387	-0.017011	-0.045022
7350	0.289654	-0.018843	-0.158281
7351	0.351503	-0.012423	-0.203867

### 3. เช็คขนาดข้อมูล

```
data.shape
```

```
(7352, 563)
```

```
print("Number of Rows",data.shape[0])  
print("Number of columns",data.shape[1])
```

```
Number of Rows 7352
```

```
Number of columns 563
```

ดูขนาดข้อมูลทั้งหมด

## 4. เช็คไฟเจอร์ชา

```
data.duplicated().any()
```

```
False
```

```
duplicated_columns = data.columns[data.T.duplicated()].tolist()
```

```
len(duplicated_columns)
```

```
21
```

เช็คข้อมูลชา

```
data = data.drop(duplicated_columns, axis=1)
```

```
data.shape
```

```
(7352, 542)
```

ขนาดข้อมูลที่เหลือ

## 5. ตรวจสอบค่าที่หายไป

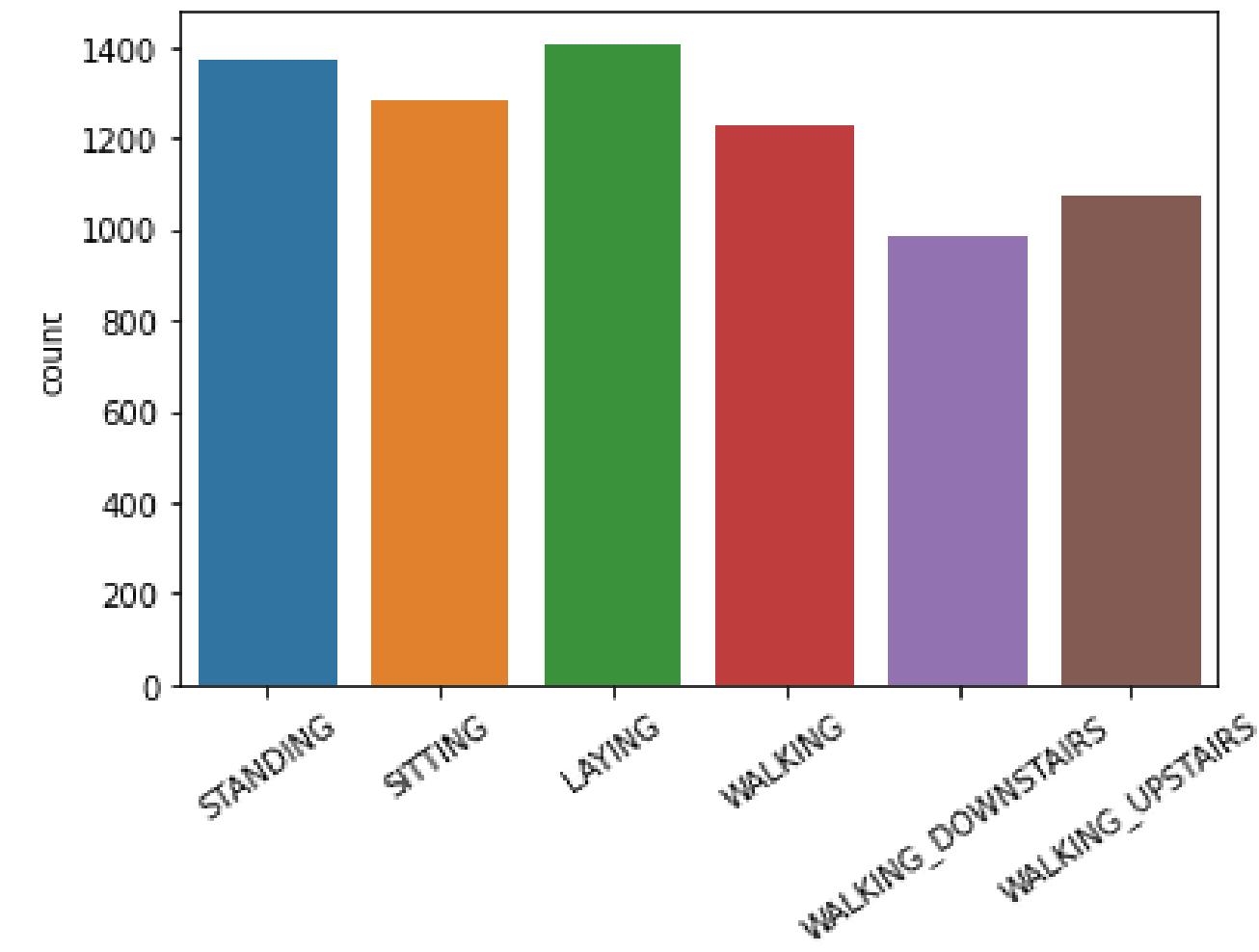
```
data.isnull().sum()

tBodyAcc-mean()-X      0
tBodyAcc-mean()-Y      0
tBodyAcc-mean()-Z      0
tBodyAcc-std()-X       0
tBodyAcc-std()-Y       0
...
angle(X, gravityMean)  0
angle(Y, gravityMean)  0
angle(Z, gravityMean)  0
subject                0
Activity               0
Length: 542, dtype: int64
```

## 6. กราฟของแต่ละการกระทำ

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(data['Activity'])
plt.xticks(rotation=35)
plt.show()
```



## 7. เปลี่ยนค่าในคอลัมน์ Activity เป็น vector

```
x = data.drop('Activity',axis=1)  
y= data['Activity']
```

```
y
```

```
0      STANDING  
1      STANDING  
2      STANDING  
3      STANDING  
4      STANDING  
     ...  
7347  WALKING_UPSTAIRS  
7348  WALKING_UPSTAIRS  
7349  WALKING_UPSTAIRS  
7350  WALKING_UPSTAIRS  
7351  WALKING_UPSTAIRS  
Name: Activity, Length: 7352, dtype: object
```

```
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
y = le.fit_transform(y)
```

```
y
```

```
array([2, 2, 2, ..., 5, 5, 5])
```

## 8. เทคนิคโมเดลด้วยวิธี logistic regression

```
from sklearn.linear_model import LogisticRegression  
from sklearn.metrics import accuracy_score
```

```
log = LogisticRegression()  
log.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
y_pred1 = log.predict(X_test)  
accuracy_score(y_test,y_pred1)
```

```
0.9802855200543847
```

หากคลาสที่ต้องการมีมากกว่า 2

Standing, Walking....Laying

Activation Function ของ Model Logistic Regression

จาก Sigmoid ที่เป็น binary classification เป็น Multiclass classification

ที่เป็น One vs Rest โดยอัตโนมัติ

Sigmoid → One vs Rest

# One vs Rest in Deep

1. สมการ Logistic Regression สำหรับคลาส 1:

$$P(Y = 1) = \frac{1}{1+e^{-(b_0^{(1)} + b_1 x_1 + b_2 x_2 + \dots + b_n x_n)}}$$

2. สมการ Logistic Regression สำหรับคลาส 2:

$$P(Y = 2) = \frac{1}{1+e^{-(b_0^{(2)} + b_1 x_1 + b_2 x_2 + \dots + b_n x_n)}}$$

3. สมการ Logistic Regression สำหรับคลาส 3:

$$P(Y = 3) = \frac{1}{1+e^{-(b_0^{(3)} + b_1 x_1 + b_2 x_2 + \dots + b_n x_n)}}$$

4. สมการ Logistic Regression สำหรับคลาส 4:

$$P(Y = 4) = \frac{1}{1+e^{-(b_0^{(4)} + b_1 x_1 + b_2 x_2 + \dots + b_n x_n)}}$$

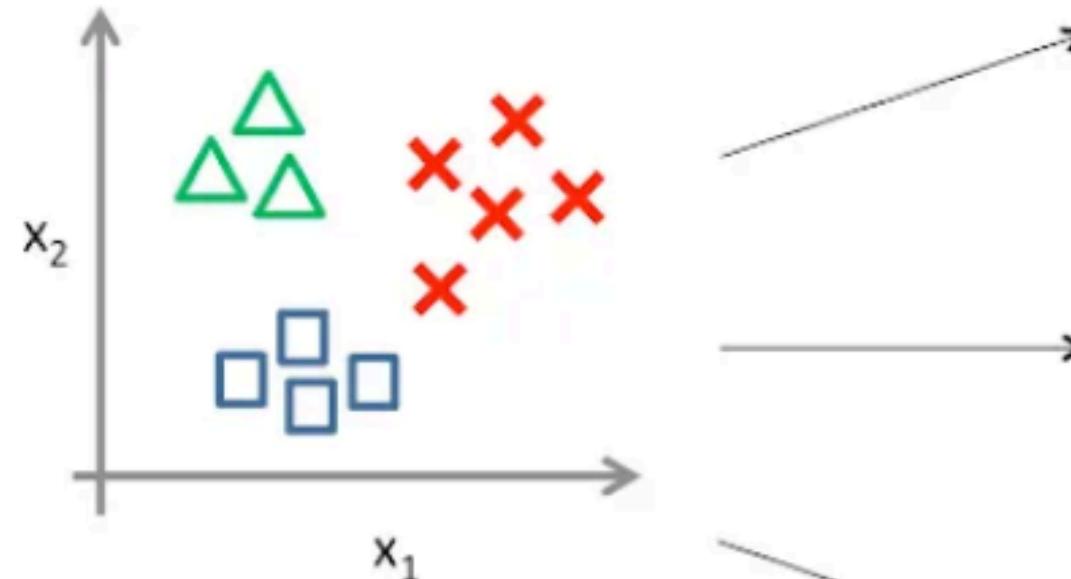
5. สมการ Logistic Regression สำหรับคลาส 5:

$$P(Y = 5) = \frac{1}{1+e^{-(b_0^{(5)} + b_1 x_1 + b_2 x_2 + \dots + b_n x_n)}}$$

6. สมการ Logistic Regression สำหรับคลาส 6:

$$P(Y = 6) = \frac{1}{1+e^{-(b_0^{(6)} + b_1 x_1 + b_2 x_2 + \dots + b_n x_n)}}$$

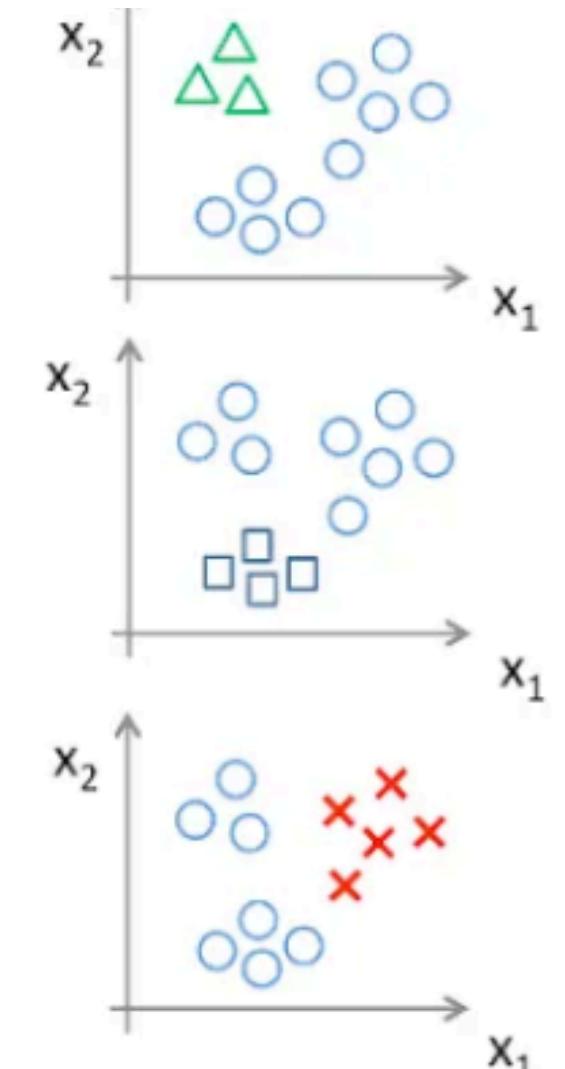
One-vs-all (one-vs-rest):



Class 1: Green

Class 2: Blue

Class 3: Red



- $P(Y = c)$  คือความน่าจะเป็นที่ตัวอย่างจะเป็นคลาส  $c$
- $b_0^{(c)}$  คือ intercept สำหรับคลาส  $c$
- $b_1, b_2, \dots, b_n$  คือ coefficients สำหรับตัวแปร  $x_1, x_2, \dots, x_n$  ตามลำดับ

## 9. Feature Selection

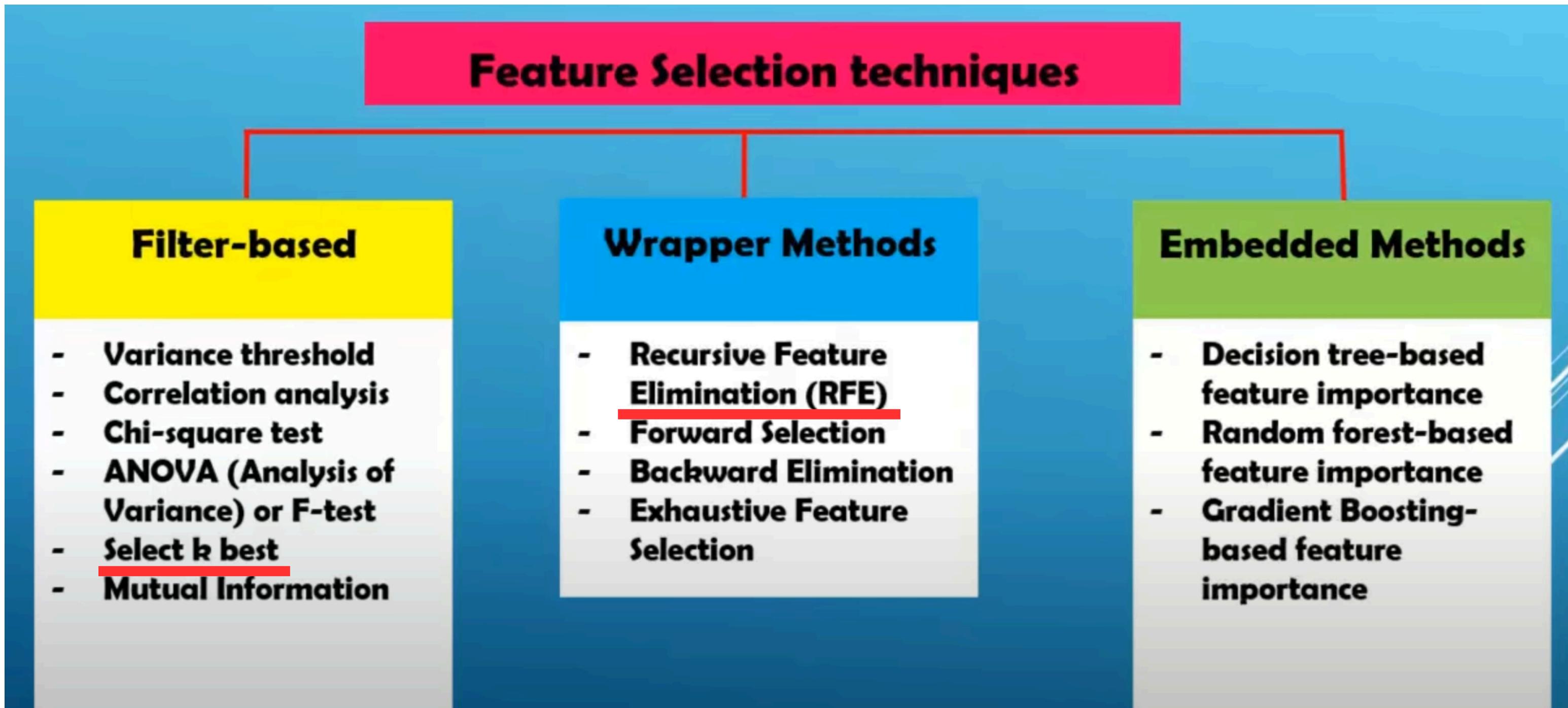
ทำไมถึงต้องใช้ Feature Selection

1. ทำให้โมเดลมีประสิทธิภาพที่ดีขึ้น

2. หลีกเลี่ยงการ Overfitting เมื่อฟีเจอร์มีมากเกินไป

3. โมเดลเบาลง และไวขึ้น

# การเลือก Feature Selection



# การรวม Feature Selection เข้าด้วยกัน

**Filter-based**

+

**Wrapper Methods**

Select k Best



Recursive Feature Elimination(RFE)

**Hybrid Method**

# 10. Filter Method

```
from sklearn.feature_selection import SelectKBest,f_classif  
  
k=200  
selector = SelectKBest(f_classif,k=k)  
X_train_selected = selector.fit_transform(X_train,y_train)  
X_test_selected = selector.transform(X_test)  
  
selected_indices=selector.get_support(indices=True)  
selected_features = X_train.columns[selected_indices]  
print(len(selected_features))
```

200

# 11. Wrapper Method

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier

estimator = RandomForestClassifier()

k=100
rfe_selector = RFE(estimator,n_features_to_select=k)
X_train_selected_rfe = rfe_selector.fit_transform(X_train_selected,y_train)
X_test_selected_rfe = rfe_selector.transform(X_test_selected)

selected_indices_rfe = rfe_selector.get_support(indices=True)
selected_features_rfe = selected_features[selected_indices_rfe]
print(selected_features_rfe)

print(len(selected_features_rfe))

100
```

```
Index(['tBodyAcc-std()-X', 'tBodyAcc-mad()-X', 'tBodyAcc-max()-X',
       'tBodyAcc-energy()-X', 'tBodyAcc-entropy()-X', 'tGravityAcc-mean()-X',
       'tGravityAcc-mean()-Y', 'tGravityAcc-max()-X', 'tGravityAcc-max()-Y',
       'tGravityAcc-min()-X', 'tGravityAcc-min()-Y', 'tGravityAcc-energy()-X',
       'tGravityAcc-energy()-Y', 'tBodyAccJerk-std()-X',
       'tBodyAccJerk-std()-Z', 'tBodyAccJerk-mad()-X', 'tBodyAccJerk-mad()-Y',
       'tBodyAccJerk-mad()-Z', 'tBodyAccJerk-max()-X', 'tBodyAccJerk-max()-Y',
       'tBodyAccJerk-max()-Z', 'tBodyAccJerk-sma()', 'tBodyAccJerk-energy()-X',
       'tBodyAccJerk-energy()-Y', 'tBodyAccJerk-iqr()-Y',
       'tBodyAccJerk-entropy()-X', 'tBodyAccJerk-entropy()-Z',
       'tBodyGyro-std()-X', 'tBodyGyro-std()-Y', 'tBodyGyro-std()-Z',
       'tBodyGyro-mad()-X', 'tBodyGyro-mad()-Y', 'tBodyGyro-mad()-Z',
       'tBodyGyro-max()-X', 'tBodyGyro-min()-X', 'tBodyGyro-iqr()-X',
       'tBodyGyro-iqr()-Y', 'tBodyGyro-iqr()-Z', 'tBodyGyroJerk-std()-X',
       'tBodyGyroJerk-std()-Z', 'tBodyGyroJerk-mad()-X',
       'tBodyGyroJerk-mad()-Z', 'tBodyGyroJerk-max()-X',
       'tBodyGyroJerk-min()-X', 'tBodyGyroJerk-sma()', 'tBodyGyroJerk-iqr()-X',
       'tBodyGyroJerk-iqr()-Z', 'tBodyAccMag-mean()', 'tBodyAccMag-std()',
       'tBodyAccMag-mad()', 'tBodyAccMag-energy()', 'tBodyAccJerkMag-mean()',
       'tBodyAccJerkMag-mad()', 'tBodyAccJerkMag-energy()',
       'tBodyAccJerkMag-iqr()', 'tBodyAccJerkMag-entropy()',
       'tBodyGyroJerkMag-mean()', 'tBodyGyroJerkMag-mad()',
       'fBodyAcc-mean()-X', 'fBodyAcc-std()-X', 'fBodyAcc-std()-Z',
       'fBodyAcc-mad()-X', 'fBodyAcc-max()-X', 'fBodyAcc-max()-Y',
       'fBodyAcc-energy()-X', 'fBodyAcc-bandsEnergy()-1,8',
       'fBodyAcc-bandsEnergy()-1,16', 'fBodyAcc-bandsEnergy()-1,24',
       'fBodyAcc-bandsEnergy()-1,8,1', 'fBodyAccJerk-mean()-Z',
       'fBodyAccJerk-std()-X', 'fBodyAccJerk-std()-Y', 'fBodyAccJerk-std()-Z',
       'fBodyAccJerk-mad()-Y', 'fBodyAccJerk-mad()-Z', 'fBodyAccJerk-sma()',
       'fBodyAccJerk-energy()-X', 'fBodyAccJerk-energy()-Y',
       'fBodyAccJerk-bandsEnergy()-1,8', 'fBodyAccJerk-bandsEnergy()-1,16',
       'fBodyAccJerk-bandsEnergy()-1,24', 'fBodyAccJerk-bandsEnergy()-1,24,1',
       'fBodyGyro-mean()-X', 'fBodyGyro-std()-X', 'fBodyGyro-std()-Y',
       'fBodyGyro-std()-Z', 'fBodyGyro-mad()-X', 'fBodyGyro-mad()-Y',
       'fBodyGyro-max()-X', 'fBodyGyro-max()-Z', 'fBodyGyro-entropy()-X',
       'fBodyAccMag-mean()', 'fBodyAccMag-std()', 'fBodyAccMag-mad()',
       'fBodyAccMag-max()', 'fBodyAccMag-energy()', 'fBodyBodyAccJerkMag-max()',
       'angle(X,gravityMean)', 'angle(Y,gravityMean)'],
      dtype='object')
```

## 12. เทคนิคใหม่โดยใช้ Hybird Method

```
log = LogisticRegression()
```

✓ 0.0s

Python

```
log.fit(X_train_selected_rfe,y_train)
```

✓ 0.2s

Python

```
accuracy_score(y_test,y_pred_log)
```

[36] ✓ 0.0s

Python

... 0.9544527532290958

# เทียบความแม่นยำและเวลาในการเทรนของทุกๆ 100 iteration

```
log = LogisticRegression()  
log.fit(X_train,y_train)
```

✓ 3.6s

100 รอบใช้เวลา  
= 3.6s

ความแม่นยำ ~98%

0.9796057104010877

หลังจากที่ใช้ Feature Selection

```
log = LogisticRegression()  
log.fit(X_train_selected_rfe,y_train)
```

✓ 0.2s

100 รอบใช้เวลา  
= 0.2s

ความแม่นยำ ~95%

0.9544527532290958

# ทดลองเทียบค่าให้เห็นภาพ

```
import joblib
```

```
joblib.dump(rf,"model_rfe")
```

```
['model_rfe']
```

```
joblib.dump(selector,"k_best_selector")
```

```
['k_best_selector']
```

```
joblib.dump(rfe_selector,"rfe_selector")
```

```
['rfe_selector']
```

```
data_test = data_test.drop("Activity",axis=1)
```

```
duplicated_columns = data_test.columns[data_test.T.duplicated()].to_list()
```

```
data_test = data_test.drop(duplicated_columns,axis=1)
```

```
model = joblib.load('model_rfe')
```

```
selector = joblib.load('k_best_selector')
```

```
rfe_selector = joblib.load('rfe_selector')
```

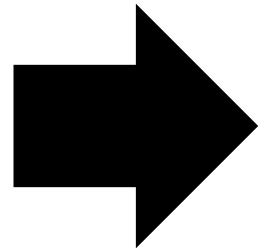
```
selector=selector.transform(data_test)
```

```
X_test_selected_rfe = rfe_selector.transform(selector)
```

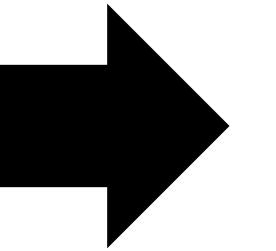
```
model.predict(X_test_selected_rfe)
```

```
array([2, 2, 2, ..., 5, 5, 5])
```

	A
1	Prediction
2	2
3	2
4	2
5	2
6	2
7	2
8	2
9	2
10	2
11	2
12	2
13	2
14	2
15	2
16	2
17	2
18	2
19	2
20	2
21	2
22	2



	UR
1	Predicted_1
2	Laying
3	Laying
4	Laying
5	Laying
6	Laying
7	Laying
8	Laying
9	Laying
10	Laying
11	Laying
12	Laying
13	Laying
14	Laying
15	Laying
16	Laying
17	Laying
18	Laying
19	Laying
20	Laying
21	Laying
22	Laying



ແທນຄ່າເປັນ

0: 'Standing'

1: 'Sitting'

2: 'Laying'

3: 'Walking\_downstairs'

4: 'Walking\_upstairs'

5: 'Walking'

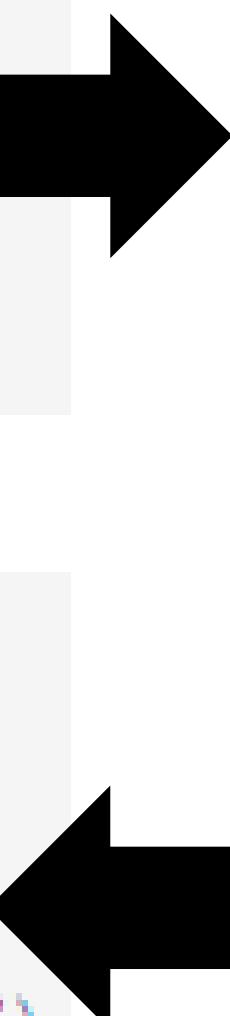
```

import tkinter as tk
from tkinter import filedialog
import pandas as pd
import joblib
from tkinter import messagebox

def open_file():
    filepath=filedialog.askopenfile(filetypes=[("CSV Files",".csv")])
    if filepath:
        try:
            data=pd.read_csv(filepath)
            process_data(data)
        except Exception as e:
            messagebox.showerror("Error",f"Failed to open file {e}")

def save_file(data):
    savepath=filedialog.asksaveasfilename(defaultextension=".csv",
                                           filetypes=[("CSV Files",".csv")])
    if savepath:
        try:
            data.to_csv(savepath)
            messagebox.showinfo("Success","File Saved Successfully")
        except Exception as e:
            messagebox.showerror("Error",f"Failed to save file:{e}")

```



```

def process_data(data):
    # Find columns with the same values
    #data= data.drop("Activity",axis=1)
    duplicated_columns = data.columns[data.T.duplicated()].tolist()
    # Remove columns with the same values

    data_test = data.drop(duplicated_columns, axis=1)

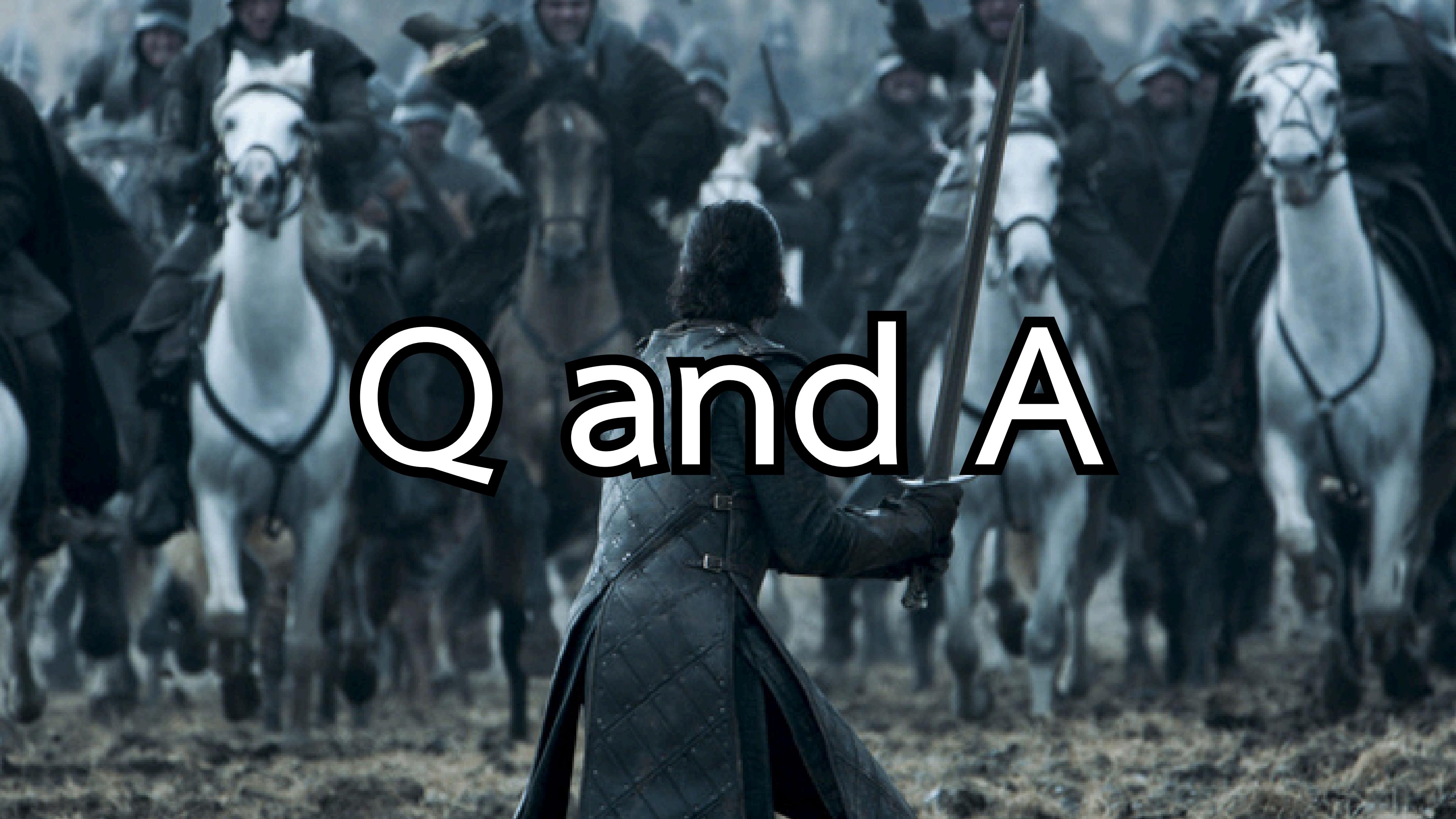
    model = joblib.load("model_rfe")
    # Load the SelectKBest object from the file
    selector = joblib.load('k_best_selector')
    rfe_selector = joblib.load('rfe_selector')

    # Transform the new data using the Loaded SelectKBest object
    X_test_selected = selector.transform(data_test)

    # Transform the new data using the Loaded RFE object
    X_test_selected_rfe = rfe_selector.transform(X_test_selected)
    y_pred=model.predict(X_test_selected_rfe)
    # standing : 0, sitting : 1, laying : 2, WALKING_DOWNSTAIRS: 3,
    # walking_upstairs:4, walking : 5
    y_pred = pd.Series(y_pred)
    y_pred = y_pred.map({0: 'Standing',1:'Sitting',2:'Laying',
                         3: 'Walking_downstairs',4: 'Walking_upstairs',
                         5:"Walking"})
    data['Predicted_target']=y_pred
    save_file(data)

def save_file(data):
    savepath=filedialog.asksaveasfilename(defaultextension=".csv",
                                           filetypes=[("CSV Files",".csv")])
    if savepath:
        try:
            data.to_csv(savepath)
            messagebox.showinfo("Success","File Saved Successfully")
        except Exception as e:
            messagebox.showerror("Error",f"Failed to save file:{e}")

```



**Q and A**

ខេត្តកណ្តាល