

# Análítica auditoría CMM-Paraguay

Período enero - abril 2024

```

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Cargamos las variables de los archivos en Google Sheets
ID_planilla = '1JpDhnHRTsY501-AaXTHi9XbKp_sJ4Muwbm7wgRuhzxA'
URL = f'https://docs.google.com/spreadsheets/d/{ID_planilla}/gviz/tq?tqx=out:csv&sheet='

```

## Inspección preliminar y limpieza

- Ver información general
- Ver tipos de datos de cada columna
- Ver formato de contenido de cada registro
- Revisar registros o filas nulas
- Revisar registros o filas duplicadas

## DataFrame de Ventas

```

# DataFrame de ventas
df_ventas = pd.read_csv(URL + 'ventas')
df_ventas.head()

```

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	Glass	P
0	583.0	\$54.510.203,61	\$45.268.636,86	\$51.579.748,25	\$21.780.180,58	\$26.576.776,52	\$0,00	\$55.872.547,77	\$18.050.385,12	\$178.
1	1104.0	\$32.438.788,20	\$25.837.100,49	\$36.603.264,50	\$21.883.374,92	\$1.473.437,08	\$0,00	\$46.239.695,30	\$33.102.840,61	\$35.
2	1384.0	\$21.780.180,58	\$78.927.599,01	\$25.837.100,49	\$33.102.840,61	\$51.579.748,25	\$0,00	\$21.883.374,92	\$78.927.599,01	\$1.
3	379.0	\$79.358.855,35	\$90.185.311,22	\$45.268.636,86	\$54.510.203,61	\$59.358.855,35	\$0,00	\$79.358.855,35	\$32.067.534,68	\$53.
4	1599.0	\$11.758.005,07	\$21.780.180,58	\$57.187.306,41	\$9.945.371,16	\$32.067.534,68	\$0,00	\$53.172.624,14	\$57.187.306,41	\$1.155.

```

# Visualizamos la informacion general
df_ventas.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 13 columns):
#   Column          Non-Null Count  Dtype
---  -
0   distributor      44 non-null    float64
1   Rubber          45 non-null    object
2   Brass           45 non-null    object
3   Vinyl           45 non-null    object
4   Granite         45 non-null    object
5   Stone           45 non-null    object
6   Brick           45 non-null    object
7   Aluminum        45 non-null    object
8   Glass           45 non-null    object
9   Plexiglass      45 non-null    object
10  Steel           45 non-null    object
11  Wood            45 non-null    object
12  Plastic         45 non-null    object
dtypes: float64(1), object(12)
memory usage: 4.7+ KB

```

```

# Buscamos los valores nulos
#Elimina los registros con id nulos
if (df_ventas["distributor"].isnull().sum() >= 1):
    df_ventas.dropna(subset="distributor", inplace=True)

# Buscamos los valores duplicados
df_dup_ventas = df_ventas[df_ventas.duplicated()]
if(df_dup_ventas.empty == False):
    df_ventas.drop_duplicates(inplace=True)

```

Se deben corregir los siguientes puntos en df\_ventas:

- distributor: debe ser un identificador numérico de tipo int
- Los valores numericos deben cambiar su signo de moneda (\$) y pasar a ser de tipo flotante (float)

```
# Convertimos columna distributor a tipo numerico (int)
df_ventas["distributor"] = df_ventas["distributor"].astype(int)

# Seleccionamos las columnas a convertir, desde "Rubber" hasta "Plastic"
cols_ventas = list(df_ventas.columns[1:])

# Convertimos los valores a tipo numerico (float) haciendo correccion de los datos contenidos en estas columnas
for columns in cols_ventas:
    df_ventas[columns] = df_ventas[columns].str.replace('$', '') # Reemplaza el $ por vacío
    df_ventas[columns] = df_ventas[columns].str.replace('.', '') # Reemplaza el . por vacío
    df_ventas[columns] = df_ventas[columns].str.replace(',', '.') # Reemplaza la , por .
    df_ventas[columns] = df_ventas[columns].astype(float) # Convertimos a tipo float

df_ventas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 44 entries, 0 to 43
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   distributor  44 non-null    int64
1   Rubber       44 non-null    float64
2   Brass        44 non-null    float64
3   Vinyl        44 non-null    float64
4   Granite      44 non-null    float64
5   Stone        44 non-null    float64
6   Brick        44 non-null    float64
7   Aluminum     44 non-null    float64
8   Glass        44 non-null    float64
9   Plexiglass   44 non-null    float64
10  Steel        44 non-null    float64
11  Wood         44 non-null    float64
12  Plastic      44 non-null    float64
dtypes: float64(12), int64(1)
memory usage: 4.8 KB
```

```
# Visualizacion del DataFrame de ventas corregido
df_ventas.head(2)
```

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	Glass	Plexiglass	Steel
0	583	54510203.61	45268636.86	51579748.25	21780180.58	26576776.52	0.0	55872547.77	18050385.12	1.789276e+08	2788.80
1	1104	32438788.20	25837100.49	36603264.50	21883374.92	1473437.08	0.0	46239695.30	33102840.61	3.556362e+07	599.01

### DataFrame de Productos Exportados a Paraguay

```
df_exportado = pd.read_csv(URL + 'exportado')
df_exportado.head()
```

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	Glass
0	1,526	\$22.431.099,00	\$36.031.577,00	\$31.118.167,00	\$21.322.223,00	\$35.382.848,00	\$35.280.292,00	\$32.362.235,00	\$36.836.190,00
1	1,553	\$27.566.922,00	\$21.996.538,00	\$39.412.316,00	\$25.681.987,00	\$41.861.783,00	\$22.408.742,00	\$40.690.302,00	\$37.958.885,00
2	1,666	\$37.577.095,00	\$41.457.655,00	\$31.467.967,00	\$37.577.926,00	\$35.845.106,00	\$42.953.168,00	\$33.817.289,00	\$41.602.183,00
3	364	\$36.012.730,00	\$41.667.692,00	\$22.837.073,00	\$29.288.200,00	\$39.553.494,00	\$33.513.588,00	\$36.827.718,00	\$29.669.764,00
4	920	\$43.416.417,00	\$36.290.780,00	\$23.679.738,00	\$21.183.706,00	\$25.210.622,00	\$30.864.041,00	\$41.173.207,00	\$36.719.169,00

```
df_exportado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   distributor  46 non-null    object
1   Rubber       46 non-null    object
2   Brass        46 non-null    object
```

```
3 Vinyl 46 non-null object
4 Granite 46 non-null object
5 Stone 46 non-null object
6 Brick 46 non-null object
7 Aluminum 46 non-null object
8 Glass 46 non-null object
9 Plexiglass 46 non-null object
10 Steel 46 non-null object
11 Wood 46 non-null object
12 Plastic 46 non-null object
dtypes: object(13)
memory usage: 4.8+ KB

# Buscamos los valores duplicados
df_dup_exportado = df_exportado[df_exportado.duplicated()]
if(df_dup_exportado.empty == False):
    df_exportado.drop_duplicates(inplace=True)

# Buscamos los valores nulos
#Elimina los registros con id nulos
if (df_exportado["distributor"].isnull().sum() >= 1):
    df_exportado.dropna(subset="distributor", inplace=True)
```

Se deben corregir los siguientes puntos en df\_exportado:

- distributor: debe ser un identificador numérico de tipo int
- Los valores numericos deben cambiar su signo de moneda (\$) y pasar a ser de tipo flotante (float)

```
df_exportado["distributor"] = df_exportado["distributor"].str.replace(',',' ')
df_exportado['distributor'] = df_exportado['distributor'].astype('int')

# Seleccionamos las columnas a convertir, desde "Rubber" hasta "Plastic"
cols_exportado = list(df_exportado.columns[1:])

# Convertimos los valores a tipo numerico (float) haciendo correccion de los datos contenidos en estas columnas
for columns in cols_exportado:
    df_exportado[columns] = df_exportado[columns].str.replace('$', '') # Reemplaza el $ por vacío
    df_exportado[columns] = df_exportado[columns].str.replace('.', '') # Reemplaza el . por vacío
    df_exportado[columns] = df_exportado[columns].str.replace(',','.') # Reemplaza la , por .
    df_exportado[columns] = df_exportado[columns].astype(float) # Convertimos a tipo float

# Visualizacion del DataFrame de ventas corregido
df_exportado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 44 entries, 0 to 45
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  -
0   distributor  44 non-null    int64
1   Rubber      44 non-null    float64
2   Brass       44 non-null    float64
3   Vinyl       44 non-null    float64
4   Granite     44 non-null    float64
5   Stone       44 non-null    float64
6   Brick       44 non-null    float64
7   Aluminum    44 non-null    float64
8   Glass       44 non-null    float64
9   Plexiglass  44 non-null    float64
10  Steel       44 non-null    float64
11  Wood        44 non-null    float64
12  Plastic     44 non-null    float64
dtypes: float64(12), int64(1)
memory usage: 4.8 KB
```

```
df_exportado.head(2)
```

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	Glass	Plexiglass	Steel
0	1526	22431099.0	36031577.0	31118167.0	21322223.0	35382848.0	35280292.0	32362235.0	36836190.0	9.440859e+09	34676425.0
1	1553	27566922.0	21996538.0	39412316.0	25681987.0	41861783.0	22408742.0	40690302.0	37958885.0	8.726745e+09	26472020.0


### DataFrame de Distribuidores

```
df_distribuidores = pd.read_csv(URL + 'distribuidores')
df_distribuidores.head()
```



	id	distributor	distributor activities	years in the construction market
0	565	Abernathy-Hayes	construction materials import/distribution, ir...	15
1	1,384	Balistreri LLC	construction materials import/distribution, si...	22
2	1,183	Brekke-Stiedemann	construction materials import/distribution	13
3	1,526	Collins LLC	construction materials import/distribution	12
4	29	Cummings-Ward	construction materials import/distribution, ir...	23

```
df_distribuidores.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 4 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                    44 non-null    object
1   distributor                          44 non-null    object
2   distributor activities                44 non-null    object
3   years in the construction market     46 non-null    int64
dtypes: int64(1), object(3)
memory usage: 1.6+ KB

#Elimina los registros con id nulos
if (df_distribuidores["id"].isnull().sum() >= 1):
    df_distribuidores.dropna(subset = "id", inplace=True)

# Busca y elimina los valores duplicados
df_dup_distribuidores = df_distribuidores[df_distribuidores.duplicated()]
if(df_dup_distribuidores.empty == False):
    df_distribuidores.drop_duplicates(inplace=True)

df_distribuidores["id"] = df_distribuidores["id"].str.replace(',',' ')
df_distribuidores['id'] = df_distribuidores['id'].astype('int')
```

```
df_distribuidores.head(3)
```



	id	distributor	distributor activities	years in the construction market
0	565	Abernathy-Hayes	construction materials import/distribution, ir...	15
1	1384	Balistreri LLC	construction materials import/distribution, si...	22
2	1183	Brekke-Stiedemann	construction materials import/distribution	13


## DataFrame de Localidades

```
df_localidades = pd.read_csv(URL + 'localidades')
df_localidades.head()
```



	PYid	id	location	department	activities
0	71	NaN	Abai	Caazapá	food and beverage products manufacture, other ...
1	224	NaN	Achay	Paraguarí	food and beverage products manufacture, other ...
2	208	NaN	Alberdi	Ñeembucú	food and beverage products manufacture, other ...
3	168	NaN	Alto Verá	Itapúa	food and beverage products manufacture, other ...
4	7	409.0	Altos	Cordillera	agriculture, livestock, hunting and related, l...

```
df_localidades.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272 entries, 0 to 271
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PYid        272 non-null   int64
1   id          11 non-null    float64
2   location    264 non-null   object
3   department  271 non-null   object
```

```
4   activities 271 non-null   object
dtypes: float64(1), int64(1), object(3)
memory usage: 10.8+ KB

# Eliminar registros donde 'activities' es nulo
if(df_localidades["activities"].isnull().sum() >= 1):
    df_localidades.dropna(subset=['activities'], inplace=True)

# Elimina duplicados si es que existen
df_dup_localidades = df_localidades[df_localidades.duplicated()]
if(df_dup_localidades.empty == False):
    df_dup_localidades.drop_duplicates(inplace=True)

# Buscamos los valores nulos
# Copiamos el dataframe original para poder determinar cuales son los distribuidores de la empresa y seguir teniendo los valores de las
if(df_localidades['id'].isnull().sum() >= 1):
    df_localidades_distribuidores = df_localidades.copy()
    df_localidades_distribuidores.dropna(subset=['id'],inplace=True)
    df_localidades_distribuidores['id'] = df_localidades_distribuidores['id'].astype('int')
```

```
df_localidades.info()
df_localidades_distribuidores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 271 entries, 0 to 271
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PYid         271 non-null   int64
1   id           10 non-null    float64
2   location     264 non-null   object
3   department   271 non-null   object
4   activities   271 non-null   object
dtypes: float64(1), int64(1), object(3)
memory usage: 12.7+ KB

<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, 4 to 246
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PYid         10 non-null    int64
1   id           10 non-null    int64
2   location     10 non-null    object
3   department   10 non-null    object
4   activities   10 non-null    object
dtypes: int64(2), object(3)
memory usage: 480.0+ bytes
```

```
df_localidades_distribuidores.head(3)
```

	PYid	id	location	department	activities
4	7	409	Altos	Cordillera	agriculture, livestock, hunting and related, l...
45	11	523	Coronel Martínez	Guairá	agriculture, livestock, hunting and related
81	5	519	Guayaybi	San Pedro	agriculture, livestock, hunting and related

## Análisis de datos

Armado del dataframe y gráfico de barras con los totales de ventas, exportaciones y el ratio entre ambos.

```
# Elimina la columna 'distributor' antes de sumar
df_export = df_exportado.drop(columns='distributor')
df_sales = df_ventas.drop(columns='distributor')

# Suma de los valores de exportación y venta por producto
sum_export = df_export.sum() / 1000000
sum_sales = df_sales.sum() / 1000000

# Cálculo del ratio ventas/exportaciones
ratio_sales_export = sum_sales / sum_export

# Crear un nuevo dataframe con los resultados
df_result = pd.DataFrame({
    'Total Exportado (en millones)': sum_export,
    'Total Vendido (en millones)': sum_sales,
    'Ratio Ventas/Exportado': ratio_sales_export
})

df_result.index.name = 'Producto'
df_result.sort_index()
```

	Total Exportado (en millones)	Total Vendido (en millones)	Ratio Ventas/Exportado
Producto			
Aluminum	1513.169630	2163.832571	1.430000
Brass	1394.438925	1840.659381	1.320000
Brick	5260.549009	0.152658	0.000029
Glass	1429.786765	2173.275883	1.520000
Granite	1337.169964	1845.294550	1.380000
Plastic	1456.143794	2047.338174	1.406000
Plexiglass	20779.232535	25974.040669	1.250000
Rubber	1465.042482	1680.453941	1.147034
Steel	13652.930026	0.058458	0.000004
Stone	1474.570889	1651.519396	1.120000
Vinyl	1396.050453	1912.589121	1.370000
Wood	1409.171577	1930.565060	1.370000

```
# Definir el valor umbral
umbral = 1.0

# Definición de colores
debajo_color = ['#ef233c' if ratio < umbral else '#74c69d' for ratio in df_result['Ratio Ventas/Exportado']]
alto_color = ['#52b788' if ratio < 1.2
              else '#40916c' if ratio < 1.4
              else '#2d6a4f' if ratio < 1.6
              else '#1b4332' if ratio < 1.8
              else '#081c15' for ratio in df_result['Ratio Ventas/Exportado']]

# Crear las barras debajo del umbral
debajo_umbral = df_result['Ratio Ventas/Exportado'].clip(upper=umbral)

# Crear las barras encima del umbral
sobre_umbral = df_result['Ratio Ventas/Exportado'] - debajo_umbral

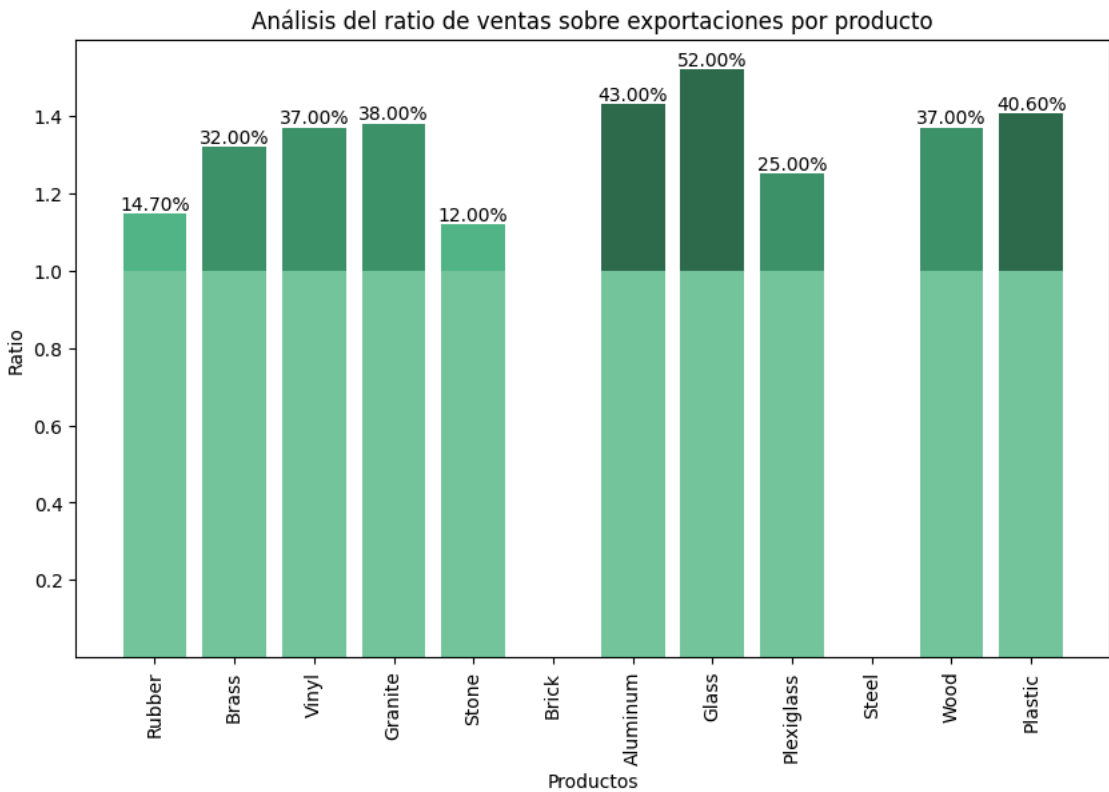
# Crear el gráfico
fig, ax = plt.subplots(figsize=(10, 6))

# Barras debajo del umbral
for i, (producto, fila) in enumerate(df_result.iterrows()):
    ax.bar(producto, debajo_umbral[i], color=debajo_color[i])

# Barras encima del umbral
for i, (producto, fila) in enumerate(df_result.iterrows()):
    ax.bar(producto, sobre_umbral[i], bottom=debajo_umbral[i], color=alto_color[i])
    if fila['Ratio Ventas/Exportado'] > umbral:
        percent = (fila['Ratio Ventas/Exportado'] - 1) * 100
        ax.text(producto, fila['Ratio Ventas/Exportado'], f'{percent:.2f}%', ha='center', va='bottom', fontsize=10)

# Crear labels y título
ax.set_xlabel("Productos")
ax.set_ylabel("Ratio")
plt.title('Análisis del ratio de ventas sobre exportaciones por producto')
plt.xticks(rotation=90)
```

```
# Mostrar gráfico
plt.show()
```



```
df_result.iloc[[5,9]]
```



Producto	Total Exportado (en millones)	Total Vendido (en millones)	Ratio Ventas/Exportado
Brick	5260.549009	0.152658	0.000029
Steel	13652.930026	0.058458	0.000004

A través del análisis del gráfico generado en la parte superior, podemos visualizar la relación entre las ventas y las exportaciones. Es evidente que existen dos productos enviados por la compañía a sus distintos distribuidores en Paraguay que no muestran los resultados esperados al compararlos con los demás productos. En contraste, otros como el vidrio muestran ganancias de alrededor del 50%.

Estos dos productos, "acero" y "ladrillos", no fueron vendidos en la proporción esperada por CMM. Esto podría deberse a la falta de un análisis previo del mercado de destino.

Por esta razón, intentamos estimar si en las localidades donde la empresa posee distribuidores existen registros de ventas y/o exportaciones.

```
# Queremos averiguar si en las locaciones donde CMM tiene distribuidores, hay datos de ventas y exportaciones.
```

```
# Seleccionar solo las columnas necesarias de df_ventas y df_exportado
df_ventas_selected = df_ventas[['distributor', 'Brick', 'Steel']]
df_exportado_selected = df_exportado[['distributor', 'Brick', 'Steel']]

# Merge df_localidades_distribuidores con df_ventas
merged_df = df_localidades_distribuidores.merge(df_ventas_selected, left_on='id', right_on='distributor')

# Eliminar la columna 'distributor' duplicada
merged_df.drop(columns='distributor', inplace=True)

# Renombrar las columnas para evitar conflictos
merged_df.rename(columns={'Brick': 'Ventas_Brick', 'Steel': 'Ventas_Steel'}, inplace=True)

# Merge del DataFrame resultante con df_exportado
final_df = merged_df.merge(df_exportado_selected, left_on='id', right_on='distributor')
final_df.drop(columns='distributor', inplace=True)
final_df.rename(columns={'Brick': 'Exportado_Brick', 'Steel': 'Exportado_Steel'}, inplace=True)

display(final_df)
```



PYid	id	location	department	activities	Ventas_Brick	Ventas_Steel	Exportado_Brick	Exportado_Steel
------	----	----------	------------	------------	--------------	--------------	-----------------	-----------------

Tras el análisis concluimos que no se tienen registros de ventas o exportaciones en las localidades proporcionadas.

Para intentar entender el motivo del bajo retorno de inversión en los materiales de acero y ladrillo, observamos las actividades industriales desarrolladas en Paraguay, distribuidas en las distintas localidades.

```
# Separar las actividades en una lista
list_actividades = df_localidades['activities'].str.split(',')

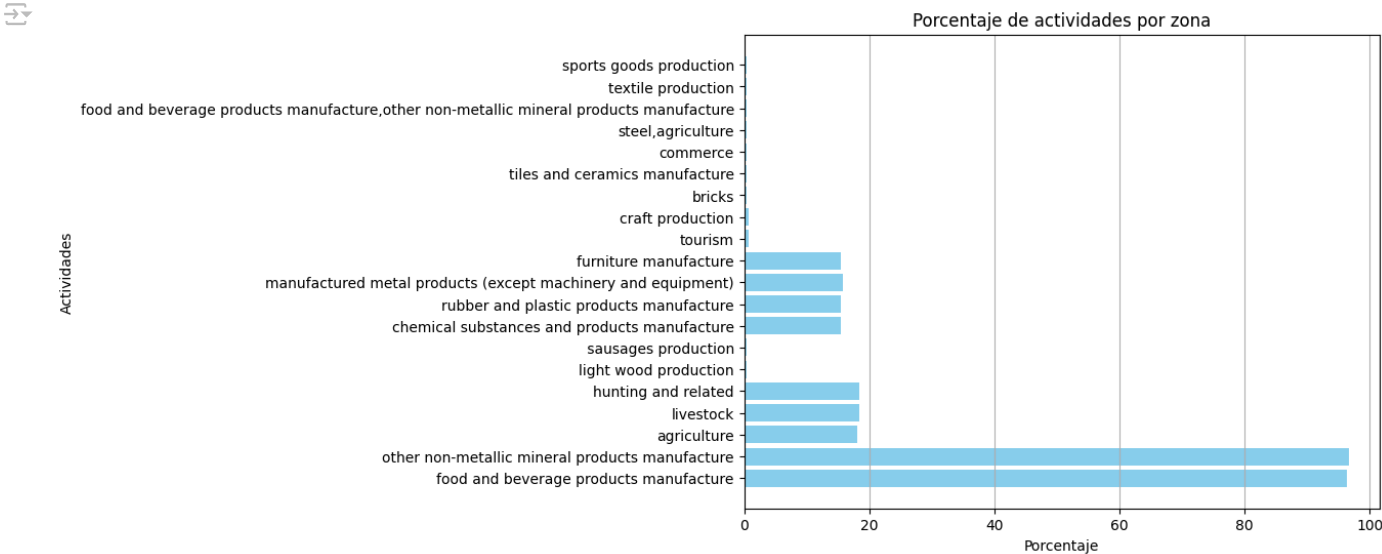
# Crear un diccionario para contar la frecuencia de cada actividad
activity_counts = {}

for activities in list_actividades:
    for activity in activities:
        if activity in activity_counts:
            activity_counts[activity] += 1
        else:
            activity_counts[activity] = 1

# Convertir el diccionario a un DataFrame
df_activity_counts = pd.DataFrame(list(activity_counts.items()), columns=['Activity', 'Count'])

# Calcular los porcentajes respecto del total de ubicaciones
total_locations = len(df_localidades)
df_activity_counts['Percentage'] = (df_activity_counts['Count'] / total_locations) * 100

plt.figure(figsize=(8, 6))
plt.barh(df_activity_counts['Activity'], df_activity_counts['Percentage'], color='skyblue')
plt.xlabel('Porcentaje')
plt.ylabel('Actividades')
plt.title('Porcentaje de actividades por zona')
plt.grid(axis='x')
plt.show()
```



Observamos que en casi la totalidad de las localidades (271) proporcionadas por el equipo de recolección de datos, se desarrollan actividades industriales relacionadas con la elaboración de productos alimenticios y bebidas, y la fabricación de otros productos minerales no metálicos. Estos datos se correlacionan con el Mapeo de Industrias del Paraguay registrado en el Ministerio de Industria y Comercio en 2018, que estima que el 26.6% de las actividades del país corresponden a estos rubros.

```
#Porcentaje de representación a nivel país de las actividades productivas en conflicto.
df_activity_counts.iloc[[9,13]]
```

	Activity	Count	Percentage
9	manufactured metal products (except machinery ...	43	15.867159
13	bricks	1	0.369004



```
#Zonas donde se desarrollan actividades relacionadas con la producción de ladrillos.
df_localidades[df_localidades["activities"].str.contains("bricks")]
```

	PYid	id	location	department	activities
226	4	102.0	Tobatí	Cordillera	bricks, tiles and ceramics manufacture, craft ...

Sin embargo, en relación a los productos que han generado un problema a CMM, podemos destacar que la fabricación de productos elaborados de metal tiene una gran predominancia en la actividad productiva del país, ocupando el 3º puesto con un 7.1% del mercado y distribuyéndose en un 15.8% del territorio.

El caso de la elaboración de ladrillos es diferente, ya que está fuertemente concentrada en la zona cordillerana. Esta actividad ocupa el 7º puesto en la producción industrial de Paraguay con un 5.1%, según el Ministerio de Industria y Comercio.

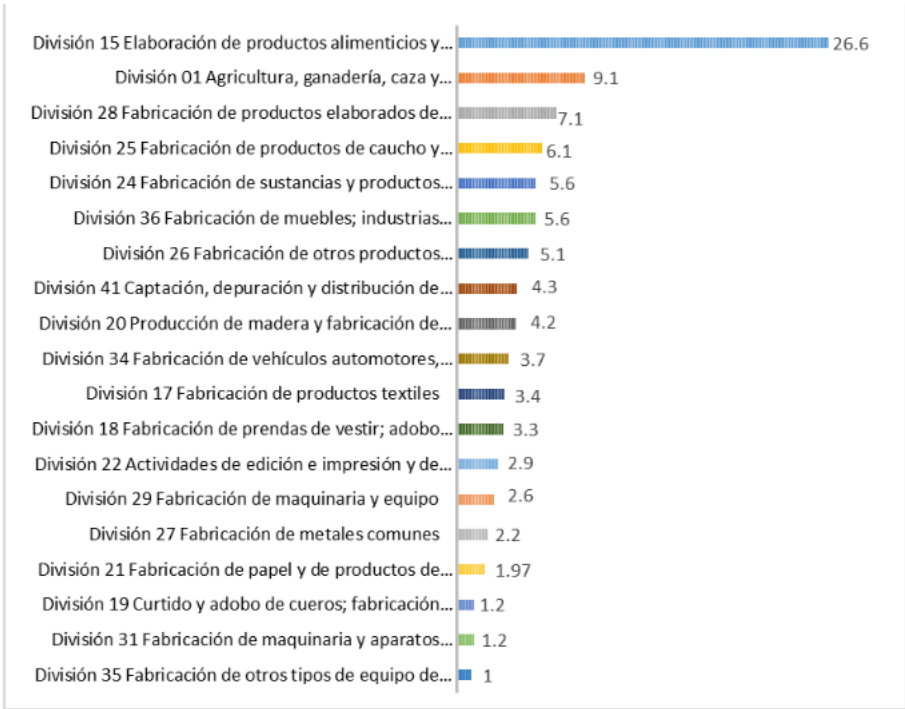


Figura 2. Distribución porcentual de industrias de mayor concentración en base al total registrado, categorizadas por División

Fuente: Mapeo de industrias del Paraguay registradas en el Ministerio de Industria y Comercio (<https://revistacientifica.sudamericana.edu.py/index.php/scientiamericana/article/view/175/194>),

Por esta razón, el hecho de que tanto el acero como los ladrillos se produzcan localmente en el país explica por qué las ventas comparadas con las exportaciones de estos materiales han sido tan bajas. Este problema se refleja en el valor del ratio ventas/exportaciones, que es extremadamente bajo. Esto sugiere que los productos de acero y ladrillos no encontraron un mercado adecuado en las localidades de destino debido a la saturación de la producción local.

En conclusión, para mejorar la efectividad de las exportaciones y ventas, es crucial realizar un análisis más detallado del mercado local y considerar factores como la producción local existente y la demanda en las regiones objetivo.