


โค้ด PySpark นี้ใช้สำหรับวิเคราะห์ข้อความและทำนายค่าเรตติ้งของรีวิวโดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ประเภทการถดถอยลอจิสติก (Logistic Regression) ผ่านการประมวลผลข้อความ และมีการประเมินผลความแม่นยำ (accuracy) ของโมเดล ด้านล่างนี้เป็นคำอธิบายทีละส่วนของโค้ด

## 1. การสร้าง SparkSession

python


 Copy code

```
spark = SparkSession.builder.appName("TextAnalytics").getOrCreate()
```

- SparkSession เป็นจุดเริ่มต้นในการทำงานกับ PySpark ซึ่งทำหน้าที่เหมือนเป็น "เอนจิน" สำหรับการประมวลผลข้อมูล

## 2. การอ่านข้อมูลจากไฟล์ CSV

python

 Copy code

```
data = spark.read.csv("reviewsRated.csv", header=True, inferSchema=True)
```

- อ่านข้อมูลจากไฟล์ CSV ที่ชื่อว่า `reviewsRated.csv` ซึ่งมีคอลัมน์สำหรับข้อความรีวิวและเรตติ้ง
- `header=True` หมายถึงไฟล์มีหัวคอลัมน์
- `inferSchema=True` คือให้ Spark สร้างโครงสร้างข้อมูลให้โดยอัตโนมัติตามประเภทของข้อมูลที่เจอในแต่ละคอลัมน์



### 3. การเลือกคอลัมน์ที่ใช้ และแปลงข้อมูล

```
python Copy code

data = data.select(data["Review Text"].alias("review_text"), data["Rating"].cast(IntegerType))
data = data.na.drop() # ลบข้อมูลที่เป็นค่าว่าง
data.show(5)
```

- เลือกคอลัมน์ "Review Text" และ "Rating" โดยเปลี่ยนชื่อคอลัมน์ "Review Text" เป็น `review_text` และแปลง `Rating` ให้เป็นประเภทจำนวนเต็ม (`IntegerType`)
- `data.na.drop()` จะทำการลบแถวที่มีค่าว่างออก
- `data.show(5)` จะแสดงข้อมูล 5 แถวแรก

### 4. การเตรียมข้อมูล (Tokenization, Stopwords Removal, HashingTF)

- **Tokenizer:** แบ่งข้อความรีวิวออกเป็นคำ

```
python Copy code

tokenizer = Tokenizer(inputCol="review_text", outputCol="words")
```

- **StopWordsRemover:** ลบคำทั่วไปที่ไม่จำเป็น (stopwords) เช่น "is", "the", "and" เป็นต้น

```
python Copy code

stopword_remover = StopWordsRemover(inputCol="words", outputCol="meaningful_words")
```

- **HashingTF:** แปลงคำให้เป็นฟีเจอร์โดยใช้ความถี่ของคำ (Term Frequency)

```
python Copy code

hashing_tf = HashingTF(inputCol="meaningful_words", outputCol="features")
```

### 5. สร้าง Pipeline สำหรับการประมวลผลข้อมูล


```
python Copy code

pipeline = Pipeline(stages=[tokenizer, stopword_remover, hashing_tf])
```

- สร้าง Pipeline ที่จะนำแต่ละขั้นตอนการแปลงข้อมูล (Tokenization, Stopword Removal, และ HashingTF) มาทำงานต่อเนื่องกันแบบอัตโนมัติ

## 6. แบ่งข้อมูลออกเป็นชุดฝึกและชุดทดสอบ

python


 Copy code

```
train_data, test_data = data.randomSplit([0.8, 0.2], seed=1234)
```

- แบ่งข้อมูลออกเป็นชุดฝึก (80%) และชุดทดสอบ (20%) โดยมีการตั้งค่า `seed` เพื่อความแน่นอนในการสุ่มข้อมูล

## 7. ฝึกโมเดลและแปลงข้อมูล

python


 Copy code

```
pipeline_model = pipeline.fit(train_data)
train_transformed = pipeline_model.transform(train_data)
test_transformed = pipeline_model.transform(test_data)
```

- นำข้อมูลชุดฝึกเข้าไปฝึก Pipeline และแปลงข้อมูลทั้งชุดฝึกและชุดทดสอบให้กลายเป็นฟีเจอร์

## 8. การสร้างและฝึก Logistic Regression

python


 Copy code

```
log_reg = LogisticRegression(labelCol="rating", featuresCol="features")
log_reg_model = log_reg.fit(train_transformed)
```

- Logistic Regression ถูกนำมาใช้สำหรับทำนายค่าเรตติ้ง โดยใช้ฟีเจอร์ที่ได้จากการแปลงข้อมูล

## 9. การทำนายและการแสดงผลลัพธ์

python

 Copy code

```
predictions = log_reg_model.transform(test_transformed)
predictions.select("meaningful_words", "rating", "prediction").show(5)
```

- ทำนายค่าจากชุดทดสอบ และแสดงผลค่าที่ถูกแปลงแล้ว (`meaningful_words`), เรตติ้งจริง (`rating`), และค่าที่ทำนายได้ (`prediction`)

## 10. การประเมินผลโมเดล

python

Copy code

```
evaluator = MulticlassClassificationEvaluator(labelCol="rating", predictionCol="prediction")
accuracy = evaluator.evaluate(predictions)
print(f"Accuracy: {accuracy}")
```

- ประเมินความแม่นยำของโมเดล ( `accuracy` ) โดยเปรียบเทียบค่าที่ทำนายได้กับค่าเรตติ้งจริง

### สรุป

โค้ดนี้เริ่มจากการเตรียมข้อมูลรีวิว โดยแบ่งคำออกจากข้อความ (Tokenization) ลบคำที่ไม่จำเป็นออก (Stopwords Removal) และแปลงคำเหล่านั้นให้กลายเป็นพีเจอร์ที่ใช้ในการเรียนรู้ (HashingTF) จากนั้นใช้โมเดล Logistic Regression เพื่อทำนายเรตติ้งของรีวิว และประเมินผลด้วยความแม่นยำ (Accuracy)

- ผลลัพธ์ที่ได้จากโค้ด:
- แสดงข้อมูลรีวิวและเรตติ้งที่ถูกเตรียมไว้
- แสดงคำที่ถูกแบ่งและพีเจอร์ที่ถูกสร้างจากคำเหล่านั้น
- แสดงผลการทำนายเรตติ้งจากโมเดล Logistic Regression
- แสดงค่าความแม่นยำของโมเดล

## Graph

โค้ดนี้ทำการประมวลผลข้อมูลเส้นทางการบิน (airline routes) โดยใช้ PySpark และ GraphFrames ซึ่งเป็นแพ็คเกจสำหรับการวิเคราะห์ข้อมูลกราฟ (Graph Analytics) บน Spark ข้อความอธิบายทีละส่วนมีดังนี้:

### 1. สร้าง SparkSession

```
python Copy code  
  
spark = SparkSession.builder \  
    .appName("AirlineRoutesGraph") \  
    .config("spark.jars.packages", "graphframes:graphframes:0.8.2-spark3.0-s_2.12") \  
    .getOrCreate()
```

- SparkSession: เป็นจุดเริ่มต้นในการทำงานกับ PySpark ในการประมวลผลข้อมูล
- GraphFrames: โค้ดนี้ใช้ GraphFrames (แพ็คเกจสำหรับการวิเคราะห์กราฟใน Spark) ดังนั้น `spark.jars.packages` ถูกตั้งค่าเพื่อนำเข้าชุดคำสั่งของ GraphFrames

### 2. การอ่านข้อมูลจากไฟล์ CSV

```
python Copy code  
  
airline_routes_df = spark.read.csv("C:/Users/sooke/Downloads/BigData/BigData/GraphAnalytic
```

- อ่านข้อมูลจากไฟล์ `airline_routes.csv` โดยตั้งค่า `header=True` เพื่อบอกว่าไฟล์มีหัวคอลัมน์ และ `inferSchema=True` เพื่อให้ Spark คาดเดาประเภทของข้อมูลในแต่ละคอลัมน์อัตโนมัติ


### 3. สร้าง DataFrame สำหรับ vertices (จุดยอด)

```
python Copy code  
  
vertices = airline_routes_df.select("source_airport").withColumnRenamed("source_airport",
```

- เลือกเฉพาะคอลัมน์ `source_airport` แล้วเปลี่ยนชื่อคอลัมน์เป็น `id` เพื่อใช้เป็น vertices (จุดยอด) สำหรับกราฟ
- `distinct()` ใช้เพื่อลบแถวที่ซ้ำกันออกจาก DataFrame

#### 4. สร้าง DataFrame สำหรับ edges (เส้นเชื่อม)

python


 Copy code

```
edges = airline_routes_df.select("source_airport", "destination_airport") \
    .withColumnRenamed("source_airport", "src") \
    .withColumnRenamed("destination_airport", "dst")
```

- เลือกคอลัมน์ `source_airport` และ `destination_airport` และเปลี่ยนชื่อเป็น `src` และ `dst` เพื่อใช้เป็น edges (เส้นเชื่อม) ในกราฟ
- `src` คือสนามบินต้นทาง และ `dst` คือสนามบินปลายทาง

#### 5. สร้าง GraphFrame

python

 Copy code

```
graph = GraphFrame(vertices, edges)
```

- GraphFrame ถูกสร้างขึ้นโดยใช้ `vertices` และ `edges` ซึ่งเป็นการสร้างกราฟจากข้อมูลสนามบินต้นทางและปลายทางในเส้นทางการบิน

## 6. แสดงจำนวน vertices และ edges

python

Copy code

```
print("Number of vertices:", graph.vertices.count())
print("Number of edges:", graph.edges.count())
```

- แสดงจำนวน vertices (จุดยอด) และ edges (เส้นเชื่อม) ที่มีอยู่ในกราฟ ซึ่งจะแสดงจำนวนสนามบินและจำนวนเส้นทางการบินทั้งหมด

## 7. กลุ่ม edges และการกรองข้อมูล

python

Copy code

```
grouped_edges = graph.edges.groupBy("src", "dst").count() \
    .filter(col("count") > 5) \
    .orderBy(desc("count")) \
    .withColumn("source_color", lit("#3358FF")) \
    .withColumn("destination_color", lit("#FF3F33"))
```

- กลุ่ม edges ตามต้นทาง (src) และปลายทาง (dst) แล้วนับจำนวนเส้นทางการบินที่เชื่อมต่อระหว่างสนามบินนั้นๆ โดยใช้ `groupBy()`
- `filter(col("count") > 5)` กรองเฉพาะเส้นทางที่มีจำนวนมากกว่า 5 เส้นทาง (หมายถึงเส้นทางที่บินมากกว่า 5 ครั้ง)
- ใช้ `withColumn()` เพื่อเพิ่มคอลัมน์ `source_color` และ `destination_color` โดยกำหนดค่าสีของต้นทางและปลายทางตามที่ระบุ

## 8. การบันทึกข้อมูลที่จัดกลุ่มลงในไฟล์ CSV

python

Copy code

```
grouped_edges.write.csv("grouped_airline_routes.csv", mode="overwrite", header=True)
```

- บันทึกผลลัพธ์ของ edges ที่ถูกจัดกลุ่มและกรองข้อมูลลงในไฟล์ CSV ชื่อ `grouped_airline_routes.csv` โดยตั้งค่า `mode="overwrite"` เพื่อให้สามารถเขียนทับไฟล์เดิมได้ และ `header=True` เพื่อบันทึกหัวข้อคอลัมน์

## สรุป

- โค้ดนี้สร้างกราฟจากข้อมูลเส้นทางการบินโดยใช้ GraphFrames โดยแต่ละสนามบินถูกมองเป็น `vertices` (จุดยอด) และแต่ละเส้นทางการบินเป็น `edges` (เส้นเชื่อม)
- มีการกรองเฉพาะเส้นทางการบินที่มีการเชื่อมต่อมากกว่า 5 ครั้ง และบันทึกข้อมูลเหล่านี้ลงในไฟล์ CSV