

Association Rules

Code 1 No RDD

```
1  from pyspark import SparkContext
2  from pyspark.sql import SparkSession
3  from pyspark.ml.fpm import FPGrowth
4  from pyspark.sql.functions import collect_list, array_distinct, explode, split, col
5
6  # Step 1: สร้าง SparkSession เพื่อเริ่มใช้งาน PySpark
7  spark = SparkSession.builder.appName("FPGrowthExample").getOrCreate()
8
9  # Step 2: อ่านข้อมูลจากไฟล์ CSV (สมมุติว่าเป็นข้อมูลเดินค้าจากร้านขายของชำ)
10 data = spark.read.csv("groceries_data.csv", header=True, inferSchema=True)
11
12 # Step 3: รวมกันข้อมูลตาม Member_number และรวมรวม itemDescription เป็นรายการเดียว จาก itemDescription แล้วนำมาเป็น Items
13 grouped_data = data.groupby("Member_number").agg(collect_list("itemDescription").alias("Items"))
14
15 # Step 4: แสดงข้อมูลที่ถูกจัดกลุ่ม (truncate=False เพื่อแสดงข้อมูลแบบเต็ม)
16 grouped_data.show(truncate=False)
17
18 # Step 5: เพิ่มคอลัมน์ 'basket' ที่มีรายการสินค้าที่ไม่ซ้ำกัน
19 # array_distinct() เป็นฟังก์ชันที่ใช้ใน PySpark SQL เพื่อทำให้อาร์เรย์ไม่มีค่าที่ซ้ำกัน
20 # โดยเมื่อใช้งาน array_distinct กับคอลัมน์ที่มีค่าที่มีน้อยกว่า 3 ฟังก์ชันนี้จะทำการลบค่าในอาร์เรย์ที่ซ้ำกันออก และลื้นค่าของรีบีฟ์ที่มีเพียงค่าที่ไม่ซ้ำกัน
21 # ถ้ามีข้อมูลเป็น [1, 2, 2, 3] การใช้ array_distinct จะได้ผลลัพธ์เป็น [1, 2, 3]
22 grouped_data = grouped_data.withColumn("basket", array_distinct(grouped_data["Items"]))
23
24 # Step 6: แสดงข้อมูลอีกครั้ง
25 grouped_data.show(truncate=False)
26
27 # Step 7: แยก (explode) รายการ Items เพื่อแยกรายการสินค้าไว้อยู่ในรูปแบบของแมต้า แล้วนำใหม่เป็น item
28 exploded_data = grouped_data.select("Member_number", explode("Items").alias("item"))
29
30 # Step 8: แทนที่เครื่องหมาย '/' ด้วย ',' เพื่อแยกประเภทสินค้าที่รวมกันในรายการเดียว
31 # milk\|eggs\|bread ถ้ามีและค่าเดียวกันในไฟล์ csv มันจะทำการแยกออกจากกันและ
32 # separated_data = exploded_data.withColumn("item", explode(split("item", "\|\|")))
33
34 separated_data = exploded_data.withColumn("item", explode(split("item", "/")))
35
36 # Step 9: รวมรายการสินค้ากลุ่มเข้าไปในรูปแบบของอาร์เรย์และให้มันจาวาในมีรายการที่ซ้ำกัน
37 final_data = separated_data.groupby("Member_number").agg(collect_list("item").alias("Items"))
38
39 # Step 10: ทำให้รายการในอาร์เรย์ Items ไม่ซ้ำกัน
40 final_data = final_data.withColumn("Items", array_distinct(col("Items")))
41
42 # Step 11: แสดงข้อมูลที่แยกเรียบร้อยแล้ว
43 final_data.show(truncate=False)
44
45 # Step 12: สร้างโมเดล FPGrowth โดยกำหนดค่า minSupport และ minConfidence
46 # การ prediction ขึ้นกับ minSupport และ minConfidence
47 minSupport = 0.1
48 minConfidence = 0.2
49
50 # itemsCol='Items': กำหนดคอลัมน์ที่มีรายการสินค้าที่จะใช้ในการฝึกโมเดล ในที่นี่คือคอลัมน์ Items
51 # predictionCol='prediction': กำหนดคื่อคอลัมน์ที่จะแสดงผลการท่านนาย
52 fp = FPGrowth(minSupport=minSupport, minConfidence=minConfidence, itemsCol='Items', predictionCol='prediction')
53
54 # Step 13: ฝึกโมเดล FPGrowth กับข้อมูลที่แยกเรียบร้อยแล้ว
55 model = fp.fit(final_data)
56
57 # Step 14: แสดง itemsets ที่พบบ่อยในข้อมูล
58 model.freqItemsets.show(10) # แสดง itemsets ที่พบบ่อย 10 อันดับแรก
59
60 # Step 15: กรองกฎความสัมพันธ์ (association rules) โดยให้ค่า confidence ที่มากกว่า 0.4
61 filtered_rules = model.associationRules.filter(model.associationRules.confidence > 0.4)
```

```

62
63 # Step 16: แสดงกฎความสัมพันธ์ที่ถูกกรอง
64 filtered_rules.show(truncate=False)
65
66 # Step 17: สร้าง DataFrame ใหม่เพื่อใช้ในการทำนาย (prediction)
67 new_data = spark.createDataFrame(
68     [
69         [
70             ("vegetable juice", "frozen fruits", "packaged fruit"), ,
71             ("mayonnaise", "butter", "buns"), ,
72         ],
73         ["Items"] # คอลัมน์ต้องเป็น "Items" เพื่อให้สอดคล้องกับข้อมูลที่ใช้ในโมเดล
74     ]
75 # Step 18: แสดงข้อมูลใหม่ที่จะใช้ในการทำนาย
76 new_data.show(truncate=False)
77
78 # Step 19: ใช้โมเดลในการทำนายสินค้าที่อาจจะซื้อร่วมกันกับข้อมูลใหม่
79 predictions = model.transform(new_data)
80
81 # Step 20: แสดงผลการทำนาย
82 predictions.show(truncate=False)

```

ผลลัพธ์

Member_number	Items
1000	[[soda, canned beer, sausage, sausage, whole milk, whole milk, pickled vegetables, misc. beverages, semi-finished bread, hygiene articles, yogurt, pastry, salted/sour cream, rolls/buns]]
1001	[[frankfurter, frankfurter, beef, sausage, whole milk, soda, curd, white bread, whole milk, soda, whipped/sour cream, rolls/buns]]
1002	[[tropical fruit, butter milk, butter, frozen vegetables, sugar, specialty chocolate, whole milk, other vegetables]]
1003	[[sausage, root vegetables, rolls/buns, detergent, frozen meals, rolls/buns, dental care, rolls/buns]]
1004	[[other vegetables, pip fruit, root vegetables, canned beer, rolls/buns, whole milk, other vegetables, hygiene articles, whole milk, whole milk, frozen fish, whipped/sour cream, rolls/buns, margarine, rolls/buns]]
1005	[[whole milk, frankfurter, chicken, frankfurter, whole milk, bottled water, flour, chocolate, bottled beer, rolls/buns, rice, softener, shopping bags, rolls/buns]]
1006	[[hamburger meat, tropical fruit, soda, liquor (appetizer), yogurt, liver loaf, root vegetables, yogurt, dessert, domestic eggs, white wine, photo/film]]
1007	[[herbs, pastry, tropical fruit, ketchup, yogurt, canned fish, newspapers, cocoa drinks]]
1008	[[pip fruit, frankfurter, pip fruit, bottled water, candles, bottled water, coffee, specialty bar, kitchen towels, rolls/buns, UHT-milk, sliced cheese]]
1009	[[whole milk, frankfurter, candles, citrus fruit, curd cheese, grapes, herbs, other vegetables, candy, pastry, bottled water, yogurt, rolls/buns]]
1010	[[whole milk, frankfurter, processed cheese, tropical fruit, root vegetables, yogurt, whole milk, rolls/buns, onions, frozen vegetables, shopping bags, brown bread]]
1011	[[frozen meals, white bread, meat, whole milk, tropical fruit, mustard, hard cheese, root vegetables, other vegetables, whole milk, domestic eggs, roll products]]
1012	[[whole milk, canned beer, sausage, hamburger meat, whole milk, bottled beer, cookware, yogurt, butter milk, rolls/buns]]
1013	[[citrus fruit, salty snack, fruit/vegetable juice, whole milk, beef, rolls/buns, chocolate]]
1014	[[UHT-milk, soft cheese, rolls/buns, chicken, frankfurter, bottled beer, pip fruit, oil, red/blend wine, pip fruit, mayonnaise]]
1015	[[soda, other vegetables, yogurt, other vegetables, dessert, rolls/buns, beef, root vegetables, brown bread, shopping bags, pet care]]
1016	[[butter milk, curd, berries, beverages, root vegetables, long life bakery product, cake bar, domestic eggs]]
1017	[[hamburger meat, red/blend wine]]
1018	[[canned beer, canned beer, frozen meals, spices, rolls/buns, shopping bags, butter, frozen fish, newspapers, yogurt]]

only showing top 20 rows

Member_number	Items
1000	[[soda, canned beer, sausage, sausage, whole milk, whole milk, pickled vegetables, misc. beverages, semi-finished bread, hygiene articles, yogurt, pastry, salted/sour cream, rolls/buns]]
1001	[[frankfurter, frankfurter, beef, sausage, whole milk, soda, curd, white bread, whole milk, soda, whipped/sour cream, rolls/buns]]
1002	[[tropical fruit, butter milk, butter, frozen vegetables, sugar, specialty chocolate, whole milk, other vegetables]]
1003	[[sausage, root vegetables, rolls/buns, detergent, frozen meals, rolls/buns, dental care, rolls/buns]]
1004	[[other vegetables, pip fruit, root vegetables, canned beer, rolls/buns, whole milk, other vegetables, hygiene articles, whole milk, whole milk, frozen fish, whipped/sour cream, rolls/buns, margarine, rolls/buns]]
1005	[[whole milk, frankfurter, chicken, frankfurter, whole milk, bottled water, flour, chocolate, bottled beer, rolls/buns, rice, softener, shopping bags, rolls/buns]]
1006	[[hamburger meat, tropical fruit, soda, liquor (appetizer), yogurt, liver loaf, root vegetables, yogurt, dessert, domestic eggs, white wine, photo/film]]
1007	[[herbs, pastry, tropical fruit, ketchup, yogurt, canned fish, newspapers, cocoa drinks]]
1008	[[pip fruit, frankfurter, pip fruit, bottled water, candles, bottled water, coffee, specialty bar, kitchen towels, rolls/buns, UHT-milk, sliced cheese]]
1009	[[whole milk, frankfurter, candles, citrus fruit, curd cheese, grapes, herbs, other vegetables, candy, pastry, bottled water, yogurt, rolls/buns]]
1010	[[whole milk, frankfurter, processed cheese, tropical fruit, root vegetables, yogurt, whole milk, rolls/buns, onions, frozen vegetables, shopping bags, brown bread]]
1011	[[frozen meals, white bread, meat, whole milk, tropical fruit, mustard, hard cheese, root vegetables, other vegetables, whole milk, domestic eggs, roll products]]
1012	[[whole milk, canned beer, sausage, hamburger meat, whole milk, bottled beer, cookware, yogurt, butter milk, rolls/buns]]
1013	[[citrus fruit, salty snack, fruit/vegetable juice, whole milk, beef, rolls/buns, chocolate]]
1014	[[UHT-milk, soft cheese, rolls/buns, chicken, frankfurter, bottled beer, pip fruit, oil, red/blend wine, pip fruit, mayonnaise]]
1015	[[soda, other vegetables, yogurt, other vegetables, dessert, rolls/buns, beef, root vegetables, brown bread, shopping bags, pet care]]
1016	[[butter milk, curd, berries, beverages, root vegetables, long life bakery product, cake bar, domestic eggs]]
1017	[[hamburger meat, red/blend wine]]
1018	[[canned beer, canned beer, frozen meals, spices, rolls/buns, shopping bags, butter, frozen fish, newspapers, yogurt]]

only showing top 20 rows

Member_number	Items
1000	[[soda, canned beer, sausage, whole milk, pickled vegetables, misc. beverages, semi-finished bread, hygiene articles, yogurt, pastry, salty snack]
1001	[[frankfurter, beef, sausage, whole milk, soda, curd, white bread, whipped, sour cream, rolls, buns]
1002	[[tropical fruit, butter milk, butter, frozen vegetables, sugar, specialty chocolate, whole milk, other vegetables]
1003	[[sausage, root vegetables, rolls, buns, detergent, frozen meals, dental care]
1004	[[other vegetables, pip fruit, root vegetables, canned beer, rolls, buns, whole milk, hygiene articles, frozen fish, red, blush wine, chocolate, shopping bags, whipped, sour cream, rolls, buns, margarine]
1005	[[whole milk, frankfurter, chicken, bottled water, flour, chocolate, bottled beer, rolls, buns, rice, softener, shopping bags, skin care]
1006	[[hamburger meat, tropical fruit, soda, liquor (appetizer), yogurt, liver loaf, root vegetables, dessert, domestic eggs, white wine, photo, film]
1009	[[herbs, pastry, tropical fruit, yogurt, ketchup, canned fish, newspapers, cocoa drinks]
1010	[[pip fruit, frankfurter, bottled water, candles, coffee, specialty bar, kitchen towels, rolls, buns, UHT-milk, sliced cheese]
1011	[[whole milk, frankfurter, candles, citrus fruit, curd cheese, grapes, herbs, other vegetables, candy, pastry, bottled water, yogurt, rolls, buns]
1012	[[frankfurter, processed cheese, tropical fruit, root vegetables, yogurt, whole milk, rolls, buns, onions, frozen vegetables, shopping bags, brown bread]
1013	[[frozen meals, white bread, meat, whole milk, tropical fruit, mustard, hard cheese, root vegetables, other vegetables, domestic eggs, roll products, candy, ff]
1014	[[whole milk, canned beer, sausage, hamburger meat, bottled beer, cookware, yogurt, butter milk, rolls, buns]
1015	[[citrus fruit, salty snack, fruit, vegetable juice, whole milk, beef, rolls, buns, chocolate]
1016	[[UHT-milk, soft cheese, rolls, buns, chicken, frankfurter, bottled beer, pip fruit, oil, red, blush wine, mayonnaise]
1017	[[soda, other vegetables, yogurt, dessert, rolls, buns, beef, root vegetables, brown bread, shopping bags, pet care]
1018	[[butter milk, curd, berries, beverages, root vegetables, long life bakery product, cake bar, domestic eggs]
1019	[[hamburger meat, red, blush wine]
1020	[[canned beer, frozen meals, spices, rolls, buns, shopping bags, butter, frozen fish, newspapers, yogurt]

only showing top 20 rows

items freq
[butter] 493
[bottled water] 833
[bottled water, w...] 438
[sour cream] 603
[sour cream, whip...] 603
[curd] 471
[pastry] 692
[coffee] 448
[rolls] 1363
[rolls, other veg...] 572

only showing top 10 rows

antecedent	consequent	confidence	lift	support
[[other vegetables]	[[whole milk]	0.5081743869209809	1.1091062487222754	0.1913801949717804
[[yogurt, rolls]	[[buns]	1.0	2.85986793837124	0.11133914828116984
[[vegetable juice]	[[fruit]	1.0	8.004106776180699	0.12493586454592098
[[yogurt]	[[other vegetables]	0.4252039891205802	1.129049829422358	0.12031811185223192
[[yogurt]	[[whole milk]	0.5321849501359928	1.1615100423460805	0.15059004617752694
[[sausage]	[[whole milk]	0.5193026151930261	1.1333939496206136	0.10697793740379682
[[rolls, whole milk]	[[buns]	1.0	2.85986793837124	0.17855310415597742
[[soda, rolls]	[[buns]	1.0	2.85986793837124	0.11980502821959979
[[whole milk]	[[other vegetables]	0.4176931690929451	1.1091062487222754	0.1913801949717804
[[rolls]	[[other vegetables]	0.41966250917094644	1.1143354637250336	0.14674191893278604
[[rolls]	[[whole milk]	0.5106382978723404	1.1144838102499344	0.17855310415597742
[[rolls]	[[buns]	1.0	2.85986793837124	0.3496664956387891
[[buns, other vegetables]	[[rolls]	1.0	2.85986793837124	0.14674191893278604
[[tropical fruit]	[[whole milk]	0.4983534577387486	1.0876717683458241	0.11646998460749101
[[rolls, other vegetables]]	[[buns]	1.0	2.85986793837124	0.14674191893278604
[[yogurt, buns]	[[rolls]	1.0	2.85986793837124	0.11133914828116984
[[soda, buns]	[[rolls]	1.0	2.85986793837124	0.11980502821959979
[[buns, whole milk]	[[rolls]	1.0	2.85986793837124	0.17855310415597742
[[whipped]	[[sour cream]	1.0	6.464344941956883	0.15469471523858389
[[buns, rolls]	[[other vegetables]	0.41966250917094644	1.1143354637250336	0.14674191893278604

only showing top 20 rows

Items	
[[vegetable juice, frozen fruits, packaged fruit]]	
[[mayonnaise, butter, buns]]	

Items	prediction
[[vegetable juice, frozen fruits, packaged fruit]]	[[fruit]]
[[mayonnaise, butter, buns]]	[[yogurt, soda, rolls, other vegetables, whole milk]]

คำอธิบาย

`collect_list` เป็นฟังก์ชันที่ใช้ใน PySpark SQL ซึ่งช่วยในการรวมค่าจากหลายแถว (rows) ให้อยู่ในรูปของอาร์เรย์ (list) โดยค่าที่ได้จะไม่ทำการกรองค่าที่ซ้ำกัน กล่าวคือสามารถมีค่าที่ซ้ำกันในอาร์เรย์ที่สร้างขึ้นได้

โดยทั่วไป `collect_list` นั้นจะถูกใช้ในการรวมค่าจากหลายๆ แถวที่อยู่ภายใต้การจัดกลุ่มเดียวกัน (groupBy) เพื่อให้สามารถนับรายการทั้งหมดมาเก็บไว้ในคอลัมน์เดียวในรูปแบบของลิสต์ เช่นในกรณีที่ต้องการรวมจำนวนค้าที่สำเนาซึ่กแต่ละคนซื้อเอาไว้

ตัวอย่าง: สมมติว่าคุณมี DataFrame ที่มีข้อมูลดังนี้:

Member_number	itemDescription
1	apple
1	banana
2	orange
2	banana
2	orange

code

```
grouped_data = data.groupBy("Member_number").agg(collect_list("itemDescription").alias("Items"))
```

ผลลัพธ์จะเป็น:

Member_number	Items
1	["apple", "banana"]
2	["orange", "banana", "orange"]

Code 2 Association RDD Prediction Test

```
1  from pyspark import SparkContext
2  from pyspark.sql import SparkSession
3  from pyspark.ml.fpm import FPGrowth
4  from pyspark.sql.functions import collect_list, array_distinct, explode, split, col
5
6  # ขั้นตอนที่ 1: สร้าง SparkSession
7  spark = SparkSession.builder.appName("FPGrowthExample").getOrCreate()
8
9  # ขั้นตอนที่ 2: อ่านข้อมูลจากไฟล์ CSV
10 data = spark.read.csv("groceries_data.csv", header=True, inferSchema=True)
11
12 # ขั้นตอนที่ 3: รวมกันข้อมูลตามหมายเลขสมาชิกและรูปแบบรายการเดิมๆ
13 grouped_data = data.groupBy("Member_number").agg(collect_list("itemDescription").alias("Items"))
14
15 # ขั้นตอนที่ 4: และข้อมูลที่รวมกันแล้ว
16 grouped_data.show(truncate=False)
17
18 # ขั้นตอนที่ 5: เพิ่มคอลัมน์ 'basket' ที่มีรายการสินค้าที่ไม่ซ้ำกันในแต่ละกรา
19 grouped_data = grouped_data.withColumn("basket", array_distinct(grouped_data["Items"]))
20
21 # ขั้นตอนที่ 6: แสดงข้อมูลที่อัปเดตแล้ว
22 grouped_data.show(truncate=False)
23
24 # ขั้นตอนที่ 7: สร้างโมเดล FPGrowth ด้วยพารามิเตอร์ที่กำหนด
25 minSupport = 0.1
26 minConfidence = 0.2
27 fp = FPGrowth(minSupport=minSupport, minConfidence=minConfidence, itemsCol='basket', predictionCol='prediction')
28
29 # ขั้นตอนที่ 8: ฝึกโมเดล FPGrowth
30 model = fp.fit(grouped_data)
31
32 # ขั้นตอนที่ 9: แสดง itemsets ที่พบบ่อย
33 model.freqItemsets.show(10)
34
35 # ขั้นตอนที่ 10: กรองกฎความลับเพื่อความค่า confidence
36 filtered_rules = model.associationRules.filter(model.associationRules.confidence > 0.4)
37
38 # ขั้นตอนที่ 11: แสดงกฎความลับที่กรองแล้ว
39 filtered_rules.show(truncate=False)
40
41 # ขั้นตอนที่ 12: สร้าง DataFrame ในที่มีจะใช้สำหรับการทำนาย
42 new_data_rdd = spark.sparkContext.parallelize([
43     ([['vegetable juice', 'frozen fruits', 'packaged fruit'],),
44     ([['mayonnaise', 'butter', 'buns'],),
45 ])
46
47 # ขั้นตอนที่ 13: สร้าง DataFrame จาก RDD
48 new_data = spark.createDataFrame(new_data_rdd, ["basket"])
49
50 # ขั้นตอนที่ 14: แสดงข้อมูลใหม่สำหรับการทำนาย
51 new_data.show(truncate=False)
52
53 # ขั้นตอนที่ 15: ใช้โมเดลในการทำนาย
54 predictions = model.transform(new_data)
55
56 # ขั้นตอนที่ 16: แสดงผลการทำนาย
57 predictions.show(truncate=False)
58
59 # ปิด SparkSession
60 spark.stop()
```

ผลลัพธ์แตกต่างกัน ตรง prediction

```
+-----+  
|basket |  
+-----+  
|[vegetable juice, frozen fruits, packaged fruit]|  
|[mayonnaise, butter, buns]|  
+-----+  
  
+-----+  
|basket |prediction |  
+-----+  
|[vegetable juice, frozen fruits, packaged fruit]|[]|  
|[mayonnaise, butter, buns] |[other vegetables, whole milk]|  
+-----+
```

ใจทาย

- Use the groceries_data.csv file as a dataset
- Assume we want to predict the item that would be purchased if the basket contains: (สมมติว่าเราต้องการท่านาสินค้าที่จะซื้อ หากตะกร้ามีรายการดังต่อไปนี้)
 - [vegetable juice, frozen fruits, packaged fruit]
 - [mayonnaise, butter, buns]
- Import Libraries:
 - `SparkContext` and `SparkSession`
 - `FPGrowth` from `pyspark.ml.fpm`
 - Create `SparkSession`
 - Read data
 - Group data based on member number (จัดกลุ่มข้อมูลตามหมายเลขสมาชิก)
 - Show data
 - Add a column basket where it contains a list of items (เพิ่มคอลัมน์ "basket" ที่มีรายการสินค้าเป็นรายการ)
 - Show data
 - Create `FPGrowth`
 - `fp = FPGrowth(minSupport = <value>, minConfidence = <value>, itemsCol = 'basket', predictionCol = <column name>)`
 - Fit the created FPGrowth into a model
 - Show frequencies using `<model>.freqItemsets.show(<value>)`
 - Filter the rules
 - `<model>.associationRules.filter(<model>.associationRules.confidence > <value>)`
 - Show data
 - Create a dataframe `<dataframe> = ['basket']`
 - Add new data to be used for the predictions (เพิ่มข้อมูลใหม่เข้าห้ามพยากรณ์)
 - Create `parallelize` RDD (สร้าง RDD โดยการ `parallelise`)
- Use the created `Dataframe` from `RDD` (ใช้ `DataFrame` ที่สร้างขึ้นจาก `RDD`)
- Show data (แสดงข้อมูล)
- Transform the model using the `Dataframe` of new data (แปลงโมเดลโดยใช้ `DataFrame` ของข้อมูลใหม่)

groceries_data.csv

A	B	C	D	E	F	G
Member_number	Date	itemDescr	year	month	day	day_of_week
1808	7/21/2015	tropical fru	2015	7	21	1
2552	5/1/2015	whole milk	2015	5	1	4
2300	9/19/2015	pip fruit	2015	9	19	5
1187	12/12/2015	other veget	2015	12	12	5
3037	1/2/2015	whole milk	2015	1	2	4
4941	2/14/2015	rolls/buns	2015	2	14	5
4501	8/5/2015	other veget	2015	8	5	2
3803	12/23/2015	pot plants	2015	12	23	2
2762	3/20/2015	whole milk	2015	3	20	4
4119	12/2/2015	tropical fru	2015	12	2	2

RecommendationSystem

```
1  from pyspark.sql import SparkSession
2  from pyspark.ml.recommendation import ALS
3  from pyspark.ml.evaluation import RegressionEvaluator
4  from pyspark.sql.functions import col
5
6  # เริ่มต้นสร้าง SparkSession ซึ่งเป็นตัวจัดการสภาพแวดล้อม Spark
7  spark = SparkSession.builder \
8      .appName("BookRecommendationALS") \
9      .getOrCreate()
10
11 # อ่านข้อมูลจากไฟล์ CSV ซึ่งเป็นข้อมูลการให้คะแนนหนังสือ
12 # กำหนดให้มีการอ่านและแยกเป็น header และให้ Spark ท่านาย schema ของคอลัมน์โดยอัตโนมัติ
13 data = spark.read.csv('book_ratings.csv', header=True, inferSchema=True)
14
15 # แสดง schema ของข้อมูลเพื่อให้เห็นโครงสร้างของข้อมูล
16 data.printSchema()
17
18 # แสดงตัวอย่างข้อมูล 5 แถวแรก
19 data.show(5)

20
21 # กำหนดค่า ALS (Alternating Least Squares) model สำหรับการแนะนำ
22 als = ALS(
23     maxIter=10, # จำนวน iteration ที่จะทำซ้ำ (ยิ่งมากยิ่ง慢 แต่ใช้เวลานานขึ้น)
24     userCol="user_id", # คอลัมน์สำหรับ user ID
25     itemCol="book_id", # คอลัมน์สำหรับ book ID
26     ratingCol="rating", # คอลัมน์ที่เก็บค่าการให้คะแนน
27     coldStartStrategy="drop" # จัดการข้อมูลที่มีการท่านายเป็น NaN โดยลบข้อมูลนั้นทิ้ง
28 )
29
30 # ฝึกโมเดล ALS โดยใช้ข้อมูลที่เราอ่านเข้ามา
31 model = als.fit(data)
32
33 # ท่านายการให้คะแนน โดยใช้โมเดลที่ฝึกมาแล้ว
34 predictions = model.transform(data)

35
36 # กำหนดตัวประเมินผลแบบ RegressionEvaluator เพื่อตรวจสอบคุณภาพของโมเดล
37 # ในที่นี่เราจะใช้ Root Mean Squared Error (RMSE)
38 evaluator = RegressionEvaluator(
39     metricName="rmse", # ตัวชี้วัดเป็น RMSE
40     labelCol="rating", # คอลัมน์ที่ใช้เป็นค่าจริง
41     predictionCol="prediction" # คอลัมน์ที่ใช้เป็นค่าท่านายจากโมเดล
42 )
43
44 # ประเมินโมเดลโดยค่านวณค่า RMSE จากการท่านาย
45 rmse = evaluator.evaluate(predictions)
46 print(f"Root Mean Squared Error (RMSE): {rmse}")
47
```

```

48 # แสดงผลการท่านายเดาของผู้ใช้ที่มี user_id เม้ากับ 53
49 user_id = 53 # user_ids = [53, 54, 55] มากกว่า 2 คน
50 user_predictions = predictions.filter(col("user_id") == user_id)
51 user_predictions = user_predictions.select("book_id", "user_id", "rating", "prediction").orderBy(col("prediction").desc())
52 # desc (descending order) มากไปน้อย - asc() ascending order น้อยไปมาก
53
54 user_predictions.show(truncate=False) # แสดงผลการท่านายเดาทั้งหมดโดยไม่ตัดตอน
55
56 # แสดง 5 หนังสือที่ไม่เคยแนะนำให้สำหรับผู้ใช้ทั้งหมด recommendForAllUsers(5) เดิมแล้วแต่จำนวนค่าหนทางได้เลือก
57 user_recommendations = model.recommendForAllUsers(5)
58 user_recommendations.show(truncate=False)
59
60 # แสดงผู้ใช้ที่แนะนำ 5 รายการที่ไม่เคยแนะนำ recommendForAllItems(5) เดิมแล้วแต่จำนวนค่าหนทางได้เลือก
61 item_recommendations = model.recommendForAllItems(5)
62 item_recommendations.show(truncate=False)
63
64 # หยุดการทำงานของ SparkSession เมื่อเสร็จสิ้น
65 spark.stop()

```

ผลลัพธ์

```

root
|-- book_id: integer (nullable = true)
|-- user_id: integer (nullable = true)
|-- rating: integer (nullable = true)

+-----+-----+-----+
|book_id|user_id|rating|
+-----+-----+-----+
|      1|     314|      5|
|      1|     439|      3|
|      1|     588|      5|
|      1|    1169|      4|
|      1|    1185|      4|
+-----+-----+-----+
only showing top 5 rows

Root Mean Squared Error (RMSE): 0.596380692581242
+-----+-----+-----+
|book_id|user_id|rating|prediction|
+-----+-----+-----+
| 8946 |    53 |    5 | 4.3467135 |
| 8882 |    53 |    2 | 2.1000404 |
| 8336 |    53 |    1 | 1.180042   |
| 8336 |    53 |    1 | 1.180042   |
+-----+-----+-----+

```

```

+-----+
|user_id|recommendations
+-----+
| 1 | [[{9531, 4.3655496}, {7844, 4.2592916}, {6024, 4.2444234}, {8013, 4.1680713}, {2205, 4.1580925}]]|
| 3 | [[{9076, 1.086818}, {6361, 1.075947}, {7844, 1.0735128}, {7254, 1.0726048}, {5207, 1.0706855}]]|
| 5 | [[{9842, 5.1694894}, {8926, 5.0142293}, {8976, 5.0042834}, {9141, 4.9983478}, {862, 4.960466}]]|
| 6 | [[{4344, 5.229495}, {7401, 5.140155}, {3753, 5.0921793}, {8548, 5.086315}, {3746, 5.0570846}]]|
| 9 | [[{4692, 4.16097}, {6971, 4.154922}, {9537, 4.04411}, {2985, 4.030314}, {7275, 3.994522}]]|
| 12 | [[{3628, 4.663629}, {1788, 4.654097}, {7401, 4.65241}, {5580, 4.576388}, {4483, 4.542922}]]|
| 13 | [[{4868, 5.0200577}, {1198, 5.001553}, {4154, 4.992331}, {7844, 4.8768063}, {6862, 4.696245}]]|
| 15 | [[{5841, 3.8900213}, {6820, 3.853875}, {8867, 3.8517556}, {3321, 3.8414254}, {8547, 3.8397245}]]|
| 16 | [[{4868, 4.9641376}, {1338, 4.881822}, {3836, 4.8664308}, {8353, 4.818584}, {9071, 4.7782474}]]|
| 17 | [[{7063, 4.904595}, {6382, 4.360248}, {8010, 4.2999086}, {3321, 4.2730255}, {5493, 4.264379}]]|
| 19 | [[{4, 4.2215424}, {8721, 4.1751547}, {7756, 4.1706543}, {1265, 4.152372}, {3628, 4.1456366}]]|
| 20 | [[{6902, 4.980991}, {4653, 4.964336}, {5028, 4.9187403}, {3628, 4.8773675}, {1788, 4.857861}]]|
| 22 | [[{9842, 4.4760876}, {8976, 4.443118}, {7305, 4.38762}, {8492, 4.370014}, {862, 4.3528705}]]|
| 26 | [[{6018, 3.9987524}, {8818, 3.9166658}, {7730, 3.8728147}, {9024, 3.85968}, {4620, 3.8422337}]]|
| 27 | [[{3953, 4.945734}, {6438, 4.92334}, {4441, 4.919877}, {7593, 4.9022017}, {6634, 4.8981915}]]|
| 28 | [[{3952, 4.6575017}, {4706, 4.5346723}, {1851, 4.4491034}, {4609, 4.4432235}, {4638, 4.4397364}]]|
| 31 | [[{6018, 4.049713}, {9864, 4.0290933}, {6932, 4.0149393}, {3692, 4.004732}, {8867, 3.9683171}]]|
| 34 | [[{4638, 3.3723059}, {8663, 3.3403244}, {7440, 3.330962}, {3628, 3.3237817}, {1788, 3.3152974}]]|
| 35 | [[{2636, 3.726225}, {3628, 3.7233858}, {7305, 3.717098}, {7401, 3.711492}, {8976, 3.6979399}]]|
| 37 | [[{5841, 5.2393107}, {1198, 5.1143513}, {2655, 5.0589375}, {7844, 4.9767284}, {5701, 4.9718823}]]|
+-----+
only showing top 20 rows

```

book_id	recommendations
1	[{50580, 5.9794426}, {43980, 5.9764295}, {4421, 5.9729986}, {27733, 5.970216}, {49517, 5.9659905}]
3	[{11472, 5.3291383}, {37765, 5.309071}, {44737, 5.211686}, {31900, 5.172024}, {3054, 5.1636953}]
5	[{51456, 5.617667}, {24552, 5.57281}, {30757, 5.5657144}, {41541, 5.5230856}, {44141, 5.4676394}]
6	[{22551, 5.6522946}, {4421, 5.618076}, {38372, 5.5826783}, {50580, 5.566371}, {20992, 5.565568}]
9	[{44737, 5.4499164}, {26587, 5.2166567}, {41085, 5.1338196}, {14766, 5.1284833}, {27401, 5.1272984}]
12	[{30126, 5.427142}, {504, 5.423197}, {23317, 5.3638735}, {4421, 5.3499403}, {22551, 5.3492227}]
13	[{19217, 5.717334}, {7850, 5.689824}, {52395, 5.6405673}, {25392, 5.6225643}, {25515, 5.613694}]
15	[{50580, 5.855026}, {24552, 5.719567}, {50234, 5.6873603}, {23176, 5.633416}, {51456, 5.5878534}]
16	[{27401, 5.6038275}, {25515, 5.556542}, {4134, 5.5059657}, {11193, 5.5033383}, {23487, 5.501045}]
17	[{8963, 5.9579115}, {25515, 5.806971}, {38263, 5.728747}, {31900, 5.7157464}, {4421, 5.7135673}]
19	[{49517, 5.8503203}, {24364, 5.721731}, {50769, 5.6944056}, {19474, 5.6625943}, {22306, 5.65011}]
20	[{50307, 5.412123}, {4421, 5.3819594}, {49517, 5.343719}, {8963, 5.3407283}, {20992, 5.3361034}]
22	[{38884, 5.246563}, {50580, 5.126454}, {49517, 5.101265}, {18322, 5.0658884}, {33750, 5.038417}]
26	[{51190, 5.2155757}, {504, 5.175779}, {47140, 5.154698}, {44737, 5.13567}, {28956, 5.1309476}]
27	[{38076, 6.52254}, {21720, 6.471094}, {21791, 6.261124}, {33340, 6.0898404}, {38263, 6.0294805}]
28	[{19657, 5.5933}, {47531, 5.5861287}, {25392, 5.58459}, {24552, 5.5584073}, {50234, 5.491508}]
31	[{30757, 5.903651}, {43980, 5.876709}, {43330, 5.857453}, {4134, 5.8144555}, {50234, 5.806565}]
34	[{21212, 5.0926576}, {30757, 5.0434575}, {11983, 5.0254045}, {29054, 5.0194874}, {6975, 4.9492016}]
35	[{30776, 5.5658393}, {21720, 5.3738475}, {38884, 5.3611956}, {35316, 5.2394147}, {53397, 5.1837664}]
37	[{50580, 6.033258}, {49517, 5.856255}, {38948, 5.731876}, {50234, 5.696304}, {43980, 5.666799}]

เพิ่มเติม

ถ้าข้อมูลมันเป็นแบบตัวอักษร rating เช่น A = ดีมาก , F = แย่มาก ๆ

เพิ่ง from pyspark.sql.functions import when

```
from pyspark.sql.functions import when

# แปลงค่าการให้คะแนนที่เป็นตัวอักษรเป็นตัวเลข
data = data.withColumn(
    "rating",
    when(col("rating") == "A", 5) # แปลง A เป็น 5 (ดีที่สุด)
    .when(col("rating") == "B", 4) # แปลง B เป็น 4
    .when(col("rating") == "C", 3) # แปลง C เป็น 3
    .when(col("rating") == "D", 2) # แปลง D เป็น 2
    .when(col("rating") == "F", 1) # แปลง F เป็น 1 (แย่ที่สุด)
    .otherwise(None) # กรณีที่ค่าไม่ตรงกับตัวอักษรที่กำหนด ให้กำหนดเป็น None
)
```

ถ้าหากต้องการเพิ่มตรงนี้ ถ้าข้อมูลนั้นเป็นแบบตัวอักษร rating เช่น A = ดีมาก , F = แย่มาก ๆ

```
RecommendationSystem > RecommendationSystem.py > ...
1  from pyspark.sql import SparkSession
2  from pyspark.ml.recommendation import ALS
3  from pyspark.ml.evaluation import RegressionEvaluator
4  from pyspark.sql.functions import col
5
6  # เริ่มต้นสร้าง SparkSession ซึ่งเป็นตัวจัดการสภาพแวดล้อม Spark
7  spark = SparkSession.builder \
8      .appName("BookRecommendationALS") \
9      .getOrCreate()
10
11 # อ่านข้อมูลจากไฟล์ CSV ซึ่งมีน้อยโมลการให้ค่าแทนหนังสือ
12 # กำหนดให้มีการอ่านแต่แรกเป็น header และให้ Spark ท่านาย schema ของคอลัมน์โดยอัตโนมัติ
13 data = spark.read.csv('book_ratings.csv', header=True, inferSchema=True)
14
15 # แสดง schema ของข้อมูลเพื่อให้เห็นโครงสร้างของข้อมูล
16 data.printSchema()
17
18
19
20
21
22
23
24 # แสดงตัวอย่างข้อมูล 5 แถวแรก
25 data.show(5)
26
27 # กำหนดค่า ALS (Alternating Least Squares) model สำหรับการแนะนำ
28 als = ALS(
29     maxIter=10, # จำนวน iteration ที่จะทำข้า (ยิ่งมากยิ่ง慢 แต่ให้ผลลัพธ์ดี)
30     userCol="user_id", # คอลัมน์สำหรับ user ID
31     itemCol="book_id", # คอลัมน์สำหรับ book ID
32     ratingCol="rating", # คอลัมน์ที่เก็บค่าการให้ค่าแทน
33     coldStartStrategy="drop" # จัดการข้อมูลที่ไม่มีการทำนายเป็น NaN โดยลบข้อมูลนั้นทิ้ง
34 )
```

| พูด |

โจทย์ (ไทย)

Import Libraries: ● `SparkSession` ● `RegressionEvaluator` ● `ALS` (จาก `pyspark.ml.recommendation`)

ในการใช้ ALS: ● กำหนด `maxIter` เป็นจำนวนการวิ่งซ้ำสูงสุด ● กำหนด `userCol` โดยระบุคอลัมน์ที่ใช้แทนผู้ใช้ ● กำหนด `itemCol` โดยระบุคอลัมน์ที่ใช้แทนรายการ ● กำหนด `ratingCol` โดยระบุคอลัมน์ที่ใช้แทนการให้ค่าแทน ● `coldStartStrategy = "drop"` ใช้สำหรับการตัดค่า `Nan` ออก

การประเมินผล: ● กำหนด `metricName` เป็น `rmse` ● กำหนด `labelCol` โดยระบุคอลัมน์ที่ใช้แทนข้อความที่ต้องการคำนวณ ● กำหนด `predictionCol` โดยระบุคอลัมน์ที่ใช้แสดงผลการท่านาย ● ประเมินผลโดยใช้ฟังก์ชัน `evaluate()`

การแสดงผู้ใช้ด้วย ID เพียง: ● ใช้ฟังก์ชัน `filter()` ○ เช่น `<yourDF>.filter(<yourDF['<column>'] == <value>)`

● แสดงผลลัพธ์เพื่อตรวจสอบผลการกรอง

การแสดงการท่านายของผู้ใช้: ● ใช้ฟังก์ชัน `transform()` โดยใช้ผลการกรองเป็นอินพุต ● แสดงผลลัพธ์ ○ `พารามิเตอร์ order_by()` สามารถใช้เพื่อจัดเรียงผลลัพธ์ตามคอลัมน์ที่กำหนด

การแสดงค่าแนะนำสำหรับผู้ใช้ทั้งหมด: ● ใช้ `recommendForAllUsers(<number of recommendations>).show()` ● หากต้องการแสดงผลเต็ม ใช้ `truncate=False`

การแสดงค่าแนะนำสำหรับรายการทั้งหมด: ● ใช้ `recommendForAllItems(<number of recommendations>).show()` ● หากต้องการแสดงผลเต็ม ใช้ `truncate=False`

ໄຈທຍໍ່ (Eng version)

```
Import Libraries:  
● SparkSession  
● RegressionEvaluator  
● ALS (from pyspark.ml.recommendation)  
To use ALS:  
● Define maxIter by the number of maximum iteration  
● Define userCol by assigning the column to be used for users  
● Define itemCol by assigning the column to be used for items  
● Define ratingCol by assigning the column to be used for ratings  
● coldStartStrategy = “drop” can be used for dropping Nan values  
To evaluate:  
● Define metricName as rmse  
● Define labelCol by assigning the column to be used for the label  
● Define predictionCol by assigning the column to be shown as a prediction  
● Evaluate using the function evaluate()  
To show a user with a specific ID:  
● The function filter() can be used.  
    ○ For example, <your DF>.filter(<yourDF[‘<column>’] == <value>)
```

- Show the result to check if the filter result is correct.
- To show the prediction of the user:
 - Transform the model by using the filtered result as an input
 - Show the result
 - The orderBy() function can be used for sorting the result based on the defined column
- To show the recommendation for all users:
 - Use recommendForAllUsers(<number of recommendations>).show()
 - To show full output, use truncate = False
 - Use recommendForAllItems(<number of recommendations>).show()
 - To show full output, use truncate = False

book_ratings.csv

book_id	user_id	rating
1	314	5
1	439	3
1	588	5
1	1169	4
1	1185	4
1	2077	4
1	2487	4
1	2900	5
1	3662	4
1	3922	5

Text Analytics

```
1  from pyspark.sql import SparkSession # ใช้สำหรับสร้าง Spark session เนื่องจากงานกับ Spark SQL
2  from pyspark.sql.types import IntegerType # ใช้สำหรับแบบชี้อ้อมูลให้มีชนิด Integer
3  from pyspark.sql.functions import trim, col # ใช้ในการตัดข้อความและอักขระส่วนที่ไม่ต้องการ
4  from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover # เครื่องมือสำหรับการสร้างฟีเจอร์
5  from pyspark.ml import Pipeline # ใช้สร้างรุ่นสร้าง Pipeline สำหรับขั้นตอนการประมวลผล
6  from pyspark.ml.classification import LogisticRegression # โมเดล Logistic Regression สำหรับการจำแนกประเภท
7  from pyspark.ml.evaluation import MulticlassClassificationEvaluator # ตัวประเมินผลสำหรับค่าความแม่นยำของโมเดล
8
9  spark = SparkSession.builder \
10    .appName("TextClassification") \
11    .getOrCreate()
12
13 # อ่านข้อมูลจากไฟล์ CSV
14 file_path = "reviews_rated.csv" # กำหนดค่าพารามิเตอร์ CSV
15 data = spark.read.csv(file_path, header=True, inferSchema=True) # อ่านไฟล์ CSV พร้อมทั้งกำหนดว่ามีส่วน header และให้เดาชนิดของข้อมูล
16
17 # เลือกและประมวลผลคอลัมน์ Review Text และ Rating
18 data = data.select(
19     trim(col("Review Text")).alias("ReviewText"), # ตัดช่องว่างจากคอลัมน์ "Review Text" และเปลี่ยนชื่อเป็น "ReviewText"
20     col("Rating").cast(IntegerType()).alias("Rating") # แปลงคอลัมน์ "Rating" เป็นชนิด Integer และเปลี่ยนชื่อเป็น "Rating"
21 )
22
23 # trim() ใน PySpark ใช้สำหรับลบช่องว่าง (space) ที่อยู่ข้างหน้าและข้างหลังของข้อความในแต่ละบรรทัดของคอลัมน์
24
```



```
25 | # กรองออกข้อมูลที่ ReviewText หรือ Rating เป็น null หรือว่างเปล่า
26 | data = data.filter(
27 |     (col("ReviewText").isNotNull() & (col("ReviewText") != "") &
28 |      (col("Rating").isNotNull() & (col("Rating") != float('nan'))))
29 | )
30
31 # แสดงข้อมูล
32 data.show(truncate=False) # แสดงข้อมูลที่ประมวลผลแล้วโดยไม่ตัดตอนเนื้อหา
33
34 # Tokenizer
35 tokenizer = Tokenizer(inputCol="ReviewText", outputCol="ReviewTextWords") # แยกข้อความรีวิวเป็นคำ ๆ
36
37 # StopWordsRemover
38 stop_word_remover = StopWordsRemover(
39     inputCol=tokenizer.getOutputCol(), # ใช้ข้อความที่ผ่านการแยกคำเป็นข้อมูลนำเข้า
40     outputCol="MeaningfulWords" # แสดงผลเป็นคำที่มีความหมาย (ลบคำที่เป็น stopwords ออก)
41 )
42
43 # HashingTF
44 hashing_tf = HashingTF(
45     inputCol=stop_word_remover.getOutputCol(), # ใช้คำที่มีความหมายเป็นข้อมูลนำเข้า
46     outputCol="features" # แสดงผลเป็นภาษาเตอร์ฟีเจอร์
47 )
48
```



```
49 # Pipeline
50 pipeline = Pipeline(stages=[tokenizer, stop_word_remover, hashing_tf]) # กำหนดลำดับขั้นตอนการประมวลผล
51
52 # แบ่งข้อมูลเป็นชุดฝึก (train) และชุดทดสอบ (test)
53 train_data, test_data = data.randomSplit([0.8, 0.2], seed=42) # แบ่งข้อมูลเป็น 80% สำหรับฝึก และ 20% สำหรับทดสอบ
54
55 # แสดงข้อมูลชุดฝึก
56 train_data.show(truncate=False) # แสดงข้อมูลชุดฝึกโดยไม่ตัดเนื้อหา
57
58 # ฝึก Pipeline ด้วยข้อมูลชุดฝึก
59 pipeline_model = pipeline.fit(train_data) # ฝึก Pipeline ด้วยข้อมูลชุดฝึก
60
```

```

61 # แปลงข้อมูลชุดเดียวกันและชุดทดสอบ
62 train_df = pipeline_model.transform(train_data) # แปลงข้อมูลชุดเดียวกับ Pipeline ที่ผ่านการฝึกแล้ว
63 test_df = pipeline_model.transform(test_data) # แปลงข้อมูลชุดทดสอบด้วย Pipeline ที่ผ่านการฝึกแล้ว
64
65 # และดึงข้อมูลชุดฝึกที่แปลงแล้ว
66 train_df.show(truncate=False) # และดึงข้อมูลชุดฝึกที่ผ่านการแปลงแล้ว
67
68 # Logistic Regression
69 lr = LogisticRegression(labelCol="Rating", featuresCol="features") # สร้างโมเดล Logistic Regression
70
71 # ฝึกโมเดลด้วย DataFrame นี้ด้วย
72 lr_model = lr.fit(train_df) # ฝึกโมเดล Logistic Regression ด้วยข้อมูลชุดฝึก
73
74 # แปลงข้อมูลชุดทดสอบโดยใช้โมเดลที่ฝึกแล้ว
75 predictions = lr_model.transform(test_df) # ทำการทำนายชุดทดสอบ
76
77
78 # แสดง MeaningfulWords, Rating (label) และ Prediction
79 predictions.select("MeaningfulWords", "Rating", "prediction").show(truncate=False) # และดึงค่าที่มีความหมาย, ค่าจริงของเรลติ๊ฟ, และค่าที่หัวน้ำอย่าง
80
81 # สร้างตัววัดคุณภาพ MulticlassClassificationEvaluator
82 evaluator = MulticlassClassificationEvaluator(
83     labelCol="Rating", # ระบุคอลัมน์ของ label
84     predictionCol="prediction", # ระบุคอลัมน์ของผลการทำนาย
85     metricName="accuracy" # กำหนดค่าที่ต้องการวัดเป็นค่าความแม่นยำ (accuracy)
86 )
87
88 # ประเมินค่าความแม่นยำของโมเดล
89 accuracy = evaluator.evaluate(predictions) # คำนวณค่าความแม่นยำของโมเดลบนชุดทดสอบ
90
91 print(' ')
92 print('*'*80)
93 print(' ')
94 print(f"Model Accuracy: {accuracy:.4f}") # แสดงค่าความแม่นยำ
95 print(' ')
96 print('*'*80)
97 print(' ')

```

คำอธิบายเพิ่มเติม

`trim()` ใน PySpark ใช้สำหรับลบช่องว่าง (space) ที่อยู่ข้างหน้าและข้างหลังของชือความในแต่ละเคานของคอลัมน์ โดยเฉพาะอย่างยิ่งเมื่อมีข้อมูลมีการเว้นช่องว่างที่ไม่จำเป็น ซึ่งอาจเกิดจากการป้อนข้อมูลโดยผู้ใช้หรือกระบวนการอื่นๆ ทำให้ข้อมูลไม่สะอาดและอาจมีผลต่อการประมวลผลต่อไปได้

```

# เลือกและประมวลผลคอลัมน์ Review Text และ Rating
data = data.select(
    trim(col("Review Text")).alias("ReviewText"), # 
    col("Rating").cast(IntegerType()).alias("Rating")
)

```

ตัวอย่าง -----→

ก่อนใช้ trim:

FruitName	Quantity
Apple	10
Banana	20
Orange	30

หลังใช้ trim:

CleanFruitName	Quantity
Apple	10
Banana	20
Orange	30

```
# กรองออกข้อมูลที่ ReviewText หรือ Rating เป็น null หรือว่างเปล่า
data = data.filter(
    (col("ReviewText").isNotNull() & (col("ReviewText") != "") &
    (col("Rating").isNotNull()) & (col("Rating") != float('nan')))
)
```

คำสั่ง `data.filter(...)` ใช้เพื่อกรองข้อมูลใน `DataFrame` โดยเลือกเฉพาะแถวที่มีค่าที่ต้องการเท่านั้น (เป็นการทำความสะอาดข้อมูล) เพื่อให้แน่ใจว่าไม่มีค่าใดที่เป็น `null` หรือค่าว่างเปล่าในคอลัมน์ที่เราจะใช้ในการประมวลผลต่อไป

สมมติว่าคุณมี `DataFrame` `data` ที่มีข้อมูลดังนี้:

ReviewText	Rating
"Good product"	5
<code>null</code>	4
---	3
"Bad quality"	<code>null</code>
"Not satisfied"	NaN
"Excellent!"	5

ผลลัพธ์ที่ได้หลังจากใช้ `filter` จะเป็น:

ReviewText	Rating
"Good product"	5
"Excellent!"	5

ผลลัพธ์

```
+-----+
|ReviewText
+-----+
|I registered on the website, tried to order a laptop, entered all the details, but instead of charging me and
|Had multiple orders one turned up and driver had to phone as no door number on packaging, then waited all day
|I informed these reprobates that I WOULD NOT BE IN as I was going to visit a sick relative, they told me they
|I have bought from Amazon before and no problems being very happy with the service and price. Amazon advertis
|If I could give a lower rate I would! I cancelled my Amazon Prime in February and subsequently found that the
|Terrible you get customer service reps that are clearly home, you hear children and family scenarios in the b
|Amazon has a way of tainting a great product due to their inability to separate used goods from the new. I re
|I love amazon! I use it for half my shopping. The prime membership is worth it as you can receive free 2-day
|I applied for a job with Amazon. I completed all the steps (including sending confidential data and personal
|I had a great experience with their customer service. They delivered my order to wrong address but didnt give
|Every time there is a problem, they fix it. I have no idea why Amazon is so poorly rated here, because they p
|I have no interest in signing up to Amazon Prime, When making a purchase, despite clicking the button not to
|Bought a Pitboss kc grill combo and received one that was previously opened and was damaged, tried to file fo
|"Sadly, Amazon no longer provides the quality customer service they once did. I've had several orders never d
|Extremely disappointed in the customer service provided by the driver in van LC21 XBN at approximately 1.30pm
|My bank card froze only with amazon so couldn't order anything.Package said delivered at the post office when
|I put onw star because I can't leave a zero ...I ordered an item for £59.99... it arrived on time but wasn't s
|Once again my delivery is late with a message saying "we're so sorry".I'm so tired of this company lying abou
|Amazon is the GARBAGE company ! When it comes to refund ANAZIN WILL STEAL YOUR MONEY & GIVE YOU B.S They will
|I bought running boards for my Jeep and they sent me ones that have already been used so when I called them i
+-----+
only showing top 20 rows
```

```

+-----+
|ReviewText
+-----+
""" you'r having a laugh!!
"""Amazon does not honour its advertised Labor Day sale price in this instance"" - According to Ven
"""Arrive tonight by 10pm" is a myth. No matter what the website says or the countdown timer says - just forget about receiving it the s
"""Blue murder"" shirts and merchandise. --"Amazon does not allow products that promote
"""Extremely frustrated with amazon. Ordered a crucial product with one-day delivery
"""I have a foreign name
"""Parcel handed to resident""""This parcel was handed to resident""NOT TRUTHis Is what it says next to the order
"""Parcel was handed to resident"" LIAR !!!!!Package was left on the pavement
"""Product Tester"" pages are rife on facebook run by chinese companies looking to move shoddy
"""The Earth's Most Customer-Centric Company"" is a total lie. Actually it has become the worst
"""want it tomorrow? No problem. Oh
*** Truly DISAPPOINTING experiences ***I ordered an item, before finalizing the order I had a total for the item, the shipping and the
***Title: Deceptive Sales Stunt - An Unfulfilled Promise**I recently had the unfortunate experience of participating in what was adverti
"+can buy items you can't buy locally+if you have a prime membership prime video is ok'ish and sometimes good.-prices usually 2x to 10x,
"......Amazon is a nefarious global Zio-corporation based in Seattle,WA.....FRAUD....MISCONDUCT.....ABUSING CUSTOMERS....ABUSING EMPLO
"1. Driver rang doorbell, but did not wait less than one minute to see if anyone was at home.2. The Amazon record says "Handed to Resid
"1. I researched online, and Amazon has upwards of 60 to 70% Asian sellers now (USA site): It's ridiculous - their clothing size measure
"1. You never get correct delivery date when ordering so if your like I am and order on certain days so you can plan delivery on a certa
"14 june, Ordered a product this morning because it said it would be delivered this evening, went to Amazon to check on delivery, got th
"1st of all I never get my stuff in 2 days.. Let's just put that out there... I'm fed up! By this time I'm only paying for Prime Video b
+-----+
only showing top 20 rows

```

```

+-----+
|ReviewText
+-----+
""" you'r having a laugh!!
"""Amazon does not honour its advertised Labor Day sale price in this instance"" - According to Ven
"""Arrive tonight by 10pm" is a myth. No matter what the website says or the countdown timer says - just forget about receiving
"""Blue murden"" shirts and merchandise. --"Amazon does not allow products that promote
"""Extremely frustrated with amazon. Ordered a crucial product with one-day delivery
"""I have a foreign name
"""Parcel handed to resident""""This parcel was handed to resident""NOT TRUTHis Is what it says next to the order
"""Parcel was handed to resident"" LIAR !!!!!Package was left on the pavement
"""Product Tester"" pages are rife on facebook run by chinese companies looking to move shoddy
"""The Earth's Most Customer-Centric Company"" is a total lie. Actually it has become the worst
"""want it tomorrow? No problem. Oh
*** Truly DISAPPOINTING experiences ***I ordered an item, before finalizing the order I had a total for the item, the shipping
***Title: Deceptive Sales Stunt - An Unfulfilled Promise**I recently had the unfortunate experience of participating in what wa
"+can buy items you can't buy locally+if you have a prime membership prime video is ok'ish and sometimes good.-prices usually 2
"......Amazon is a nefarious global Zio-corporation based in Seattle,WA.....FRAUD....MISCONDUCT.....ABUSING CUSTOMERS....ABUS
"1. Driver rang doorbell, but did not wait less than one minute to see if anyone was at home.2. The Amazon record says "Handed
"1. I researched online, and Amazon has upwards of 60 to 70% Asian sellers now (USA site): It's ridiculous - their clothing siz
"1. You never get correct delivery date when ordering so if your like I am and order on certain days so you can plan delivery o
"14 june, Ordered a product this morning because it said it would be delivered this evening, went to Amazon to check on deliver
"1st of all I never get my stuff in 2 days.. Let's just put that out there... I'm fed up! By this time I'm only paying for Prim
+-----+
only showing top 20 rows

```

```

+-----+
|MeaningfulWords
+-----+
[["amazon, truly, revolutionized, way, shop, online., platform, offers, incredible, variety, products]
[["horrible, experience, amazon, -, recommend, worst, enemies, get, associated, them""amazon, delivered, bosch, front, load, washing, mach
[["indian, space, chimps, mars""awful]
[["we, publish, review, meet, guidelines""amazon, liars., publish, honest, review, blocked, protect, 🚫, chinese, suppliers, scamming, u
[["1, amazon, listing, seemed, like, counterfeit, product, (<$500, price, tag, $2k, item.)2), amazon, representative, told, personally, vet
[["1.7, rating, time, review,, really?, problems, amazon., huge, selection,, price, generally, lower, always,, shipping, great,, returning,
[["6.4.23, finally, got, call., best, person, deal, someone, apparently, top., , get, refund, unless, a-z, may, get, money, seller, says, re
[["a, review, states,, ""i, understand, people, give, amazon, many, bad, reviews""., person., bought, loads, goods, u.k., u.s., website, ran
[["aws, re/start, training, scam, promises, job, placement, support,, training, says, ""no, guarantees, connect, job, opportunities, open, p
[["about, buy, firestick, offer., amazon, offered, bigger, discount, trading, old, firestick,, did., whole, trade, took, 4, days., received,
[["absolute, joke, customer, service., chatted, customer, support, named, samina, noorahmed., delayed, reply, terrible, customer, service, r
[["absolutely, appalling, customer, service., amazon, customer, 18, years,, mostly, without, incident, although, hiccups, along, way., , mid
[["absolutely, awful!, amazon, customer, support, representatives, often, rude, frequently, unhelpful., so-called, ""supervisor""., informed
[["absolutely, seething., bought, christmas, presents, website, dear, family, members., believe, tell, arrived, today, (06/02/2024).., asked,
[["after, ""publishing"", kdp]
[["almost, customer, services, calls, answered, overseas, call, center., even, request, ""native"", english, speaking, representative., many
[["amazon, prime, gotten, horrible., last, 3, times, ordered, shipment, delayed, last, minute., calling, associates, condescending, customer
[["amazon, continues, mislead, buyers, thier, ""overnite""., deliveries, pay, 2.99, quicker, delivery.i, bought, item, overnite, delivery..
[["amazon, eero, pro, 6e, mesh, wi-fi, 6e, router, system:world-class, customer, support:, questions?, expert, wi-fi, troubleshooters, ready
[["amazon, groceries.i, order, amazon, (groceries), delivery, 2200hrs, 2400hrs.the, delivery, marked, ""packed, ready""., midnight, came, we
+-----+
only showing top 20 rows

```

Model Accuracy: 0.6151

ใจที่ไทย

- Import libraries
 - `SparkSession`
 - `IntegerType` from `pyspark.sql.types`
 - `All` from `pyspark.sql.functions`
 - `HashingTF`, `Tokenizer`, `StopWordRemover` from `pyspark.ml.feature`
 - `Pipeline` from `pyspark.ml`
 - `LogisticRegression` from `pyspark.ml.classification`
 - `MulticlassClassificationEvaluator` from `pyspark.ml.evaluation`
- (นี่เข้าใจยากมาก)
- Create `SparkSession` (สร้าง `SparkSession`)
- Read data from file (`reviews_rated.csv`) (อ่านข้อมูลจากไฟล์ `reviews_rated.csv`)
- Select Review Text and Rating columns where Review Text can be trimmed using `trim()` and Rating should be converted to `IntegerType`
(เลือกคอลัมน์ `Review Text` และ `Rating` โดยที่ `Review Text` ต้องถูกตัดหัวท้ายด้วย `trim()` และ `Rating` ต้องแปลงเป็น `IntegerType`)
- Show data(แสดงข้อมูล)
- Create Tokenizer using:
 - `<your tokenizer> = Tokenizer(inputCol=<your review text column>, outputCol=<your review text words column>)`
- (สร้าง `Tokenizer` โดยใช้)

- Create `StopWordsRemover` using:
 - `<your stop word remover> = StopWordsRemover(inputCol=<your tokenizer>.getOutputCol(), outputCol=<your meaningful words column>)`
- (สร้าง `StopWordsRemover` โดยใช้)
- Create `HashingTF` using:
 - `<your hashing TF> = HashingTF(inputCol=<your stop word remover>.getOutputCol(), outputCol="features")`
- (สร้าง `HashingTF` โดยใช้)
- Create Pipeline by using `<your tokenizer>`, `<your stopword remover>`, and `<your hashing TF>` as stages
(สร้าง Pipeline โดยใช้ `<your tokenizer>`, `<your stopword remover>`, และ `<your hashing TF>` เป็นชั้นตอน)
- Create train and test datasets (สร้างชุดข้อมูล train และ test)
- Show train dataset (แสดงชุดข้อมูล train)
- Fit train data to the pipeline (ปรับข้อมูล train ให้เข้ากับ pipeline)
- Transform train data to a new train Dataframe (แปลงข้อมูล train เป็น Dataframe ใหม่)
- Transform test data to a new test Dataframe (แปลงข้อมูล test เป็น Dataframe ใหม่)

- Create `LogisticRegression` (สร้าง `LogisticRegression`)
- Fit train Dataframe to the created LogisticRegression (ปรับ Dataframe ของ train ให้เข้ากับ `LogisticRegression` ที่สร้างขึ้น)
- Transform the test Dataframe to the model (แปลง Dataframe ของ test ด้วยโมเดลที่สร้างขึ้น)
- Show `<"your meaningful words column">`, `<"your label column">`, and `<"your prediction column">`
(แสดง `<"คอลัมน์คำที่มีความหมาย">`, `<"คอลัมน์ label ของคุณ">`, และ `<"คอลัมน์การทำนายของคุณ">`)
- Create `MulticlassClassificationEvaluator` (สร้าง `MulticlassClassificationEvaluator`)
- Evaluate the accuracy using your created `MulticlassClassificationEvaluator`
(ประเมินความแม่นยำโดยใช้ `MulticlassClassificationEvaluator` ที่สร้างขึ้น)

ໄຈທຍໍ່ ENG

- Import libraries
 - `SparkSession`
 - `IntegerType` from `pyspark.sql.types`
 - All from `pyspark.sql.functions`
 - `HashingTF`, `Tokenizer`, `StopWordRemover` from `pyspark.ml.feature`
 - Pipeline from `pyspark.ml`
 - `LogisticRegression` from `pyspark.ml.classification`
 - `MulticlassClassificationEvaluator` from `pyspark.ml.evaluation`
- Create SparkSession
- Read data from file (`reviews_rated.csv`)
- Select Review Text and Rating columns where Review Text can be trimmed using `trim()` and Rating should be converted to `IntegerType`
- Show data
- Create Tokenizer using:
 - `<your tokenizer> = Tokenizer(inputCol=<“your review text column”>, outputCol=<“your review text words column”>`
- Create StopWordsRemover using:
 - `<your stop word remover> = StopWordsRemover(inputCol=<yourtokenizer>.getOutputCol(), outputCol=<“your meaningful words column”>`

- Create HashingTF using:
 - `<your hashing TF> = HashingTF(inputCol=<your stop word remover>.getOutputCol(), outputCol=“features”`
- Create Pipeline by using `<your tokenizer>`, `<your stopword remover>`, and `<your hashing TF>` as stages
- Create train and test datasets
- Show train dataset
- Fit train data to the pipeline
- Transform train data to a new train `Dataframe`
- Transform test data to a new test `Dataframe`
- Create `LogisticRegression`
- Fit train Dataframe to the created `LogisticRegression`
- Transform the test `Dataframe` to the model
- Show `<“your meaningful words column”>`, `< “your label column”>`, and `<“your prediction column”>`
- Create `MulticlassClassificationEvaluator`
- Evaluate the accuracy using your created `MulticlassClassificationEvaluator`

reviews_rated.csv

A	B	C	D	E	F	G	H	I
Reviewer Name	Profile Link	Country	Review Count	Review Date	Rating	Review Title	Review Text	Date of Experience
Eugene ath	/users/66e8185ff1598352 US		1 review	2024-09-16T13:44:26.00	1	A Store That Doesn't Work	I registered on the website, tried to make a purchase, but it didn't work.	16-Sep-24
Daniel ohalloran	/users/5d75e460200c1f6 GB		9 reviews	2024-09-16T18:26:46.00	1	Had multiple orders	One	16-Sep-24
p fisher	/users/546ccfcf10000640C GB		90 reviews	2024-09-16T21:47:39.00	1	I informed these people	I informed these people	16-Sep-24
Greg Dunn	/users/62c35cdabacc0eaAU		5 reviews	2024-09-17T07:15:49.00	1	Advertise one price	Then I have bought from Amazon before	17-Sep-24
Sheila Hannah	/users/5ddbe429478d882 GB		8 reviews	2024-09-16T18:37:17.00	1	If I could give a lower rate	If I could give a lower rate	16-Sep-24
Tamesha Hamilton	/users/61c9cae376c3900 US		4 reviews	2024-09-17T10:07:38.00	1	Terrible, I had to hang up	Terrible you get customer service	16-Aug-24
Irene Rousseau	/users/5951a1dd0000ff0C US		30 reviews	2024-09-16T23:43:32.00	1	Yet again	Amazon has a way of tainting	16-Sep-24
Jacqueline Novak	/users/66e7036d940b04e US		2 reviews	2024-09-15T18:07:45.00	5	I love amazon	I love amazon! I use it for half	15-Sep-24
Robert Field	/users/66e96137389eefc1 GB		1 review	2024-09-17T13:00:12.00	1	I applied for a job with Amazon	I applied for a job with Amazon	17-Sep-24
MH	/users/66e87dbc940b047 US		3 reviews	2024-09-16T21:04:07.00	5	I had a great experience	I had a great experience with	13-Sep-24
Jimmie Detrick	/users/66e81e7ffb6f87ec JP		3 reviews	2024-09-16T14:06:17.00	5	Every time there is a problem	Every time there is a problem,	5-Sep-24
Michael Mac	/users/652e72b82182a0GB		1 review	2024-09-17T13:16:22.00	1	Amazon aggressive mark	I have no interest in signing up	17-Sep-24

Regression

ElasticNetParam

ตัวอย่าง

สมมติว่าคุณมีโมเดลกี่ต้องทำนายยอดขายสินค้า และข้อมูลที่ใช้ในการทำนายมีฟีเจอร์หลายตัว เช่น:

- price (ราคา)
- advertising_spend (งบโฆษณา)
- number_of_reviews (จำนวนรีวิว)
- average_review_score (คะแนนรีวิวเฉลี่ย)
- stock_availability (จำนวนสินค้าคงคลัง)

ใช้ ElasticNetParam = 0 (L2 Regularization) ** เอา Columns มาทั้งหมดแต่ไม่ให้ความสำคัญ หากเราตั้งค่า ElasticNetParam เป็น 0 โมเดลจะใช้ L2 Regularization เต็มรูปแบบ (Ridge Regression) ซึ่งจะพยายามลดค่าเบี้ยนักของฟีเจอร์ที่ไม่สำคัญให้ใกล้ศูนย์ โดยยังคงรวมฟีเจอร์ กันหมดไว้

ตัวอย่างเช่น:

- โมเดลอาจลดความสำคัญของ average_review_score แต่ไม่กำจัดมันไปจากโมเดล ค่าเบี้ยนัก ของมันอาจใกล้ศูนย์ เช่น 0.02

ใช้ ElasticNetParam = 1 (L1 Regularization) ** ตัด Columns บางส่วนออกไปเลยที่ไม่สำคัญ หากเราตั้งค่า ElasticNetParam เป็น 1 โมเดลจะใช้ L1 Regularization เต็มรูปแบบ (Lasso Regression) ซึ่งจะพยายามบีบอัดน้ำหนักของฟีเจอร์ที่ไม่สำคัญให้เป็นศูนย์ไปเลย ซึ่งหมายความว่า ฟีเจอร์บางตัวจะถูกตัดออกไปจากโมเดล

ตัวอย่างเช่น:

- โมเดลอาจพบว่า number_of_reviews และ stock_availability ไม่มีความสำคัญต่อการทำนาย ยอดขาย ดังนั้นค่าเบี้ยนักของฟีเจอร์เหล่านี้จะถูกตัดออกโดยเป็นศูนย์ โมเดลจึงไม่พิจารณาฟีเจอร์เหล่านี้อีกต่อไป

ใช้ ElasticNetParam = 0.5 (ผสมระหว่าง L1 และ L2 Regularization)

หากตั้งค่า ElasticNetParam เป็น 0.5 โมเดลจะใช้ กึ่ง L1 และ L2 Regularization แบบผสมผสาน ดังนั้นฟีเจอร์บางตัวอาจถูกตัดออกไป (ตาม L1) และฟีเจอร์บางตัวอาจยังคงอยู่แต่มีน้ำหนักลดลงใกล้ ศูนย์ (ตาม L2)

ตัวอย่างเช่น:

- โมเดลอาจลดค่าเบี้ยนักของ number_of_reviews ให้เป็นศูนย์ (ตัดออก) และค่าเบี้ยนักของ stock_availability อาจถูกลดลงเหลือ 0.1

Dataframe

ใช้ filter() หรือ where():

- ใช้ในการกรอง||ถ้าที่ไม่มีค่า null โดยระบุเงื่อนไขอย่างชัดเจน

ตัวอย่าง

- df = df.filter(df["column_name"].isNotNull()) # กรอง||ถ้าที่ไม่มีค่า null ในคอลัมน์ "column_name"
- df = df.where(df["column_name"].isNotNull()) # ใช้ where ได้เช่นกัน

fillna():

- เหมือนกับ na.fill() โดยสามารถแทนค่า null ด้วยค่าที่ระบุได้

ตัวอย่าง

- df = df.fillna(0) # แทนค่าที่เป็น null ด้วย 0 ในทุกคอลัมน์
- df = df.fillna({"age": 18, "name": "unknown"}) # แทนค่าที่เป็น null ในคอลัมน์ที่ระบุด้วยค่าที่ต้องการ

ใช้ replace():

- ใช้แทนค่าหลาย ๆ ค่า รวมถึงค่า null หรือค่าอื่นที่ไม่ต้องการในคอลัมน์ด้วยค่าใหม่

ตัวอย่าง

- df = df.replace(float("nan"), None) # แทนค่า NaN ด้วย None หรือ null
- df = df.replace("unknown", None) # แทนค่าที่ไม่ต้องการด้วย None

ใช้ dropna():

- ฟังก์ชันนี้เหมือน na.drop() แต่รองรับการระบุเงื่อนไขเพิ่มเติม เช่น จะลบ||ถ้ามี null ในบางคอลัมน์เท่านั้น หรือจะลบ||ถ้าเมื่อเจอ null มากกว่าจำนวนที่กำหนด

ตัวอย่าง

- df = df.dropna(subset=["column1", "column2"]) # ลบ||ถ้ามีค่า null ในคอลัมน์ที่ระบุเท่านั้น
- df = df.dropna(thresh=2) # ลบ||ถ้ามีคอลัมน์ที่ไม่ใช่ null น้อยกว่า 2 คอลัมน์

ใช้ SQL สำหรับการจัดการ null:

- สามารถใช้คำสั่ง SQL ผ่าน Spark SQL เพื่อจัดการกับค่า null

ตัวอย่าง

- df.createOrReplaceTempView("data")
- df_non_null = spark.sql("SELECT * FROM data WHERE column_name IS NOT NULL")

coalesce():

- ใช้ในการแทนค่าที่เป็น null ด้วยค่าจากคอลัมน์อื่น โดยเลือกค่าที่ไม่ใช่ null ตัวแรกที่เจอ

ตัวอย่าง

- from pyspark.sql.functions import coalesce
- df = df.withColumn("new_column", coalesce(df["column1"], df["column2"])) # ใช้ค่าจาก column2 หาก column1 เป็น null

Linear Regression Code

```
1  from pyspark.sql import SparkSession
2  from pyspark.ml.feature import StringIndexer, VectorAssembler
3  from pyspark.ml.regression import LinearRegression
4  from pyspark.ml.evaluation import RegressionEvaluator
5  from pyspark.ml import Pipeline
6  from pyspark.sql.types import IntegerType
7  from pyspark.sql.functions import col, desc
8  import seaborn as sns
9  import matplotlib.pyplot as plt
10
11 #Linear
12
13 # Linear Regression Pipeline Example
14 # สร้าง SparkSession เพื่อใช้ในการทำงานต้น PySpark
15 spark = SparkSession.builder \
16     .appName("LinearRegressionPipelineExample") \
17     .getOrCreate()
18
19 # ขั้นตอนที่ 2: โหลดข้อมูลจากไฟล์ CSV ลงใน DataFrame
20 data = spark.read.csv('fb_live_thailand.csv', header=True, inferSchema=True)
21
22 # ขั้นตอนที่ 3: ใช้ StringIndexer เพื่อเตรียมข้อมูล
23 # แปลงคอลัมน์ num_reactions เป็นค่าที่เป็นตัวเลข
24 indexer_reactions = StringIndexer(inputCol="num_reactions", outputCol="num_reactions_ind")
25 # แปลงคอลัมน์ num_loves เป็นค่าที่เป็นตัวเลข
26 indexer_loves = StringIndexer(inputCol="num_loves", outputCol="num_loves_ind")
27
28 # ขั้นตอนที่ 4: ใช้ VectorAssembler เพื่อสร้างเวกเตอร์ฟีเจอร์
29 assembler = VectorAssembler(
30     inputCols=["num_reactions_ind", "num_loves_ind"],
31     outputCol="features"
32 )
33
34 # ปรับค่า โนಡ
35 # ขั้นตอนที่ 5: กำหนดโนಡ Linear Regression
36 lr = LinearRegression(
37     featuresCol="features",
38     labelCol="num_loves_ind",
39     maxIter=10,           # จำนวนรอบการฝึกฝน 10 รอบ
40     regParam=0.1,         # พารามิเตอร์การลดความซ้ำซ้อน
41     elasticNetParam=0.8 # ค่าที่ใช้ในการควบคุมระหว่าง L1 และ L2 regularization
42 )
43
```

```

44 # ค่า ElasticNetParam สามารถกำหนดได้ระหว่าง 0 ถึง 1:
45 # หากค่าทั้งคู่เป็น 0 โนดจะใช้เฉพาะ L2 Regularization (Ridge) ** เวลา Columns มาทั้งหมดแต่ไม่ให้ความสำคัญ
46 # หากค่าทั้งคู่เป็น 1 โนดจะใช้เฉพาะ L1 Regularization (Lasso) ** ตัว Columns บังส่วนของไปเลยที่ไม่สำคัญ
47 # หากค่าทั้งคู่ ระหว่าง 0 และ 1 โนดจะใช้การผสมผสานระหว่าง L1 และ L2 , 0.5 0.8 0.7 ประมาณนี้
48
49 # ขั้นตอนที่ 6: สร้าง Pipeline ตัวแอลซึ่งขั้นตอน
50 # ในการ indexer_reactions และ indexer_loves ใน pipeline
51 pipeline = Pipeline(stages=[assembler, lr]) # stages=[assembler, lr] ทำงานตามลำดับขั้นตอน
52 # assembler --> lr
53
54 # ขั้นตอนที่ 7: แบ่งข้อมูลเป็นชุดฝึกและชุดทดสอบ
55 train_data, test_data = data.randomSplit([0.8, 0.2], seed=1234) # randomSplit ต้านมีจังเก็ต Overfitting
56
57 # ขั้นตอนที่ 8: ผูกโนดเดียวชุดฝึก
58 # ใช้ StringIndexers เพื่อแปลงข้อมูลในชุดฝึก
59 train_data = indexer_reactions.fit(train_data).transform(train_data)
60 train_data = indexer_loves.fit(train_data).transform(train_data)
61 pipeline_model = pipeline.fit(train_data)

62
63 # ขั้นตอนที่ 9: แปลงข้อมูลชุดทดสอบด้วยโนดเดล pipeline
64 # ใช้ StringIndexers เพื่อแปลงข้อมูลในชุดทดสอบ
65 test_data = indexer_reactions.fit(test_data).transform(test_data)
66 test_data = indexer_loves.fit(test_data).transform(test_data)
67 predictions = pipeline_model.transform(test_data)
68
69 # ขั้นตอนที่ 10: สร้าง RegressionEvaluator
70 evaluator = RegressionEvaluator(
71     labelCol="num_loves_ind",
72     predictionCol="prediction"
73 )
74
75 print()
76 print('='*50)
77 print()

78
79 # MSE ใกล้เคียงกับ 0 ดีสุด
80 # ประเมิน Mean Squared Error (MSE)
81 evaluator.setMetricName("mse")
82 mse = evaluator.evaluate(predictions)
83 print(f"Mean Squared Error (MSE): {mse}")

84
85 # R2 ใกล้เคียงกับ 1 ดีสุด
86 # ประเมิน R-squared (R2)
87 evaluator.setMetricName("r2")
88 r2 = evaluator.evaluate(predictions)
89 print(f"R-squared (R2): {r2}")
90 print()
91 print('='*50)
92 print()
93

```

```

94 # ขั้นตอนที่ 11: เลือกคอลัมน์และแปลงประเภท, เรียงลำดับตาม 'prediction' ในลำดับลดลง
95 # เลือกเฉพาะคอลัมน์ num_loves_ind และ prediction พร้อมแปลงเป็นประเภท Integer
96 # และเรียงข้อมูลตาม prediction ในลำดับลดลง
97
98 selected_data = predictions.select(
99     col("num_loves_ind").cast(IntegerType()).alias("num_loves"),
100    col("prediction").cast(IntegerType()).alias("prediction")
101 ).orderBy(col("prediction").desc())
102
103 # แปลงเป็น Pandas DataFrame สำหรับการวิเคราะห์เพิ่มเติม
104 selected_data_pd = selected_data.toPandas()
105 print(selected_data_pd)
106
107 # ขั้นตอนที่ 12: การสร้างกราฟโดยใช้ Seaborn's lmplot
108 plt.figure(figsize=(12, 6))
109 sns.regplot(
110     data=selected_data_pd,
111     x='num_loves',
112     y='prediction',
113     scatter_kws={'s': 50, 'alpha': 0.5}, # กำหนดคุณสมบัติของจุดในกราฟ
114     line_kws={'color': 'red'} # กำหนดคุณสมบัติของเส้นในกราฟ
115 )
116
117 plt.title('Linear Regression: Actual vs Predicted Values') # ชื่อกราฟ
118 plt.xlabel('Actual num_loves') # ชื่อแกน x
119 plt.ylabel('Predicted num_loves') # ชื่อแกน y
120 plt.show() # แสดงกราฟ
121

```

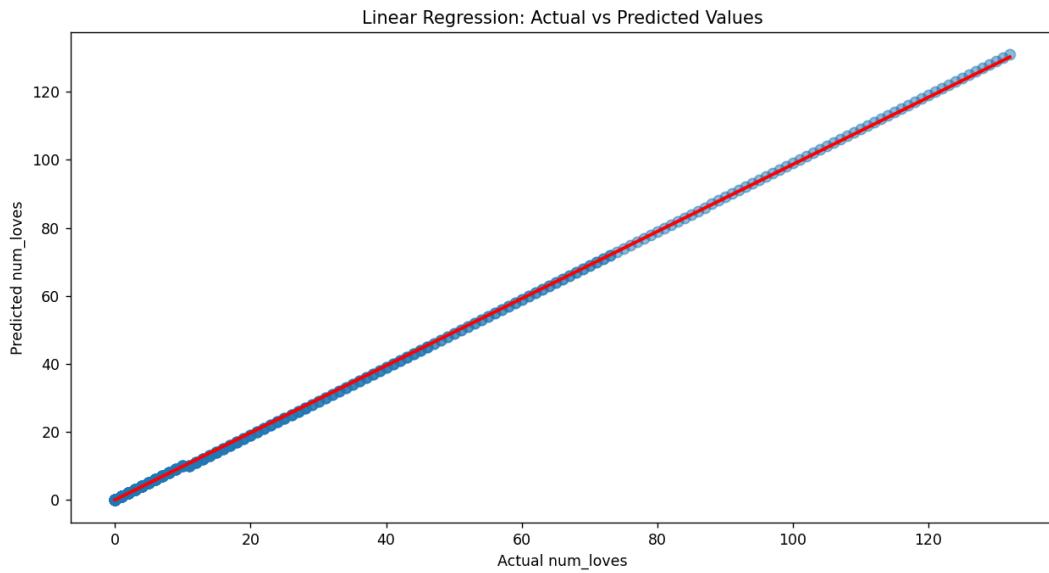
ผลลัพธ์

```

=====
Mean Squared Error (MSE): 0.007204820474657998
R-squared (R2): 0.9999878577178806
=====

      num_loves  prediction
0          132        131
1          131        130
2          130        129
3          129        128
4          128        127
...
1369         0         0
1370         0         0
1371         0         0
1372         0         0
1373         0         0

```



Decision Tree Regression Code

```

1  # ນັບໃນເສີ່ມ ດ້ວຍເລືດ 1
2
3  from pyspark.sql import SparkSession
4  from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
5  from pyspark.ml.regression import DecisionTreeRegressor
6  from pyspark.ml.evaluation import RegressionEvaluator
7  from pyspark.ml import Pipeline
8
9  # ສ້າງ SparkSession
10 spark = SparkSession.builder.appName("DecisionTreeRegression").getOrCreate()
11
12 # ໂອດຊອບຂໍ້ມູນຈາກໄຟສ CSV
13 df = spark.read.format("csv").option("header", True).load("fb_live_thailand.csv")
14
15 # ພຳລັງຄອລິນນີ້ num_reactions ແລະ num_loves (ເປັນຂັບຕື່ອນດຸ)
16 df = df.select(df.num_reactions.cast("Double"), df.num_loves.cast("Double")) # Float 32 ມີຄ Double 64 ມີຄ
17
18 # ລັບແກວທີ່ມີຄ່າ null ຕ່າງວ່າ (ຕ້າງnot null ໃນວ່າງ)
19 df = df.na.drop()
20 # df = df.na.drop() # ລັບແກວທີ່ມີຄ່າ null ທັງໝົດ
21 # df = df.na.fill(0) # ແກນຄ່າທີ່ເປັນ null ດັວນ 0 ໃນທຸກຄອລິນນີ້
22 # df = df.na.fill({"age": 18}) # ແກນຄ່າທີ່ເປັນ null ໃນຄອລິນນີ້ "age" ດັວນ 18
23
24
25 # ໃຊ້ StringIndexer ເພື່ອແປ່ງຄອລິນນີ້ num_reactions ແລະ num_loves ເປັນຄາດໜີ້
26 indexer_reactions = StringIndexer(inputCol="num_reactions", outputCol="num_reactions_ind")
27 indexer_loves = StringIndexer(inputCol="num_loves", outputCol="num_loves_ind")
28
29 # ໃຊ້ OneHotEncoder ເພື່ອແປ່ງຄາດໜີ້ທີ່ເປັນແກຣດອຣີທີ່ບໍ່ມີການແນບ One-Hot
30 encoder_reactions = OneHotEncoder(inputCols=["num_reactions_ind"], outputCols=["reactions_encoded"])
31 encoder_loves = OneHotEncoder(inputCols=["num_loves_ind"], outputCols=["loves_encoded"])
32
33 # ໃຊ້ VectorAssembler ເພື່ອຮັມພືເຈອຣີຂໍ້ຕົວກິດແປ່ງແກຣດີຍ່າງ
34 vecAssembler = VectorAssembler(inputCols=["reactions_encoded", "loves_encoded"], outputCol="features")
35
36 # ຄ່າວັດ Pipeline ທີ່ກະກອບຄ່າຍື່ນຕອນຄ່າງ ຖ້າ
37 pipeline = Pipeline(stages=[indexer_reactions, indexer_loves, encoder_reactions, encoder_loves, vecAssembler])
38
39 # ຢືນ Pipeline model ໂດຍໃຫ້ມູດຂໍ້ມູນດີ
40 pipeline_model = pipeline.fit(df)

```

```

41 # ทำการแปลงข้อมูลตัว变量 Pipeline model
42 df_transformed = pipeline_model.transform(df)
43
44 # แบ่งข้อมูลเป็นชุดฝึกและชุดทดสอบ
45 train_df, test_df = df_transformed.randomSplit([0.8, 0.2], seed=1234)
46
47 # สร้างโมเดล Decision Tree Regressor พร้อมตั้งค่า hyperparameters
48 dt_regressor = DecisionTreeRegressor(featuresCol="features", labelCol="num_loves_ind", maxDepth=30, minInstancesPerNode=2)
49 # tune Model maxDepth=30, minInstancesPerNode=2 ปรับตั้งค่า
50
51 # ฝึกโมเดลตัวอย่างฝึก
52 dt_model = dt_regressor.fit(train_df)
53
54
55 # ทำการท่านายด้วยโมเดลที่ฝึกมา
56 predictions = dt_model.transform(test_df)
57
58 # สร้างตัวประมวลผลเพื่อประเมินโมเดล
59 evaluator = RegressionEvaluator(labelCol="num_loves_ind", predictionCol="prediction")
60
61 # คำนวณคะแนน R2
62 r2 = evaluator.setMetricName("r2").evaluate(predictions)
63 print()
64 print('='*50)
65 print()
66 print(f'R2: {r2}')
67 print()
68 print('='*50)
69 print()
70
71 spark.stop()

```

ผลลัพธ์

```

=====
R2: 0.7039013416577182
=====
```

จ้อย Linear Regression

- Import library SparkSession (นำเข้าไลบรารี SparkSession)
- Import StringIndexer and VectorAssembler from pyspark.ml.feature (นำเข้า StringIndexer และ VectorAssembler จาก pyspark.ml.feature)
- Import LinearRegression from pyspark.ml.regression (นำเข้า LinearRegression จาก pyspark.ml.regression)
- Import RegressionEvaluator from pyspark.ml.evaluation (นำเข้า RegressionEvaluator จาก pyspark.ml.evaluation)
- Import Pipeline from pyspark.ml (นำเข้า Pipeline จาก pyspark.ml)
- Create SparkSession (สร้าง SparkSession)
- Load data file into DataFrame (FBLiveTH) (โหลดไฟล์ข้อมูลเข้า DataFrame ชื่อ FBLiveTH)
- Use StringIndexer to prepare data where the columns num_reactions and num_loves are used as input to create indexes (num_reactions_ind and num_loves_ind), then fit and transform (ใช้ StringIndexer เพื่อเตรียมข้อมูลให้เข้าสู่คอลัมน์ num_reactions และ num_loves เป็นชื่อของคอลัมน์ (num_reactions_ind และ num_loves_ind) จากนั้นทำการ fit และ transform)
- Use VectorAssembler to create vector of num_reactions_ind and num_loves_ind resulting in the features (ใช้ VectorAssembler เพื่อสร้างคอลัมน์ num_reactions_ind และ num_loves_ind เป็นสร้าง features)
- Create linear regression where the num_loves_ind is used as a label and setMaxIter (set maximum iteration), setRegParam (set regularisation parameter [0...1]), and set ElasticNetParam (set ElasticNet parameter [0...1]) are also used (สร้างการ回帰โดยใช้ num_loves_ind เป็น label และตั้งค่า setMaxIter (กำหนดจำนวนรอบสูงสุด), setRegParam (กำหนดพารามิเตอร์การเรียนรู้ [0...1]), และ setElasticNetParam (กำหนดพารามิเตอร์ ElasticNet [0...1]))
- Create a pipeline where the stages include output from the vector assembler and the created linear regression (สร้าง pipeline ที่มีขั้นตอนประกอบด้วย vector assembler และการ回帰โดยใช้ค่าที่สร้างขึ้น)
- Create train and test datasets using randomSplit function where the output from string indexer is used as an input (สร้างตัวชี้ของ train และ test โดยใช้ฟังก์ชัน randomSplit โดยผลลัพธ์จาก string indexer ใช้เป็นชื่อของคอลัมน์)
- Fit train data into the created pipeline to create the pipeline model (ฝึกชุดข้อมูล train ด้วย pipeline ที่สร้างขึ้นเพื่อสร้างโมเดล pipeline)

```

- Use the created pipeline model to transform test data resulting in the predictions DataFrame (ใช้โมเดล pipeline ที่สร้างขึ้น
  แปลงชุดข้อมูล test เพื่อให้ได้ DataFrame ค่าท่านาย)
- Show 5 rows of the predictions DataFrame (แสดง 5 แถวแรกของ DataFrame ค่าท่านาย)
- Use RegressionEvaluator to create the evaluator where the num_loves_ind is used as the label column and the prediction
  column is used as the prediction column (ใช้RegressionEvaluator เพื่อสร้างตัวประเมินโดยใช้ num_loves_ind เป็นคอลัมน์
  prediction เป็นคอลัมน์ค่าท่านาย)
- Show MSE (evaluator.setMetricName("mse").evaluate(predictions)) และ R2
  (evaluator.setMetricName("r2").evaluate(predictions)) (แสดงค่า MSE (evaluator.setMetricName("mse").evaluate(predictions))
  และ R2 (evaluator.setMetricName("r2").evaluate(predictions)))
- Import seaborn (นำเข้า seaborn)
- Import IntegerType from pyspark.sql.types (นำเข้า IntegerType จาก pyspark.sql.types)
- Import matplotlib.pyplot (นำเข้า matplotlib.pyplot)
- Import col and desc from pyspark.sql.functions (นำเข้า col และ desc จาก pyspark.sql.functions)
- Select num_loves and prediction column where the values in these columns are converted to IntegerType and order by the
  column prediction in descending order (เลือกคอลัมน์ num_loves และ prediction โดยค่าที่มีผลลัพธ์มากที่สุดจะอยู่บนบน
  ตามคอลัมน์ prediction และลง)
- Use selected data as data and convert it to Pandas, then use lmplot function of seaborn where x is num_loves and y is
  prediction column (ใช้โมเดลที่เลือกเป็น data และแปลงเป็น Pandas จากนั้นใช้ฟังก์ชัน lmplot ของ seaborn โดยกำหนดให้ x เป็นคอลัมน์ num_loves และ y
  เป็นคอลัมน์ prediction)
- Use matplotlib to show the plot (ใช้matplotlib เพื่อแสดงกราฟ)

```

โจทย์ Decision Tree Regression

```

- Import library SparkSession
- Import StringIndexer, VectorAssembler, and OneHotEncoder from pyspark.ml.feature
- Import DecisionTreeRegressor from pyspark.ml.regression
- Import RegressionEvaluator from pyspark.ml.evaluation
- Import Pipeline from pyspark.ml
- Create SparkSession
- Load data file into DataFrame (FBLiveTH)
  (โหลดไฟล์ข้อมูลชุด DataFrame ชื่อ FBLiveTH)
- Use StringIndexer to prepare data where the columns num_reactions and num_loves are used as inputs to create indexes
  (num_reactions_ind and num_loves_ind)
  (ใช้StringIndexer เพื่อเตรียมข้อมูลโดยใช้คอลัมน์ num_reactions และ num_loves เป็นข้อมูลนำเข้าในการสร้างตัวหนี่ (num_reactions_ind และ
  num_loves_ind))
- Use OneHotEncoder to create Boolean flag of num_reactions_ind and num_loves_ind as they will be used as a component of
  a vector in the next steps
  (ใช้OneHotEncoder เพื่อสร้างตัวกำหนด Boolean flag สำหรับ num_reactions_ind และ num_loves_ind เนื่องจากจะใช้เป็นองค์ประกอบของเวกเตอร์ในขั้นตอน
  ถัดไป)
- Use VectorAssembler to create a vector of encoded num_reactions_ind and num_loves_ind resulting in the column features
  (ใช้VectorAssembler เพื่อสร้างเวกเตอร์จาก num_reactions_ind และ num_loves_ind ที่ถูกเข้ารหัส ซึ่งได้ผลลัพธ์เป็นคอลัมน์ features)
- Create a pipeline where the stages include output from string indexer, encoder, and vector assembler
  (สร้าง pipeline ที่ขั้นตอนประกอบด้วยผลลัพธ์จาก string indexer, encoder และ vector assembler)
- Fit DataFrame into the created pipeline to create the pipeline model
  (ฝึก DataFrame ด้วย pipeline ที่สร้างขึ้นเพื่อสร้างโมเดล pipeline)
- Use the created pipeline model to transform the DataFrame data resulting in another DataFrame
  (ใช้โมเดล pipeline ที่สร้างขึ้นแปลงข้อมูล DataFrame เพื่อให้ได้ DataFrame อีกชุดหนึ่ง)

```

```

- Create train and test datasets using randomSplit function where the output from the previous step is used as an input
  (สร้างชุดข้อมูล train และ test โดยใช้ฟังก์ชัน randomSplit โดยผลลัพธ์จากขั้นตอนก่อนหน้าถูกใช้เป็นข้อมูลฝึกหัด)
- Create decision tree regression where the num_loves_ind is used as a label and the output is features
  (สร้างการถอดความแบบต้นไม้ตัดสินใจโดยใช้ num_loves_ind เป็น label และใช้ features เป็นข้อมูลนำเข้า)
- Fit train data into the created decision tree to create the model (ฝึกชุดข้อมูล train ด้วยต้นไม้ตัดสินใจที่สร้างขึ้นเพื่อสร้างโมเดล)
- Use the created model to transform the test data resulting in a prediction DataFrame
  (ใช้โมเดลที่สร้างขึ้นแปลงชุดข้อมูลทดสอบเพื่อให้ได้ DataFrame ค่าท่านาย)
- Use RegressionEvaluator to create the evaluator where the num_loves_ind is used as the label column and the prediction
  column is used as the prediction column
  (ใช้RegressionEvaluator เพื่อสร้างตัวประเมินโดยใช้ num_loves_ind เป็นคอลัมน์ label และใช้คอลัมน์ prediction เป็นคอลัมน์ค่าท่านาย)
- Show R2 (evaluator.setMetricName("r2").evaluate(predictions))
  (แสดงค่า R2 (evaluator.setMetricName("r2").evaluate(predictions)))

```

Classification

Logistic Classification Code - 1

```
1  from pyspark.sql import SparkSession # ນໍາເຫັນ SparkSession ທີ່ມີນຸດເຕີມທີ່ໃນການໃຊ້ PySpark
2  from pyspark.ml.feature import StringIndexer, VectorAssembler # ນໍາເຫັນ StringIndexer ແລະ VectorAssembler
3  from pyspark.ml.classification import LogisticRegression # ນໍາເຫັນ LogisticRegression ເພື່ອໃຊ້ໃນກາສ່ວນໄນ້ຄວາມຈຳກັງປະກັບ
4  from pyspark.ml.evaluation import MulticlassClassificationEvaluator # ນໍາເຫັນ Evaluator ສໍາຮັບກາຍປະເມີນຜົນໄແລດ
5  from pyspark.ml import Pipeline # ນໍາເຫັນ Pipeline ສໍາຮັບຈຳກາງກະບຽນການສ່ວນໄຟເຕັມ
6
7  # ສ້າງ SparkSession ແລະຕັ້ງໜີ້ຂອງພາບໄຟເຕັມ
8  spark = SparkSession.builder \
9      .appName("LogisticRegressionPipelineExample") \
10     .getOrCreate()
11
12 # ລັບຂ່າຍມູນຈາກໄຟ່ CSV
13 data = spark.read.csv('fb_live_thailand.csv', header=True, inferSchema=True)
14
15 # ສ້າງ StringIndexer ສໍາຮັບອົດລົມທີ່ status_type ແລະ status_published
16 indexer_status_type = StringIndexer(inputCol="status_type", outputCol="status_type_ind")
17 indexer_status_published = StringIndexer(inputCol="status_published", outputCol="status_published_ind")
18
19 # ປຶບໃໝ່ StringIndexer ຕ່າງໆອື່ນ
20 data = indexer_status_type.fit(data).transform(data)
21 data = indexer_status_published.fit(data).transform(data)
22
23 # ສ້າງ VectorAssembler ເພື່ອຮົມທີ່ເຈັອຣເປັນເວັກເຕັກ
24 assembler = VectorAssembler(
25     inputCols=["status_type_ind", "status_published_ind"],
26     outputCol="features"
27 )
28
29 # ສ້າງໂມເຕັລ LogisticRegression
30 log_reg = LogisticRegression(
31     labelCol="status_type_ind", # ຄວລັມນີ້ label ສໍາຮັບກາຈ່າແນກປະກັບ
32     featuresCol="features", # ຄວລັມນີ້ເຈັອຣ
33     maxIter=10, # ຈຳນວນຮອບການຝຶກ
34     regParam=0.1, # ພາຣາມີເຕອກການປັບປຸງ
35     elasticNetParam=0.8 # ພາຣາມີເຕອກ ElasticNet
36 )
37
38 # ສ້າງ Pipeline ທີ່ປະກອບດ້ວຍ assembler ແລະ log_reg
39 pipeline = Pipeline(stages=[assembler, log_reg])
40
41 # ແບ່ງຂ່າຍມູນເປັນຊຸດຝຶກ (80%) ແລະຊຸດທດສອນ (20%)
42 train_data, test_data = data.randomSplit([0.8, 0.2], seed=1234)
43
44 # ຝຶກໂມເຕັລດ້ວຍຂ່າຍມູນຊຸດຝຶກ
45 pipeline_model = pipeline.fit(train_data)
46
47 # ສ້າງການທ່ານາຍຈາກຊຸດຂ່າຍມູນທດສອນ
48 predictions = pipeline_model.transform(test_data)
49
50 # ແສດຜລລັພທີ່ການທ່ານາຍ 5 ແລະ
51 predictions.select("status_type_ind", "prediction", "probability", "features").show(5)
52
53 # ສ້າງ Evaluator ສໍາຮັບປະເມີນຜົນໄຟເຕັມ
54 evaluator = MulticlassClassificationEvaluator(
55     labelCol="status_type_ind", # ຄວລັມນີ້ label
56     predictionCol="prediction" # ຄວລັມນີ້ prediction
57 )
```

```

58 print()
59 print('*'*50)
60 print()
61 # ค่าวนยนความแม่นยำของโมเดล
62 accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})
63 print(f"Accuracy: {accuracy}")
64
65 # ค่าวนยนความแม่นยำแบบน้ำหนัก
66 precision = evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"})
67 print(f"Precision: {precision}")
68
69 # ค่าวนยนความรู้จำแนกน้ำหนัก
70 recall = evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"})
71 print(f"Recall: {recall}")
72
73 # ค่าวนยน F1 Score
74 f1 = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
75 print(f"F1 Score: {f1}")
76 print()
77 print('*'*50)
78 print()
79
80 # หยุด SparkSession
81 spark.stop()
82

```

ผลลัพธ์

status_type_ind	prediction	probability	features
0.0	0.0	[0.87592035643515...]	[0.0,6627.0]
0.0	0.0	[0.87592035643515...]	[0.0,6693.0]
0.0	0.0	[0.87592035643515...]	[0.0,766.0]
0.0	0.0	[0.87592035643515...]	[0.0,848.0]
0.0	0.0	[0.87592035643515...]	[0.0,925.0]

only showing top 5 rows

=====

Accuracy: 0.9395924308588064
Precision: 0.8878293931636803
Recall: 0.9395924308588064
F1 Score: 0.9117163288502892

=====

Dataset fb_live_thailand.csv

A	B	C	D	E	F	G	H	I	J	K	L
status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_haha	num_sads	num_angrys
24667554	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0
24667554	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0
24667554	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0
24667554	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0
24667554	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0
24667554	photo	4/18/2018 2:14	217	6	0	211	5	1	0	0	0
24667554	video	4/18/2018 0:24	503	614	72	418	70	10	2	0	3
24667554	video	4/17/2018 7:42	295	453	53	260	32	1	1	0	1
24667554	photo	4/17/2018 3:33	203	1	0	198	5	0	0	0	0
24667554	photo	4/11/2018 4:53	170	9	1	167	3	0	0	0	0
24667554	photo	4/10/2018 1:01	210	2	3	202	7	1	0	0	0
24667554	photo	4/9/2018 2:06	222	4	0	213	5	4	0	0	0

Logistic Classification Code – 2

```

1  from pyspark.sql import SparkSession # นี่คือ SparkSession ที่มีอยู่แล้วใน PySpark
2  from pyspark.ml.feature import StringIndexer, VectorAssembler # นี่คือ StringIndexer และ VectorAssembler
3  from pyspark.ml.classification import LogisticRegression # นี่คือ LogisticRegression เพื่อใช้ในการสร้างโมเดลและการจำแนกประเภท
4  from pyspark.ml.evaluation import MulticlassClassificationEvaluator # นี่คือ Evaluator สำหรับการประเมินผลในโมเดล
5  from pyspark.ml import Pipeline # นี่คือ Pipeline สำหรับจัดการกระบวนการเร่งรัดในโมเดล
6
7  # สร้าง SparkSession และตั้งชื่อและพารามิเตอร์
8  spark = SparkSession.builder \
9      .appName("LogisticRegressionPipelineExample") \
10     .getOrCreate()
11
12 # อ่านข้อมูลจากไฟล์ CSV
13 data = spark.read.csv('fb_live_thailand.csv', header=True, inferSchema=True)
14
15 # สร้าง StringIndexer สำหรับคอลัมน์ status_type และ status_published
16 indexer_status_type = StringIndexer(inputCol="status_type", outputCol="status_type_ind")
17 indexer_status_published = StringIndexer(inputCol="status_published", outputCol="status_published_ind")
18
19 # ปรับใช้ StringIndexer กับข้อมูล
20 data = indexer_status_type.fit(data).transform(data)
21 data = indexer_status_published.fit(data).transform(data)
22
23 # สร้าง VectorAssembler เพื่อรวมฟีดเจอร์เป็นเวกเตอร์
24 assembler = VectorAssembler(
25     inputCols=["status_type_ind", "status_published_ind"],
26     outputCol="features"
27 )
28
29 # สร้างโมเดล LogisticRegression
30 log_reg = LogisticRegression(
31     labelCol="status_type_ind", # คอลัมน์ label สำหรับการจำแนกประเภท
32     featuresCol="features", # คอลัมน์ฟีดเจอร์
33     maxIter=10, # จำนวนรอบการฝึก
34     regParam=0.1, # พารามิเตอร์การบีบอัด
35     elasticNetParam=0.8 # พารามิเตอร์ ElasticNet
36 )
37
38 # สร้าง Pipeline ที่ประกอบด้วย assembler และ log_reg
39 pipeline = Pipeline(stages=[assembler, log_reg])
40
41 # แบ่งข้อมูลเป็นชุดตัด (80%) และชุดทดสอบ (20%)
42 train_data, test_data = data.randomSplit([0.8, 0.2], seed=1234)
43

```

```

44 # ฝึกโมเดลด้วยข้อมูลที่ได้
45 pipeline_model = pipeline.fit(train_data)
46
47 # สร้างการทำนายจากชุดข้อมูลทดสอบ
48 predictions = pipeline_model.transform(test_data)
49
50 # แสดงผลลัพธ์การทำนาย 5 ผล
51 predictions.select("status_type_ind", "prediction", "probability", "features").show(5)
52
53 # สร้าง Evaluator สำหรับประเมินผลโมเดล
54 evaluator = MulticlassClassificationEvaluator(
55     labelCol="status_type_ind",          # คอลัมน์ label
56     predictionCol="prediction"        # คอลัมน์ prediction
57 )
58 print()
59 print('*'*50)
60 print()
61 # คำนวณความแม่นยำของโมเดล
62 accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})
63 print(f"Accuracy: {accuracy}")
64
65 # คำนวณความแม่นยำแบบน้ำหนัก
66 precision = evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"})
67 print(f"Precision: {precision}")
68
69 # คำนวณความรู้จำแนกน้ำหนัก
70 recall = evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"})
71 print(f"Recall: {recall}")
72
73 # คำนวณ F1 Score
74 f1 = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
75 print(f"F1 Score: {f1}")
76 print()
77 print('*'*50)
78 print()
79
80 # หยุด SparkSession
81 spark.stop()
82

```

ผลลัพธ์

features	status_type_ind	prediction
[0.0,6645.0]	0.0	0.0
[0.0,6695.0]	0.0	0.0
[0.0,6698.0]	0.0	0.0
[0.0,1161.0]	0.0	0.0
[0.0,752.0]	0.0	0.0

only showing top 5 rows

Accuracy: 0.6025260029717682
 Precision: 0.38812437522645277
 Recall: 0.6025260029717682
 F1 Measure: 0.47212424000976233

จอย Logistic Classification

```
- Create SparkSession
- Import library SparkSession
- Import StringIndexer and VectorAssembler from pyspark.ml.feature
- Import LogisticRegression from pyspark.ml.classification
- Import MulticlassClassificationEvaluator from pyspark.ml.evaluation
- Import Pipeline from pyspark.ml

- Load data file into DataFrame (FBLiveTH) (โหลดไฟล์ข้อมูลเข้าเป็น DataFrame ชื่อ FBLiveTH)
- Use StringIndexer to prepare data where the columns status_type and status_published are used as input to create indexes (status_type ind and status_published ind), then fit and transform
(ใช้ StringIndexer เพื่อเตรียมข้อมูลโดยใช้คอลัมน์ status_type และ status_published เป็นอินพุตในการสร้างอินเดกซ์ (status_type ind และ status_published ind) จากนั้น fit และ transform)
- Use VectorAssembler to create a vector of status_type ind and status_published ind resulting in the column features
(ใช้ VectorAssembler เพื่อสร้างเก็งเตอร์จาก status_type ind และ status_published ind ให้ได้คอลัมน์ features)
- Create logistic regression where status_type ind is used as a label and setMaxIter (set maximum iteration), setRegParam (set regularisation parameter [0...1]), and setElasticNetParam (set ElasticNet parameter [0...1]) are also used
(สร้าง logistic regression โดยใช้ status_type ind เป็น label และค่าหนด setMaxIter (กำหนดจำนวนครั้งสูงสุดในการวนลูป), setRegParam (กำหนดพารามิเตอร์การปรับค่าปกติ [0...1]), และ setElasticNetParam (กำหนดพารามิเตอร์ ElasticNet [0...1]))

- Create a pipeline where the stages include output from the vector assembler and the created logistic regression
(สร้าง pipeline ที่ประกอบไปด้วยขั้นตอนที่รวมเอาผลลัพธ์จาก vector assembler และ logistic regression ที่สร้างขึ้น)
- Create train and test datasets using randomSplit function where the output from string indexer is used as input
(สร้างชุดข้อมูลฝึกและทดสอบโดยใช้ฟังก์ชัน randomSplit ซึ่งใช้ผลลัพธ์จาก string indexer เป็นอินพุต)
- Fit train data into the created pipeline to create the pipeline model
(นำข้อมูลฝึกเข้า pipeline ที่สร้างขึ้นเพื่อสร้าง pipeline model)
- Use the created pipeline model to transform test data resulting in the predictions DataFrame
(ใช้ pipeline model ที่สร้างขึ้นเพื่อแปลงข้อมูลทดสอบให้ได้เป็น DataFrame ชื่อ predictions)
- Show 5 rows of the predictions DataFrame (แสดง 5 แถวจาก DataFrame ชื่อ predictions)
- Use MulticlassClassificationEvaluator to create an evaluator where status_type ind is used as the label column and the prediction column is used as the prediction column
(ใช้ MulticlassClassificationEvaluator เพื่อสร้าง evaluator โดยใช้ status_type ind เป็น label และ prediction column เป็น prediction column)
- Show accuracy (evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})), precision (metricName: "weightedPrecision"), recall (metricName: "weightedRecall"), and F1 measure (metricName: "f1")
(แสดงค่า accuracy, precision (metricName: "weightedPrecision"), recall (metricName: "weightedRecall") และ F1 measure (metricName: "f1"))
```

DecistionTree Classification Code

```
1  from pyspark.sql import SparkSession
2  from pyspark.ml.feature import StringIndexer, VectorAssembler, OneHotEncoder
3  from pyspark.ml.classification import DecisionTreeClassifier
4  from pyspark.ml.evaluation import MulticlassClassificationEvaluator
5  from pyspark.ml import Pipeline
6  |
7  spark = SparkSession.builder.appName("DecisionTree_Classification").getOrCreate()
8
9  data = spark.read.csv("fb_live_thailand.csv", header=True, inferSchema=True)
10 # โหลดข้อมูลจากไฟล์ CSV เป็น DataFrame ให้สามารถเข้ามือถือได้
11
12 status_type_indexer = StringIndexer(inputCol="status_type", outputCol="status_type_ind")
13 # สร้าง StringIndexer เพื่อแปลงอักษร 'status_type' เป็นตัวเลข 'status_type_ind'
14
15 status_published_indexer = StringIndexer(inputCol="status_published", outputCol="status_published_ind")
16 # สร้าง StringIndexer เพื่อแปลงอักษร 'status_published' เป็นตัวเลข 'status_published_ind'
17
18 status_type_encoder = OneHotEncoder(inputCol="status_type_ind", outputCol="status_type_encoded")
19 # สร้าง OneHotEncoder เพื่อแปลงตัวเลข 'status_type_ind' เป็นค่า Boolean ในอักษร 'status_type_encoded'
20
21 status_published_encoder = OneHotEncoder(inputCol="status_published_ind", outputCol="status_published_encoded")
22 # สร้าง OneHotEncoder เพื่อแปลงตัวเลข 'status_published_ind' เป็นค่า Boolean ในอักษร 'status_published_encoded'
23
24 assembler = VectorAssembler(inputCols=["status_type_encoded", "status_published_encoded"], outputCol="features")
25 # รวมฟีดอินที่ซึ่งกันและกันไว้ในเวกเตอร์ในอักษร 'features'
26
27 pipeline = Pipeline(stages=[status_type_indexer, status_published_indexer, status_type_encoder, status_published_encoder, assembler])
28 # สร้าง pipeline ที่รวมทุกขั้นตอนที่ต้องใช้ในรูปแบบลำดับขั้นตอน
29
30 pipeline_model = pipeline.fit(data)
31 # ฝึกหัดโมเดล pipeline เพื่อสร้างโมเดล pipeline
32
33 transformed_data = pipeline_model.transform(data)
34 # ใช้ pipeline model เพื่อแปลงข้อมูลและสร้าง DataFrame ใหม่ที่มีฟีดอินใหม่
35
36 train_data, test_data = transformed_data.randomSplit([0.8, 0.2])
37 # แบ่งข้อมูลที่แปลงแล้วออกเป็นชุดการฝึก (80%) และชุดทดสอบ (20%)
38
39 decision_tree = DecisionTreeClassifier(labelCol="status_type_ind", featuresCol="features")
40 # สร้างโมเดล Decision Tree โดยใช้ 'status_type_ind' เป็นคอลัมน์เป้าหมายและ 'features' เป็นฟีดอิน
41
42 decision_tree_model = decision_tree.fit(train_data)
43 # ฝึกโมเดลด้วยชุดข้อมูลการฝึก
44
45 predictions = decision_tree_model.transform(test_data)
46 # ใช้โมเดลที่ฝึกแล้วเพื่อทำการท่านนายชุดข้อมูลทดสอบ
47
48 evaluator = MulticlassClassificationEvaluator(labelCol="status_type_ind", predictionCol="prediction")
49 # สร้าง evaluator สำหรับประเมินผลลัพธ์จากการท่านนายโดยใช้คอลัมน์เป้าหมายและคอลัมน์ที่การทำนาย
50
51 accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})
52 # ประเมินความแม่นยำของโมเดล
53
54 precision = evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"})
55 # ประเมินความแม่นยำเชิงลับพันธ์ของโมเดล
56
57 recall = evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"})
58 # ประเมินการเรียกคืนของโมเดล
59
60 f1 = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
61 # ประเมินค่า F1 ของโมเดล
62
63 test_error = 1.0 - accuracy
64 # ค่าความผิด Test Error โดยการลบความแม่นยำจาก 1
```

```
65  print()
66  print('='*50)
67  print()
68
69  print(f"Accuracy: {accuracy}")
70  # แสดงผลความแม่นยำ
71
72  print(f"Precision: {precision}")
73  # แสดงผลความแม่นยำเชิงล้มเหลว
74
75  print(f"Recall: {recall}")
76  # แสดงผลการเรียกคืน
77
78  print(f"F1 Measure: {f1}")
79  # แสดงผลค่า F1
80
81  print(f"Test Error: {test_error}")
82  # แสดงผลค่า Test Error
83  print()
84  print('='*50)
85  print()
86
87  spark.stop()
```

ผลลัพธ์

```
=====
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Measure: 1.0
Test Error: 0.0
=====
```

ໂຈນຍໍ DecistionTree Classification

```
- Create SparkSession
- Import library SparkSession
- Import StringIndexer, VectorAssembler, and OneHotEncoder from pyspark.ml.feature
- Import DecisionTreeClassifier from pyspark.ml.classification
- Import MulticlassClassificationEvaluator from pyspark.ml.evaluation
- Import Pipeline from pyspark.ml

- Load data file into DataFrame (FBLiveTH) (ໂລດໄຟຣ໌ຂອ້ມູນເຊັ່ງ DataFrame ສ່ວນ FBLiveTH)
- Use StringIndexer to prepare data where the columns status_type and status_published are used as inputs to create indexes (status_type_ind and status_published_ind)
(ໃໝ່ StringIndexer ເພື່ອເຕີມຂໍ້ມູນໄດ້ໃຫ້ຄວາມນິ້ນ status_type ແລະ status_published ເປັນອິນພຸດເພື່ອສ້າງຕົ້ນນີ້ (status_type_ind ແລະ status_published_ind)
- Use OneHotEncoder to create Boolean flag of status_type_ind and status_published_ind as they will be used as a component of vector in the next steps
(ໃໝ່ OneHotEncoder ເພື່ອສ້າງ Boolean flag ຂອງ status_type_ind ແລະ status_published_ind ເນື່ອຈາກຈະຖຸກໃຫ້ເປັນສ່ານປະກອບຂອງເວກເຕີຣີໃນຂັ້ນຕອນຕົ້ນໄປ)
- Use VectorAssembler to create vector of encoded status_type_ind and status_published_ind resulting in the column features
(ໃໝ່ VectorAssembler ເພື່ອສ້າງເງິນເຕີຣີຂອງ status_type_ind ແລະ status_published_ind ທີ່ຖຸກເຫັນທັສ ສັງເກດໃຫ້ໄດ້ຄວາມນິ້ນຂອງ features)
- Create pipeline where the stages include output from StringIndexer, encoder, and vector assembler
(ສ້າງ Pipeline ໂດຍມີ stages ສໍາຮັບເຖິງຜົນລົງທຶນຈາກ StringIndexer, encoder, ແລະ vector assembler)
```

```
- Fit DataFrame into the created pipeline to create the pipeline model
(ໃໝ່ DataFrame ປັບໃຫ້ເຂົ້າກັບ Pipeline ທີ່ສ້າງຂຶ້ນເພື່ອສ້າງ Pipeline model)
- Use the created pipeline model to transform the DataFrame data resulting in another DataFrame
(ໃໝ່ Pipeline model ທີ່ສ້າງຂຶ້ນໃນການແປງຂໍ້ມູນ DataFrame ເພື່ອໃຫ້ໄດ້ DataFrame ໄທ)
- Create train and test datasets using randomSplit function where the output from the previous step is used as an input
(ສ້າງຂຶ້ນຂໍ້ມູນ train ແລະ test ໂດຍໃຫ້ຝຶກຕົ້ນ randomSplit ໂດຍໃຫ້ຜົນລົງທຶນຕ້ອນໜ້າເປັນອິນພຸດ)
- Create decision tree classification where the status_type_ind is used as a label and the output is features
(ສ້າງການຈຳແກກປະເທດແບບ Decision Tree ໂດຍໃຫ້ status_type_ind ເປັນ label ແລະ features ເປັນອິນພຸດ)
- Fit train data into the created decision tree to create the model
(ໃຫ້ຂໍ້ມູນ train ປັບໃຫ້ເຂົ້າກັບ Decision Tree ທີ່ສ້າງຂຶ້ນເພື່ອສ້າງໂມເດລ)
- Use the created model to transform the test data resulting in a prediction DataFrame
(ໃຫ້ໂມເດລທີ່ສ້າງຂຶ້ນໃນການແປງຂໍ້ມູນ test ເພື່ອໃຫ້ໄດ້ DataFrame ການທ່ານາຍົດ)
- Use MulticlassClassificationEvaluator to create the evaluator where the status_type_ind is used as the label column and prediction column is used as the prediction column
(ໃໝ່ MulticlassClassificationEvaluator ໃນການສ້າງຕົ້ນປະເມີນ ໂດຍໃຫ້ status_type_ind ເປັນຄວາມນິ້ນ label ແລະ ຄວາມນິ້ນ prediction ເປັນຄວາມນິ້ນການທ່ານາຍ)
```



```
- Show accuracy, precision, recall (metricName: "weightedRecall"), and F1 measure (metricName: "f1")
(ແສດງຄ່າ accuracy, precision, recall (metricName: "weightedRecall") ແລະ F1 measure (metricName: "f1"))
- Also show the Test Error where it is calculated as 1.0 - accuracy
(ແສດງ Test Error ສິ້ນຄ່າວານຈາກ 1.0 - accuracy)
```

Dataset fb_live_thailand.csv

A	B	C	D	E	F	G	H	I	J	K	L
status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_haha	num_sads	num_angrys
24667554	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0
24667554	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0
24667554	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0
24667554	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0
24667554	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0
24667554	photo	4/18/2018 2:14	217	6	0	211	5	1	0	0	0
24667554	video	4/18/2018 0:24	503	614	72	418	70	10	2	0	3
24667554	video	4/17/2018 7:42	295	453	53	260	32	1	1	0	1
24667554	photo	4/17/2018 3:33	203	1	0	198	5	0	0	0	0
24667554	photo	4/11/2018 4:53	170	9	1	167	3	0	0	0	0
24667554	photo	4/10/2018 1:01	210	2	3	202	7	1	0	0	0
24667554	photo	4/9/2018 2:06	222	4	0	213	5	4	0	0	0

Time Series

Time Series Code - 1

```

1 import pandas as pd # สำหรับการจัดการข้อมูล
2 import numpy as np # สำหรับการทำงานกับอาเรย์และคณิตศาสตร์
3 import matplotlib.pyplot as plt # สำหรับการสร้างกราฟ
4 from pmdarima import auto_arima # สำหรับการสร้างโมเดล ARIMA อัตโนมัติ
5 from pmdarima.arima import ADFTest # สำหรับการตรวจสอบความเป็นสถานะ
6
7 # Step 3: Load your dataset from the specified path
8 df = pd.read_csv('year_sales.csv') # โหลดข้อมูลจากไฟล์ CSV
9
10 # Step 4: Convert the 'Year' column to datetime and set it as the index
11 df['Year'] = pd.to_datetime(df['Year']) # แปลงคอลัมน์ Year เป็น datetime
12 df.set_index('Year', inplace=True) # ตั้งคอลัมน์ Year เป็น index ของ DataFrame
13 # inplace=False Default DataFrame ในกรณีที่สร้างขึ้นโดยมี Year เป็น index
14 # inplace=True จะถูกเปลี่ยนแปลงให้ Year เป็น index ของมันทันที (ไม่มีการสร้าง DataFrame ใหม่)
15
16 # Step 5: Plot the data to see the time series
17 df.plot() # แสดงกราฟของข้อมูลที่อุดมแนวโน้ม
18 plt.title("Original Time Series") # ตั้งชื่อกราฟ
19 plt.show() # แสดงกราฟ
20
21 # Step 6: Perform the ADF test to check for stationarity
22 adf_test = ADFTest(alpha=0.05) # สร้างอject ADFTest โดยใช้ระดับความเชื่อมั่นที่ 0.05
23 adf_result = adf_test.should_diff(df['Sales']) # ตรวจสอบว่าต้องทำการ differencing หรือไม่
24 print(f"ADF Test Result: {adf_result}") # แสดงผลการทดสอบ ADF
25
26 # Step 7: Split the data into training and testing sets
27 train_size = int(len(df) * 0.8) # กำหนดขนาดของชุดข้อมูลฝึกหัด 80% ของข้อมูลทั้งหมด
28 train = df[:train_size] # สร้างชุดข้อมูลฝึกหัด train จะประกอบด้วยแคนท์ 0 ถึง 79 (80 แคน)
29 test = df[train_size:] # สร้างชุดข้อมูลทดสอบ test จะประกอบด้วยแคนท์ 80 ถึง 99 (20 แคน)
30
31 print(train)
32 print(test)

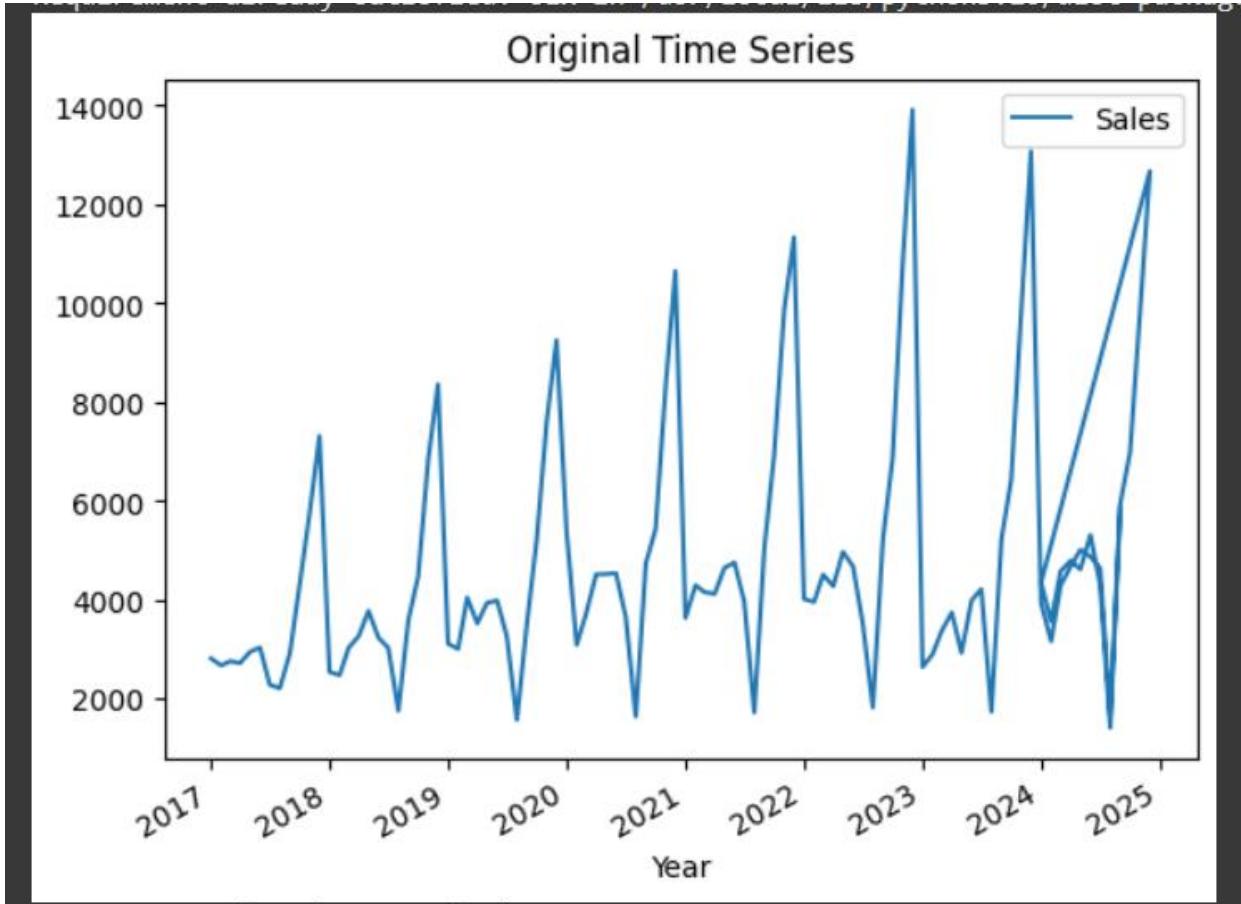
```

```

33
34 # Step 8: Fit the ARIMA model using auto_arima
35 model = auto_arima(train, start_p=0, d=1,
36                     start_q=0, max_p=5, max_d=5, max_q=5,
37                     start_P=0, D=1, start_Q=0, max_P=5,
38                     max_D=5, max_Q=5, m=12,
39                     seasonal=True, error_action='warn',
40                     trace=True, suppress_warnings=True,
41                     stepwise=True, random_state=20,
42                     n_fits=50) # สร้างโมเดล ARIMA โดยใช้ auto_arima พัฒนาโดยที่ระบุ
43
44 # สรุป model
45 print(model.summary())
46
47
48 # Step 9: Forecast future values
49 n_periods = len(test) # จำนวนช่วงเวลาที่จะพยากรณ์เท่ากับขนาดของชุดข้อมูลทดสอบ
50 forecast = model.predict(n_periods=n_periods) # ทำการพยากรณ์ค่าต่อไป
51
52 # Step 10: Convert forecast into a DataFrame and assign dates from the test set
53 forecast_df = pd.DataFrame(forecast, index=test.index, columns=['Predicted'])
54 # columns=['Predicted'] สร้าง DataFrame สำหรับผลการพยากรณ์
55
56 # Step 11: Plot the training, test, and predicted data
57 plt.figure(figsize=(10, 6)) # กำหนดขนาดของกราฟ
58 plt.plot(train, label='Train Data') # แสดงชุดข้อมูลฝึก
59 plt.plot(test, label='Test Data') # แสดงชุดข้อมูลทดสอบ
60 plt.plot(forecast_df, label='Predicted Data') # แสดงผลการพยากรณ์
61 plt.legend(loc='best') # แสดงในตำแหน่งที่ดีที่สุด
62 plt.title('Train, Test, and Predicted Data') # ตั้งชื่อกราฟ
63 plt.show() # แสดงกราฟ

```

ผลลัพธ์



Year	Sales
2017-01-01	2815
2017-02-01	2672
2017-03-01	2755
2017-04-01	2721
2017-05-01	2946
...	...
2023-08-01	1738
2023-09-01	5221
2023-10-01	6424
2023-11-01	9842
2023-12-01	13076

Performing stepwise search to minimize aic

```

ARIMA(0,1,0)(0,1,0)[12] : AIC=1183.693, Time=0.04 sec
ARIMA(1,1,0)(1,1,0)[12] : AIC=1173.736, Time=0.17 sec
ARIMA(0,1,1)(0,1,1)[12] : AIC=1157.042, Time=0.33 sec
ARIMA(0,1,1)(0,1,0)[12] : AIC=1155.109, Time=0.09 sec
ARIMA(0,1,1)(1,1,0)[12] : AIC=1157.009, Time=0.29 sec
ARIMA(0,1,1)(1,1,1)[12] : AIC=1158.348, Time=0.68 sec
ARIMA(1,1,1)(0,1,0)[12] : AIC=1155.379, Time=0.16 sec
ARIMA(0,1,2)(0,1,0)[12] : AIC=1155.138, Time=0.16 sec
ARIMA(1,1,0)(0,1,0)[12] : AIC=1173.612, Time=0.04 sec
ARIMA(1,1,2)(0,1,0)[12] : AIC=1155.790, Time=0.25 sec
ARIMA(0,1,1)(0,1,0)[12] intercept : AIC=inf, Time=0.13 sec

```

Best model: ARIMA(0,1,1)(0,1,0)[12]
Total fit time: 2.369 seconds

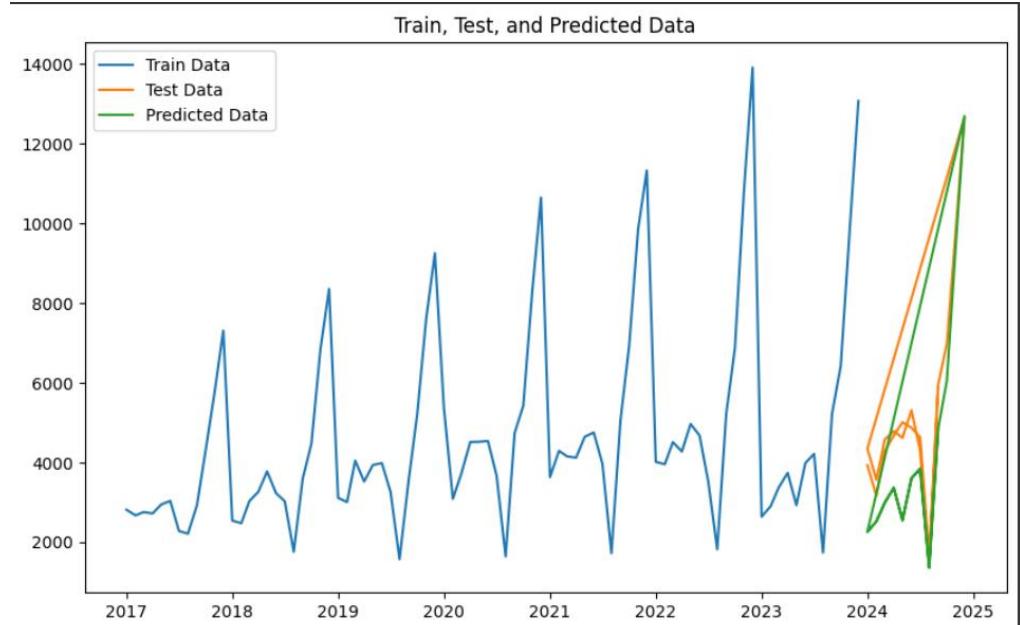
[84 rows x 1 columns]

Year	Sales
2024-01-01	3934
2024-02-01	3162
2024-03-01	4286
2024-04-01	4676
2024-05-01	5010
2024-06-01	4874
2024-07-01	4633
2024-08-01	1659
2024-09-01	5951
2024-10-01	6981
2024-11-01	9851
2024-12-01	12670
2024-01-01	4348
2024-02-01	3564
2024-03-01	4577
2024-04-01	4788
2024-05-01	4618

SARIMAX Results

```

=====
Dep. Variable:                      y      No. Observations:                 84
Model:                SARIMAX(0, 1, 1)x(0, 1, [], 12)   Log Likelihood:            -575.554
Date:                  Sun, 27 Oct 2024    AIC:                         1155.109
Time:                      12:17:13           BIC:                         1159.634
Sample:                 01-01-2017    HQIC:                        1156.908
                           - 12-01-2023
Covariance Type:             opg
=====
            coef    std err      z   P>|z|      [0.025      0.975]
-----
ma.L1     -0.8756     0.060   -14.663    0.000    -0.993     -0.759
sigma2    5.86e+05  7.15e+04     8.195    0.000    4.46e+05    7.26e+05
=====
Ljung-Box (L1) (Q):                   0.16  Jarque-Bera (JB):          6.95
Prob(Q):                            0.69  Prob(JB):              0.03
Heteroskedasticity (H):               2.13  Skew:                  0.01
Prob(H) (two-sided):                 0.07  Kurtosis:              4.53
=====
```



Time Series Code – 2

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from pmdarima.arima import auto_arima
from pmdarima.arima import ADFTest

# Load the dataset
df = pd.read_csv('year_sales.csv')

# Convert 'Year' to datetime format and set as index
df['Year'] = pd.to_datetime(df['Year'])
df.set_index('Year', inplace=True)

# Sort the DataFrame by Year to ensure proper slicing
df = df.sort_index()

# Plot the entire dataset
df.plot()
plt.title("Yearly Sales Data")
plt.show()

# Perform the ADF Test to check stationarity
adf_test = ADFTest(alpha=0.05)
result = adf_test.should_diff(df)
print(f"ADF Test result: {result}")

# Split data into train (up to 2020) and test (from 2021 onwards)
train = df[:'2020']
test = df['2021':]
```

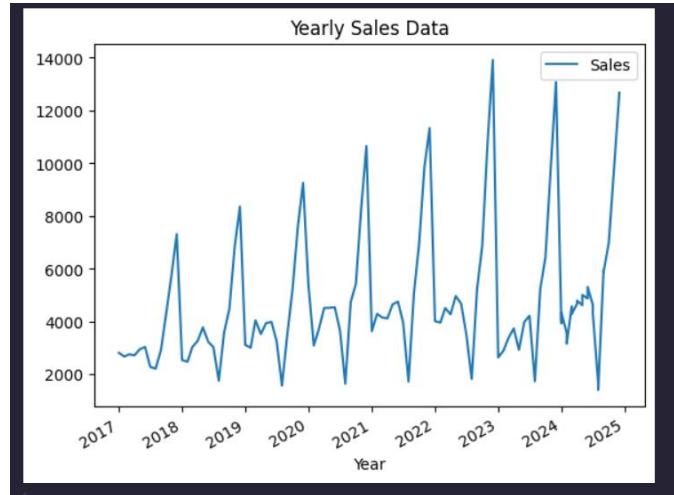
```
# Train the ARIMA model on the train data
model = auto_arima(train, start_p=0, d=1, start_q=0,
                    max_p=5, max_d=5, max_q=5,
                    start_P=0, D=1, start_Q=0, max_P=5, max_D=5, max_Q=5,
                    m=12, seasonal=True, error_action='warn',
                    trace=True, suppress_warnings=True, stepwise=True,
                    random_state=20, n_fits=50)

# Print model summary
print(model.summary())

# Predict the values for the test set
predictions = pd.DataFrame(model.predict(n_periods=len(test)),
                             index=test.index)
predictions.columns = ['Predicted']

# Plot the train, test, and predicted data
plt.figure(figsize=(10, 6))
plt.plot(train, label='Train', color='blue')
plt.plot(test, label='Test', color='orange')
plt.plot(predictions, label='Predicted', color='green')
plt.title("Train, Test, and Predicted Sales Data")
plt.xlabel('Year')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

ນາລັກວິທະຍາ

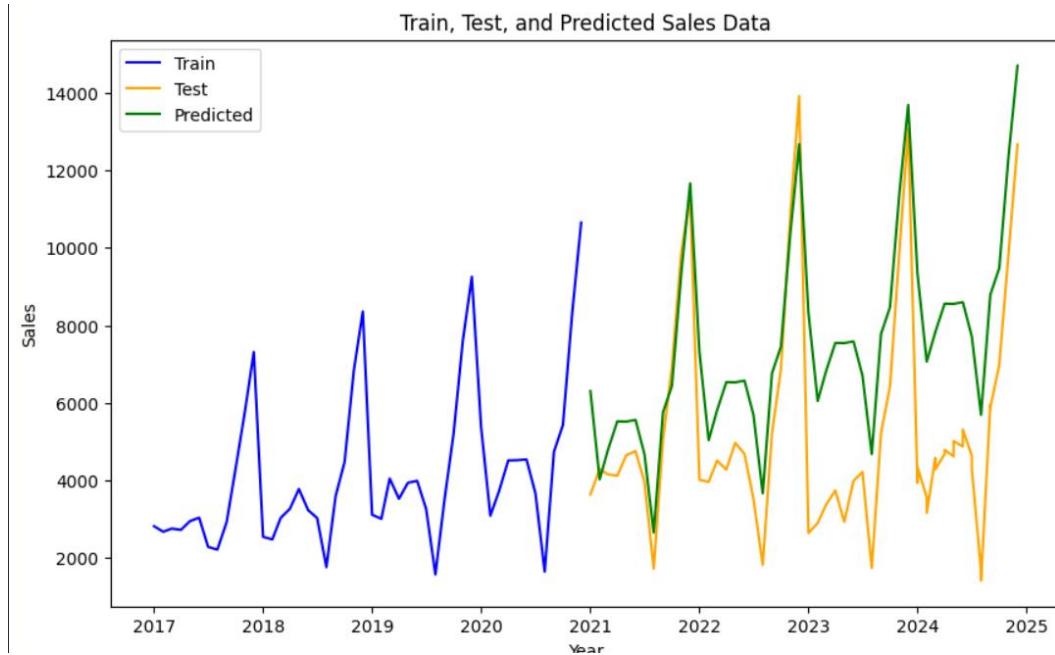


```
ADF Test result: (0.01, False)
Performing stepwise search to minimize aic
ARIMA(0,1,0)(0,1,0)[12] : AIC=563.896, Time=0.02 sec
ARIMA(1,1,0)(1,1,0)[12] : AIC=562.739, Time=0.12 sec
ARIMA(0,1,1)(0,1,1)[12] : AIC=inf, Time=0.13 sec
ARIMA(1,1,0)(0,1,0)[12] : AIC=560.756, Time=0.03 sec
ARIMA(1,1,0)(0,1,1)[12] : AIC=562.740, Time=0.05 sec
ARIMA(1,1,0)(1,1,1)[12] : AIC=inf, Time=0.22 sec
ARIMA(2,1,0)(0,1,0)[12] : AIC=557.899, Time=0.04 sec
ARIMA(2,1,0)(1,1,0)[12] : AIC=559.654, Time=0.08 sec
ARIMA(2,1,0)(0,1,1)[12] : AIC=559.648, Time=0.17 sec
ARIMA(2,1,0)(1,1,1)[12] : AIC=inf, Time=0.44 sec
ARIMA(3,1,0)(0,1,0)[12] : AIC=559.097, Time=0.04 sec
ARIMA(2,1,1)(0,1,0)[12] : AIC=inf, Time=0.13 sec
ARIMA(1,1,1)(0,1,0)[12] : AIC=inf, Time=0.09 sec
ARIMA(3,1,1)(0,1,0)[12] : AIC=inf, Time=0.16 sec
ARIMA(2,1,0)(0,1,0)[12] intercept : AIC=559.805, Time=0.11 sec

Best model: ARIMA(2,1,0)(0,1,0)[12]
Total fit time: 1.842 seconds
```

SARIMAX Results

```
=====
Dep. Variable:                      y      No. Observations:                 48
Model:                SARIMAX(2, 1, 0)x(0, 1, 0, 12)   Log Likelihood:            -275.949
Date:          Mon, 07 Oct 2024        AIC:                         557.899
...
=====
```



ໂຄງ

```
Import libraries:  
● pandas  
● auto_arima and ADFTest from pmdarima.arima (ນໍາເຊົາ auto_arima ແລະ ADFTest ຈາກ pmdarima.arima)  
● matplotlib.pyplot (ນໍາເຊົາ matplotlib.pyplot)  
  
1. Import data (year_sales.csv) (ນໍາຫາຂໍ້ອມລາຍກຳໄຟສ year_sales.csv)  
2. Use to_datetime() function to convert Year to datetime  
(ໃໝ່ຝຶກຮັບ to_datetime() ເພື່ອແປລ Year ເປັນ datetime)  
3. Use set_index() function as <yourdataframe>.set_index('Year', inplace=True)  
(ໃໝ່ຝຶກຮັບ set_index() ໂດຍເຫັນແປນ <yourdataframe>.set_index('Year', inplace=True)) Show plot (ແສດງການ)  
4. Set adf_test = ADFTest(alpha = 0.05) (ຕັ້ງຄ່າ adf_test = ADFTest(alpha = 0.05))  
5. adf_test.should_diff(<your_dataframe>) (ເຮັດວຽກ adf_test.should_diff(<your_dataframe>))  
6. Create train dataset from <your_dataframe> (ສ້າງชຸດຂໍ້ອມລັກິກ (train) ຈາກ <your_dataframe>)  
7. Create test dataset from <your_dataframe> (ສ້າງໜຸດຂໍ້ອມລັດສອບ (test) ຈາກ <your_dataframe>)  
  
8. Create ARIMA model using auto_arima function (ສ້າງໂມເດລ ARIMA ໂດຍໃໝ່ຝຶກຮັບ auto_arima) (For parameter setting details, please find from here) (ຮາຍລະເວີຍດການຕັ້ງຄ່າພາຣາມີເຕີເອງ ARIMA)  
ARIMA parameter setting example: (ຕ້ອມງານຕັ້ງຄ່າພາຣາມີເຕີເອງ ARIMA)  
auto_arima(<your train dataset>, start_p=0, d=1, start_q=0, max_p=5, max_d=5, max_q=5, start_P=0,  
D=1, start_Q=0, max_P=5, max_D=5, max_Q=5, m=12, seasonal=True, error_action='warn', trace=True,  
suppress_warnings=True, stepwise=True, random_state=20, n_fits=50) (auto_arima(<your train dataset>,  
start_p=0, d=1, start_q=0, max_p=5, max_d=5, max_q=5, start_P=0, D=1, start_Q=0, max_P=5, max_D=5,  
max_Q=5, m=12, seasonal=True, error_action='warn', trace=True, suppress_warnings=True,  
stepwise=True, random_state=20, n_fits=50))  
9. Use summary() function to summarise the ARIMA model (ໃໝ່ຝຶກຮັບ summary() ເພື່ອສຸບປະໂຫຼດ ARIMA)  
10. Create prediction: <your prediction> = pd.DataFrame(<your ARIMA model>.predict(n_periods=<number>), index=<your test dataset>.index)  
(ສ້າງການທຳນາຍ: <your prediction> = pd.DataFrame(<your ARIMA model>.predict(n_periods=<number>),  
index=<your test dataset>.index)) <your prediction>.columns = ['<predicted>'] (<your  
prediction>.columns = ['<predicted>'])  
11. Create and show the plot that includes train, test, and predicted data.  
(ສ້າງແລະແສດງການທີ່ຮ່ວມຂໍ້ອມລັກິກ (train), ຂໍ້ອມລັດສອບ (test) ແລະຂໍ້ອມທີ່ທ່ານຍິວ (predicted))
```

Dataset

Year	Sales
2017-01	2815
2017-02	2672
2017-03	2755
2017-04	2721
2017-05	2946
2017-06	3036
2017-07	2282
2017-08	2212

Clustering

Clustering Code - 1

```
Clustering > 🐍 TEST-Clustering-2.py > ...
 1  from pyspark.sql import SparkSession
 2  from pyspark.sql.types import DoubleType
 3  from pyspark.ml.feature import VectorAssembler, StandardScaler
 4  from pyspark.ml import Pipeline
 5  from pyspark.ml.clustering import KMeans
 6  from pyspark.ml.evaluation import ClusteringEvaluator
 7  import matplotlib.pyplot as plt
 8
 9  # Start Spark session
10 spark = SparkSession.builder.appName("testClustering").getOrCreate()
11
12 # Load and preprocess data
13 df = spark.read.csv("fb_live_thailand.csv", header=True, inferSchema=True)
14 df = df.select(df.num_sads.cast(DoubleType()), df.num_reactions.cast(DoubleType()))
15
16 # Set up feature vector
17 vecAssembler = VectorAssembler(inputCols=["num_sads", "num_reactions"], outputCol="features")
18 scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=True, withMean=False)
19
20 # Find the optimal k using silhouette score
21 k_values = []
22 for i in range(2, 6):
23     kmeans = KMeans(featuresCol="scaledFeatures", predictionCol="prediction", k=i)
24     pipeline = Pipeline(stages=[vecAssembler, scaler, kmeans])
25     model = pipeline.fit(df)
26     output = model.transform(df)
27     evaluator = ClusteringEvaluator(predictionCol="prediction", featuresCol="scaledFeatures",
28                                     metricName="silhouette", distanceMeasure="squaredEuclidean")
29
30     score = evaluator.evaluate(output)
31     k_values.append(score)
32     print('Silhouette Score for k =', i, ':', score)
33
34 # Adjust `best_k` to actual k value
35 best_k = k_values.index(max(k_values)) + 2
36 print("The best k is:", best_k, "with a Silhouette Score of:", max(k_values))
37
38 # Initialize KMeans with the best k
39 kmeans = KMeans(featuresCol="scaledFeatures", predictionCol="prediction", k=best_k)
40 pipeline = Pipeline(stages=[vecAssembler, scaler, kmeans])
41
42 # Fit model
43 model = pipeline.fit(df)
44
45 # Predictions
46 predictions = model.transform(df)
47
48 # Convert predictions to pandas for plotting
49 clustered_data_pd = predictions.toPandas()
50
51 # Plot each cluster
52 plt.figure(figsize=(10, 6))
53 for cluster in clustered_data_pd["prediction"].unique():
54     clustered_subset = clustered_data_pd[clustered_data_pd["prediction"] == cluster]
55     plt.scatter(clustered_subset["num_reactions"], clustered_subset["num_sads"], label=f"Cluster {cluster}")
56
57 plt.xlabel("num_reactions")
58 plt.ylabel("num_sads")
59 plt.title("K-Means Clustering")
60 plt.legend(title="Clusters")
61 plt.show()
62
63 # Count the number of points in each cluster
64 cluster_counts = clustered_data_pd["prediction"].value_counts().sort_index()
65
66 # Plotting the bar chart for cluster sizes
67 plt.figure(figsize=(8, 6))
68 bars = plt.bar(cluster_counts.index, cluster_counts.values, color='skyblue')
69 plt.xlabel("Cluster")
70 plt.ylabel("Count")
71 plt.title("Number of Points in Each Cluster")
72 plt.xticks(cluster_counts.index)
73
```

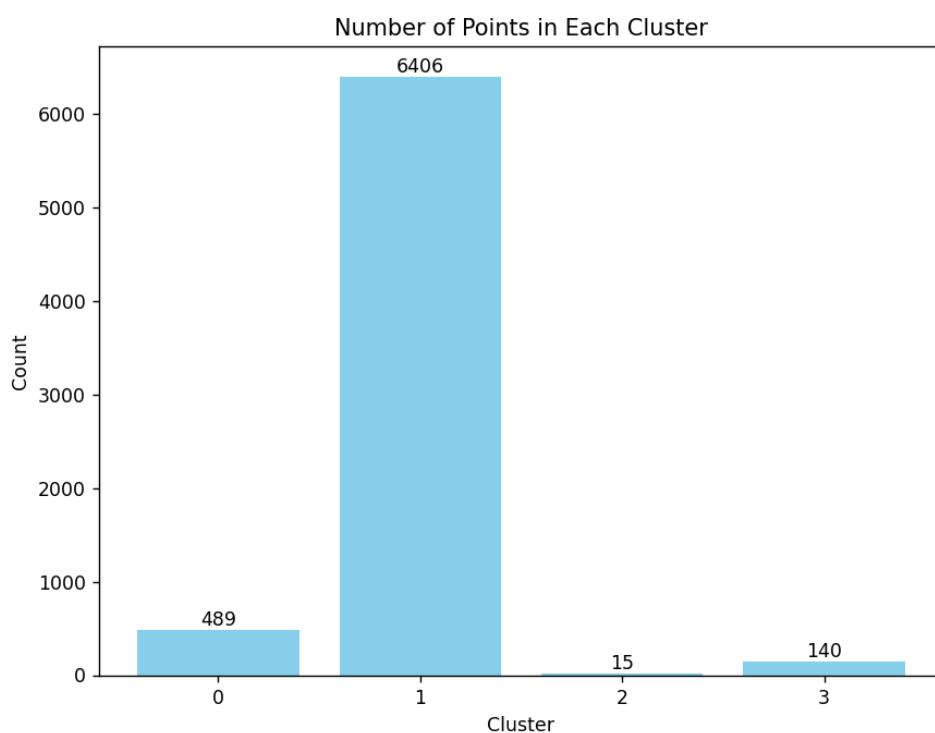
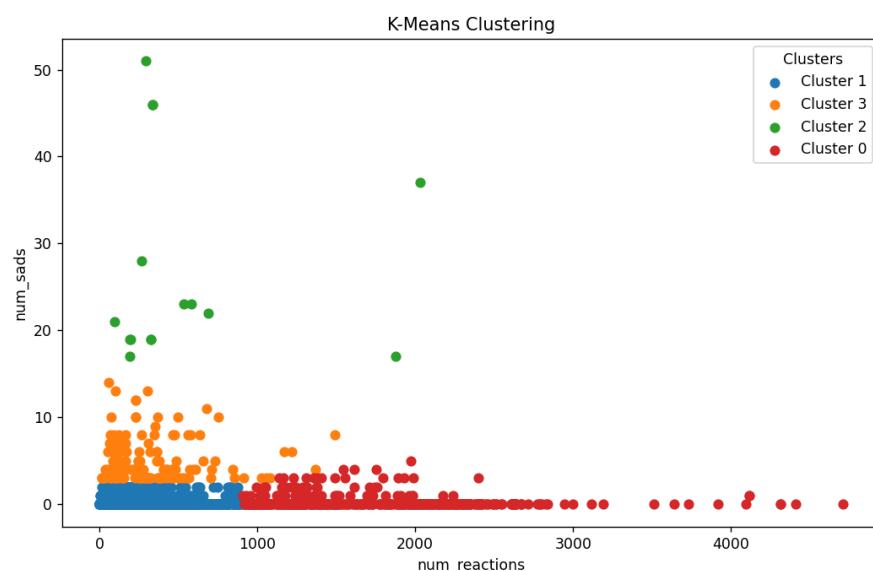
```

74 # Add count labels on top of each bar
75 for bar in bars:
76     yval = bar.get_height()
77     plt.text(bar.get_x() + bar.get_width() / 2, yval, int(yval), ha='center', va='bottom')
78
79 plt.show()
80

```

ผลลัพธ์

Silhouette Score for k = 2 : 0.8870485911324794
 Silhouette Score for k = 3 : 0.9201883344331407
 Silhouette Score for k = 4 : 0.9281101063954209
 Silhouette Score for k = 5 : 0.8774149540457314
 The best k is: 4 with a Silhouette Score of: 0.9281101063954209



Clustering Code – 2 MAIN

```
1  from pyspark.sql import SparkSession
2  from pyspark.ml.feature import VectorAssembler, StandardScaler
3  from pyspark.ml.clustering import KMeans
4  from pyspark.ml import Pipeline
5  from pyspark.ml.evaluation import ClusteringEvaluator
6  import matplotlib.pyplot as plt
7  from pyspark.sql.types import *
8  import numpy as np
9
10 spark = SparkSession.builder \
11     .appName("KMeans Clustering") \
12     .getOrCreate()
13
14 df = spark.read.csv("fb_live_thailand.csv", \
15     header=True, inferSchema=True)
16 df = df.select(df.num_sads.cast(DoubleType()), df.num_reactions.cast(DoubleType()))
17
18 vecAssembler = VectorAssembler(inputCols=["num_sads", "num_reactions"], outputCol="features")
19
20 # Scaling for making columns comparable
21 scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=True, withMean=False)
22
23 # Initialize k values list
24 k_values = []
25
26 # Loop for finding the optimal k in range 2 to 5
27 for i in range(2, 5):
28     kmeans = KMeans(featuresCol="scaledFeatures", predictionCol="prediction_col", k=i)
29     pipeline = Pipeline(stages=[vecAssembler, scaler, kmeans])
30     model = pipeline.fit(df)
31     output = model.transform(df)
32     evaluator = ClusteringEvaluator(predictionCol="prediction_col", \
33                                     featuresCol="scaledFeatures", \
34                                     metricName="silhouette", \
35                                     distanceMeasure="squaredEuclidean")
36     score = evaluator.evaluate(output)
37     k_values.append(score)
38     print("Silhouette Score for k =", i, ":", score)
39
40 # Get the best k
41 best_k = k_values.index(max(k_values)) + 2
42 print("The best k:", best_k, "with Silhouette Score:", max(k_values))
43
44 # Initialize KMeans with the best k
45 kmeans = KMeans(featuresCol="scaledFeatures", predictionCol="prediction_col", k=best_k)
46
47 # Create pipeline
48 pipeline = Pipeline(stages=[vecAssembler, scaler, kmeans])
49
50 # Fit the model
51 model = pipeline.fit(df)
52
53 # Prediction
54 predictions = model.transform(df)
```

```

55 # Evaluate
56 evaluator = ClusteringEvaluator(predictionCol="prediction_col", \
57                                 featuresCol="scaledFeatures", \
58                                 metricName="silhouette", \
59                                 distanceMeasure="squaredEuclidean")
60 silhouette = evaluator.evaluate(predictions)
61 print("Silhouette with squared euclidean distance =", str(silhouette))
62
63 # Converting to Pandas DataFrame
64 clustered_data_pd = predictions.toPandas()
65
66 # Visualizing the results (Scatter Plot)
67 plt.scatter(clustered_data_pd["num_reactions"], \
68             clustered_data_pd["num_sads"], \
69             c=clustered_data_pd["prediction_col"])
70 plt.xlabel("num_reactions")
71 plt.ylabel("num_sads")
72 plt.title("K-means Clustering")
73 plt.colorbar().set_label("Cluster")
74 plt.show()
75
76 # Visualizing the cluster count (Bar Chart)
77 unique, counts = np.unique(clustered_data_pd["prediction_col"], return_counts=True)
78 plt.bar(unique, counts, color='skyblue')
79 plt.xlabel("Cluster ID")
80 plt.ylabel("Number of Data Points")
81 plt.title("Number of Data Points per Cluster")
82
83
84 # Adding text on top of bars
85 for i, count in zip(unique, counts):
86     plt.text(i, count + 0.5, str(count), ha='center')
87
88 plt.show()
89

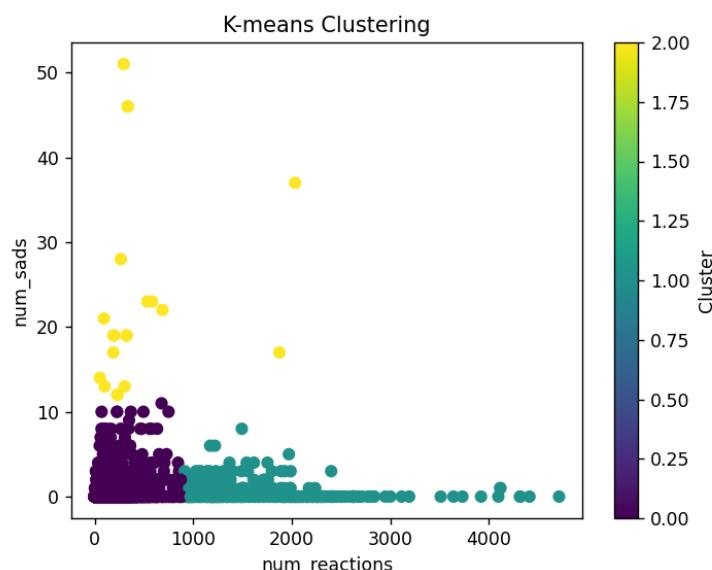
```

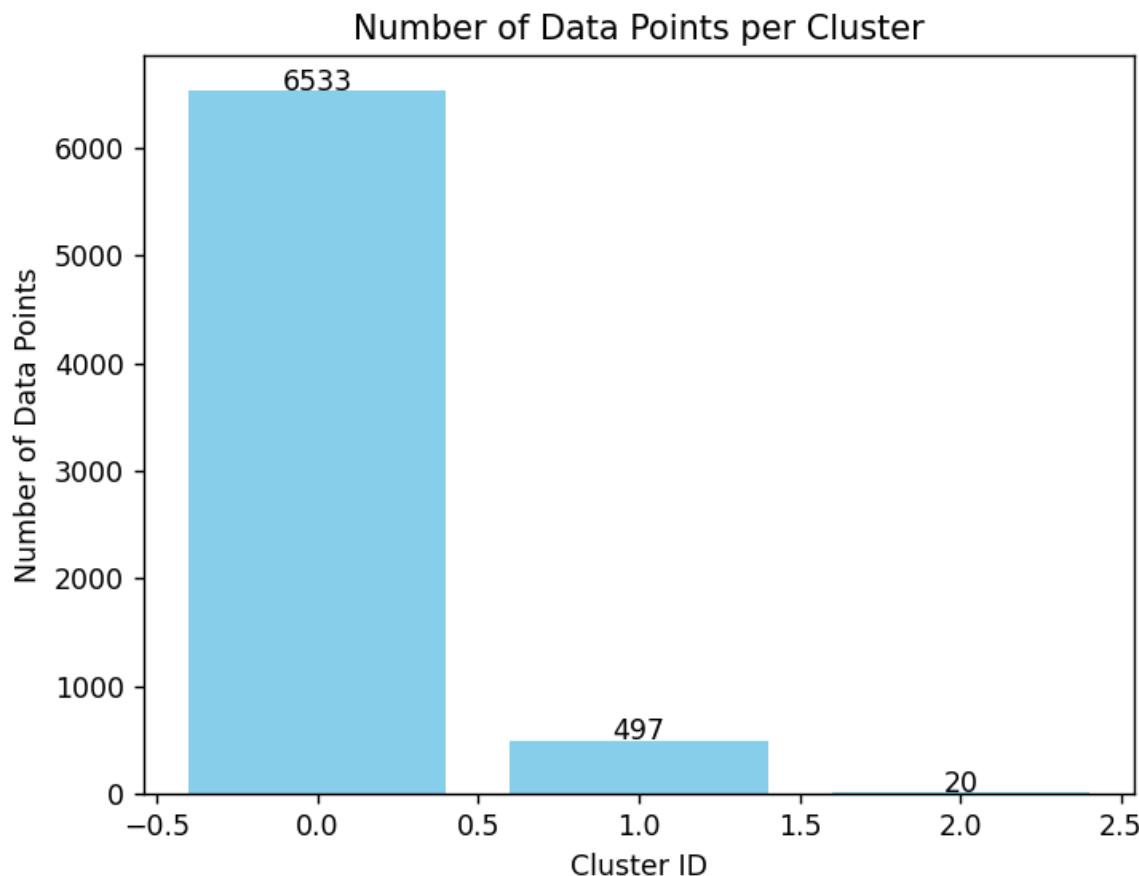
ผลลัพธ์

```

Silhouette Score for k = 2 : 0.8870485911324794
Silhouette Score for k = 3 : 0.9222692618451699
Silhouette Score for k = 4 : 0.9166494988012941
The best k: 3 with Silhouette Score: 0.9222692618451699
Silhouette with squared euclidean distance = 0.9222692618451699

```





เพิ่มเติม การ Save output รูปภาพ

```

plt.savefig("scatter_plot.png")
67 # Visualizing the results (Scatter Plot)
68 plt.scatter(clustered_data_pd["num_reactions"], \
69 | | | clustered_data_pd["num_sads"], \
70 | | | c=clustered_data_pd["prediction_col"])
71 plt.xlabel("num_reactions")
72 plt.ylabel("num_sads")
73 plt.title("K-means Clustering")
74 plt.colorbar().set_label("Cluster")
75
76
77
78 plt.show()

```

```

plt.savefig("bar_chart.png")
87  # Adding text on top of bars
88  for i, count in zip(unique, counts):
89      plt.text(i, count + 0.5, str(count), ha='center')
90
91 plt.show()
92

```

เปลี่ยนสี Scatter plot

```

68  colors = ['red', 'green', 'blue', 'orange', 'purple']
69  plt.scatter(clustered_data_pd["num_reactions"], \
70             clustered_data_pd["num_sads"], \
71             c=clustered_data_pd["prediction_col"].apply(lambda x: colors[x % len(colors)]))

```

เปลี่ยนสี Bar Chart

```

81  bar_colors = [colors[i % len(colors)] for i in unique]
82  plt.bar(unique, counts, color=bar_colors)
83  plt.xlabel("Cluster ID")
84  plt.ylabel("Number of Data Points")
85  plt.title("Number of Data Points per Cluster")
86

```

ตัวอย่างการประเมิน K-Mean

- การวัดค่า Silhouette ใช้ในการวัดคุณภาพของกลุ่มข้อมูล
- บันทึกแสดงให้เห็นว่าข้อมูลอยู่ห่างไกลกลุ่มมาก
- ค่าการวัดอยู่ในช่วง [-1, 1]:
 - 1 หมายถึงข้อมูลอาจถูกจัดให้อยู่ในกลุ่มที่ผิด
 - 0 หมายถึงข้อมูลในกลุ่มอยู่ใกล้กันมาก
 - 1 หมายถึงข้อมูลในกลุ่มอยู่ห่างจากกันมาก

Dataset fb_live_thailand.csv

A	B	C	D	E	F	G	H	I	J	K	L
status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_haha	num_sads	num_angrys
24667554	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0
24667554	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0
24667554	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0
24667554	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0
24667554	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0
24667554	photo	4/18/2018 2:14	217	6	0	211	5	1	0	0	0
24667554	video	4/18/2018 0:24	503	614	72	418	70	10	2	0	3
24667554	video	4/17/2018 7:42	295	453	53	260	32	1	1	0	1
24667554	photo	4/17/2018 3:33	203	1	0	198	5	0	0	0	0
24667554	photo	4/11/2018 4:53	170	9	1	167	3	0	0	0	0
24667554	photo	4/10/2018 1:01	210	2	3	202	7	1	0	0	0
24667554	photo	4/9/2018 2:06	222	4	0	213	5	4	0	0	0