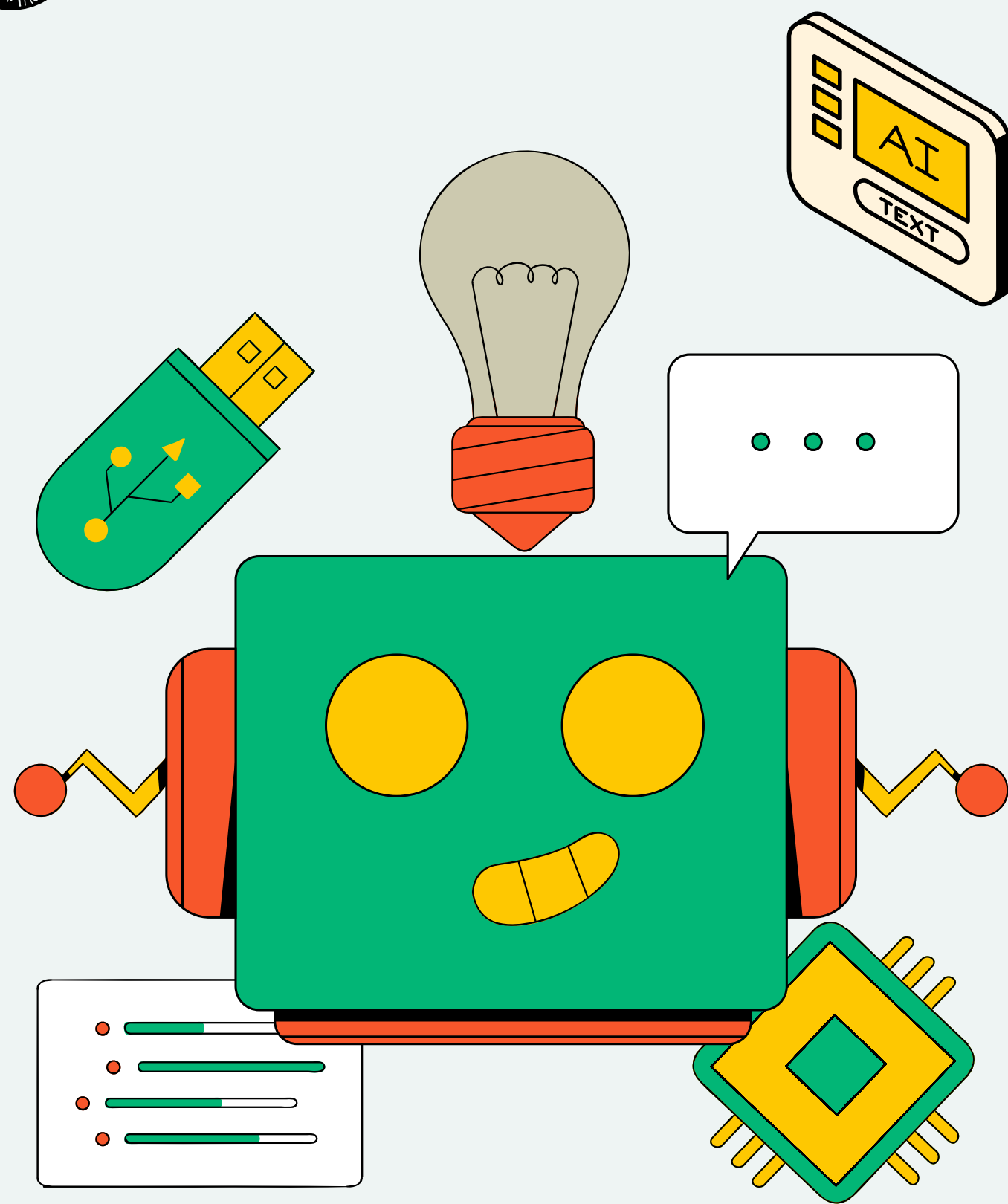




**SPACEMAN**



# **MUSHROOM CLASSIFICATION**

## **INTRODUCTION TO MACHINE LEARNING**



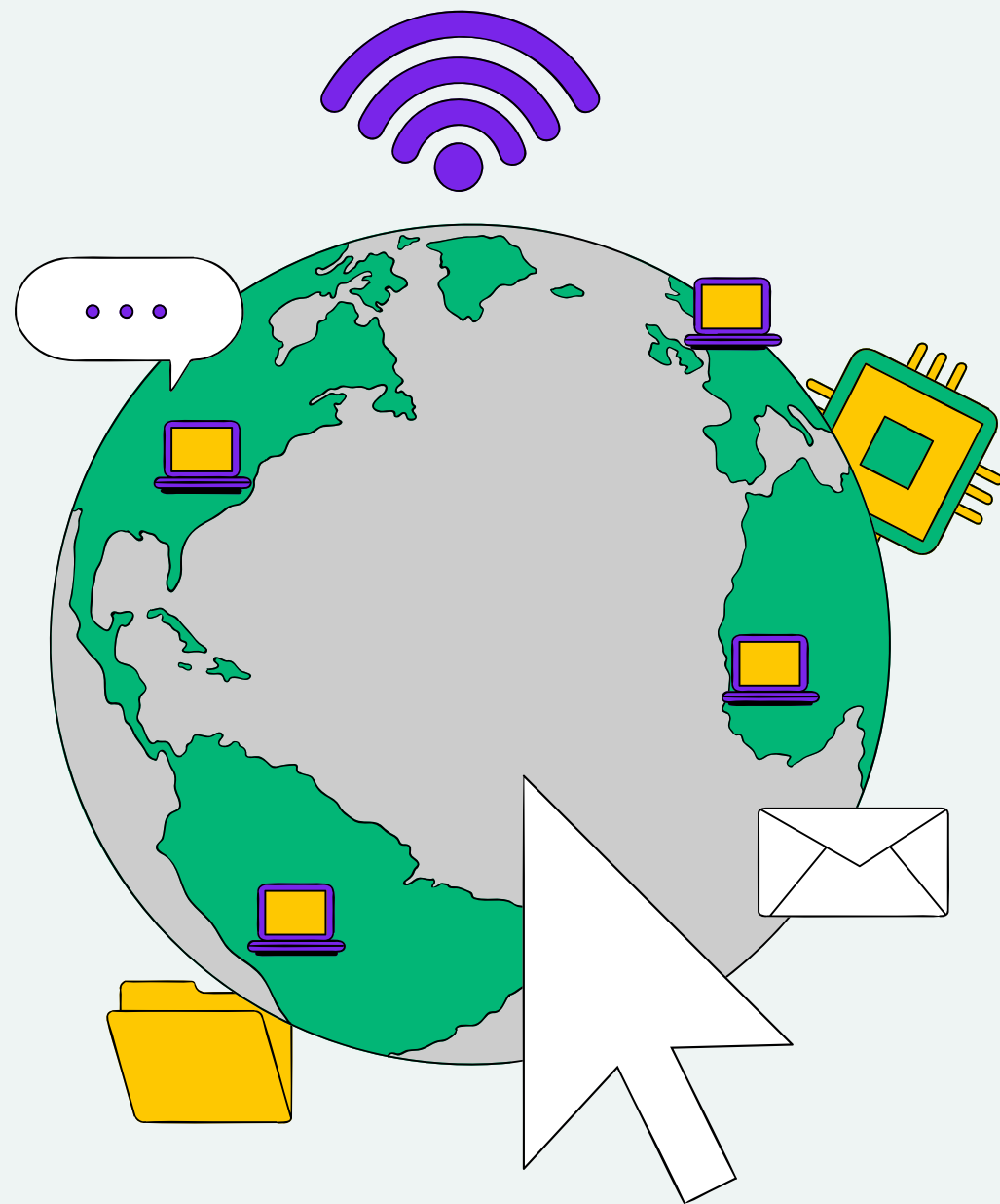


# PRESENTATION OUTLINE

- Introduction
- Objective
- Dataset
- Attribute Information
- Data Preprocessing
- Exploratory Data Analysis
- Machine Learning Models
- Model Evaluation
- Model Comparison
- Results & Discussion
- Conclusion



# INTRODUCTION



ในโปรเจกต์นี้ เรามุ่งเน้นการพัฒนาโมเดลสำหรับ การจำแนกประเภทเห็ด (Mushroom Classification) ว่าเป็น เห็ดที่กินได้ (edible) หรือ เห็ดพิษ (poisonous) โดยอาศัยเทคนิค Machine Learning เพื่อช่วยให้การระบุชนิดของเห็ดมีความแม่นยำและรวดเร็วขึ้น ซึ่งสามารถนำไปใช้ประโยชน์ในด้านต่าง ๆ เช่น การเกษตร การแพทย์ และการป้องกันความเสี่ยงจากการบริโภคเห็ดที่เป็นพิษ

ชุดข้อมูลที่ใช้ในโปรเจกต์นี้ประกอบด้วย คุณสมบัติ (features) ต่าง ๆ ของเห็ด เช่น รูปร่างหมวกเห็ด (cap-shape), สี (cap-color), กลิ่น (odor), ลักษณะเหงือก (gill-attachment) เป็นต้น โดยข้อมูลเหล่านี้จะถูกนำมาใช้ในการฝึกโมเดลเพื่อเรียนรู้และจำแนกประเภทได้อย่างมีประสิทธิภาพ

เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เราได้ทำการทดลองกับหลายโมเดล ได้แก่ Decision Tree, Random Forest และ Support Vector Machine (SVM) พร้อมทั้งวิเคราะห์และเปรียบเทียบประสิทธิภาพของแต่ละโมเดล



# OBJECTIVE

01

เพื่อพัฒนาโมเดลที่สามารถจำแนก  
เห็นได้อย่างถูกต้องและแม่นยำ

02

เพื่อศึกษาประสิทธิภาพของ  
โมเดล Machine Learning  
ที่แตกต่างกัน

03

เพื่อให้ความรู้และแนวทางในการ  
ใช้ AI ในการตรวจสอบความ  
ปลอดภัยของอาหาร



# DATASET



Mushroom Classification

Safe to eat or deadly poison?

 [kaggle.com](https://www.kaggle.com/datasets/uciml/mushroom-classification)

ชุดข้อมูลนี้ประกอบด้วยข้อมูลจำนวน 8,124 รายการ

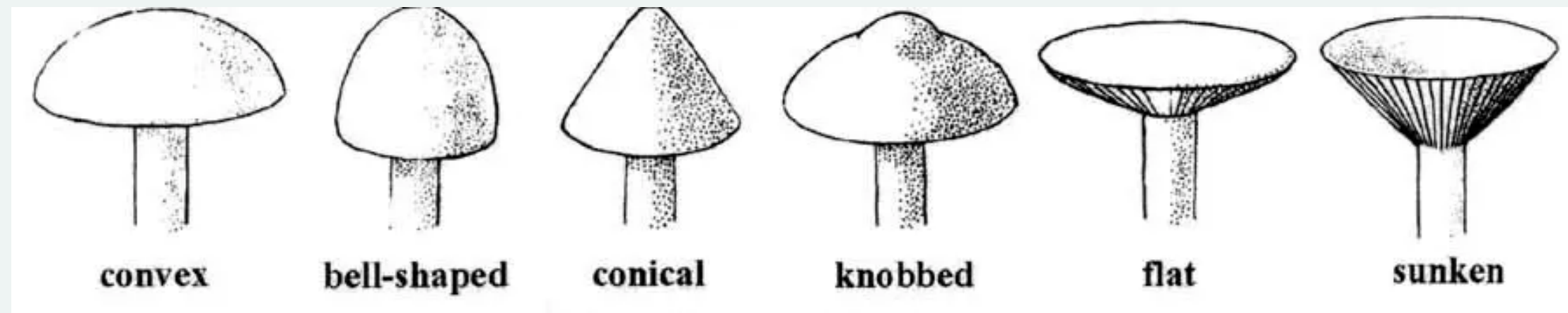
class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attach...	gill-spacing	gill-size	gill-color
p	x	s	n	t	p	f	c	n	k
e	x	s	y	t	a	f	c	b	k
e	b	s	w	t	l	f	c	b	n
p	x	y	w	t	p	f	c	n	n
e	x	s	g	f	n	f	w	b	k
e	x	y	y	t	a	f	c	b	n
e	b	s	w	t	a	f	c	b	g
e	b	y	w	t	l	f	c	b	n
p	x	y	w	t	p	f	c	n	p
e	b	s	y	t	a	f	c	b	g
e	x	y	y	t	l	f	c	b	g
e	x	y	y	t	a	f	c	b	n

Source:

Mushroom Classification: UCI MACHINE LEARNING



# ATTRIBUTE INFORMATION



## Class (ประเภทของเห็ด):

edible(กินได้) = e, poisonous(พิษ) = p

## Cap Shape (รูปร่างของหมวกเห็ด):

bell(ทรงระฆัง) = b, conical(ทรงกรวย) = c, convex(นูน) = x,

flat(แบน) = f, knobbed(มีปุ่ม) = k, sunken(เว้า) = s

## Cap Surface (พื้นผิวของหมวกเห็ด):

fibrous(เป็นเส้นใย) = f, grooves(เป็นร่อง) = g, scaly(เป็นเกล็ด) = y, smooth(เรียบเนียน) = s



### **Cap Color (สีของหมวกเห็ด):**

brown(น้ำตาล) = n, buff(เนื้อ) = b, cinnamon(อบเชย) = c, gray(เทา) = g, green(เขียว) = r, pink(ชมพู) = p, purple(ม่วง) = u, red(แดง) = e, white(ขาว) = w, yellow(เหลือง) = y

### **Bruises (รอยช้ำบนเห็ด):**

bruises(มีรอยช้ำ) = t, no(ไม่มีรอยช้ำ) = f

### **Odor (กลิ่นของเห็ด):**

almond(กลิ่นอัลมอนด์) = a, anise(กลิ่นโป๊ยกั๊ก) = l, creosote(กลิ่นน้ำมันดิน) = c, fishy(กลิ่นคาวปลา) = y, foul(กลิ่นเหม็น) = f, musty(กลิ่นอับชื้น) = m, none(ไม่มีกลิ่น) = n, pungent(กลิ่นฉุน) = p, spicy(กลิ่นเผ็ด) = s

### **Gill Attachment (การติดของเหงือกเห็ด)**

attached(ติดแน่น) = a, descending(ลาดลง) = d, free(หลุดง่าย) = f, notched(หยัก) = n

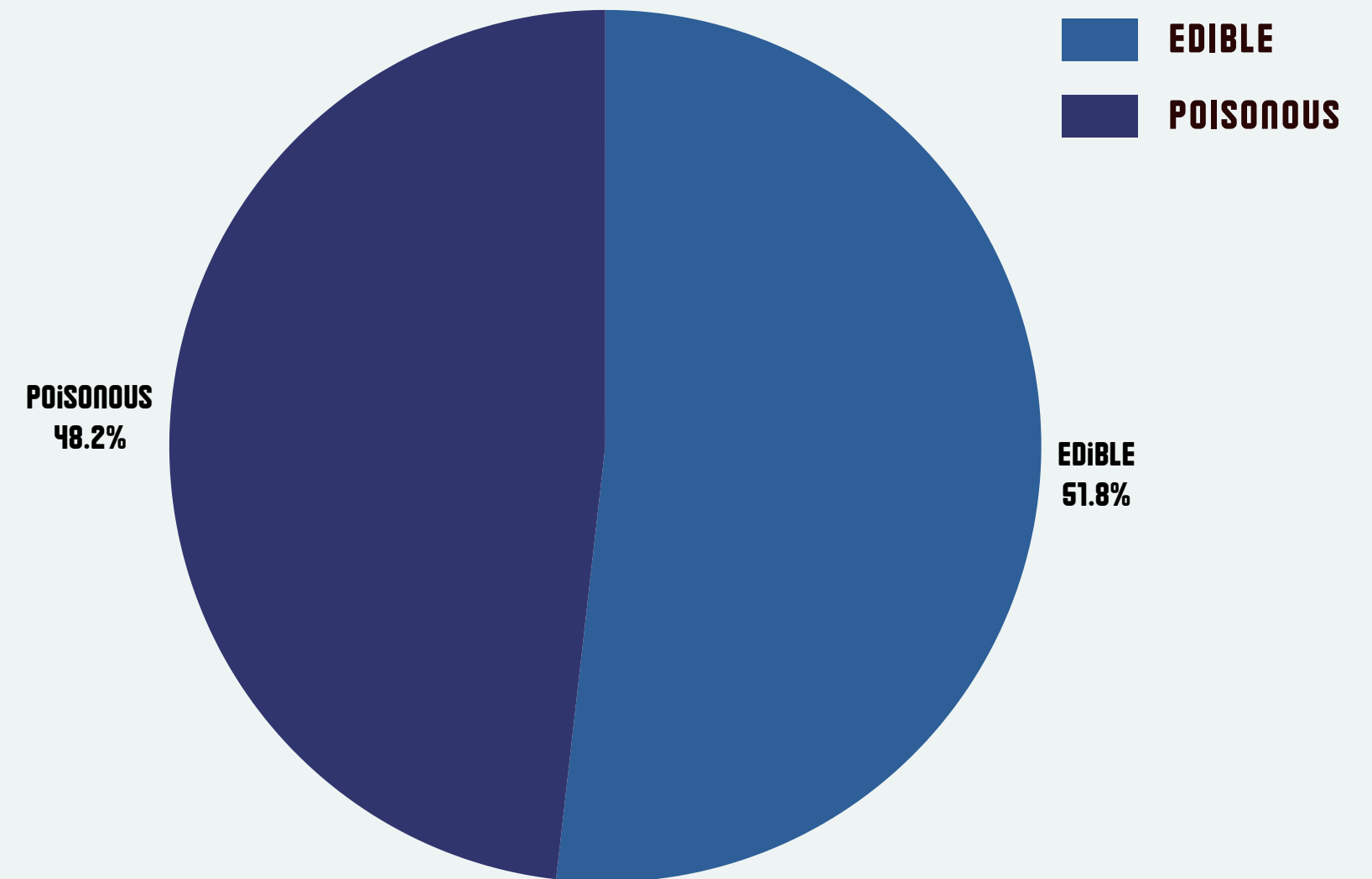
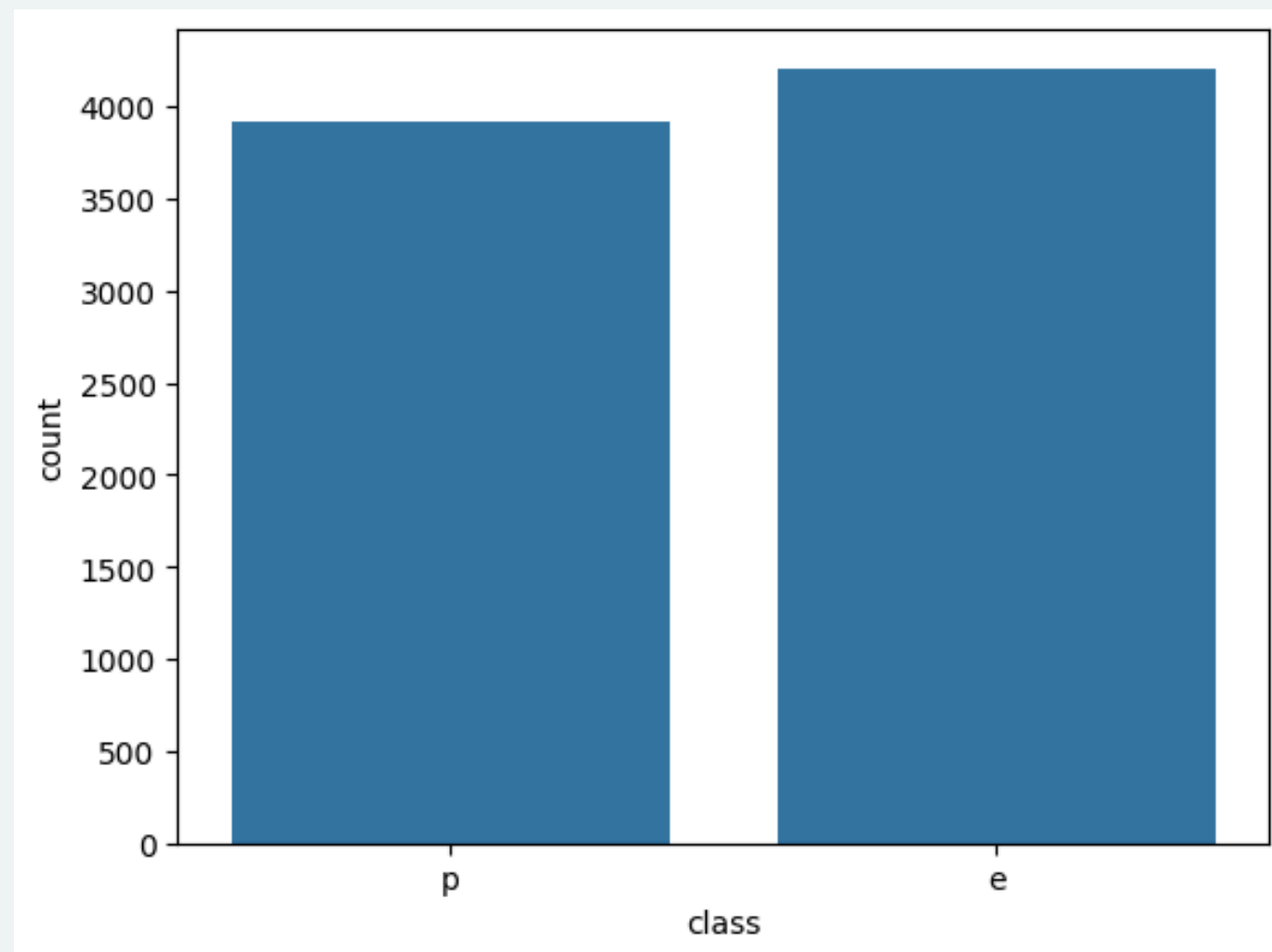
### **Gill Spacing (ระยะห่างของเหงือกเห็ด):**

close(ใกล้) = c, crowded(เบียดกันแน่น) = w, distant(ห่าง) = d





# EXPLORATORY DATA ANALYSIS

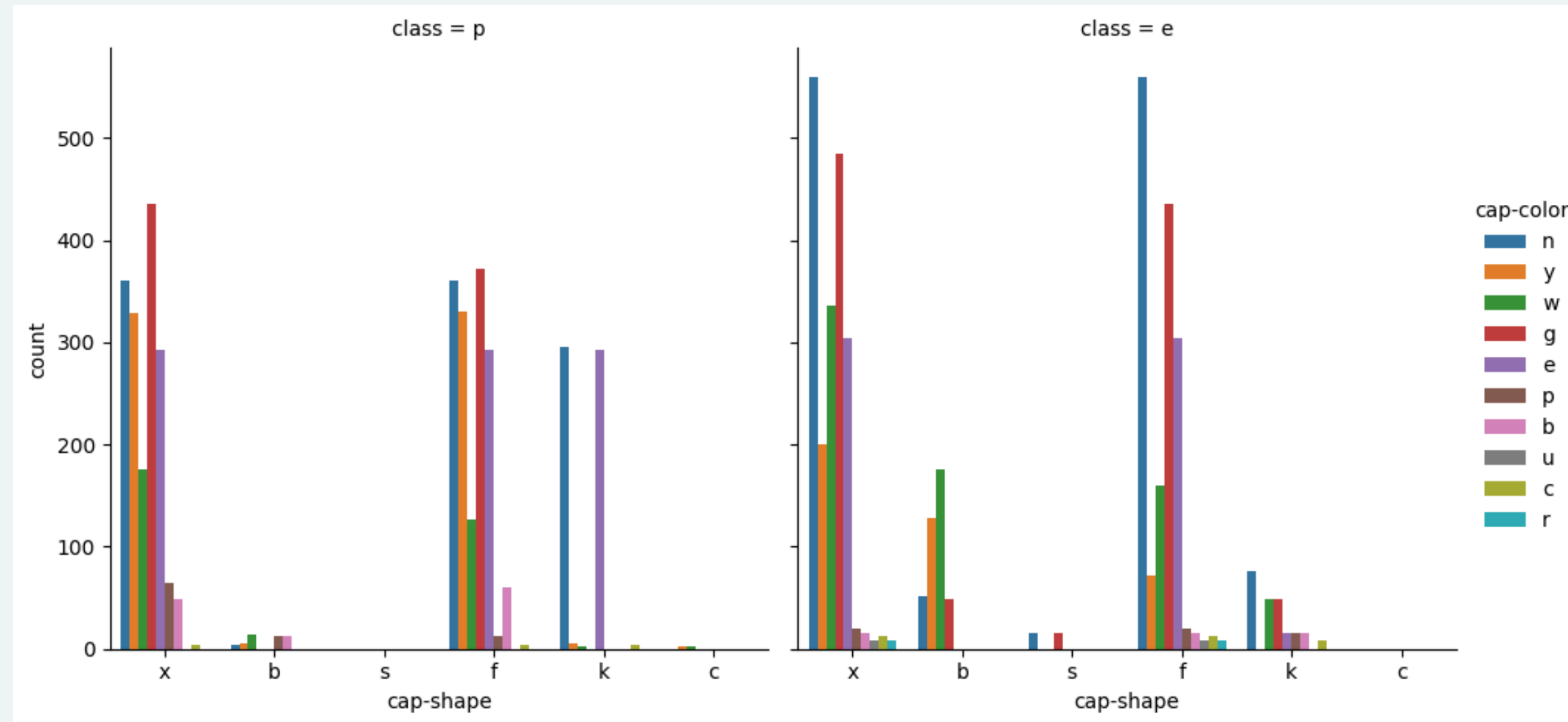


จากกราฟแสดงจำนวนเห็ดที่กินได้ (edible) เทียบกับเห็ดพิษ (poisonous)  
มีเห็ดที่กินได้ 51.80% และเห็ดพิษ 48.20%





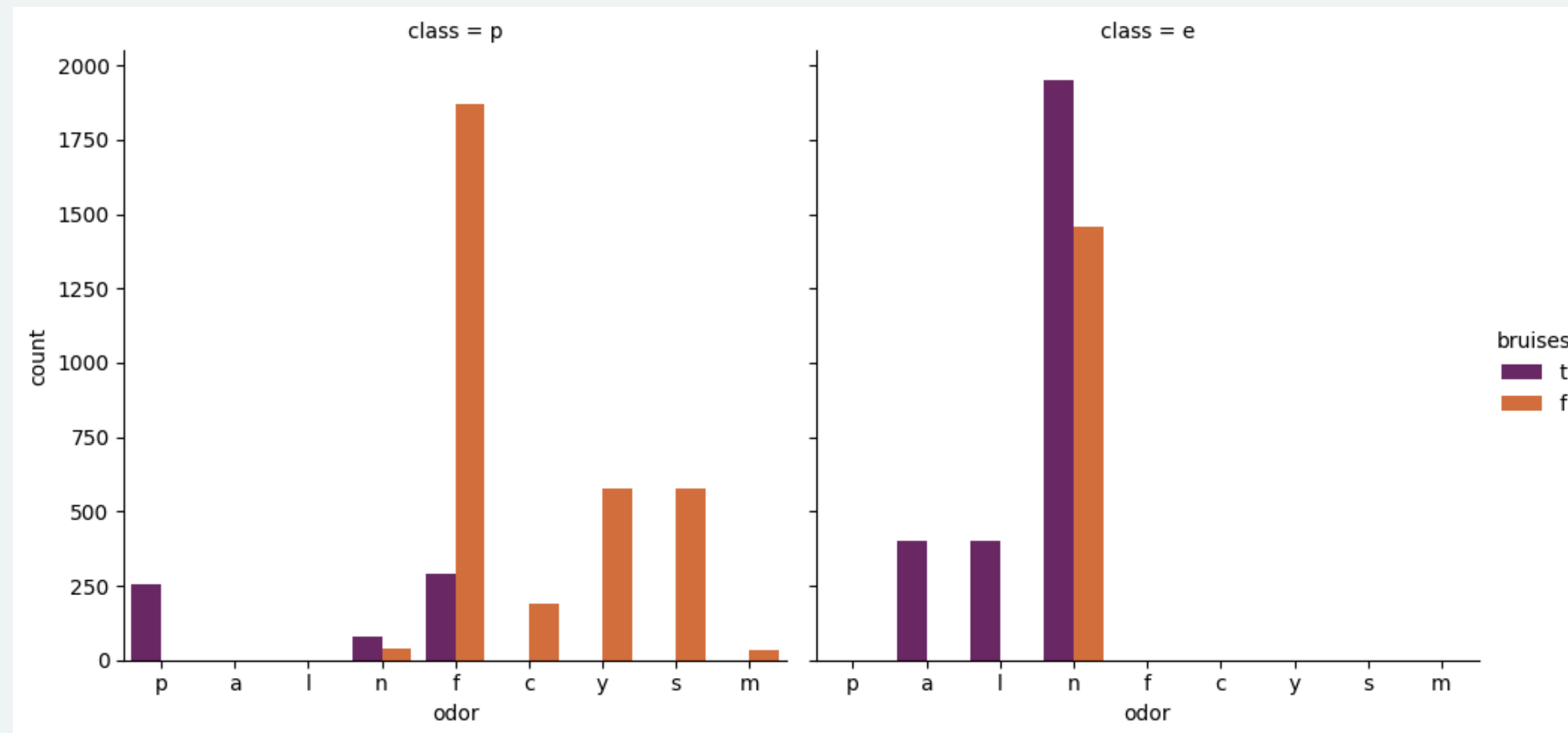
# EXPLORATORY DATA ANALYSIS



1. จากกราฟพบว่าเห็ดสีขาวทรงระฆังหรือ (White Bell Shape Mushroom) นั้นพบได้น้อยในคลาสพิษ (p) ซึ่งบ่งชี้ว่า หากเราไปเจอเห็ดสีขาวทรงระฆัง ส่วนใหญ่นั้นก็มักจะกินได้ปลอดภัย
2. สำหรับเห็ดที่มีพิษจะเห็นได้เลยว่า เห็ดสีน้ำตาล (n) และเห็ดสีแดง (e) นั้นพบได้ในคลาสพิษ (p) อย่างมาก ซึ่งบ่งชี้ว่า หากเราไปเจอเห็ดสีน้ำตาลหรือสีแดง ส่วนใหญ่มักจะมีพิษ และไม่ควายนำมาให้กิน



# EXPLORATORY DATA ANALYSIS



1. จากกราฟพบว่าเห็ดที่ไม่มีกลิ่นและมีรอยช้ำ หรือ Oderless and Bruised Mushroom นั้นปลอดภัยต่อการกิน
2. จากกราฟก็พบว่าอีกว่าเห็ดที่มีกลิ่นเหม็นและไม่มีรอยช้ำ หรือ Foul and no Bruised Mushroom นั้นส่วนใหญ่มักจะมีพิษ ไม่ปลอดภัยแก่การกิน



# DATA PREPROCESSING

## Data Cleaning

ดูว่ามีข้อมูลซ้ำกันหรือไม่

Check Duplicated

[ ] 1 df[df.duplicated(keep=False)]

class cap-shape cap-surface cap-color bruises odor gill-attachment gill-spacing gill-size gill-color ... stalk-surface-below-ring stalk-color-above-ring stalk-color-below-ring veil-type

0 rows x 23 columns

ดูค่า null ของ dataset

1 df.isnull().sum()

Check Null Value

0

class 0

cap-shape 0

cap-surface 0

cap-color 0


bruises 0



# DATA PREPROCESSING

Data Encoding: ใช้ Label Encoding (แปลงค่าหมวดหมู่เป็นตัวเลข)

```
1 from sklearn.preprocessing import LabelEncoder
2 le = LabelEncoder()
3
4 df.iloc[:, 1:] = df.iloc[:, 1:].apply(le.fit_transform)
5
6 df.head()
```



	class	cap- shape	cap- surface	cap- color	bruises	odor	gill- attachment	gill- spacing	gill- size	gill- color	...	stalk- surface- below- ring	stalk- color- above- ring	stalk- color- below- ring	veil- type
0	p	5	2	4	1	6	1	0	1	4	...	2	7	7	0
1	e	5	2	9	1	0	1	0	0	4	...	2	7	7	0
2	e	0	2	8	1	3	1	0	0	5	...	2	7	7	0
3	p	5	3	8	1	6	1	0	1	5	...	2	7	7	0
4	e	5	2	3	0	5	1	1	0	4	...	2	7	7	0

5 rows x 23 columns

+ Code

+ Text



# DATA PREPROCESSING

**Train-Test Split:** แบ่งข้อมูลออกเป็นสองส่วนเพื่อใช้สำหรับการฝึกโมเดลและทดสอบ

```
[ ] 1 x = df.drop(['class'], axis = 1)
    2 y = df['class']
    3
    4 class_names = y.unique()

[ ] 1 from sklearn.model_selection import train_test_split
    2 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

- 70% ใช้ในการฝึก (TRAINING SET)
- 30% ใช้ในการทดสอบ (TEST SET)



# MACHINE LEARNING MODELS

**Decision Tree**

**Random Forest**

**Support Vector Machine (SVM)**



# DECISION TREE CONCEPT

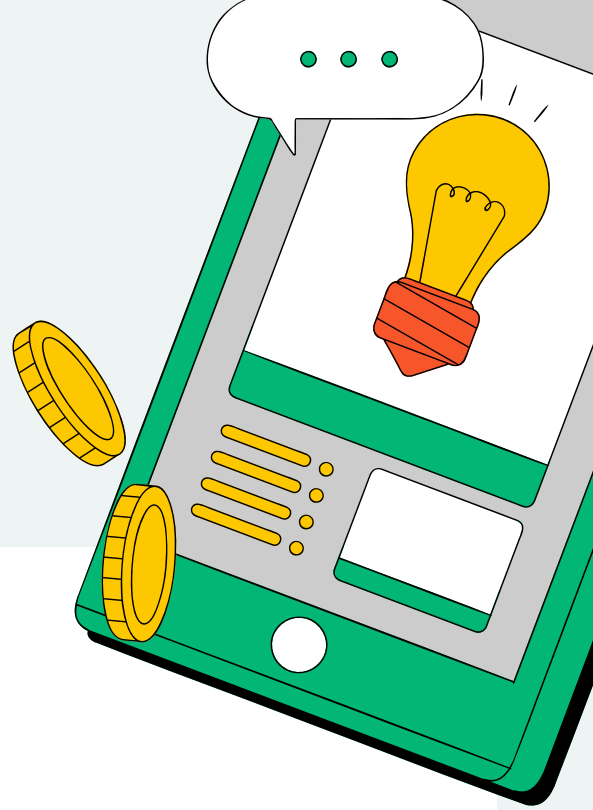
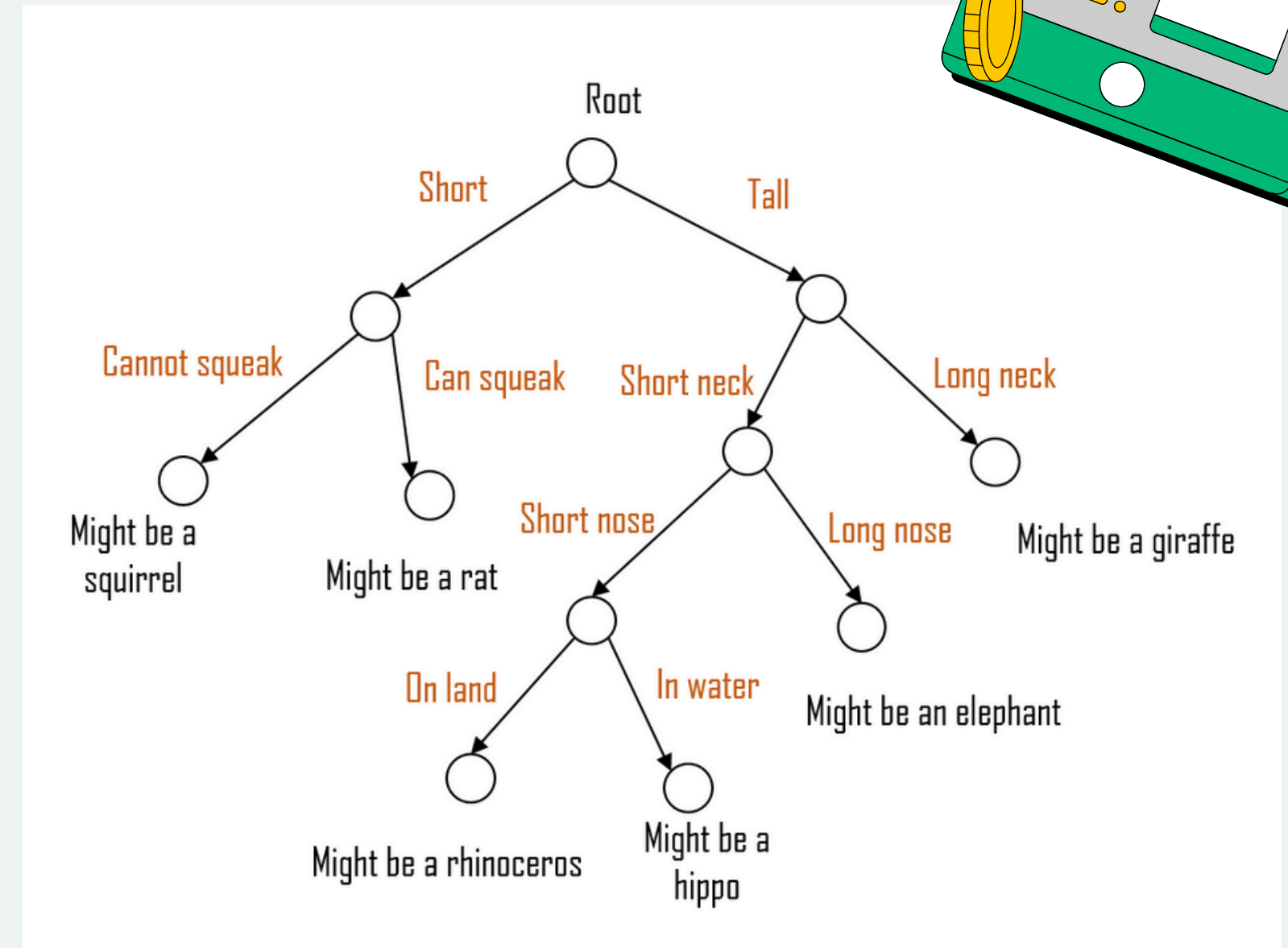
- ใช้หลักการแบ่งข้อมูลออกเป็น กิ่งก้าน (nodes) ตามคุณลักษณะที่สำคัญที่สุดโดยอาศัยเงื่อนไขแบบ ถ้าหาก (if-else)
- ต้นไม้จะถูกสร้างขึ้นจากการเลือกฟีเจอร์ที่ช่วยแยกกลุ่มข้อมูลได้ดีที่สุด (เช่น ใช้เกณฑ์ Gini Index หรือ Entropy)
- การตัดสินใจเกิดขึ้นจากการไล่ลงไปตามโครงสร้างของต้นไม้จนถึงใบไม้ (leaf node) ซึ่งระบุผลลัพธ์สุดท้าย

## ข้อดี:

- ตีความได้ง่าย (Interpretability)
- จัดการข้อมูลประเภทตัวเลขและตัวอักษรได้ดี

## ข้อเสีย:

- อาจเกิดการ Overfitting ได้ง่าย
- มีความอ่อนไหวต่อการเปลี่ยนแปลงของข้อมูล





# DECISION TREE CODE

## DECISION TREE MODEL

```
[ ] 1 from sklearn.tree import DecisionTreeClassifier
    2 dt = DecisionTreeClassifier(max_depth = 5)
    3 dt.fit(x_train, y_train)
```

```
⇒ DecisionTreeClassifier
   DecisionTreeClassifier(max_depth=5)
```

```
[ ] 1 dt.score(x_train, y_train)
```

```
⇒ 0.9785437917692579
```

```
▶ 1 prediction = dt.predict(x_test)
   2 prediction
```

```
⇒ array(['e', 'e', 'e', ..., 'p', 'p', 'e'], dtype=object)
```



# RANDOM FOREST CONCEPT

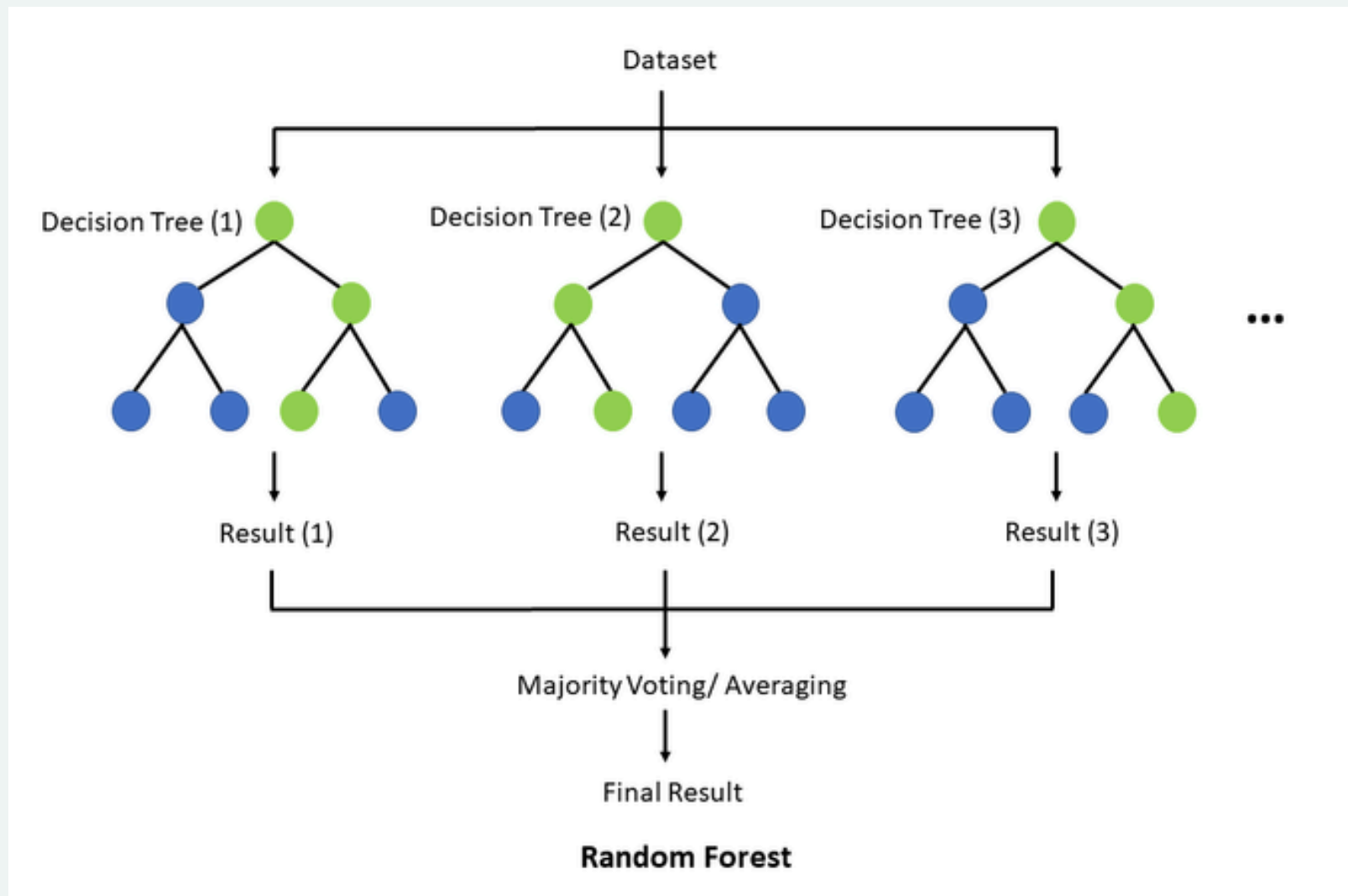
- เป็นการรวมต้นไม้ตัดสินใจ (Decision Trees) เพื่อให้ผลลัพธ์มีความแม่นยำและเสถียรขึ้น
- ใช้เทคนิค Bagging เพื่อสุ่มตัวอย่างข้อมูลและฟีเจอร์บางส่วนมาใช้ในการสร้างแต่ละต้นไม้
- ผลลัพธ์สุดท้ายเกิดจากการโหวต (Voting) ของต้นไม้ทั้งหมด (กรณี classification) หรือค่าเฉลี่ย (กรณี regression)

## ข้อดี:

- ลดปัญหา Overfitting ของ Decision Tree และทำให้ผลลัพธ์มีความน่าเชื่อถือมากขึ้น

## ข้อเสีย:

- ต้นไม้ 70% บอกว่าเห็นกินได้ และ 30% บอกว่าเห็นเห็นพิษ → ระบบเลือกตามเสียงข้างมาก



# RANDOM FOREST CODE

## RANDOM FOREST MODEL

```
1 from sklearn.ensemble import RandomForestClassifier
2 rf = RandomForestClassifier(max_depth = 5)
3 rf.fit(x_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=5)
```

```
[ ] 1 rf.score(x_train, y_train)
```

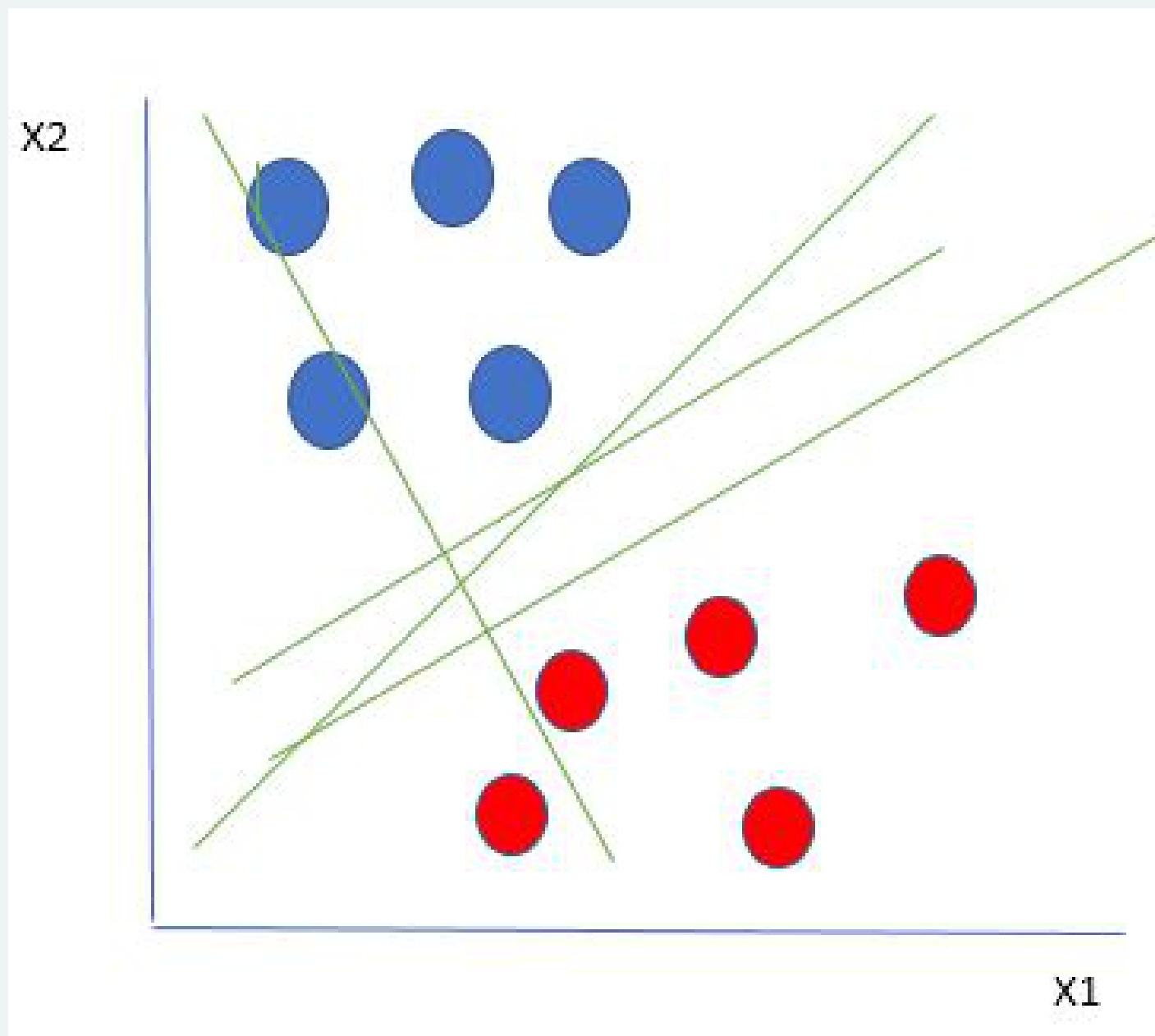
```
0.9922616953921913
```

```
[ ] 1 prediction = rf.predict(x_test)
    2 rf.score(x_test, y_test)
```

```
0.9885151763740772
```



# SUPPORT VECTOR MACHINE CONCEPT



- ใช้แนวคิดการหาขอบเขต (Hyperplane) ที่สามารถแบ่งกลุ่มข้อมูลให้ออกจากกันได้ดีที่สุด
- ระบบพยายามหาขอบเขตที่มี ระยะห่าง (Margin) ระหว่างคลาสมากที่สุดเพื่อเพิ่มความสามารถในการจำแนก
- ถ้าข้อมูลไม่สามารถแบ่งได้ง่าย จะใช้ Kernel Trick เพื่อแปลงข้อมูลไปยังมิติที่สูงขึ้นเพื่อให้แบ่งกลุ่มได้ดีขึ้น

## ข้อดี:

- ทำงานได้ดีในชุดข้อมูลที่มีมิติสูง
- ให้ผลลัพธ์ที่แม่นยำแม้ข้อมูลไม่เป็นเชิงเส้น

## ข้อเสีย:

- ต้องการการจูนค่าพารามิเตอร์ (เช่น Kernel)
- คำนวณช้าเมื่อตัวอย่างมีขนาดใหญ่



# SUPPORT VECTOR MACHINE CODE

## SUPPORT VECTOR MACHINE

```
[ ] 1 from sklearn.svm import SVC
```

```
[ ] 1 svm_model = SVC(C = 1.0, kernel = 'rbf')  
2 svm_model.fit(x_train, y_train)
```

↗

▼ SVC ⓘ ?

SVC ( )

▶ 1 svm\_model.score(x\_train, y\_train)

↗ 0.9901512486809708

```
[ ] 1 prediction = svm_model.predict(x_test)  
2 svm_model.score(x_test, y_test)
```

↗ 0.9868744872846595



# MODEL EVALUATION



**Confusion Matrix**

**Accuracy**

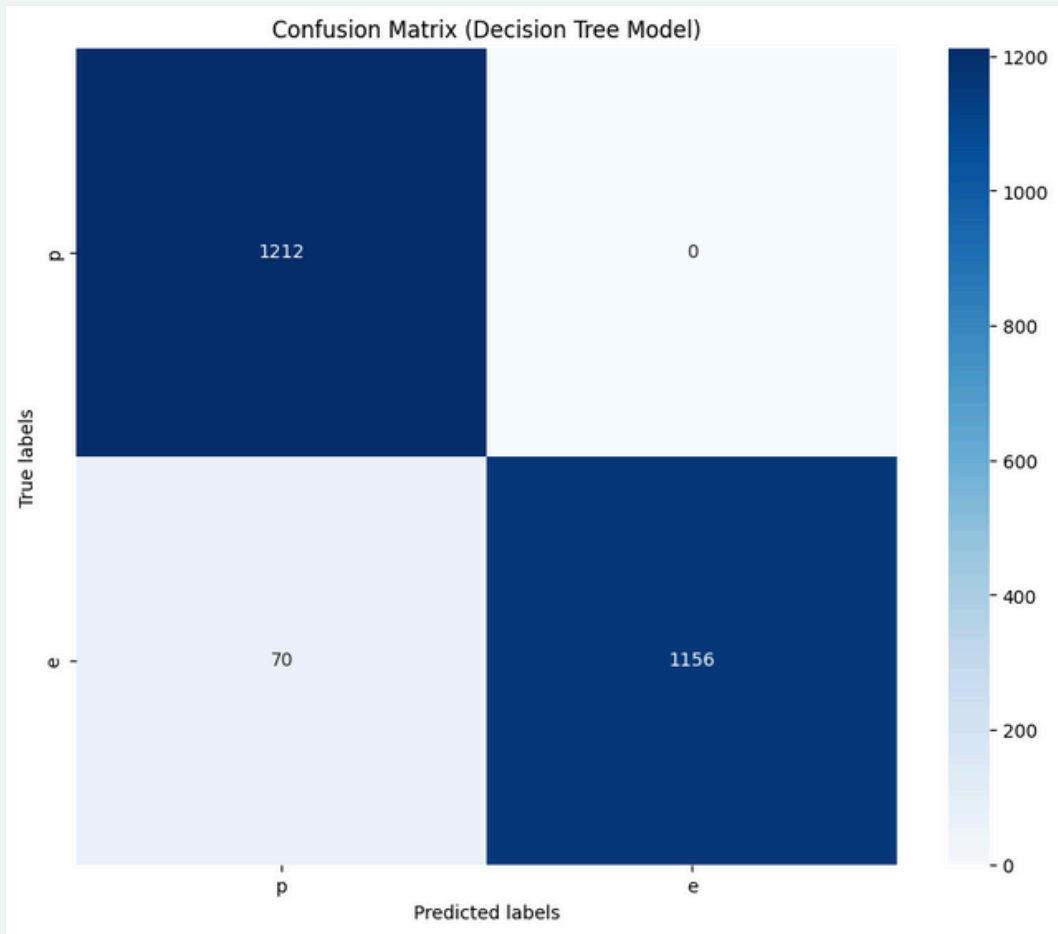
**Precision /  
Recall / F1-score**



# MODEL COMPARISON

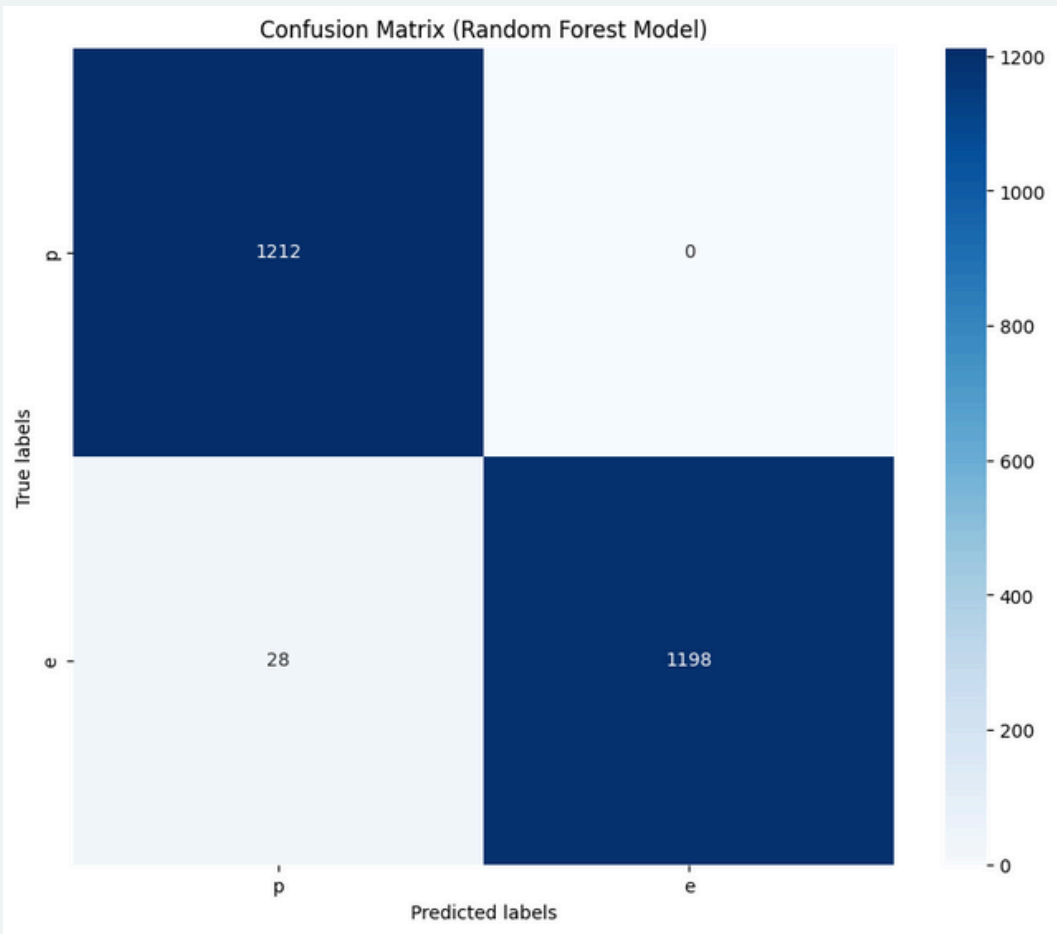
## Confusion Matrix

Decision Tree



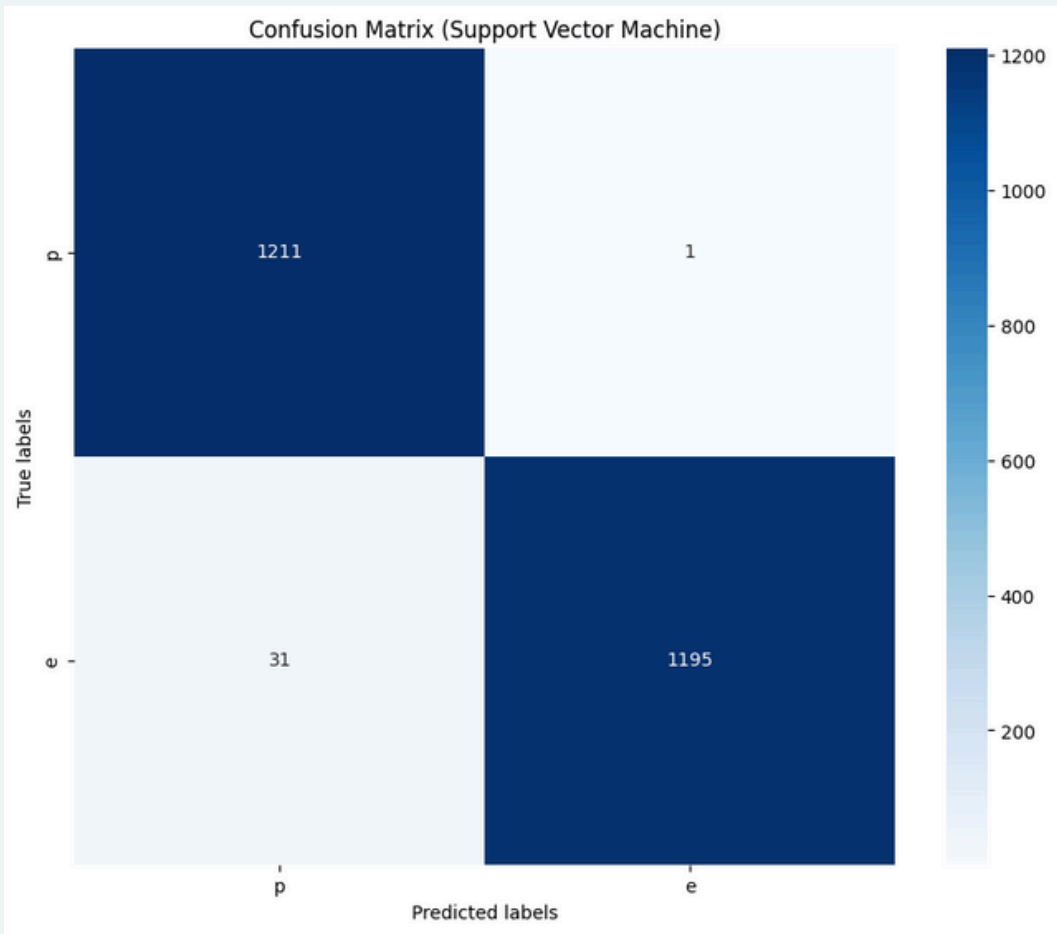
TP = 1212    FN = 0  
FP = 70    TN = 1156

Random Forest



TP = 1212    FN = 0  
FP = 28    TN = 1198

Support Vector Machine



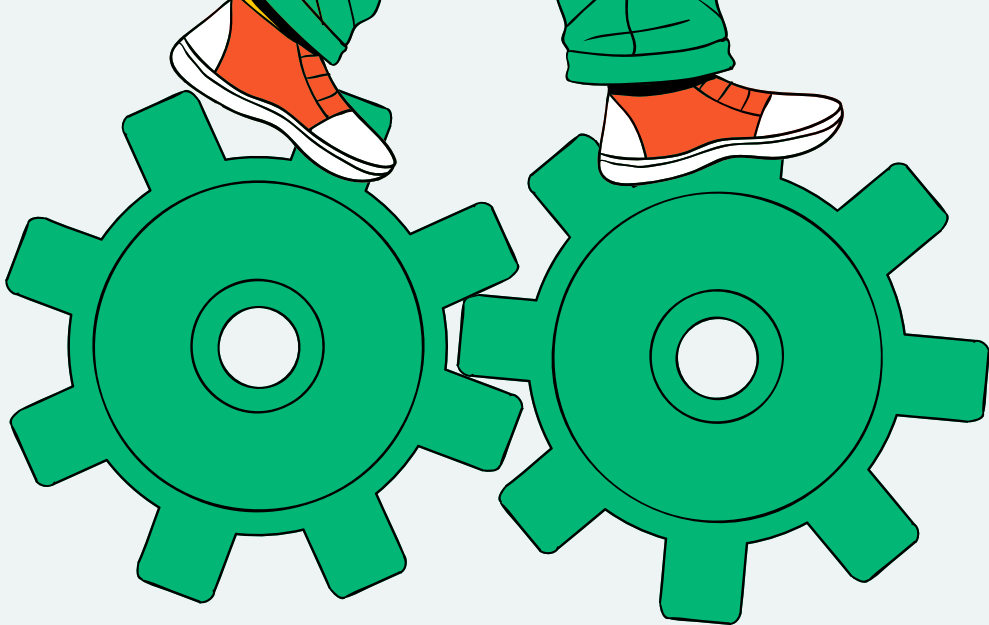
TP = 1211    FN = 1  
FP = 31    TN = 1195





# MODEL COMPARISON

Accuracy / Precision / Recall / F1-score



Accuracy

Precision

Recall

F1-score

Decision Tree

0.9713

0.9727

0.9715

0.9713

Random Forest

0.9885

0.9887

0.9886

0.9885

Support Vector  
Machine

0.9869

0.9871

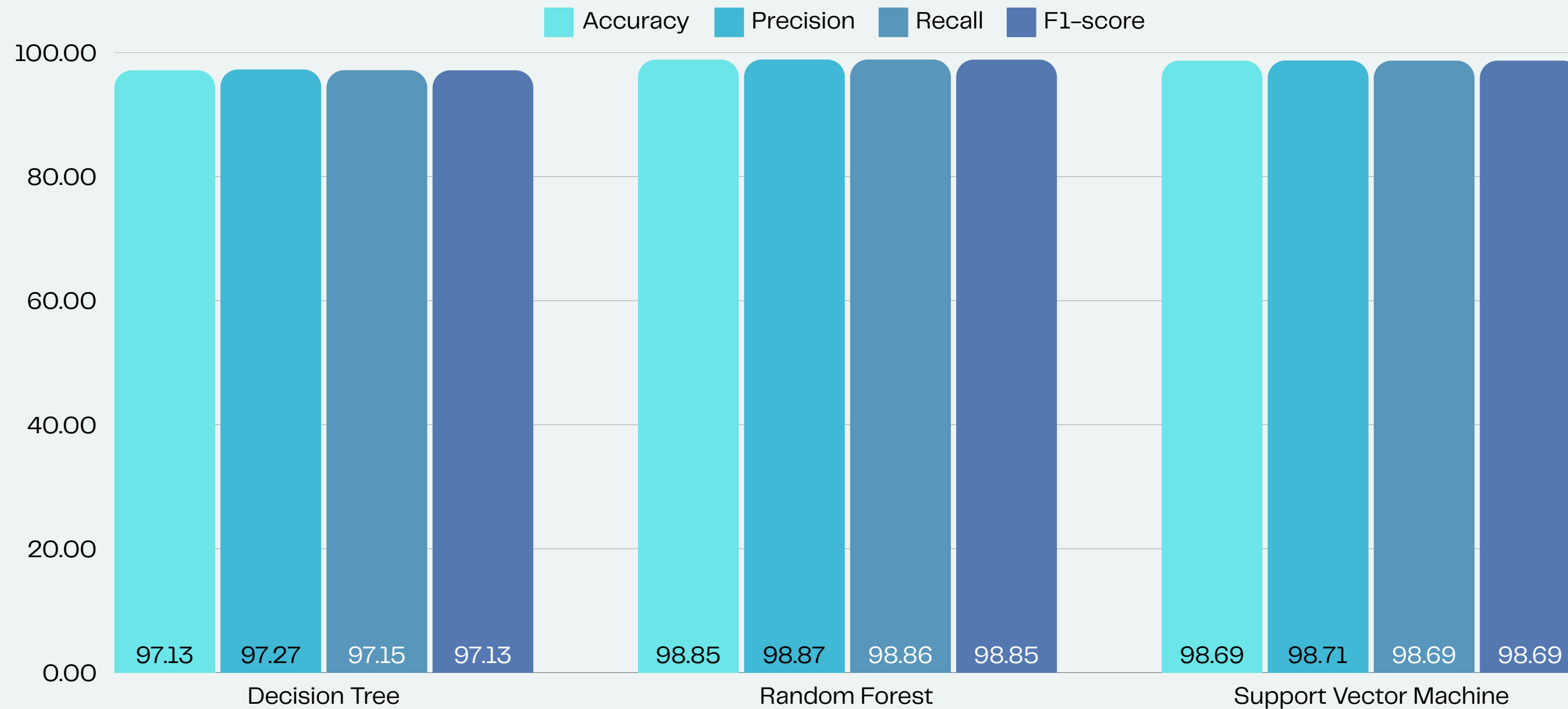
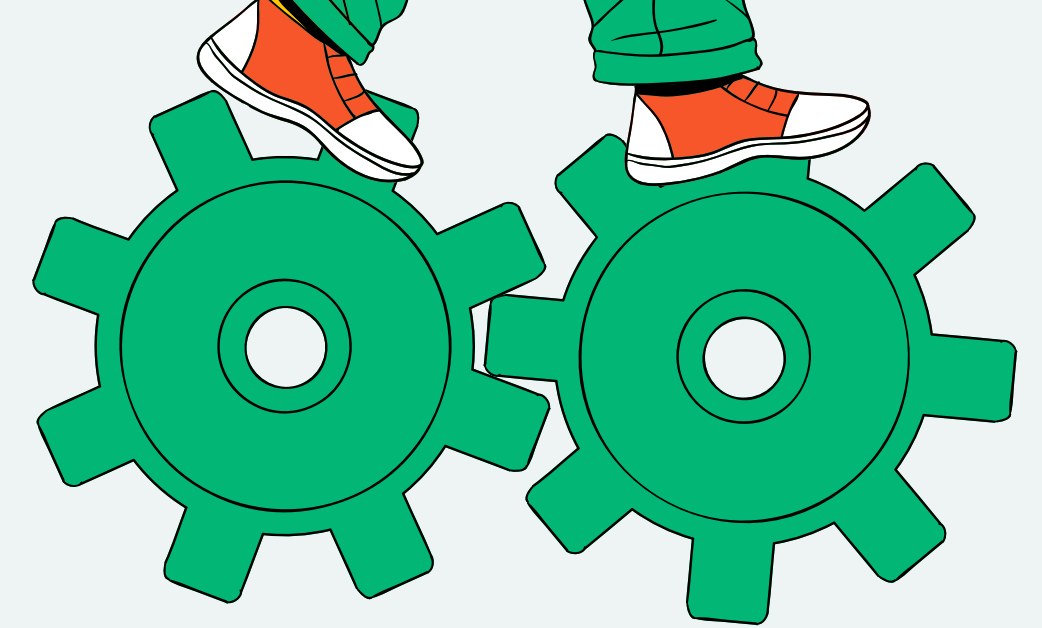
0.9869

0.9869

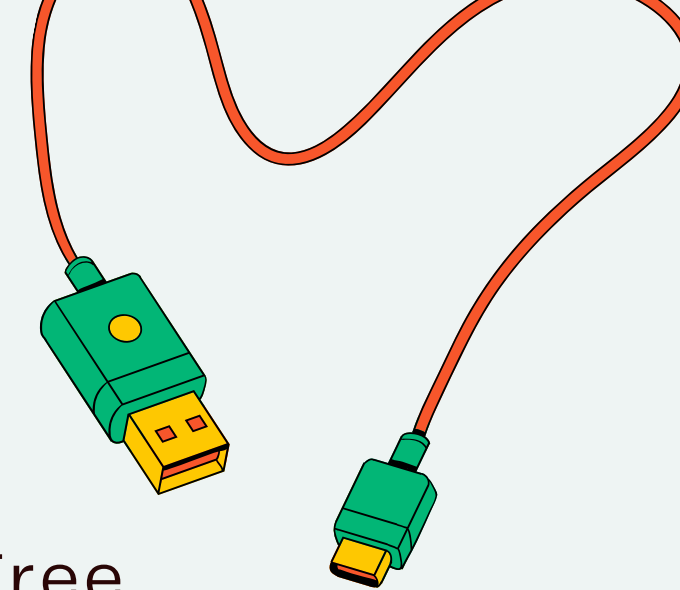


# MODEL COMPARISON

Accuracy / Precision / Recall / F1-score



# RESULTS & DISCUSSION



จากการทำโปรเจก Mushroom Classification โดยการเลือกใช้สามโมเดลหลัก ได้แก่ Decision Tree, Random Forest, และ Support Vector Machine (SVM) ผลลัพธ์ที่ได้แสดงให้เห็นถึงประสิทธิภาพที่สูงในการจำแนกประเภทเห็ดว่าเป็น "กินได้" หรือ "พิษ" โดยใช้ค่าประสิทธิภาพหลักดังนี้:

## Decision Tree:

- Accuracy: 97.13%
- Precision: 97.27%
- Recall: 97.15%
- F1-score: 97.13%

โมเดล Decision Tree มีประสิทธิภาพสูงในการจำแนกเห็ด แต่ยังมีค่า False Positive (FP) ที่สูงกว่าโมเดลอื่น ๆ ซึ่งอาจส่งผลต่อการจำแนกเห็ดที่พิษในบางกรณี



# RESULTS & DISCUSSION

## Random Forest:

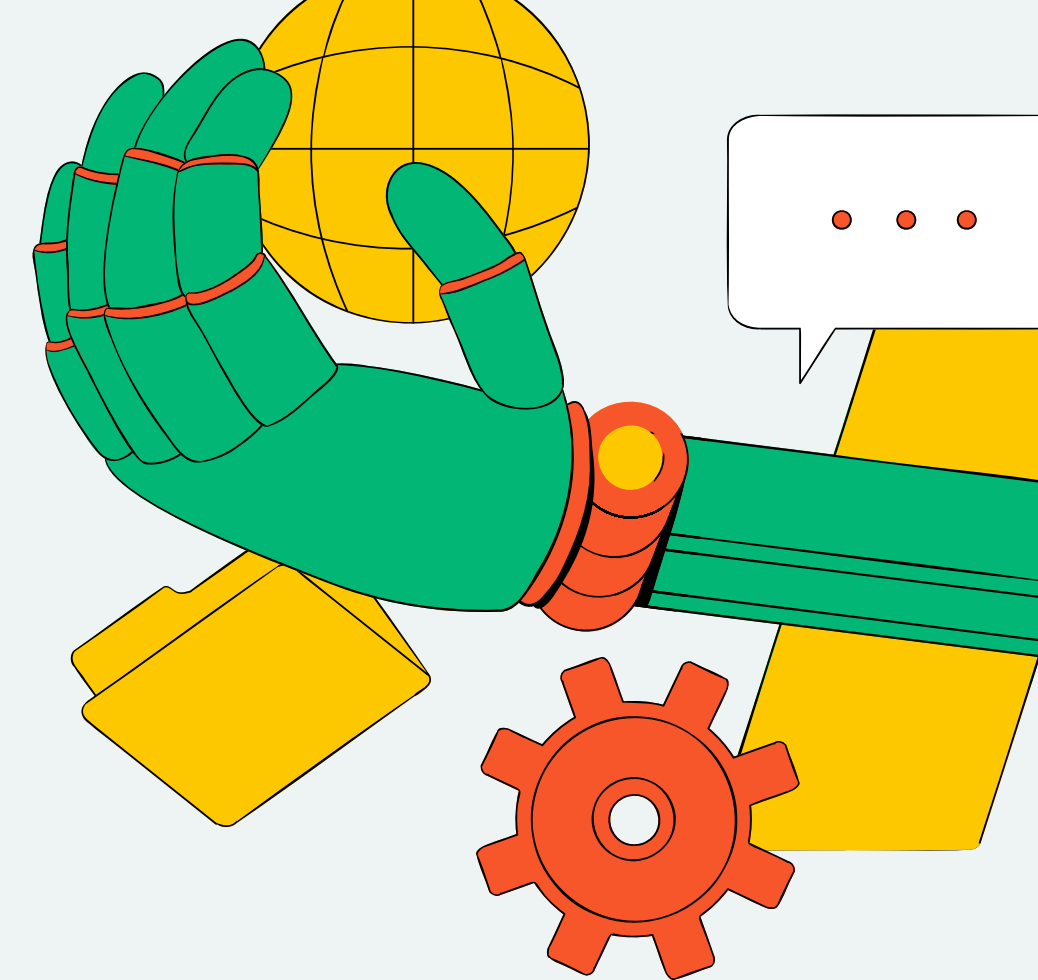
- Accuracy: 98.85%
- Precision: 98.87%
- Recall: 98.86%
- F1-score: 98.85%

**Random Forest** ให้ผลลัพธ์ที่ดีที่สุดในแง่ของความแม่นยำและความสมดุลระหว่าง Precision และ Recall โดยมีค่า False Positive (FP) ต่ำที่สุด ซึ่งแสดงถึงความสามารถในการจำแนกเห็นพ้องได้แม่นยำกว่า

## Support Vector Machine (SVM):

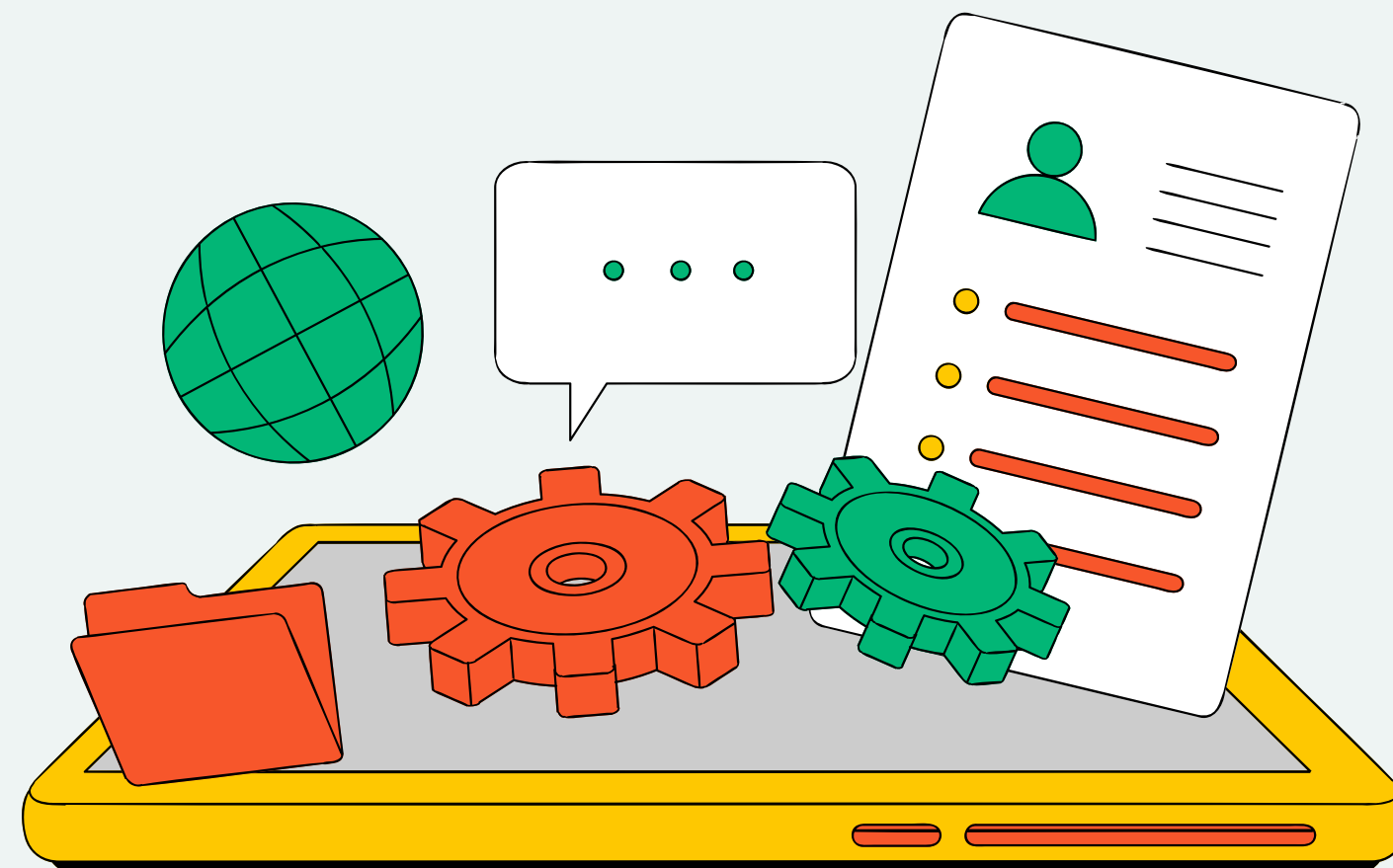
- Accuracy: 98.69%
- Precision: 98.71%
- Recall: 98.69%
- F1-score: 98.69%

โมเดล SVM มีความแม่นยำสูง แต่ไม่สูงเท่ากับ Random Forest โดยมีค่า False Positive (FP) และ False Negative (FN) ที่น้อยมาก แสดงถึงการจำแนกที่แม่นยำในกรณีที่ข้อมูลมีความซับซ้อน



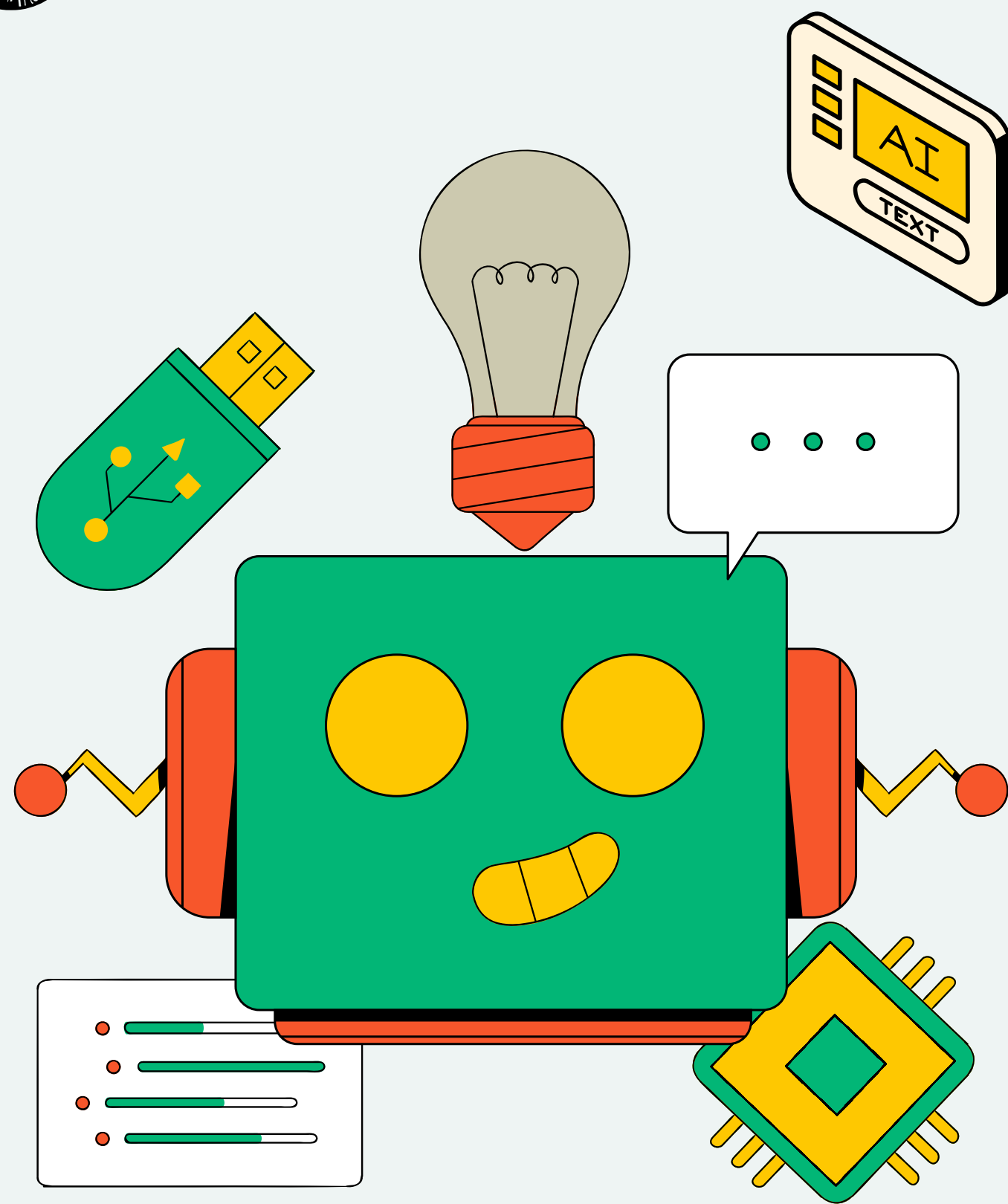
# CONCLUSION

จากการประเมินผลทั้งสามโมเดล โมเดล **Random Forest** มีประสิทธิภาพดีที่สุดในทุก ๆ ด้าน โดยมีค่า **Accuracy, Precision, Recall, และ F1-score** ที่สูงที่สุด ซึ่งเหมาะสมที่สุดสำหรับการจำแนกประเภทเห็นในโปรเจกต์นี้ ขณะที่ Decision Tree และ SVM ก็ยังสามารถให้ผลลัพธ์ที่ดีและมีความแม่นยำสูง แต่ยังคงมีข้อจำกัดในบางกรณี เช่น ค่า False Positive ที่สูงใน Decision Tree และการใช้เวลาฝึกโมเดลที่ยาวนานใน SVM





**SPACEMAN**



**THANK YOU  
FOR LISTENING**

