ชุดข้อมูล productOnline.csv

รายงานการทดลองการจำแนกความนิยมของ Product Online

1. ข้อมูลฝึก (Dataset)

ชุดข้อมูลที่นำมาใช้เพื่อให้เครื่องเรียนรู้ คือ ข้อมูลของบริษัทแห่งหนึ่งที่ขายสินค้าออนไลน์ และได้เก็บข้อมูลความนิยมการซื้อสินค้าในช่วงคริสต์มาสโดยแบ่งเป็น 4 คลาส ดังนี้

- Class A หมายถึงสินค้าที่ได้รับความนิยมการสั่งซื้อสูงมากที่สุด
- Class B หมายถึงสินค้าที่ได้รับความนิยมการสั่งซื้อสูงมาก
- Class C หมายถึงสินค้าที่ได้รับความนิยมการสั่งซื้อสูงปานกลาง
- Class D หมายถึงสินค้าที่ได้รับความนิยมการสั่งซื้อน้อย

โดยบริษัทได้จัดเก็บลักษณะของสินค้า ได้แก่ ยี่ห้อ (brand) ชื่อสินค้า (product name) หน่วยประมวลผล (processor) ลักษณะ CPU ขนาด RAM ขนาด ROM ชนิด RAM ชนิด ROM ระบบปฏิบัติการ (OS) GPU ขนาดหน้าจอ (display size) และจำนวนปีที่รับประกัน (warranty)

ในชุดข้อมูลมีข้อมูลทั้งหมด 848 ระเบียน (Record) โดยมีลักษณะตาราง ดังภาพที่ 1.1

	brand	product name	class	processor	CPU	Ram	Ram_type	ROM	ROM_type	GPU	display_size	OS	warranty
475	HP	Victus 15-fa0666TX Gaming Laptop	C	12th Gen Intel Core i5 12450H	Octa Core (4P + 4E), 12 Threads	16GB	DDR4	512GB	SSD	4GB NVIDIA GeForce RTX 3050	15.6	Windows 11 OS	2
690	Lenovo	IdeaPad 5 Pro 82SN00F0IN Gaming Laptop	C	6th Gen AMD Ryzen 7 6800HS	Octa Core, 16 Threads	16GB	LPDDR5	512GB	SSD	4GB NVIDIA GeForce RTX 3050	16.0	Windows 11 OS	3
725	LG	Gram 14 2023 ??14Z90R-G.CH54A2 Laptop	D	13th Gen Intel Core i5 1340P	12 Cores (4P + 8E), 16 Threads	16GB	LPDDR5	512GB	SSD	Intel Iris Xe Graphics	14.0	Windows 11 OS	1
530	HP	Omen 16-xf0059AX Gaming Laptop	D	7th Gen AMD Ryzen 7 7840HS	Octa Core, 16 Threads	16GB	DDR5	512GB	SSD	8GB NVIDIA GeForce RTX 4060	16.0	Windows 11 OS	3
329	Dell	Inspiron 7430 IC7430FD64T001ORS1 Laptop	C	13th Gen Intel Core i7 1355U	10 Cores (2P + 8E), 12 Threads	16GB	LPDDR5	512GB	SSD	Intel Iris Xe Graphics	14.0	Windows 11 OS	1

ภาพที่ 1.1 ตารางแสดงข้อมูลความนิยมการซื้อสินค้าในช่วงคริสต์มาส

2. จุดประสงค์

การทดลองครั้งนี้มีจุดประสงค์เพื่อจำแนกข้อมูลด้วยวิธีต่าง ๆ ได้แก่ Decision Tree, Naïve Bayes (Multinomial Naïve Byes), Support Vector Machine (SVM) และ K-Nearest Neighbors โดยมีเป้าหมายที่ต้องการจำแนกคือ ระดับความนิยม โดยให้เครื่องวิเคราะห์จากข้อมูลลักษณะของสินค้า

3. การกำหนด Target แล Feature

Target คือ ระดับความนิยม

Features คือ ลักษณะของสินค้า ได้แก่ ยี่ห้อ (brand) ชื่อสินค้า (product name) หน่วยประมวลผล (processor) ลักษณะ CPU ขนาด RAM ขนาด ROM ชนิด RAM ชนิด ROM ระบบปฏิบัติการ (OS) GPU ขนาดหน้าจอ (display size) และจำนวนปีที่รับประกัน (warranty)

4. การเตรียมข้อมูล (Data Preprocessing)

ในทุกการทดลองจะมีการทำ Preprocessing ซึ่งมีบางขั้นตอนที่ทำเหมือนและต่างกัน โดยขั้นตอนที่ทำเหมือนกันในทุกการทดลอง มีดังนี้

1. **เปลี่ยนข้อมูลตัวอักษรให้เป็นตัวอักษรพิมพ์เล็กทั้งหมด**

   เนื่องจากแม้ว่าข้อมูลจะมีความหมายเดียวกัน แต่ถ้าเขียนตัวอักษรพิมพ์ใหญ่หรือเล็กต่างกัน คอมพิวเตอร์จะเข้าใจว่ามันคือข้อมูลที่แตกต่างกัน เลยต้องทำเพื่อให้เข้าใจว่าเป็นข้อมูลที่เหมือนกัน

2. **ทำให้ Multiple Spaces กลายเป็น Single Space**

   วิธีนี้จะทำกับข้อมูลที่เป็นตัวอักษร เพราะมีข้อมูลบางตัวที่มีความหมายเหมือนกัน แต่มีจำนวนเว้นวรรคต่างกัน ทำให้คอมพิวเตอร์เข้าใจว่าเป็นข้อมูลที่แตกต่างกัน เลยต้องทำเพื่อให้เข้าใจว่าเป็นข้อมูลที่เหมือนกัน

3. **เข้ารหัสข้อความเป็นตัวเลข (Encode String Data) Target**

   เปลี่ยนข้อมูลในคอลัมน์ Class ให้เป็นตัวเลขเพื่อความเข้าใจที่ตรงกัน และง่ายต่อการวิเคราะห์ข้อมูลเพราะมันจะถูกนำไปเปลี่ยนเป็นตัวเลขอยู่แล้วในภายหลัง แต่ถ้าเราเปลี่ยนเองเราจะรู้ว่าตัวเลขใดหมายถึงข้อมูลใด โดยจะเปลี่ยนข้อมูล A, B, C, D ให้เป็น 0, 1, 2, 3 ตามลำดับ

ต่อมาการทำ preprocessing ที่ทำเฉพาะใน Decision Tree และ SVM คือ

1. **เข้ารหัสข้อความเป็นตัวเลข (Encode String Data) Feature**

   เนื่องจากคอมพิวเตอร์ไม่สามารถนำข้อมูลที่เป็นข้อความ หรือตัวอักษรไปคำนวณได้ จึงต้องเปลี่ยนข้อมูลที่เป็นตัวอักษรให้เป็นตัวเลขก่อน โดยเปลี่ยนให้ข้อมูลที่เหมือนกันเป็นตัวเลข

เดียวกัน โดยใช้คลาส sklearn.preprocessing.LabelEncoder เปลี่ยนข้อมูลให้เป็นตัวเลข โดยจะเริ่มต้นที่ 0 และนับต่อไปเรื่อย ๆ

และขั้นตอนที่ทำเฉพาะในการทดลอง Naïve Bayes และ K-Nearest Neighbors คือ

1. ทำ One-Hot encoding (Feature)

เนื่องจากข้อมูลที่มีอยู่มีลักษณะเป็นกลุ่ม ๆ (Categorical) เพื่อให้เครื่องทำงานง่ายขึ้นจึงทำ One-Hot encoding โดยใช้ Library sklearn.preprocessing.OneHotEncoder ซึ่งก็คือการเปลี่ยนข้อมูลให้อยู่ในรูปแบบของ Binary values ที่มีค่า 0 หรือ 1 โดยจะนำข้อมูลในแต่ละคอลัมน์มาสร้างเป็นคอลัมน์ใหม่ ซึ่งจะกลายเป็น Feature แทน โดยหากข้อมูลมีลักษณะเป็นดังคอลัมน์นั้นจะมีค่าเป็น 1 แต่ถ้าไม่จะมีค่าเป็น 0 เนื่องจากทำทุกคอลัมน์ที่เป็น Feature ทำให้มีคอลัมน์เพิ่มขึ้นมาเป็นจำนวนมาก ดังนั้นแล้ว Feature จึงมีจำนวนเพิ่มขึ้นเช่นกัน คือมีจำนวน 1090 Features แต่ยังคงเป็นลักษณะของสินค้าเหมือนเดิม เพียงแต่แจกแจงออกมา ดังภาพที่ 4.1

	brand_acer	brand_apple	brand_asus	brand_avita	brand_axl	brand_chuwi	brand_dell	brand_fujitsu	brand_gigabyte	brand_honor	...
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

5 rows × 1090 columns

ภาพที่ 4.1 After One-Hot encoding

5. การเรียนรู้ของเครื่อง

ในขั้นตอนนี้เราต้องการจะสร้าง Model ที่สามารถจำแนกระดับความนิยมออกมาให้ได้ โดยเราจะเริ่มที่การแบ่งข้อมูลออกเป็น 2 ชุด คือชุดข้อมูลฝึก ชุดข้อมูลทดสอบ โดยใช้ library ของ sklearn.model_selection.train_test_split ซึ่งจำเป็นต้องใส่ random state เพื่อให้ทุกครั้งที่มีการรันใหม่การแบ่งข้อมูลจะแบ่งเหมือนเดิม และต้องกำหนด test_size เพื่อระบุขนาดของชุดข้อมูลทดสอบว่าเป็นร้อยละเท่าใดของชุดทั้งหมด และที่เหลือจะเป็นชุดข้อมูลฝึก

ต่อมาเราจะทำการเลือกค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมเพื่อให้ Model ของเรามีประสิทธิภาพ โดยในการทดลองนี้มีการเลือกไฮเปอร์พารามิเตอร์ 2 วิธี คือ

1. สุ่ม คือการสุ่มค่า ๆ หนึ่งขึ้นมา จากนั้นก็ค่อย ๆ ปรับเพิ่มหรือลดค่าให้ได้ Accuracy ที่สูงที่สุด
2. ใช้ Library GridSearchCV คือการใช้เทคนิค k-fold Cross-validation และ Library GridSearchCV มาช่วยในการหาค่า Hyperparameter โดยจะสร้าง Dictionary ที่ Key คือ ชื่อ Hyperparameter และ Value คือ List ของค่าที่ต้องการนำมาเปรียบเทียบเพื่อหาค่าที่ดีที่สุด จากนั้นให้ GridSearchCV เปรียบเทียบเอาค่าที่ดีที่สุดมาให้ ต้องมีการกำหนด cv ซึ่งเป็นไฮเปอร์พารามิเตอร์ของ GridSearchCV เป็นตัวระบุ k-fold Cross-validation

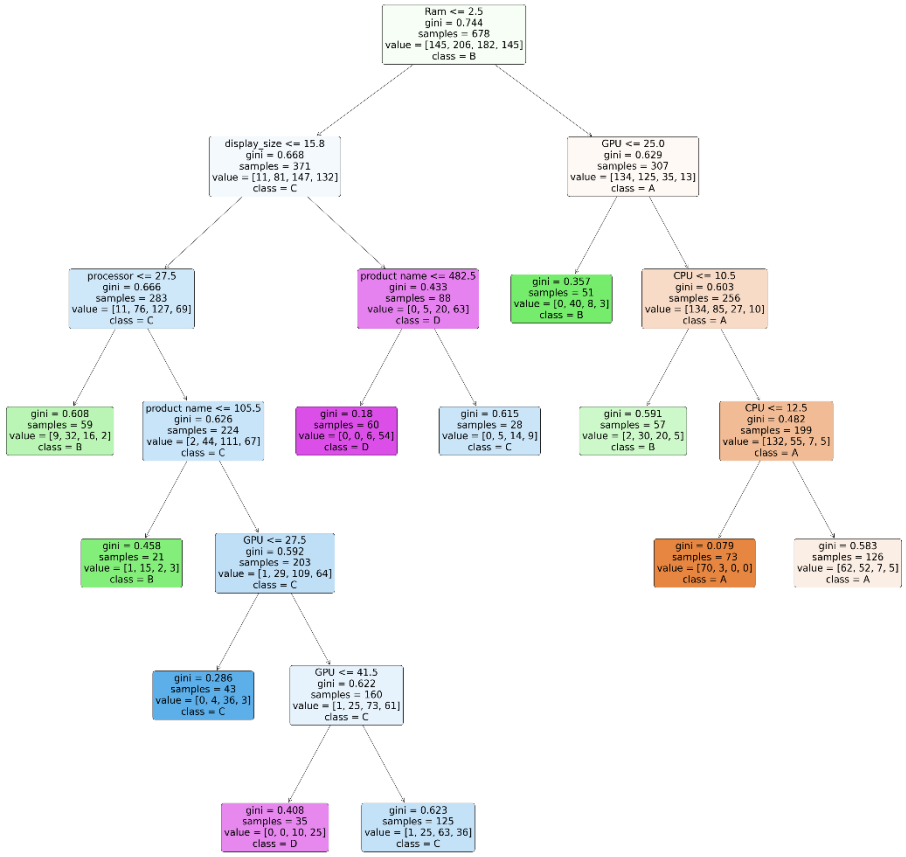
หลังจากเลือกพารามิเตอร์แล้วก็นำมาสร้างเป็น Model ซึ่ง Model แต่ละเทคนิคก็จะเรียกใช้ Library ที่แตกต่างกัน จากนั้นก็นำชุดข้อมูลฝึกที่ได้แบ่งไว้นำไปให้ Model เรียนรู้ รายละเอียดการสร้าง Model และการกำหนดไฮเปอร์พารามิเตอร์ในแต่ละเทคนิคมีรายละเอียดดังนี้

1. Decision Tree

1.1 แบ่งชุดข้อมูลฝึกด้วย train_test_split โดยกำหนดให้ test_size = 0.2 และ random_state = 12 คือกำหนดให้ชุดข้อมูลทดสอบมีขนาด 20 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด ดังนั้นชุดข้อมูลฝึกจึงมีขนาดเท่ากับ 80 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด โดยการสุ่มจะสุ่มตาม random state ที่ 12

1.2 ค่าของไฮเปอร์พารามิเตอร์เลือกมาแบบสุ่มแล้วจึงปรับให้ได้ Accuracy สูงสุด ผลลัพธ์คือ max_depth = 8 และ ccp_alpha = 0.01

1.3 ใช้คำสั่ง DecisionTreeClassifier() จาก sklearn.tree.DecisionTreeClassifier โดยกำหนดค่าไฮเปอร์พารามิเตอร์ตามที่ได้เลือกไว้ จากนั้นนำข้อมูลฝึกไปให้ Model เรียนรู้ แล้วได้ต้นไม้การตัดสินใจดังภาพที่ 5.1



ภาพที่ 5.1 ต้นไม้การตัดสินใจ

2. SVM

2.1 แบ่งชุดข้อมูลฝึกด้วย train_test_split โดยกำหนดให้ test_size = 0.15 และ random_state = 189 คือกำหนดให้ชุดข้อมูลทดสอบมีขนาด 15 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด ดังนั้นชุดข้อมูลฝึกจึงมีขนาดเท่ากับ 85 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด โดยการสุ่มจะสุ่มตาม random state ที่ 189

2.2 ใช้ GridSearchCV โดยกำหนด Dictionary ของค่าที่จะนำไปเปรียบเทียบดังภาพที่ 5.2 และกำหนดค่า cv = 2 หรือก็คือกำหนดให้เป็น 2-fold Cross-Validation จากนั้นก็นำชุดข้อมูลฝึกให้เรียนรู้ หลังการประมวลผลพบว่า Hyperparameter ที่ดีที่สุดคือ C = 1000, decision_function_shape = ovo, gamma = 0.0001, kernel = rbf, shrinking = True, และ tol = 0.1

```
grid = {
    'decision_function_shape': ['ovo'],
    'kernel': ['rbf'],
    'gamma': [0.1, 0.001, 0.0001],
    'C': [0.1, 1, 10, 50, 100, 1000],
    'tol': [0.1, 0.01, 0.001],
    'shrinking': [True, False]
}
```

ภาพที่ 5.2 Possible Hyperparameter Dictionary (SVM)

2.3 สร้าง Instance of SVC ขึ้นมาใหม่ด้วยคำสั่ง SVC() จาก sklearn.svm.SVC โดยกำหนดค่า Hyperparameter ตามผลลัพธ์ที่ได้จากการทำขั้นตอนที่ 2.2 โดยสามารถใช้คำสั่ง “best_svm = svm_cv.best_estimator_” ได้เลย ต่อมาจึงนำชุดข้อมูลฝึกให้ Model (best_svm) ที่เพิ่งสร้างเรียนรู้

3. Naïve Byes (Multinomial Naïve Byes)

3.1 แบ่งชุดข้อมูลฝึกด้วย train_test_split โดยกำหนดให้ test_size = 0.15 และ random_state = 3750 คือกำหนดให้ชุดข้อมูลทดสอบมีขนาด 15 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด ดังนั้นชุดข้อมูลฝึกจึงมีขนาดเท่ากับ 85 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด โดยการสุ่มจะสุ่มตาม random state ที่ 3750

3.2 ไม่ได้กำหนดค่าของไฮเปอร์พารามิเตอร์

3.3 ใช้คำสั่ง MultinomialNB() จาก sklearn.naive_bayes.MultinomialNB เพื่อสร้าง Model จากนั้นนำข้อมูลฝึกไปให้ Model เรียนรู้

4. K-Nearest Neighbors

4.1 แบ่งชุดข้อมูลฝึกด้วย train_test_split โดยกำหนดให้ test_size = 0.15 และ random_state = 680 คือกำหนดให้ชุดข้อมูลทดสอบมีขนาด 15 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด ดังนั้นชุดข้อมูลฝึกจึงมีขนาดเท่ากับ 85 เปอร์เซ็นต์ของชุดข้อมูลทั้งหมด โดยการสุ่มจะสุ่มตาม random state ที่ 680

4.2 ใช้ GridSearchCV โดยกำหนด Dictionary ของค่าที่จะนำไปเปรียบเทียบดังภาพที่ 5.3 และกำหนดค่า cv = 10 หรือก็คือกำหนดให้เป็น 10-fold Cross-Validation จากนั้นก็นำชุดข้อมูลฝึกให้เรียนรู้ หลังการประมวลผลพบว่า Hyperparameter ที่ดีที่สุดคือ n_neighbors = 6

```
grid = {
    'n_neighbors': [4,5,6,7,8,9]
}
```

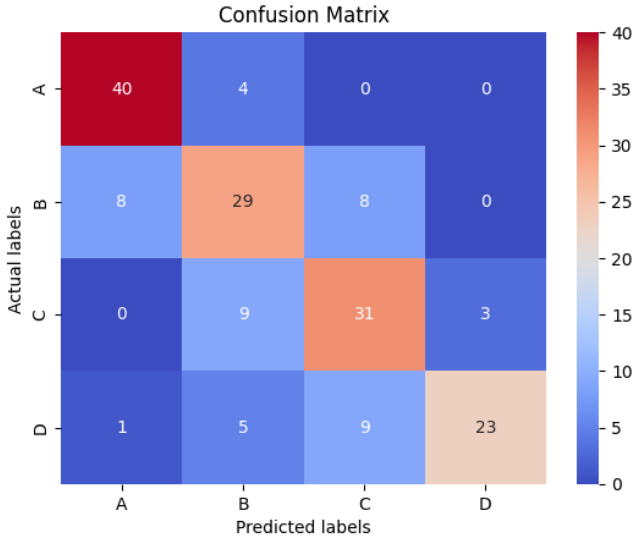
ภาพที่ 5.3 Possible Hyperparameter Dictionary (KNN)

4.3 สร้าง Instance of KNeighborsClassifier ด้วยคำสั่ง KNeighborsClassifier() จาก sklearn.neighbors.KNeighborsClassifier โดยกำหนดค่า Hyperparameter ตามผลลัพธ์ที่ได้จากการทำขั้นตอนที่ 4.2 โดยสามารถใช้คำสั่ง “best_ classifier = classifier_cv.best_estimator_” ได้เลย โดยที่ classifier_cv คือ ผลลัพธ์ที่ได้จาก GridSearchCV ต่อมาจึงนำชุดข้อมูลฝึกให้ Model (best_ classifier) ที่เพิ่งสร้างเรียนรู้

6. การวิเคราะห์ผลลัพธ์

เพื่อประเมินว่า Model มีประสิทธิภาพหรือไม่ จึงทดลองให้ Model จำแนกข้อมูลที่ไม่เคยเจอมาก่อน โดยข้อมูลที่จะนำมาทดสอบคือชุดข้อมูลทดสอบที่ได้แบ่งไว้ด้วย train_test_split หลังจากที่ Model ได้จำแนกข้อมูลแล้ว จะนำผลลัพธ์มาเปรียบเทียบกับผลเฉลยได้เป็น Confusion Matrix และนำมาวิเคราะห์ค่า Accuracy Precision Recall และ F1-score แต่ละ Model ได้ผลลัพธ์ดังนี้

6.1. Decision Tree



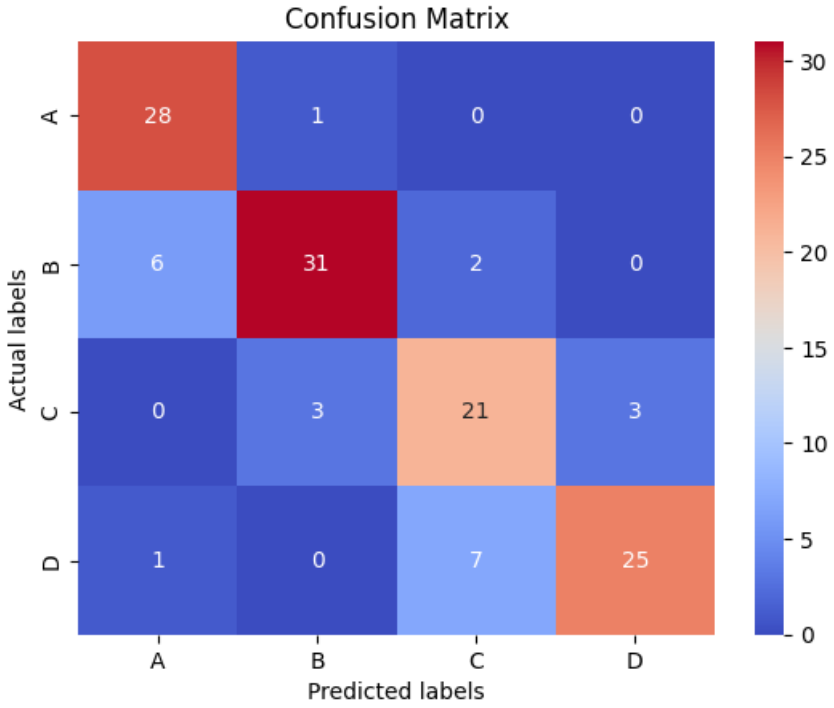
ภาพที่ 6.1 Confusion Metrix (Decision Tree)

	precision	recall	f1-score	support
A	0.8163	0.9091	0.8602	44
B	0.6170	0.6444	0.6304	45
C	0.6458	0.7209	0.6813	43
D	0.8846	0.6053	0.7188	38
accuracy			0.7235	170
macro avg	0.7409	0.7199	0.7227	170
weighted avg	0.7357	0.7235	0.7225	170

ภาพที่ 6.2 Classification Report (Decision Tree)

จากภาพที่ 6.2 จะเห็นได้ว่า Model ที่ใช้เทคนิค Decision Tree ได้ Accuracy เท่ากับ 0.7235 ค่าเฉลี่ยถ่วงน้ำหนักของ Precision เท่ากับ 0.7357 ค่าเฉลี่ยถ่วงน้ำหนักของ Recall เท่ากับ 0.7235 และค่าเฉลี่ยถ่วงน้ำหนักของ F1-Score เท่ากับ 0.7225

6.2. SVM



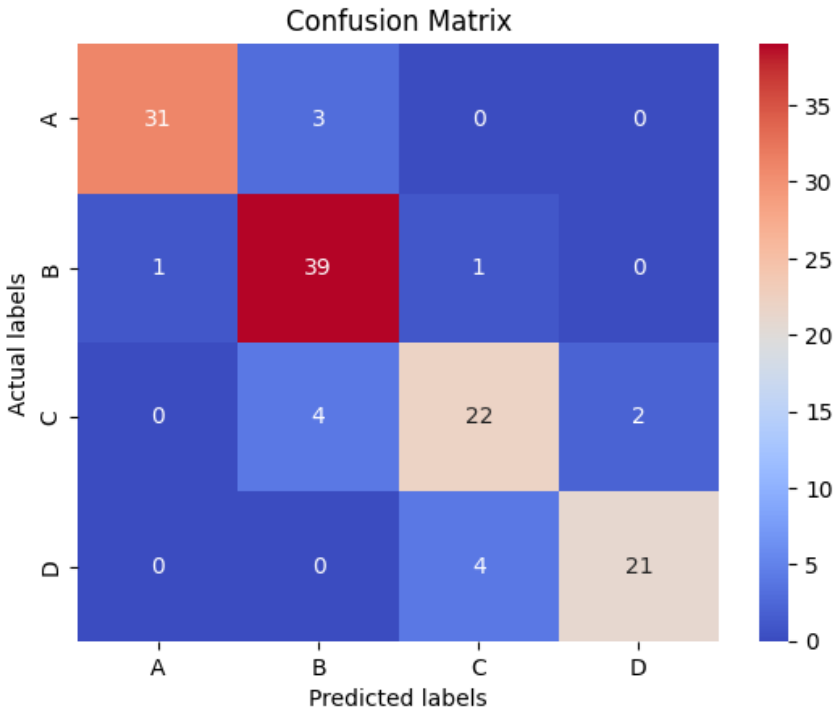
ภาพที่ 6.3 Confusion Metrix (SVM)

	precision	recall	f1-score	support
A	0.8000	0.9655	0.8750	29
B	0.8857	0.7949	0.8378	39
C	0.7000	0.7778	0.7368	27
D	0.8929	0.7576	0.8197	33
accuracy			0.8203	128
macro avg	0.8196	0.8239	0.8173	128
weighted avg	0.8290	0.8203	0.8203	128

ภาพที่ 6.4 Classification Report (SVM)

จากภาพที่ 6.4 จะเห็นได้ว่า Model ที่ใช้เทคนิค SVM มี Accuracy เท่ากับ 0.8203 ค่าเฉลี่ยถ่วงน้ำหนักของ Precision Recall และ F1-score คือ 0.8290 0.8203 และ 0.8203 ตามลำดับ

6.3. Naïve Bayes



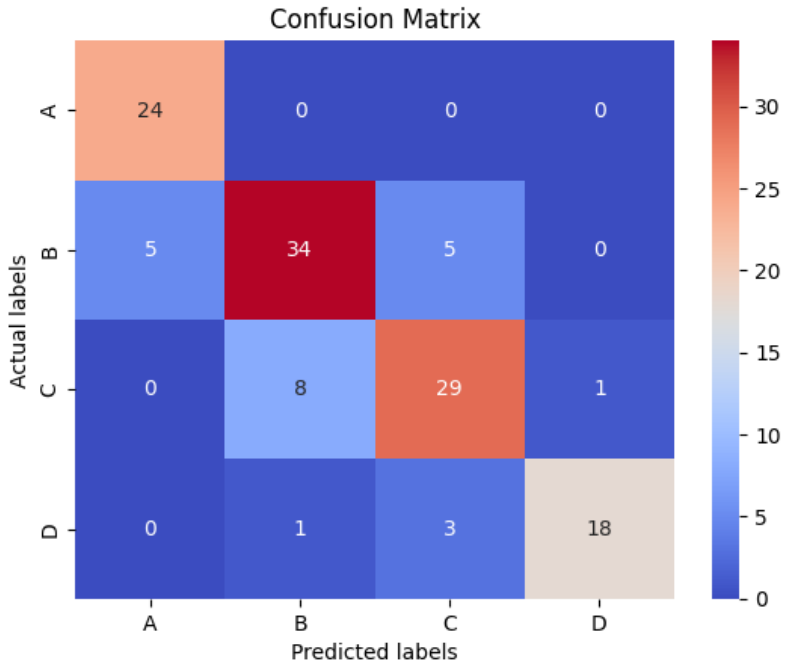
ภาพที่ 6.5 Confusion Metrix (Naïve Bayes)

	precision	recall	f1-score	support
A	0.9688	0.9118	0.9394	34
B	0.8478	0.9512	0.8966	41
C	0.8148	0.7857	0.8000	28
D	0.9130	0.8400	0.8750	25
accuracy			0.8828	128
macro avg	0.8861	0.8722	0.8777	128
weighted avg	0.8855	0.8828	0.8826	128

ภาพที่ 6.6 Classification Report (Naïve Bayes)

จากภาพที่ 6.6 จะเห็นได้ว่า Model ที่ใช้เทคนิค Naïve Bayes ได้ Accuracy มีค่าเท่ากับ 0.8828 ซึ่งถือได้ว่ามีความแม่นยำในระดับที่ดี ค่าเฉลี่ยถ่วงน้ำหนักของ Precision Recall และ F1-score คือ 0.8855 0.8828 และ 0.8826 ตามลำดับ

6.4. K-Nearest Neighbors



ภาพที่ 6.7 Confusion Metrix (K-Nearest Neighbors)

	precision	recall	f1-score	support
A	0.8276	1.0000	0.9057	24
B	0.7907	0.7727	0.7816	44
C	0.7838	0.7632	0.7733	38
D	0.9474	0.8182	0.8780	22
accuracy			0.8203	128
macro avg	0.8374	0.8385	0.8347	128
weighted avg	0.8225	0.8203	0.8190	128

ภาพที่ 6.8 Classification Report (K-Nearest Neighbors)

จากภาพที่ 6.8 จะเห็นได้ว่า Model ที่ใช้เทคนิค K-Nearest Neighbors มี Accuracy เท่ากับ 0.8203 ค่าเฉลี่ยถ่วงน้ำหนักของ Precision Recall และ F1-score คือ 0.8225 0.8203 และ 0.8190 ตามลำดับ

7. สรุปผลการทดลอง

จากการทดลองจำแนกข้อมูล Product Online ที่มีข้อมูล 848 ระเบียน ด้วยเทคนิคทั้งหมด 4 เทคนิค คือ ได้แก่ Decision Tree, Naïve Bayes, Support Vector Machine และ K-Nearest Neighbors

ค้นพบว่า Model ที่มีประสิทธิภาพในการจำแนกข้อมูลมากที่สุดคือ Model ที่สร้างด้วยเทคนิค Naïve Bayes (Multinomial Naïve Byes) ที่มีการทำ One-hot encoder ให้กับ Feature ได้ค่า Accuracy เท่ากับ 0.8828 ซึ่งสูงที่สุด

สาเหตุที่ Model ของ Multinomial Naïve Byes มีค่า Accuracy สูงที่สุด สามารถสันนิษฐานได้ว่า เนื่องจากข้อมูลที่นำมาใช้เป็น Feature ส่วนใหญ่มีลักษณะเป็น Categorical ที่ไม่สามารถวัดค่าเป็นตัวเลขได้ ทำให้เมื่อใช้เทคนิค Multinomial Naïve Byes ทำให้ได้ Model ที่มีประสิทธิภาพสูงมากกว่าเทคนิคอื่น เพราะ Multinomial Naïve Byes ใช้ประโยชน์จากการนับความถี่ของคำหรือค่าที่มีอยู่ในแต่ละคลาสมาคำนวณเพื่อจำแนกข้อมูล

ในส่วนของ Model ที่มีค่า Accuracy ลดหลั่นลงมาคือ Model ที่ใช้เทคนิค SVM, K-Nearest Neighbors และ Decision Tree มี Accuracy เท่ากับ 0.8203 0.8203 และ 0.7235 ตามลำดับ

หากพิจารณาและเปรียบกันระหว่าง SVM และ K-Nearest Neighbors ที่มี Accuracy เท่า ๆ กัน จะเห็นว่าค่าเฉลี่ย recall แบบถ่วงน้ำหนักของทั้งสอง Model มีค่าเท่ากัน แต่ค่าเฉลี่ย precision แบบถ่วงน้ำหนักของ SVM สูงกว่า แต่ในทางกลับกันค่าเฉลี่ย f1-score แบบถ่วงน้ำหนักของ K-Nearest Neighbors สูงกว่า จึงสามารถอนุมานได้ว่า SVM มีความสามารถในการแยกแยะคลาสบางคลาสได้ดีกว่า K-Nearest Neighbors แต่ K-Nearest Neighbors มีประสิทธิภาพในการทำนายคลาสทั้ง 4 คลาสดีกว่า SVM

	precision	recall	f1-score	support
A	0.9688	0.9118	0.9394	34
B	0.8478	0.9512	0.8966	41
C	0.8148	0.7857	0.8000	28
D	0.9130	0.8400	0.8750	25
accuracy			0.8828	128
macro avg	0.8861	0.8722	0.8777	128
weighted avg	0.8855	0.8828	0.8826	128

ภาพที่ 7.1 Classification Report (Naïve Bayes) 2

	precision	recall	f1-score	support
A	0.8276	1.0000	0.9057	24
B	0.7907	0.7727	0.7816	44
C	0.7838	0.7632	0.7733	38
D	0.9474	0.8182	0.8780	22
accuracy			0.8203	128
macro avg	0.8374	0.8385	0.8347	128
weighted avg	0.8225	0.8203	0.8190	128

ภาพที่ 7.2 Classification Report (K-Nearest Neighbors) 2

ในส่วน Model ที่มีประสิทธิภาพน้อยที่สุด คือ Decision Tree มีค่า Accuracy เพียง 0.7235 ซึ่งไม่เหมาะสมต่อการนำมาใช้งาน เพราะมีโอกาสที่จะจำแนกข้อมูลผิดค่อนข้างสูง