actividad2

May 1, 2025

0.1 # Proyecto | lectura, escritura, archivos de Big Data PySpark - Implementación

MAESTRÍA EN INTELIGENCIA ARTIFICIAL APLICADA

Curso: TC4034.10 - Análisis de grandes volúmenes de datos

Tecnológico de Monterrey

- Dr. Iván Olmos Pineda
- Mtra. Verónica Sandra Guzmán de Valle
- Mtro. Alberto Daniel Salinas Montemayor

Proyecto lectura, escritura, archivos de Big Data PySpark

Equipo 37

NOMBRE COMPLETO	MATRÍCULA
Alejandro Díaz Villagómez	A01276769
Alonso Pedrero Martínez	A01769076
César Iván Pedrero Martínez	A01366501
Emiliano Saucedo Arriola	A01659258

0.2 ## Descripción del tema de interés

El presente proyecto tiene como objetivo principal analizar las reseñas de productos publicadas en Amazon utilizando herramientas de Big Data, específicamente PySpark. Amazon, como empresa de reconocimiento mundial tanto en el ámbito del comercio electrónico como en el desarrollo de soluciones tecnológicas, destacando su plataforma AWS (Amazon Web Services), constituye un referente global en el sector. Además, su posicionamiento como líder indiscutible del mercado de marketplaces la convierte en una fuente valiosa de datos masivos para análisis de comportamiento del consumidor.

La selección de este tema responde a intereses tanto profesionales como personales. Como programadores y consultores de software, frecuentemente trabajamos con clientes que solicitan el desarrollo de plataformas de venta en línea, mostrando un creciente interés en conocer la percepción y aceptación de sus productos en el mercado digital. En este contexto, resulta fundamental contar con herramientas que permitan identificar tendencias, analizar sentimientos y extraer patrones ocultos en las opiniones de los usuarios.

Por tanto, el objetivo general de este proyecto es construir una solución que, a partir del procesamiento y análisis de grandes volúmenes de reseñas, permita:

- Identificar el sentimiento general asociado a los productos (positivo, negativo o neutral).
- Detectar tendencias y patrones recurrentes en las opiniones de los consumidores.
- Apoyar en la toma de decisiones estratégicas de marketing y mejora de productos, proporcionando información valiosa y procesable a partir de los datos analizados.

Este enfoque no solo fortalece nuestras competencias técnicas en Big Data (e incluso en temas como procesamiento de lenguaje natural - NLP), sino que también aporta una herramienta práctica y directamente aplicable a nuestro entorno laboral actual.

Algunas aplicaciones potenciales son:

- **Mejora de Productos**: Analizar las reseñas puede ayudar a identificar problemas recurrentes o aspectos destacados de los productos, ayudando a los fabricantes a mejorar los diseños o la experiencia del cliente.
- **Atención al Cliente y Soporte**: Identificar rápidamente las reseñas negativas permite a las empresas abordar las preocupaciones de los clientes de manera eficiente, mejorando la satisfacción del cliente.
- **Sistemas de Recomendación**: El conjunto de datos puede usarse para mejorar los sistemas de recomendación, correlacionando las calificaciones de los productos, los sentimientos y las preferencias de los clientes.

0.3 ## Selección del dataset

0.3.1 Selección

Campo	Detalle
Nombre del dataset	Amazon-reviews
Enlace	Amazon-reviews en Kaggle
Publicador	Machha Ravi Kiran
Última actualización	Hace aproximadamente 2 años
Formato - Tamaño	CSV - 3.68 GB
Fuente	Repositorio Kaggle (acceso público)

0.3.2 Atributos del Conjunto de Datos

Columna	Tipo de dato	Descripción
marketplace	string	Mercado donde se realizó la compra
customer_id	integer	Identificador único del cliente
review_id	string	Identificador único de la reseña
product_id	string	Identificador del producto
product_parent	integer	Identificador de grupo o “padre” para productos similares o relacionados
product_title	string	Nombre del producto
product_category	string	Categoría del producto

Columna	Tipo de dato	Descripción
star_rating	integer	Calificación otorgada por el cliente (1-5)
helpful_votes	integer	Número de votos útiles recibidos por la reseña
total_votes	integer	Número total de votos recibidos
vine	string	Participación en el programa Vine de reseñas (Y para sí, N para no)
verified_purchase	string	Indicador de compra verificada (Y para sí, N para no)
review_headline	string	Encabezado de la reseña
review_body	string	Cuerpo del texto de la reseña
review_date	date	Fecha en que se realizó la reseña
sentiment	integer	Valor de sentimiento asignado a la reseña (1 para positivo, 0 para negativo)

0.3.3 Descripción General

Este conjunto de datos contiene reseñas de productos publicadas en Amazon, con un enfoque principal en productos electrónicos. La información es detallada, incluyendo las calificaciones de los usuarios, comentarios, votos útiles, y otros aspectos que permiten realizar un análisis profundo del comportamiento del consumidor y el sentimiento de las reseñas.

El dataset seleccionado cumple con los criterios establecidos para este proyecto, ya que está directamente relacionado con el análisis de reseñas de productos, presenta un tamaño adecuado para técnicas de Big Data, y su formato tabular favorece su procesamiento mediante PySpark. Además, la procedencia del dataset, proveniente de una fuente reconocida como Kaggle, garantiza además su calidad y fiabilidad para los fines académicos y profesionales de este trabajo.

0.4 ## Análisis con PySpark

0.4.1 Librerías

> Note that to run this notebook, you'll need to install some dependencies. It's HIGHLY suggested to run this command to actually install it in a virtual environment by following this command:

```
python3 -m venv venv && source venv/bin/activate && pip install findspark pyspark
matplotlib seaborn setuptools kagglehub
```

[52]:
```
import findspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, isnan, when, count, lit
from pyspark.sql.types import StringType, IntegerType, LongType, DoubleType,␣
 ↪FloatType, DecimalType, ShortType, NumericType

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

```
from IPython.display import display
```

0.4.2 Inicializar Pyspark

[53]:
```
findspark.init()
findspark.find()
```

[53]:
```
'/Users/emisaar/anaconda3/envs/pyspark_env/lib/python3.10/site-packages/pyspark'
```

[54]:
```
spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.sparkContext.setLogLevel("ERROR")
spark.conf.set("spark.sql.repl.eagerEval.enabled", True)

spark
```

[54]:
```
<pyspark.sql.session.SparkSession at 0x15a536170>
```

0.4.3 Cargar Dataset

[55]:
```
import kagglehub

# Download latest version
FILE_PATH = kagglehub.dataset_download("machharavikiran/amazon-reviews")

print("Path to dataset files:", FILE_PATH)
```

```
Path to dataset files:
/Users/emisaar/.cache/kagglehub/datasets/machharavikiran/amazon-
reviews/versions/1
```

0.4.4 Definición de clases

[56]:
```
class FileManager():
    @staticmethod
    def open_csv_file(file_path: str):
        """
        This method opens a csv file with pyspark
        """
        csv_df = spark.read.csv(
            file_path,
            header=True,
            inferSchema=True,
            multiLine=True,
            escape="\"",
            quote="\""
        )
```

```
        # csv_df.show(truncate=20)

        return csv_df
```

[57]:
```
class StatisticalAnalysisHelper():
    @staticmethod
    def dataset_dimensions(df_input):
        print("columns in the dataset:", len(df_input.columns))
        print("rows in the dataset:", df_input.count())

    @staticmethod
    def schema_information(df_input):
        """
        This method shows the current schema of the data.
        """
        df_input.printSchema()

    @staticmethod
    def descriptive_statistics(df_input, variable_type="numeric"):
        """
        Shows descriptive statistics for the dataset.

        Parameters:
        - df_input: Spark DataFrame
        - variable_type: 'numeric' to show numeric columns, 'non-numeric' to␣
↪show others
        """
        numeric_cols = [field.name for field in df_input.schema.fields if␣
↪isinstance(field.dataType, NumericType)]
        non_numeric_cols = [field.name for field in df_input.schema.fields if␣
↪field.name not in numeric_cols]

        if variable_type == "numeric":
            print("\n=== Numeric Variables ===")
            if numeric_cols:
                display(df_input.select(numeric_cols).summary().toPandas())
            else:
                print("No numeric columns found.")

        elif variable_type == "non-numeric":
            print("\n=== Non-numeric Variables ===")
            if non_numeric_cols:
                for col_name in non_numeric_cols:
                    print(f"Columna: {col_name}")
                    print("--" * 20)
                    display(df_input.groupBy(col(col_name)).count().
↪orderBy("count", ascending=False).limit(5).toPandas())
```

```
            else:
                print("No non-numeric columns found.")

        else:
            print(f"Invalid variable_type: {variable_type}. Choose 'numeric' or␣
↪'non-numeric'.")

    @staticmethod
    def missing_values_table(df_input):
        """
        Displays a table with the count of missing values per column.
        """
        missing_exprs = []

        for c in df_input.schema.fields:
            field_name = c.name
            field_type = c.dataType

            if isinstance(field_type, (IntegerType, LongType, DoubleType,␣
↪FloatType, DecimalType, ShortType, NumericType)):
                missing_exprs.append(
                    count(when(col(field_name).isNull() |␣
↪isnan(col(field_name)), field_name)).alias(field_name)
                )
            elif isinstance(field_type, StringType):
                missing_exprs.append(
                    count(when(col(field_name).isNull() | (col(field_name) ==␣
↪""), field_name)).alias(field_name)
                )
            else:
                missing_exprs.append(
                    count(when(col(field_name).isNull(), field_name)).
↪alias(field_name)
                )

        df_missing_values = df_input.select(missing_exprs)

        return df_missing_values
```

[58]:
```
class VisualAnalysisHelper:
    @staticmethod
    def missing_values_plot(df_missing_values):
        """
        This method shows the missing values and plots them.
        """
        df_missing_values_pd = df_missing_values.toPandas()
```

```
        df_missing_values_pd = df_missing_values_pd.melt(var_name="column",␣
↪value_name="missing_count")

        plt.figure(figsize=(10,6))
        sns.barplot(x="column", y="missing_count", data=df_missing_values_pd)
        plt.xticks(rotation=45, ha='right')
        plt.title("Missing Values per Column")
        plt.tight_layout()
        plt.show()

    @staticmethod
    def column_distribution_plot(df_input, column_name, x_label, y_label):
        """
        This method plots the distribution of the 'star_rating' column.
        """

        rating_counts = (
            df_input.groupBy(column_name)
            .count()
            .orderBy(column_name)
        )

        plot_data = rating_counts.toPandas()

        plt.figure(figsize=(8, 5))
        sns.barplot(x=column_name, y="count", data=plot_data)
        plt.xlabel(x_label)
        plt.ylabel(y_label)
        plt.title("Distribution of Values")
        plt.xticks(rotation=0)
        plt.tight_layout()
        plt.show()
```

0.4.5 Exploratory Data Analysis (EDA)

[59]:
```
df_reviews = FileManager.open_csv_file(FILE_PATH)
df_reviews.limit(10).toPandas()
```

[59]:

	marketplace	customer_id	review_id	product_id	product_parent	\
0	US	22873041	R3ARRMDEGED8RD	B00KJWQIIC	335625766	
1	US	30088427	RQ28TSA020Y6J	B013ALA9LA	671157305	
2	US	20329786	RUXJRZCT6953M	B00PML2GQ8	982036237	
3	US	14215710	R7EO0UO6BPB71	B001NS00Z4	576587596	
4	US	38264512	R39NJY2YJ1JFSV	B00AQMTND2	964759214	
5	US	30548466	R31SR7REWNX7CF	B00KX4TORI	170101802	

```
6          US       589298   RVBP8I1R0CTZ8  B00P17WEMY       206124740
7          US     49329488  R1QF6RS1PDLU18  B00TR05L9Y       778403103
8          US     50728290  R23AICGEDAJQL1  B0098Y77OG       177098042
9          US     37802374  R2EY3N4K9W19UP  B00IFYEYXC       602496520

                                      product_title product_category  \
0  Plemo 14-Inch Laptop Sleeve Case Waterproof Fa…               PC
1         TP-Link OnHub AC1900 Wireless Wi-Fi Router                 PC
2  AmazonBasics USB 3.0 A Male to A Male Cable - …               PC
3  Transcend P8 15-in-1 USB 2.0 Flash Memory Card…               PC
4  Aleratec SATA Data Cable 2.0 20in Serial ATA S…               PC
5  Kingston Digital MobileLite G4 USB 3.0 Multi-F…               PC
6  White 9 Inch Unlocked Dual Sim Card Phone Tabl…               PC
7  Lenovo TAB2 A10 - 10.1" 2-in-1 Tablet (1.5Ghz,…               PC
8                                             Acer                 PC
9  AzureWave Broadcom BCM94352HMB 802.11/ac/867Mb…               PC

   star_rating  helpful_votes  total_votes vine verified_purchase  \
0            5              0            0    N                Y
1            5             24           31    N                N
2            1              2            2    N                N
3            1              0            0    N                Y
4            5              0            0    N                Y
5            5              0            0    N                Y
6            3              1            2    N                Y
7            4              1            1    N                Y
8            1              0            0    N                Y
9            5              3            4    N                Y

                                  review_headline  \
0                              Pleasantly surprised
1  OnHub is a pretty no nonsense type router that…
2     None of them worked.  No functionality at all.
3                              just keep searching.
4                                        Five Stars
5               Good quality, works well and compact
6        in fact this is third China good. Demn s***
7                                             Good
8                          You get what you pay for
9                        Great for Windows 7 Laptop!

                                       review_body review_date  sentiment
0  I was very surprised at the high quality of th…  2015-08-31          1
1  I am a Google employee and had to chance to us…  2015-08-31          1
2  Bought cables in 3ft, 6ft and 9ft.  NONE of th…  2015-08-31          0
3                              nope, cheap and slow  2015-08-31          0
4           Excellent! Great value and does the job.  2015-08-31          1
```

```
5          Good quality,works well and compact size  2015-08-31          1
6  This demn tablet is just a Real Chinese produc…  2015-08-31          0
7  I am not sure I don't know if it is the tablet…  2015-08-31          1
8  After exactly 45 days, the screen went dark. P…  2015-08-31          0
9  Replaced my Intel Centrino 2230 with the BCM94…  2015-08-31          1
```

[60]:
```python
from pyspark.sql import functions as F

df_star = df_reviews

# Conteo por estrella
star_rating_counts = df_star.groupBy("star_rating").count().
 ↪orderBy("star_rating")

# Estadísticas de star_rating
star_rating_stats = df_star.agg(
    F.min("star_rating"), F.max("star_rating"),
    F.avg("star_rating"), F.stddev("star_rating")
)
```

[61]:
```python
df_with_ratio = df_star.withColumn(
    "helpful_ratio",
    F.when(F.col("total_votes") > 0, F.col("helpful_votes") / F.
 ↪col("total_votes")).otherwise(0)
)

helpful_ratio_stats = df_with_ratio.agg(
    F.min("helpful_ratio"), F.max("helpful_ratio"),
    F.avg("helpful_ratio"), F.stddev("helpful_ratio")
)
helpful_ratio_stats.show()
```

```
[Stage 3:>                                                          (0 + 1) / 1]
+------------------+------------------+------------------+--------------------
+
|min(helpful_ratio)|max(helpful_ratio)|
avg(helpful_ratio)|stddev(helpful_ratio)|
+------------------+------------------+------------------+--------------------
+
|               0.0|               1.0|0.23128070845636955|
0.3967627166047604|
+------------------+------------------+------------------+--------------------
+
```

[62]:
```
total_votes_stats = df_with_ratio.agg(
    F.min("total_votes"), F.max("total_votes"),
    F.avg("total_votes"), F.stddev("total_votes")
)
total_votes_stats.show()
```

```
[Stage 6:>                                                          (0 + 1) / 1]
```

```
+---------------+---------------+------------------+------------------+
|min(total_votes)|max(total_votes)|  avg(total_votes)|stddev(total_votes)|
+---------------+---------------+------------------+------------------+
|              0|          48362|1.9620770907212328|  43.07079218821245|
+---------------+---------------+------------------+------------------+
```

[63]:
```
sentiment_counts = df_with_ratio.groupBy("sentiment").count().
 ↪orderBy("sentiment")
sentiment_stats = df_with_ratio.agg(F.avg("sentiment").
 ↪alias("proporcion_positiva"))
sentiment_counts.show()
sentiment_stats.show()
```

```
+---------+-------+
|sentiment|  count|
+---------+-------+
|        0|1632669|
|        1|5273895|
+---------+-------+
```

```
[Stage 12:>                                                         (0 + 1) / 1]
```

```
+-------------------+
|proporcion_positiva|
+-------------------+
| 0.7636061868101128|
+-------------------+
```

[64]:
```
customer_top10 = df_with_ratio.groupBy("customer_id").count().orderBy(F.
 ↪desc("count")).limit(10)
customer_top10.show()
```

```
[Stage 17:>                                                         (0 + 8) / 8]
```

```
+-----------+-----+
|customer_id|count|
+-----------+-----+
|   17957446|  458|
|   44834233|  442|
|   52938899|  366|
|   45664110|  275|
|   49452274|  261|
|   50820654|  256|
|   12200139|  251|
|   45070473|  251|
|   32038204|  241|
|   49266466|  240|
+-----------+-----+
```

[65]:
```
vine_counts = df_with_ratio.groupBy("vine").count()
total_reviews = df_with_ratio.count()

vine_proportions = vine_counts.withColumn(
    "proportion", F.round(F.col("count") / F.lit(total_reviews), 4)
)
vine_proportions.show()
```

```
[Stage 21:>                                                         (0 + 1) / 1]
```

```
+----+-------+----------+
|vine|  count|proportion|
+----+-------+----------+
|   Y|  36228|    0.0052|
|   N|6870336|    0.9948|
+----+-------+----------+
```

[66]:
```
verified_counts = df_with_ratio.groupBy("verified_purchase").count()

verified_proportions = verified_counts.withColumn(
    "proportion", F.round(F.col("count") / F.lit(total_reviews), 4)
)
verified_proportions.show()
```

```
[Stage 24:>                                                         (0 + 1) / 1]
```

```
+-----------------+-------+----------+
|verified_purchase|  count|proportion|
+-----------------+-------+----------+
|                Y|6047075|    0.8756|
```

```
|                N| 859489|    0.1244|
+-----------------+-------+----------+
```

[67]:
```
df_with_year = df_with_ratio.withColumn("review_year", F.year("review_date"))
reviews_by_year = df_with_year.groupBy("review_year").count().
  ↪orderBy("review_year")
reviews_by_year.show()
```

```
[Stage 27:>                                                         (0 + 1) / 1]
+-----------+-------+
|review_year|  count|
+-----------+-------+
|       1999|    384|
|       2000|   3596|
|       2001|   6588|
|       2002|  10125|
|       2003|  13619|
|       2004|  14124|
|       2005|  18171|
|       2006|  26277|
|       2007|  59870|
|       2008|  81409|
|       2009| 129840|
|       2010| 213170|
|       2011| 397182|
|       2012| 661742|
|       2013|1396819|
|       2014|1996666|
|       2015|1876982|
+-----------+-------+
```

[68]:
```
from pyspark.sql.functions import col, count, round

# Total de registros para calcular proporciones
total = df_reviews.count()

# === Caracterización de product_category ===
carac_categoria = (
    df_reviews.groupBy("product_category")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("count").desc())
```

```
)
carac_categoria.show(truncate=False)

# === Caracterización de verified_purchase ===
carac_verificado = (
    df_reviews.groupBy("verified_purchase")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("count").desc())
)
carac_verificado.show()
```

```
+----------------+-------+----------+
|product_category|count  |proporcion|
+----------------+-------+----------+
|PC              |6906564|1.0       |
+----------------+-------+----------+
```

```
[Stage 36:>                                                         (0 + 1) / 1]
```

```
+-----------------+-------+----------+
|verified_purchase|  count|proporcion|
+-----------------+-------+----------+
|                Y|6047075|    0.8756|
|                N| 859489|    0.1244|
+-----------------+-------+----------+
```

[69]:
```
carac_marketplace = (
    df_reviews.groupBy("marketplace")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("marketplace"))
)
carac_marketplace.show()
```

```
[Stage 39:>                                                         (0 + 1) / 1]
```

```
+-----------+-------+----------+
|marketplace|  count|proporcion|
+-----------+-------+----------+
|         US|6906564|       1.0|
+-----------+-------+----------+
```

[70]:
```python
from pyspark.sql.functions import col, count, round

# Total de registros para calcular proporciones
total = df_reviews.count()

# === Caracterización de product_category ===
carac_star = (
    df_reviews.groupBy("star_rating")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("star_rating"))
)
carac_star.show(truncate=False)

# === Caracterización de sentiment ===
carac_sentiment = (
    df_reviews.groupBy("sentiment")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("sentiment"))
)
carac_sentiment.show()
```

```
+-----------+-------+----------+
|star_rating|count  |proporcion|
+-----------+-------+----------+
|1          |756857 |0.1096    |
|2          |362156 |0.0524    |
|3          |513656 |0.0744    |
|4          |1168208|0.1691    |
|5          |4105687|0.5945    |
+-----------+-------+----------+
```

```
[Stage 48:>                                                         (0 + 1) / 1]
```

```
+---------+-------+----------+
|sentiment|  count|proporcion|
+---------+-------+----------+
|        0|1632669|    0.2364|
|        1|5273895|    0.7636|
+---------+-------+----------+
```

[71]:
```python
from pyspark.sql.functions import col, count, round

# === Calcular total de registros ===
total = df_reviews.count()

# === Agrupar por combinaciones de star_rating y sentiment ===
combinaciones = (
    df_reviews.groupBy("star_rating", "sentiment")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("star_rating").asc(), col("sentiment").desc())
)

# === Mostrar combinaciones con probabilidad ===
combinaciones.show(truncate=False)
```

```
[Stage 54:>                                                         (0 + 1) / 1]
```

```
+-----------+---------+-------+----------+
|star_rating|sentiment|count  |proporcion|
+-----------+---------+-------+----------+
|1          |0        |756857 |0.1096    |
|2          |0        |362156 |0.0524    |
|3          |0        |513656 |0.0744    |
|4          |1        |1168208|0.1691    |
|5          |1        |4105687|0.5945    |
+-----------+---------+-------+----------+
```

entonces, combinar sentiment y star_rating no aporta mucha variedad para el particionamiento (porque una variable se infiere de la otra).

[72]:
```python
from pyspark.sql.functions import col, count, round

# === Calcular total de registros ===
total = df_reviews.count()

# === Agrupar por combinaciones de star_rating y verified_purchase ===
combinaciones_rating_verificado = (
    df_reviews.groupBy("star_rating", "verified_purchase")
    .count()
    .withColumn("proporcion", round(col("count") / total, 4))
    .orderBy(col("star_rating").asc(), col("verified_purchase").desc())
)

# === Mostrar resultados ===
combinaciones_rating_verificado.show(truncate=False)
```

```
[Stage 60:>                                                         (0 + 1) / 1]
```

star_rating	verified_purchase	count	proporcion
1	Y	603373	0.0874
1	N	153484	0.0222
2	Y	300549	0.0435
2	N	61607	0.0089
3	Y	443372	0.0642
3	N	70284	0.0102
4	Y	1019771	0.1477
4	N	148437	0.0215
5	Y	3680010	0.5328
5	N	425677	0.0616

[ ]:
```
spark.stop()
```

Las estadísticas descriptivas de las variables numéricas indican que el dataset está concentrado en calificaciones altas (`star_rating` promedio de 4.08) y en sentimientos mayoritariamente positivos (`sentiment` promedio de 0.76). La mayoría de los productos recibieron pocos votos útiles y totales.