Proyecto de evaluación

Autor: Juan Carlos González Ibarra

Correo Institucional: juan.gonzalez.dti@imfe.mx

Fecha: 15 de mayo, 2023

Contenido

1. Introducción

La tecnología está presente cada vez más en nuestras vidas, podemos notarlo de tal forma que nuestras actividades diarias están vinculadas a compartir información por medios digitales, sin duda esto hace que la información recolectada por estos medios digitales sea visualizada como una herramienta en la toma de decisiones y/o en la solución de problemas. Esta información para poder ser utilizada con estos propósitos previamente debe haber pasado por un proceso de análisis; este proceso de análisis ya tiene una base científica y aplicativa llamada ciencia de datos. La ciencia de datos conlleva una serie de etapas para el análisis de la información como: el entendimiento de los datos, la extracción de sus propiedades, el modelado y análisis del problema, la presentación de resultados y el desarrollo de software para explotar el conocimiento extraído. La **Ciencia de Datos** proporciona las herramientas y en ayuda del **Big Data** provee nuevas oportunidades tecnológicas para análisis masivos de información, por lo que las Tecnologías Big Data ayudan a mejorar la eficiencia, calidad y los productos y servicios personalizados ofrecidos por las organizaciones y a integrar datos estructurados y no estructurados con respuestas en tiempo real, abriendo nuevos caminos a la innovación y desarrollo.

Por lo que el **Big Data** se refiere al manejo y análisis de enormes volúmenes de datos que no pueden ser procesados de manera efectiva con las técnicas tradicionales debido a su volumen, velocidad y variedad. Los datos pueden provenir de varias fuentes, como transacciones comerciales, sensores de IoT, redes sociales, datos de salud, etc. Las técnicas de Big Data permiten descubrir patrones ocultos, correlaciones y otras ideas útiles que pueden informar la toma de decisiones empresariales y estratégicas.

Por lo que para realizar este análisis masivo de datos se debe identificar patrones que ayuden a la toma de decisiones tenemos que llevar acabo el **Análisis Exploratorio de Datos (AED)**.

1.1 Análisis Exploratorio de Datos

Se utiliza para analizar e investigar conjuntos de datos y resumir sus características principales, a menudo empleando métodos de visualización de datos que ayuda a entender la estructura de estos. Es un paso crucial antes de realizar modelos más complejos ya que ayuda a gestionar las fuentes de datos, descubrir patrones, detectar anomalías, probar una hipótesis, identificar relaciones y tendencias, etc.

El **AED** es un paso esencial en cualquier flujo de trabajo de **análisis de datos**, implica el uso de algoritmos de muestreo y agregación para reducir la escala de los datos a un tamaño manejable, y luego explorar estos datos resumidos para obtener insights.

Source: https://www.ibm.com/mx-es/topics/exploratory-data-analysis

El **AED** forma parte de cualquier **metodología para el análisis de datos**. Es una fase inicial en la que los analistas examinan los datos para comprender sus características principales antes de aplicar modelos más complejos. Aquí es cómo el AED se vincula con una metodología general para el **análisis de datos**.

1.2 Metodología en el Análisis de Datos

Para establecer una metodología en el análisis de datos, debemos definir que el análisis de datos es el proceso de convertir datos sin procesar en información práctica e incluye una serie de herramientas, tecnologías y procesos para encontrar tendencias y resolver problemas mediante datos.

Source: https://aws.amazon.com/es/what-is/data-analytics/

Existen metodologías para el análisis de datos, en esta guía básica realizada por Non-Profit Evaluation & Resource Center, Inc. (NPERCI) presenta los pasos generales para procesar y analizar datos.

Definición del problema: Antes de comenzar cualquier análisis, es crucial definir claramente el problema que se está tratando de resolver. Esto implica entender el contexto del problema y los objetivos del análisis.

Recolección de datos: Una vez definido el problema, el siguiente paso es recoger los datos necesarios para el análisis. Esto puede implicar la extracción de datos de bases de datos, archivos, APIs, servicios web, etc.

Limpieza de datos: Los datos rara vez están limpios y listos para analizar. Es probable que necesites limpiarlos, lo que puede implicar tratar con datos faltantes, eliminar duplicados, corregir errores, etc.

Análisis Exploratorio de Datos (AED): Una vez que los datos están limpios, el siguiente paso es explorarlos para entender sus características principales. Esto puede implicar visualizar los datos, calcular estadísticas descriptivas, identificar outliers, examinar correlaciones entre variables, etc. El objetivo del AED es obtener una comprensión clara de los datos y formular hipótesis sobre posibles relaciones y patrones en los datos.

Modelado de datos: Después del AED, puedes comenzar a construir modelos de datos, como modelos de regresión, modelos de clasificación, clusters, etc., dependiendo del problema que estés tratando de resolver.

Evaluación del modelo: Una vez que tienes un modelo, necesitas evaluar su rendimiento. Esto puede implicar técnicas como la validación cruzada, el uso de conjuntos de prueba y validación, y el cálculo de métricas de rendimiento.

Interpretación y comunicación de resultados: Finalmente, interpretas los resultados de tu modelo y los comunicas a otras partes interesadas. Esto puede implicar la creación de visualizaciones, informes o presentaciones.

Source: https://nperci.org

2. Objetivo del Proyecto

El objetivo es analizar los datos y obtener información valiosa que permita a la compañía mejorar su estrategia de ventas y aumentar su rentabilidad.

3. Descripción de los datos

En este proyecto se cuenta con un conjunto de datos de Instacart Market Basket Analysis, que se encuentra en varios archivos CSV.

La información que contiene los archivos CSV es la siguiente:

• **orders.csv** : Contiene información de los pedidos realizados por los usuarios. Cada fila representa un pedido y contiene la siguiente información:

- **order_id** : ID del pedido.
- **user_id** : ID del usuario que realizó el pedido.
- **order_number**: Número de pedido para cada usuario (1 = primer pedido, 2 = segundo pedido, etc.).
- **order_dow** : Día de la semana en que se realizó el pedido.
- **order_hour_of_day** : Hora del día en que se realizó el pedido.
- **days_since_prior_order** : Días transcurridos desde el último pedido del usuario.

• **order_products__prior.csv**: Contiene información de los productos comprados en cada pedido. Cada fila representa un producto en un pedido y contiene la siguiente información:

- **order_id** : ID del pedido.
- **product_id** : ID del producto comprado.
- **add_to_cart_order** : Orden en el que se agregó el producto al carrito en el momento del pedido.
- **reordered** : Indica si el producto ha sido ordenado por el usuario anteriormente.

• **products.csv** : Contiene información de los productos vendidos en la tienda. Cada fila representa un producto y contiene la siguiente información:

- **product_id** : ID del producto.
- **product_name** : Nombre del producto.
- **aisle_id**: ID del pasillo donde se encuentra el producto.
- **department_id** : ID del departamento al que pertenece el producto.

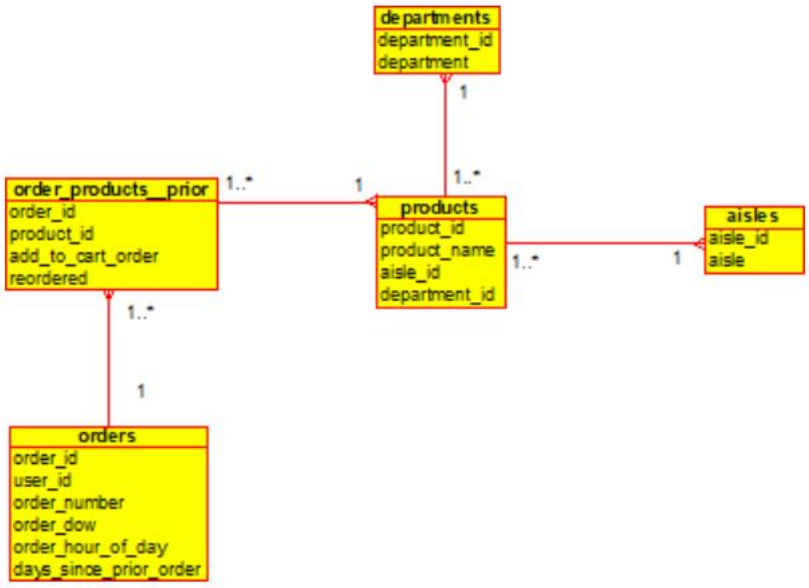
• **aisles.csv** : Contiene información de los pasillos de la tienda. Cada fila representa un pasillo y contiene la siguiente información:

- **aisle_id** : ID del pasillo.
- **aisle** : Nombre del pasillo.

• **departments.csv** : Contiene información de los departamentos de la tienda. Cada fila representa un departamento y contiene la siguiente información:

- **department_id** : ID del departamento.
- **department** : Nombre del departamento.

Diagrama Entidad Relacion



4. Análisis exploratorio

En el análisis exploratorio se analizan los archivos con la información proporcionada en la sección anterior, el propósito es analizar e investigar los conjuntos de datos, resumir sus características principales, detectar anomalías, identificar relaciones, eliminar variables irrelevantes, imputar valores faltantes y limpiar los nombres de los productos.

Se definen las tareas a realizar:

1. Realizar un análisis exploratorio de los datos para entender la distribución de los pedidos, la cantidad de productos por pedido y la frecuencia de compras por día y hora.
2. Identificar los productos más vendidos en la tienda y visualizar los resultados utilizando un gráfico de barras.
3. Identificar los productos más comprados en cada departamento y visualizar los resultados utilizando un gráfico de barras.
4. Identificar los productos que más se compran juntos, es decir, aquellos que aparecen en los mismos pedidos con mayor frecuencia, y visualizar los resultados utilizando un gráfico de red.

En base a las tareas que se define y al analisis exploratorio se realizaran las siguientes acciones:

- **No ocupar la tabla aisles**.
- Eliminar los siguientes atributos:
- - **eval_set** de la tabla **orders**: Este atributo indica a cuál conjunto de datos (entrenamiento, prueba, etc.) pertenece cada pedido, pero no se usa en este análisis.
- - **days_since_prior_order** de la tabla **orders**: Este atributo indica cuántos días han pasado desde el último pedido del mismo usuario, pero no se usa en este análisis.
- - **aisle** de la tabla **products**: Este atributo indica el pasillo en el que se encuentra cada producto, pero no se usa en este análisis.
- - **reordered** de la tabla **order_products**: Este atributo indica si el producto fue reordenado por el mismo usuario, pero no se usa en este análisis.
- Imputar valores faltantes y limpiar los nombres de los productos.
- Se van a eliminar los espacios en blanco despues del nombre del producto.
- Se van a convertir a minusculas el nombre del producto.

5. Desarrollo

Para el desarrollo se va trabajar cada tarea de la siguiente forma:

1. Preprocesar los datos eliminando variables irrelevantes, imputando valores faltantes y limpiando los nombres de los productos.
2. Realizar un análisis exploratorio de los datos para entender la distribución de los pedidos.

- Visualización cantidad de productos por pedido
- Visualización frecuencia de compras por día
- Visualización frecuencia de compras por hora.

3. Identificar los productos más vendidos en la tienda y visualizar los resultados utilizando un gráfico de barras.
4. Identificar los productos más comprados en cada departamento y visualizar los resultados utilizando un gráfico de barras.
5. Identificar los productos que más se compran juntos, es decir, aquellos que aparecen en los mismos pedidos con mayor frecuencia, y visualizar los resultados utilizando un gráfico de red.

5.1. Preprocesar los datos eliminando variables irrelevantes, imputando valores faltantes y limpiando los nombres de los productos.

```
#Instalación de módulos
!pip install pandas
!pip install matplotlib
!pip install networkx
```

```
Requirement already satisfied: pandas in c:\users\jc\anaconda3\lib\site-packages (1.4.4)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\jc\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\jc\anaconda3\lib\site-packages (from pandas) (1.21.5)
Requirement already satisfied: pytz>=2020.1 in c:\users\jc\anaconda3\lib\site-packages (from pandas) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\jc\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.
0)
Requirement already satisfied: matplotlib in c:\users\jc\anaconda3\lib\site-packages (3.5.2)
Requirement already satisfied: numpy>=1.17 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: cycler>=0.10 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\jc\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: six>=1.5 in c:\users\jc\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.1
6.0)
Requirement already satisfied: networkx in c:\users\jc\anaconda3\lib\site-packages (2.8.4)
```

```
#Importar módulos
#Modulo para procesamiento de datos
import pandas as pd
#Modulo para visualizacion de datos
import matplotlib.pyplot as plt
#Modulo para visualizar los resultados utilizando un gráfico de red
import networkx as nx
from itertools import combinations
from collections import Counter
```

```
#Carga de datos de los archivos *.csv
df_department = pd.read_csv('data/departments.csv')
df_orders = pd.read_csv('data/orders.csv')
df_products = pd.read_csv('data/products.csv')
df_order_products = pd.read_csv('data/order_products__prior.csv')
```

```
#Eliminar atributos
df_orders = df_orders.drop('eval_set', axis=1)
df_orders = df_orders.drop('days_since_prior_order', axis=1)
df_products = df_products.drop('aisle_id', axis=1)
df_order_products = df_order_products.drop('reordered', axis=1)
```

```
#Unir los datos
data = df_orders.merge(df_order_products, on="order_id").merge(df_products, on="product_id").merge(df_department, on="department_
```

```
#Imputar valores faltantes y limpiar los nombres de los productos.
#Verificar si hay valores nulos o faltantes
valor_imputar = data.isnull().sum()
print(valor_imputar)
```

```
order_id             0
user_id              0
order_number         0
order_dow            0
order_hour_of_day    0
product_id           0
add_to_cart_order    0
product_name         0
department_id        0
department           0
dtype: int64
```

No hay valores NULOS

```
#Verificar si hay nombres de productos duplicados o mal escritos
valor_duplicados = data['product_name'].value_counts()
print(valor_duplicados)
```

```
Banana                                    472565
Bag of Organic Bananas                    379450
Organic Strawberries                      264683
Organic Baby Spinach                      241921
Organic Hass Avocado                      213584
                                           ...
Cajun Sides Dirty Rice                         1
Orangemint Flavored Water                      1
Florentine Spinach and Cheese Pasta Sauce      1
Bite Size Caramel Chocolates                   1
Dynostix Rawhide Chew With Meat                1
Name: product_name, Length: 49677, dtype: int64
```

```
#Se van a eliminar los espacios en blanco despues del nombre del producto
#Se van a convertir a minusculas el nombre del producto
# Corregir nombres de productos (si es necesario)
data['product_name'] = data['product_name'].str.strip()  # Eliminar espacios extra
data['product_name'] = data['product_name'].str.lower()  # Convertir a minúsculas
```

```
# Verificar si hay nombres de productos duplicados o mal escritos
valor_duplicados = data['product_name'].value_counts()
print(valor_duplicados)
```

```
banana                                    472565
bag of organic bananas                    379450
organic strawberries                      264683
organic baby spinach                      241921
organic hass avocado                      213584
                                           ...
orange flavored ice cubes                      1
ginseng vitality tea                           1
orange recharge                                1
pasta shapes in tomato sauce                   1
lindor peppermint white chocolate truffles     1
Name: product_name, Length: 49577, dtype: int64
```

5.2. Realizar un análisis exploratorio de los datos para entender la distribución de los pedidos.

- Visualización cantidad de productos por pedido.
- Visualización frecuencia de compras por día.
- Visualización frecuencia de compras por hora.

5.2.1 Visualización cantidad de productos por pedido

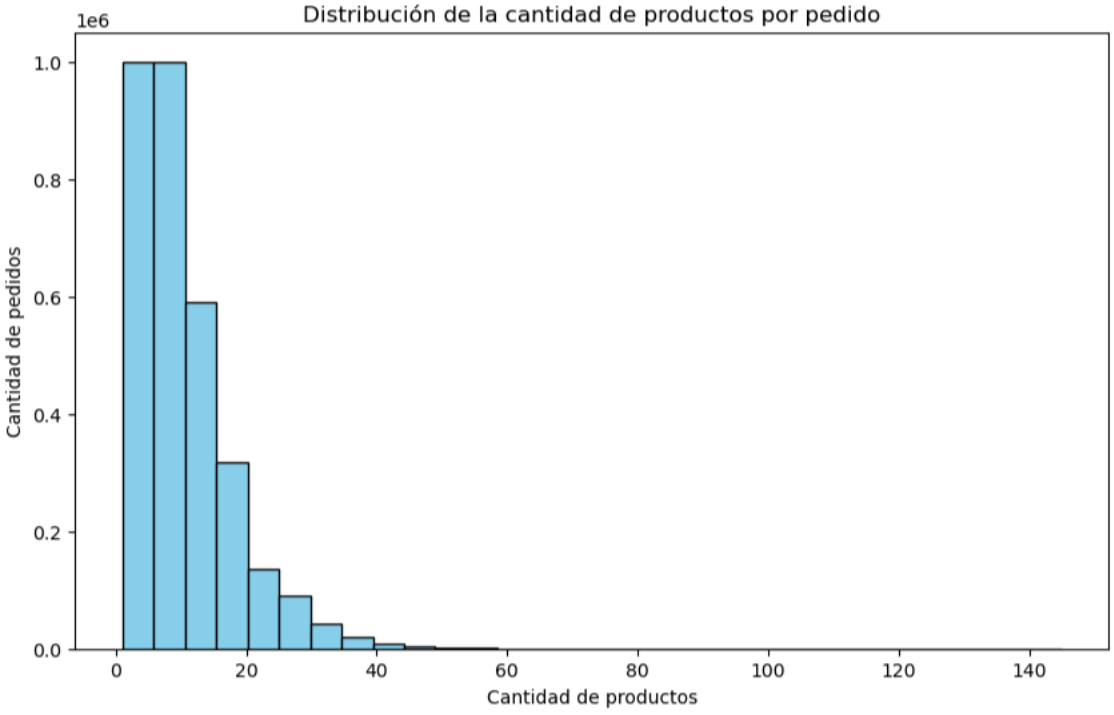
```
# Número de productos por pedido
prod_x_orden = data.groupby('order_id')['product_id'].count()
```

```
#Gráfico de la cantidad de productos por pedido
plt.figure(figsize=(10,6))
plt.hist(prod_x_orden, bins=30, edgecolor='black', color='skyblue')
plt.title('Distribución de la cantidad de productos por pedido')
plt.xlabel('Cantidad de productos')
plt.ylabel('Cantidad de pedidos')
plt.show()
```

5.2.1 Visualización cantidad de productos por pedido

```
# Número de productos por pedido
prod_x_orden = data.groupby('order_id')['product_id'].count()
```

```
#Gráfico de la cantidad de productos por pedido
plt.figure(figsize=(10,6))
plt.hist(prod_x_orden, bins=30, edgecolor='black', color='skyblue')
plt.title('Distribución de la cantidad de productos por pedido')
plt.xlabel('Cantidad de productos')
plt.ylabel('Cantidad de pedidos')
plt.show()
```

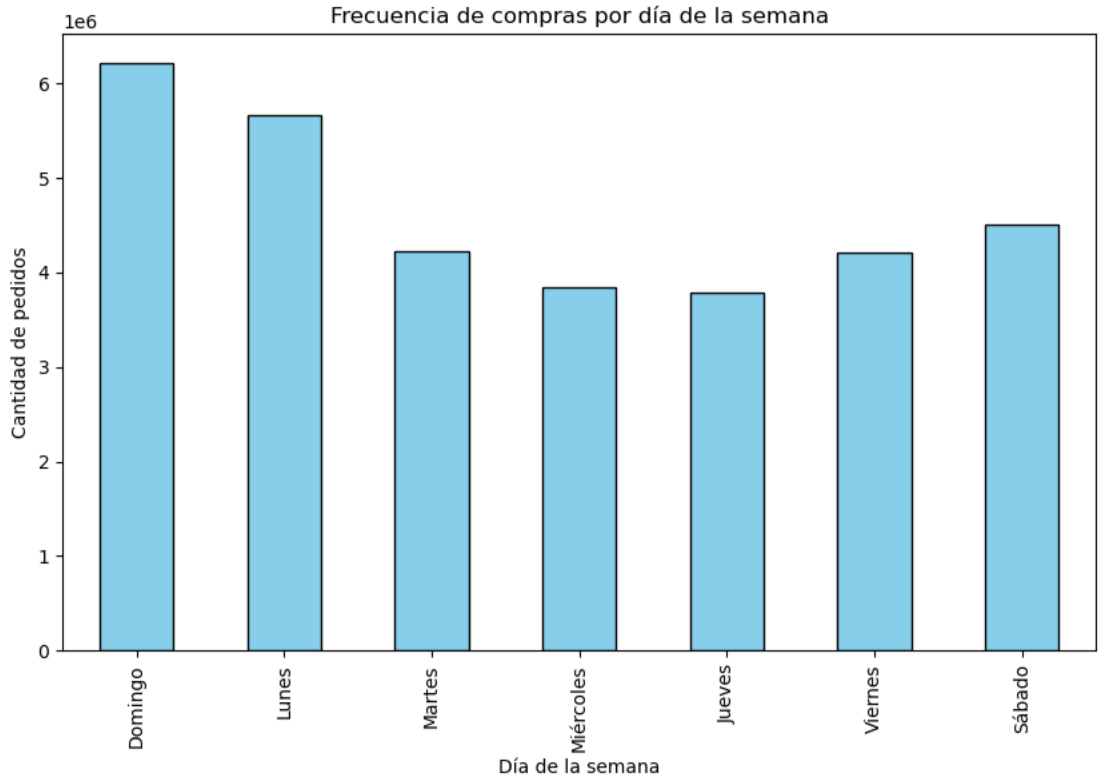


5.2.2 Visualización frecuencia de compras por día.

Convertir el tipo de dato numerico a un string para identificar los dias por nombre

```
dow_dict = {
    0: 'Domingo',
    1: 'Lunes',
    2: 'Martes',
    3: 'Miércoles',
    4: 'Jueves',
    5: 'Viernes',
    6: 'Sábado'
}

#Usar el método map para reemplazar los números por los nombres de los días
data['order_dow'] = data['order_dow'].map(dow_dict)
```

```
#Gráfico de pedidos por día de la semana
#Gráfico de pedidos por día de la semana
orden_x_dia = data['order_dow'].value_counts().loc[dow_dict.values()]
orden_x_dia.plot.bar(figsize=(10,6), edgecolor='black', color='skyblue')
plt.title('Frecuencia de compras por día de la semana')
plt.xlabel('Día de la semana')
plt.ylabel('Cantidad de pedidos')
plt.show()
```

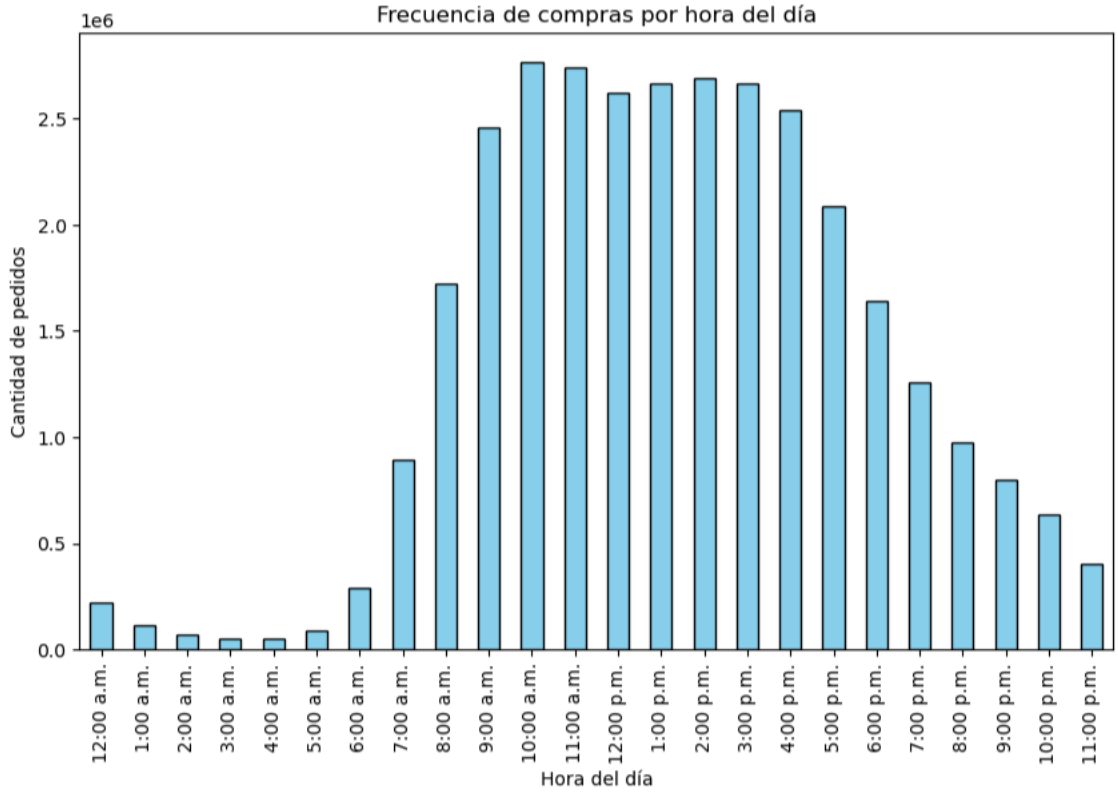


5.2.3 Visualización frecuencia de compras por hora.

Convertir el tipo de dato numerico a un string para identificar los horarios en a.m. o p.m.

```
# Pedidos por hora del día
dow_dict = {
    0: '12:00 a.m.',
    1: '1:00 a.m.',
    2: '2:00 a.m.',
    3: '3:00 a.m.',
    4: '4:00 a.m.',
    5: '5:00 a.m.',
    6: '6:00 a.m.',
    7: '7:00 a.m.',
    8: '8:00 a.m.',
    9: '9:00 a.m.',
    10: '10:00 a.m.',
    11: '11:00 a.m.',
    12: '12:00 p.m.',
    13: '1:00 p.m.',
    14: '2:00 p.m.',
    15: '3:00 p.m.',
    16: '4:00 p.m.',
    17: '5:00 p.m.',
    18: '6:00 p.m.',
    19: '7:00 p.m.',
    20: '8:00 p.m.',
    21: '9:00 p.m.',
    22: '10:00 p.m.',
    23: '11:00 p.m.'
}

# Usar el método map para reemplazar los números por los nombres de los días
data['order_hour_of_day'] = data['order_hour_of_day'].map(dow_dict)
orden_x_hora = data['order_hour_of_day'].value_counts().loc[dow_dict.values()]
```

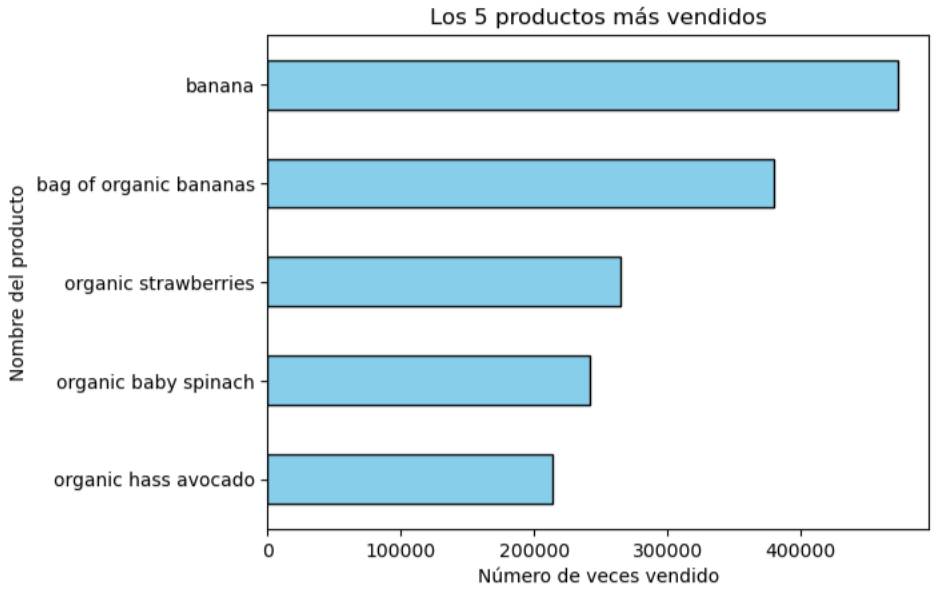
```
# Gráfico de pedidos por hora del día
orden_x_hora.plot.bar(figsize=(10,6), edgecolor='black', color='skyblue')
plt.title('Frecuencia de compras por hora del día')
plt.xlabel('Hora del día')
plt.ylabel('Cantidad de pedidos')
plt.xticks(range(24))
plt.show()
```



5.3. Identificar los productos más vendidos en la tienda y visualizar los resultados utilizando un gráfico de barras.

```
#Obtener la relacion entre numero del producto con nombre del prodcuto
nombre_prod = data[['product_id', 'product_name']]
#Contar la cantidad de cada producto vendido
cant_prod = nombre_prod['product_name'].value_counts()
#Obtener los nombres de los 5 productos más vendidos
top_prod = cant_prod.head(5)
```

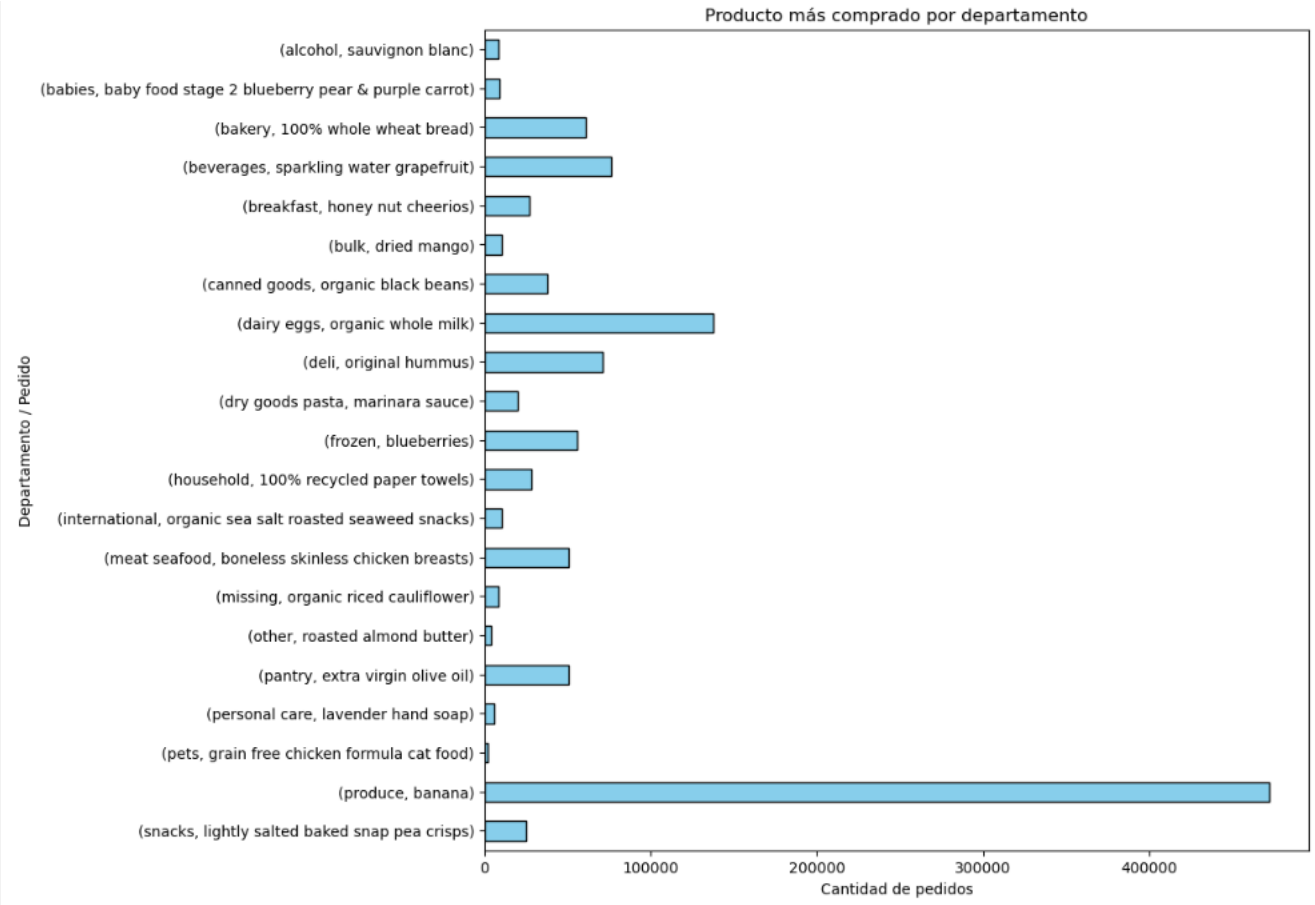
```
# Gráfico de los productos más vendidos
top_prod.plot(kind='barh', edgecolor='black', color='skyblue')
# Añadir etiquetas y título
plt.xlabel('Número de veces vendido')
plt.ylabel('Nombre del producto')
plt.title('Los 5 productos más vendidos')
plt.gca().invert_yaxis()
# Invertir el eje y para que el producto más vendido esté en la parte superior
plt.show()
```



5.4. Identificar los productos más comprados en cada departamento y visualizar los resultados utilizando un gráfico de barras.

```
# Producto más vendido por departamento
top_prod_x_dep = data.groupby('department')['product_name'].value_counts().groupby(level=0).nlargest(1)
# Restablecer el primer nivel de índice
top_prod_x_dep = top_prod_x_dep.reset_index(level=0, drop=True)
```

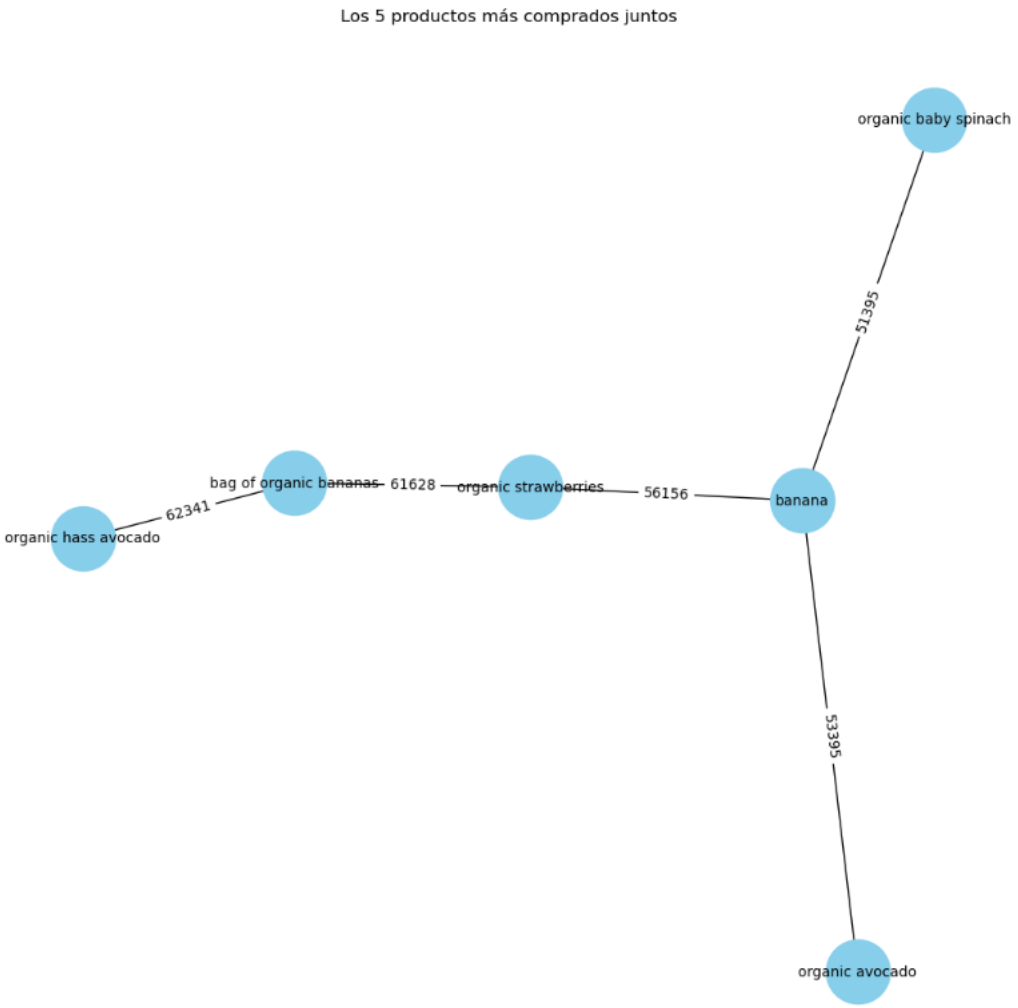
```
# Crear el gráfico de barras
plt.figure(figsize=(10,10))
top_prod_x_dep.plot(kind='barh', edgecolor='black', color='skyblue')
# Configurar los ejes y el título
plt.title('Producto más comprado por departamento')
plt.xlabel('Cantidad de pedidos')
plt.ylabel('Departamento / Pedido')
# Invertir el eje y para que el departamento con el producto más vendido esté en la parte superior
plt.gca().invert_yaxis()
plt.show()
```



5.5. Identificar los productos que más se compran juntos, es decir, aquellos que aparecen en los mismos pedidos con mayor frecuencia, y visualizar los resultados utilizando un gráfico de red.

```
#Agrupar los datos por 'order_id' y obtener la lista de productos en cada pedido
orden_prod = data.groupby('order_id')['product_name'].apply(list)

#Generar todas las posibles combinaciones de productos en cada pedido y contar la frecuencia de cada combinación
comb_prod = Counter()

for products in orden_prod:
    comb_prod.update(combinations(products, 2))

# Obtener las 5 combinaciones de productos más comunes
top_comb_prod = comb_prod.most_common(5)
```

```
# Creamos un grafo
G = nx.Graph()

# Añadimos nodos (productos) y bordes (conexiones entre productos) al grafo
for (p1, p2), count in top_comb_prod:
    G.add_node(p1)
    G.add_node(p2)
    G.add_edge(p1, p2, weight=count)

# Dibujamos el grafo
plt.figure(figsize=(10,10))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='skyblue', node_size=2000, font_size=10)
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
plt.title('Los 5 productos más comprados juntos')
plt.show()
```



5.6. Utilizar Spark para realizar un análisis de los datos en paralelo y obtener la frecuencia de compras por día y hora.

```
#Instalacion de modulos
!pip install pyspark
```

```
Requirement already satisfied: pyspark in c:\users\jc\anaconda3\lib\site-packages (3.4.0)
Requirement already satisfied: py4j==0.10.9.7 in c:\users\jc\anaconda3\lib\site-packages (from pyspark) (0.10.9.7)
```

```
#Importar modulos
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
```

```
# Crear una sesión de Spark
spark = SparkSession.builder.appName("Proyecto_Evaluacion").getOrCreate()
```

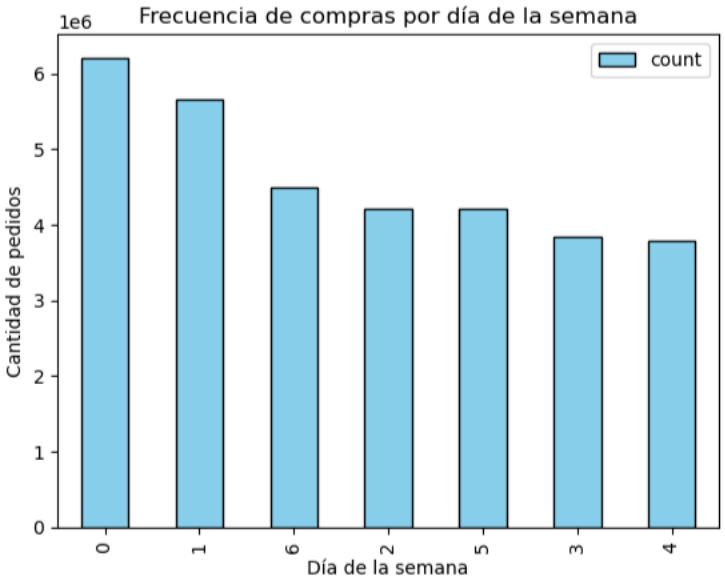
```
# Cargar los datos en DataFrames de Spark
df_department = spark.read.csv('data/departments.csv', header=True, inferSchema=True)
df_order_products = spark.read.csv('data/order_products__prior.csv', header=True, inferSchema=True)
df_orders = spark.read.csv('data/orders.csv', header=True, inferSchema=True)
df_products = spark.read.csv('data/products.csv', header=True, inferSchema=True)
```

```
#Eliminar atributos
df_orders = df_orders.drop('eval_set')
df_orders = df_orders.drop('days_since_prior_order')
df_products = df_products.drop('aisle_id')
df_order_products = df_order_products.drop('reordered')
```

```
#Unir los datos
data = df_orders.join(df_order_products, "order_id").join(df_products, "product_id").join(df_department, "department_id")
```

```
# Calcular la frecuencia de compras por día
orden_x_dia_sp = data.groupBy('order_dow').count().orderBy('count', ascending=False)
orden_x_dia_pd = orden_x_dia_sp.toPandas()
orden_x_dia_pd.set_index('order_dow', inplace=True)
```

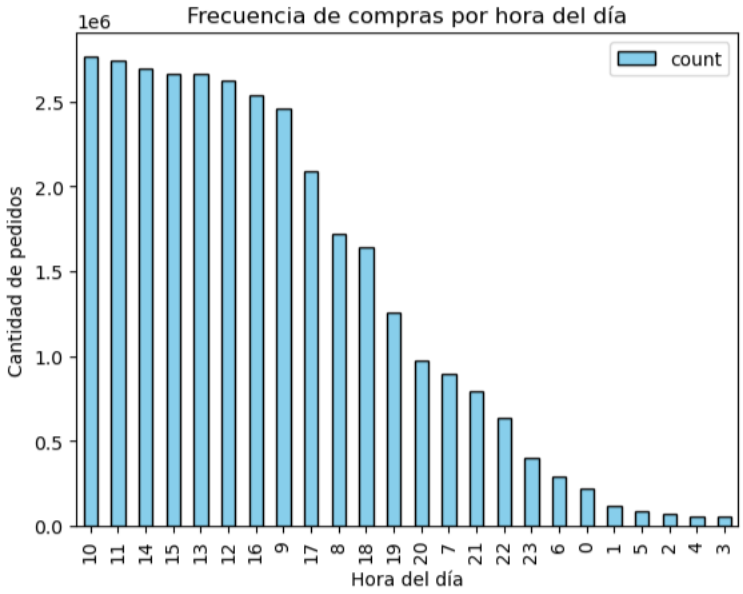
```
# Visualizar la frecuencia de compras por día
orden_x_dia_pd.plot(kind='bar', title='Frecuencia de compras por día de la semana', edgecolor='black', color='skyblue')
plt.xlabel('Día de la semana')
plt.ylabel('Cantidad de pedidos')
plt.show()
```



- 0: 'Domingo',
- 1: 'Lunes',
- 2: 'Martes',
- 3: 'Miércoles',
- 4: 'Jueves',
- 5: 'Viernes',
- 6: 'Sábado'

```
# Calcular la frecuencia de compras por hora
orden_x_hora_sp = data.groupBy('order_hour_of_day').count().orderBy('count', ascending=False)
orden_x_hora_pd = orden_x_hora_sp.toPandas()
orden_x_hora_pd.set_index('order_hour_of_day', inplace=True)
```

```
# Visualizar la frecuencia de compras por hora
orden_x_hora_pd.plot(kind='bar', edgecolor='black', color='skyblue')
plt.title('Frecuencia de compras por hora del día')
plt.xlabel('Hora del día')
plt.ylabel('Cantidad de pedidos')
plt.xticks(range(24))
plt.show()
```



- 0: '12:00 a.m.',
- 1: '1:00 a.m.',
- 2: '2:00 a.m.',
- 3: '3:00 a.m.',
- 4: '4:00 a.m.',
- 5: '5:00 a.m.',
- 6: '6:00 a.m.',
- 7: '7:00 a.m.',
- 8: '8:00 a.m.',
- 9: '9:00 a.m.',
- 10: '10:00 a.m.',
- 11: '11:00 a.m.',
- 12: '12:00 p.m.',
- 13: '1:00 p.m.',
- 14: '2:00 p.m.',
- 15: '3:00 p.m.',
- 16: '4:00 p.m.',
- 17: '5:00 p.m.',
- 18: '6:00 p.m.',
- 19: '7:00 p.m.',
- 20: '8:00 p.m.',
- 21: '9:00 p.m.',
- 22: '10:00 p.m.',
- 23: '11:00 p.m.'

Intente implementar los diccionarios de dias y horarios como en pandas, pero no tuve exito

6. Resultados

Para las solicitudes de análisis en Pandas implemente el archivo ADE.ipynb. Para las solicitudes de análisis en Spark implemente el archivo ADESpark.ipynb en:

Evaluacion eficencia Python con Pandas.

Se realizaron pruebas en:

Sistema Operativo	Tiempo de ejecucion	Uso de CPU	Uso de memoria
Windows 10	112.2 segundos	26.8%	40.4%
macOS Catalina	153.05 segundos	19.8%	37.3%
Red Hat 9	Error Kernel	Error Kernel	Error Kernel

- El resultado de Red Hat de Python con Pandas se detuvo.

Evaluacion eficencia Spark.

Se realizaron pruebas en:

Sistema Operativo	Tiempo de ejecucion	Uso de CPU	Uso de memoria
Windows 10	86.69 segundos	75.7%	47.8%
macOS Catalina	181.68 segundos	27.8%	51.9%
Red Hat 9	Aprox 120 segundos	Aprox 36.66%	Aprox 31%

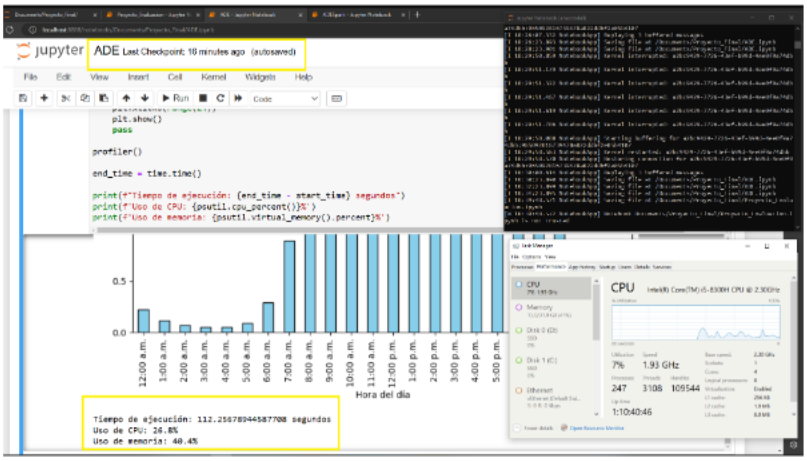
- El resultado de Red Hat con Spark fue aproximado.
- Solo genero una grafica y se paro el servicio.
- Posiblemente el error con Red Hat 9 se debio a su virtualización y restriccion en el tamaño del disco duro virtual con tenia 31 GB de espacio.

Detalles tecnicos y Demostración de ejecución:

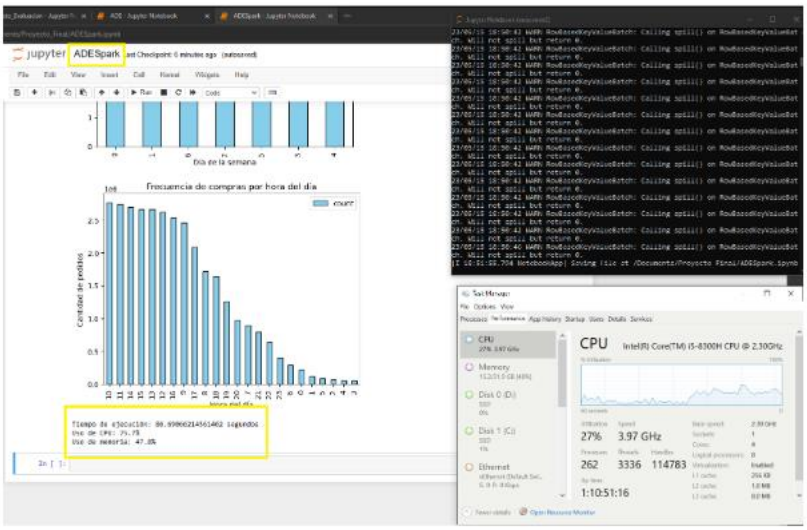
- Windows 10

- ▪ Procesador i5 de 9th Generacion 2.30 GHz con 8 nucleos
- ▪ Memoria RAM 32 GB
- ▪ Disco Duro NVME de 1TB

Python con pandas Win 10



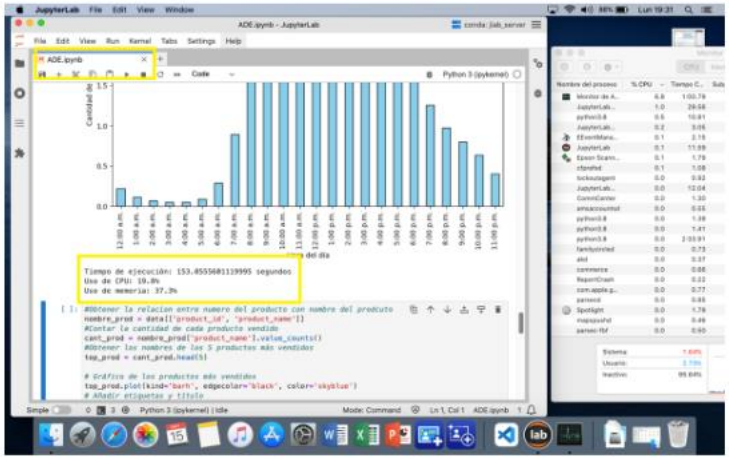
Spark Win 10



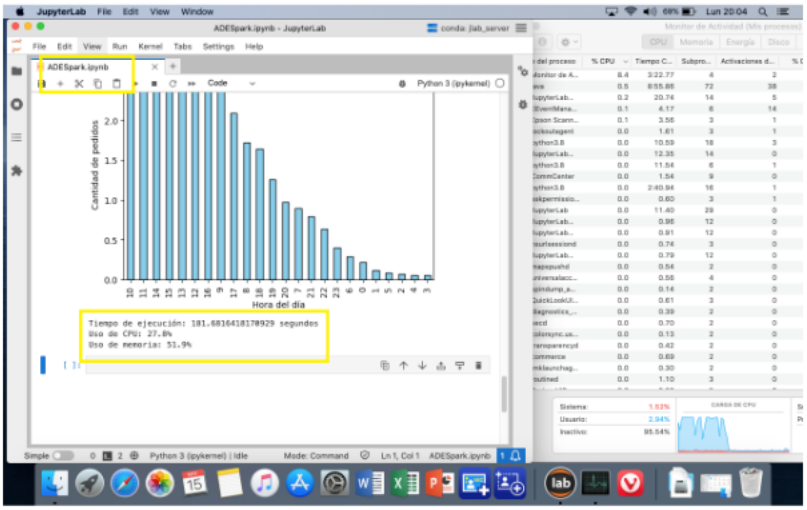
- Osx Catalina

- ▪ Procesador i5 de 5th Generacion 1.7 GHz con 2 nucleos
- ▪ Memoria RAM 16 GB
- ▪ Disco Duro SSD de 512GB

Python con pandas Osx Catalina



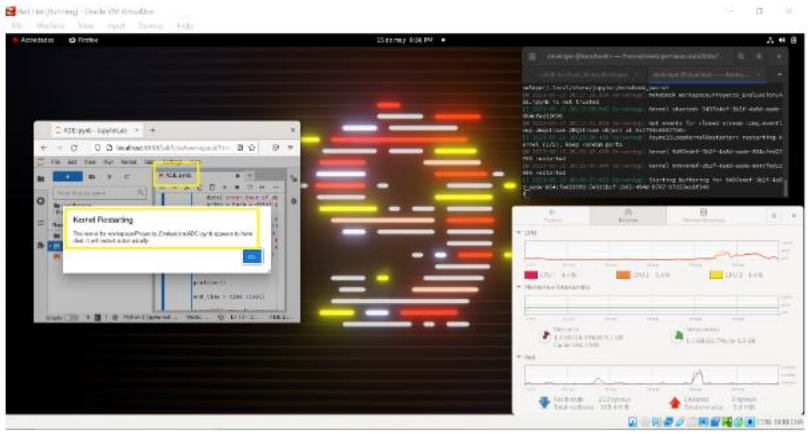
Spark Osx Catalina



- Red Hat 9 (Maquina Virtual en Virtual Box)

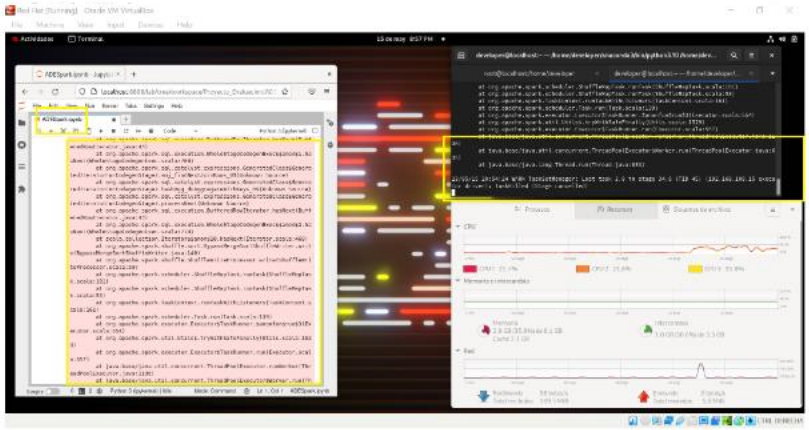
- Procesador i5 de 9th Generacion 2.30 GHz con 4 nucleos
- Memoria RAM 8 GB
- Disco Duro NVME de 31GB

Error en Python con pandas Red Hat 9



Spark Red Hat 9

Error en Spark Red Hat 9



7. Conclusión

El uso e implementación de técnicas de Big Data para el análisis de datos aporta al conocimiento teórico y técnico una comprensión del comportamiento de la información y el potencial que tiene para poder ayudar a la solución de problemas reales que tiene la sociedad y/o en la toma de decisiones dentro de empresas públicas y privadas, por lo que la información resulta ser invaluable para las empresas de cualquier sector productivo.

El desarrollo del análisis de datos de la información proporcionada, se tuvo en primer parte una visión global de los datos y la relación que tienen entre estos, lo que llevó a establecer el tipo de atributos que los relacionaban y el giro de servicios y/o productos que la empresa utiliza esta información. Así se tiene una comprensión más a detalle de las tareas que se van a desarrollar y los resultados que se pretenden obtener.

En el análisis de datos se establece una tienda de servicios de productos que están separados por departamentos y pasillos, también las órdenes de compra que nos dicen el cliente, la frecuencia y la cantidad de productos por cada orden de compra diaria, así se identifica los productos que se compran juntos, con más frecuencia por día o por hora y por departamento, con el resultado se puede ayudar a la empresas a desarrollar estrategias de venta cruzada más efectivas y a optimizar el diseño de sus tiendas, tanto físicas como en línea. Al mismo tiempo, conocer la frecuencia de las compras por día y hora puede ayudar a mejorar la gestión de inventario y la programación del personal, lo que a su vez puede conducir a una mayor eficiencia y ahorro de costos.

En la parte técnica del análisis de datos se utilizó las herramientas de Python con Pandas y Spark, ambas son herramientas para el manejo y análisis de datos, pero se utilizan de manera diferente dependiendo del tamaño de los datos y el entorno de procesamiento.

Por lo tanto, en la implementación de estas técnicas en este problema el uso de Python con Pandas tuvo como ventajas:

- Manipulación de datos sencilla.
- Análisis exploratorio de datos a pequeña y mediana escala
- Poder trabajar en un solo equipo de cómputo con requisitos técnicos mínimos.

Y como desventajas:

- La limitante de analizar conjuntos de datos masivos debido a las restricciones de memoria del equipo.

Por otra parte, Spark está diseñado para ser rápido y manejar grandes cantidades de datos distribuidos a través de nodos en un clúster, por lo que, lo hace más adecuado para conjuntos de datos muy grandes que no caben en la memoria del equipo y que se puede implementar en una infraestructura de cloud computing (Amazon, Azure, Google Cloud, etc.).

En la implementación de Spark en este problema tuvo ventajas:

- El tiempo de carga de datos fue más rápido.
- El análisis exploratorio mejora el rendimiento del procesador.
- El poder aumentar la cantidad de información para realmente visualizar su potencial.

Y como desventajas:

- La limitante de analizar conjuntos de datos masivos debido a las restricciones de memoria del equipo.

Por otra parte, Spark está diseñado para ser rápido y manejar grandes cantidades de datos distribuidos a través de nodos en un clúster, por lo que, lo hace más adecuado para conjuntos de datos muy grandes que no caben en la memoria del equipo y que se puede implementar en una infraestructura de cloud computing (Amazon, Azure, Google Cloud, etc.).

En la implementación de Spark en este problema tuvo ventajas:

- El tiempo de carga de datos fue más rápido.
- El análisis exploratorio mejora el rendimiento del procesador.
- El poder aumentar la cantidad de información para realmente visualizar su potencial.

Y como desventajas:

- Es complicado en su codificación, si no se tienen conocimientos previos del lenguaje de programación.
- La manipulación de datos es más compleja retomando lo dicho en el punto anterior.
- No se exploran métodos de ploteo o graficación.
- No se visualiza el rendimiento en recursos ya que todo se ejecutó en un solo equipo de cómputo.

En conclusión, el uso de Big Data y técnicas de análisis de datos (Python con Pandas y Spark) en este proyecto, ayuda a entender y clarificar el poder aplicativo como desarrollador de software en lo que es el proceso de análisis de información, y como una herramienta para la dirección de empresas (públicas, privadas, investigación, fundaciones, etc.,) en la toma de decisiones. Asi que el Big Data se posiciona como una de las áreas del conocimiento con más aplicación en el presente y como una base científica para futuras áreas de la inteligencia artificial como el Machine Learning. Y en una ventaja para mi proyecto de investigación, ya que será la herramienta que me ayude a plantear y validar mi pregunta de investigación e hipótesis en mi proyecto de Doctorado en Tecnologías de la Información.

8. Bibliográfia

- Documentación oficial de Python: https://docs.python.org/3/
- Documentación oficial de Pandas: https://pandas.pydata.org/docs/
- Documentación oficial de Matplotlib: https://matplotlib.org/stable/contents.html
- Documentación oficial de Apache Spark: https://spark.apache.org/docs/latest/
- Analisis Exploratorio de Datos https://www.aprendemachinelearning.com/analisis-exploratorio-de-datos-pandas-python/
- Conceptos en Python https://www.geeksforgeeks.org
- Dudas https://stackoverflow.com/questions/tagged/pandas+python
- Analisis de datos https://ocw.uc3m.es/course/view.php?id=230
- Diccionarios de datos en data frame https://github.com/nsheikh23/COVID_StockMarket_Analysis/blob/master/52_Week.ipynb
- Procesamiento de data frames en pandas https://barcelonageeks.com/eliminar-una-o-varias-columnas-de-pyspark-dataframe/
- Data Clean https://github.com/mramshaw/Data-Cleaning
- Ploteo de datos https://github.com/tomimester/python-histogram/blob/master/plot-histogram-python-pandas.ipynb
- Creación de grafos con networkx https://ernestocrespo13.wordpress.com/2012/11/25/creacion-de-grafos-con-networkx-parte-1/
- Data Analysis Techniques with PySpark https://github.com/sedaatalay/Sample-Data-Analysis-Techniques-with-PySpark