

Modelo Predictivo de Ingresos Netos para la Planificación Estratégica de Deutsche Bank

Introducción

El siguiente proyecto de Ciencia de Datos consiste en desarrollar un modelo predictivo de los Ingresos Netos de Deutsche Bank en los próximos cuatro trimestres con la intención de conocer si invertir en Deutsche Bank es seguro o riesgoso y para ser una herramienta para Deutsche Bank que sirva para definir su planeación estratégica en base a los Ingresos Netos.

Poder predecir los Ingresos Netos es fundamental dado que se vive en un entorno dinámico y una economía altamente competitiva, por lo que, la falta de un modelo de pronóstico sistemático y basado en datos para el Ingreso Neto (Net_Income) expone a la organización a riesgos significativos, incluyendo:

- Planificación ineficaz: Asignación de capital y recursos basada en proyecciones imprecisas.
- Pérdida de confianza de los inversionistas: Incumplimiento de las expectativas del mercado debido a resultados financieros inesperados.
- Reacciones tardías a tendencias del mercado: Incapacidad para identificar y actuar sobre patrones emergentes que afectan la rentabilidad.

Este proyecto aborda directamente la necesidad de pasar de un enfoque reactivo a uno proactivo y predictivo en la gestión financiera, lo que permite a Deutsche Bank ser altamente competitivo.

Los datos a utilizar para realizar el análisis y el desarrollo del modelo predictivo será el dataset de Kaggle "Rendimiento Financiero del Deutsche Bank", que contiene 15 variables financieras registradas desde enero de 2015. La variable objetivo es Net_Income. Variables como Ingresos, Gastos y Operating_Income se utilizarán como posibles variables exógenas (regresores) para mejorar la precisión del modelo.

Metodología

• Proceso de Limpieza de datos

El proceso de limpieza del dataset fue riguroso ya que existían valores nulos, texto en columnas, filas duplicadas y valores nulos. Con un análisis exploratorio previo se pudo localizar todos estos hallazgos, lo que me permitió tener claridad de los pasos a seguir para realizar una limpieza exitosa. Las técnicas usadas para realizar la limpieza fueron:

1. Eliminar filas duplicadas con "drop_duplicates"
2. Coercionar las filas a formato numérico (a excepción de la columna "Date")
3. Sustituir los valores de texto por el valor promedio de la columna
4. Sustituir los valores nulos por el valor promedio de la columna
5. Traducir los nombres de las columnas al español
6. Visualizar los outliers para su tratamiento en la fase de análisis

Durante el proceso aprendí a realizar una limpieza ordenada y estructurada que me permitió diseñar el proceso a seguir para lograr limpiar el dataset de manera correcta y lista para su análisis.

• Análisis Exploratorio de Datos (EDA)

```
In [63]: # Cargar bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [64]: # Cargar dataset de Github
url = "https://raw.githubusercontent.com/Edgardo2501/Proyecto_Ciencia_de_Datos/main/df_deutsche_bank.csv"
df = pd.read_csv(url)

# Mostrar las primeras filas del dataset
print(df.head())
```

```

      Fecha  Ingreso_Total  Gastos_Operativos  Ingreso_Operativo \
0  2015-01-01        9435946.42        3682573.85        4370861.07
1  2015-01-02          NaN            1186425.69        9556428.76
2  2015-01-03          NaN            3093297.62        7587945.48
3  2015-01-04       4000698.76        3230217.71          NaN
4  2015-01-05       3940243.11        2408588.02        2404167.76

      Gastos_Financieros  Impuestos  Ingreso_Neto  Dividendos  Activos \
0           609472.14  1042103.16       688287.22     1170151.42  1.363403e+08
1           699770.11  1329539.73      8370003.06      492998.93  1.955172e+08
2           337977.34          NaN      4494647.86     1603358.92  1.519954e+08
3           1345468.95  1316373.62      3157708.65      426566.77  2.097483e+08
4           175614.54  136655.30      -4420.26     2808563.51   8.124073e+07

      Pasivos  Patrimonio_Neto  Flujo_Efectivo  Ratio_deuda_patrimonio \
0  2.095012e+08      59803407.08      1428845.20             2.50
1  4.725052e+07      55281922.53      1029017.28             0.85
2  1.756590e+08          NaN      7320721.28             6.40
3  1.382626e+08      87293509.41      1925965.75             1.58
4  2.773058e+08      70924670.47      6659376.16             3.91

      ROA  Margen_Utilidad
0  0.01          0.07
1  0.04          0.67
2  0.03          0.38
3  0.02          0.79
4  0.00          0.00

```

Descripción General de los Datos

In [65]: # Número de filas y columnas en el dataset

```
print("El dataset contiene {} filas y {} columnas.".format(df.shape[0], df.shape[1]))
```

El dataset contiene 6800 filas y 15 columnas.

In [66]: # Tipos de variables en el dataset
Crear un dataframe con Los tipos de datos
tipos = df.dtypes.reset_index()
tipos.columns = ['Variable', 'Tipo de Dato']
print(tipos.to_markdown(index=False))

Variable	Tipo de Dato
Fecha	object
Ingreso_Total	float64
Gastos_Operativos	float64
Ingreso_Operativo	float64
Gastos_Financieros	float64
Impuestos	float64
Ingreso_Neto	float64
Dividendos	float64
Activos	float64
Pasivos	float64
Patrimonio_Neto	float64
Flujo_Efectivo	float64
Ratio_deuda_patrimonio	float64
ROA	float64
Margen_Utilidad	float64

In [67]: # Convertir la variable Fecha a tipo datetime
df['Fecha'] = df['Fecha'].astype('datetime64[ns]')

In [68]: # Resumen estadístico del dataset
print(df.describe(include='all').to_markdown())

	Fecha	Ingreso_Total	Gastos_Operativos	Ingresa_Operativo	Gastos_Financieros	Impuestos	Ingreso_Neto	Dividendos	Activos	Pasivos	Patrimonio_Neto
	Flujo_Efectivo	Ratio_deuda_patrimonio	ROA	Margen_Utilidad							
count	6800	6463	6466	6332	6465	6800	6334	6800	6461	6800	6465
6454		6333	6459	6465	5.48331e+06	1.05021e+06	766076		2.75491e+06	1.48551e+06	2.77717e+08
mean	2024-04-22 12:00:00	8.43581e+06	2.7489e+06	5.48331e+06	0.433497				2.75491e+06	1.48551e+06	2.09501e+08
+07		4.22339e+06	5.46768	0.013629							5.44224e
min	2015-01-01 00:00:00	2.0004e+06	520844		1.04555e+06	107445	50349.5		-3.75134e+06	101164	5.07043e+07
+07	507076		0.24	-0.04	-1.53						2.00044e+07
25%	2019-08-27 18:00:00	5.22008e+06	1.54832e+06	3.18223e+06	583165	399226	670263	747441		1.7659e+08	1.25584e+08
+07		2.44393e+06	2.06	0	0.06						3.10983e
50%	2024-04-22 12:00:00	8.57406e+06	2.8297e+06	5.56849e+06		1.05021e+06	750370		2.75491e+06	1.46528e+06	2.77717e+08
+07		4.11951e+06	3.88	0.01	0.33						2.09501e+08
75%	2028-12-17 06:00:00	1.15468e+07	3.84579e+06	7.75784e+06		1.50849e+06	1.14734e+06		4.78007e+06	2.15542e+06	3.81879e+08
07		6.05498e+06	6.9	0.02	0.63						2.98439e+08
max	2033-08-13 00:00:00	1.49707e+07	4.99736e+06	9.99746e+06		1.99898e+06	1.49885e+06		9.11147e+06	2.99682e+06	4.99256e+08
+07		7.99095e+06	35.45	0.13	3.59						3.99172e+08
std	nan	3.71908e+06	1.32281e+06	2.63123e+06	535953	418770			2.85528e+06	831432	
+07		2.14384e+06	5.26072	0.0209903	0.614244						1.24828e+08

Visualización y Distribución de variables

```
In [69]: # Visualización y Distribución de variables que componen el Estado de Resultados (Histogramas)
# Configurar el estilo de las gráficas
fig, axes = plt.subplots(4, 2, figsize=(12, 9))

# Aplanar el arreglo de ejes para indexarlos fácilmente
axes = axes.ravel()

# Distribución del Ingreso Total
sns.histplot(df['Ingreso_Total'], bins=30, color="#00008B", kde=True, ax=axes[0])
axes[0].set_title('Gráfica 1. Distribución del Ingreso Total')
axes[0].set_xlabel('Ingreso Total')
axes[0].set_ylabel('Frecuencia')

# Distribución de los Gastos Operativos
sns.histplot(df['Gastos_Operativos'], bins=30, color="#104E8B", kde=True, ax=axes[1])
axes[1].set_title('Gráfica 2. Distribución de los Gastos Operativos')
axes[1].set_xlabel('Gastos Operativos')
axes[1].set_ylabel('Frecuencia')

# Distribución del Ingreso Operativo
sns.histplot(df['Ingreso_Operativo'], bins=30, color="#8B2500", kde=True, ax=axes[2])
axes[2].set_title('Gráfica 3. Distribución del Ingreso Operativo')
axes[2].set_xlabel('Ingreso Operativo')
axes[2].set_ylabel('Frecuencia')

# Distribución de los Gastos Financieros
sns.histplot(df['Gastos_Financieros'], bins=30, color="#8B4789", kde=True, ax=axes[3])
axes[3].set_title('Gráfica 4. Distribución de los Gastos Financieros')
axes[3].set_xlabel('Gastos Financieros')
axes[3].set_ylabel('Frecuencia')

# Distribución de los Impuestos
sns.histplot(df['Impuestos'], bins=30, color="#8B0000", kde=True, ax=axes[4])
axes[4].set_title('Gráfica 5. Distribución de los Impuestos')
axes[4].set_xlabel('Impuestos')
axes[4].set_ylabel('Frecuencia')

# Distribución del Ingreso Neto
sns.histplot(df['Ingreso_Neto'], bins=30, kde=True, ax=axes[5])
axes[5].set_title('Gráfica 6. Distribución del Ingreso Neto')
axes[5].set_xlabel('Ingreso Neto')
axes[5].set_ylabel('Frecuencia')

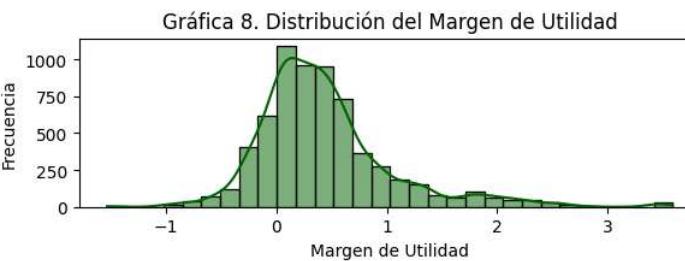
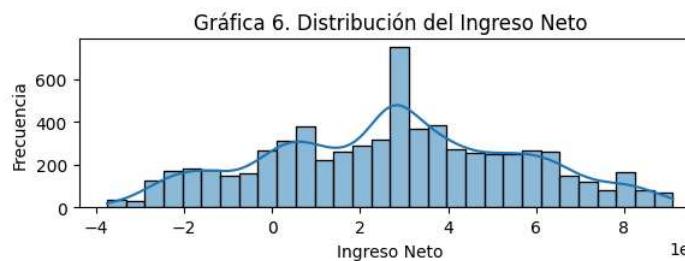
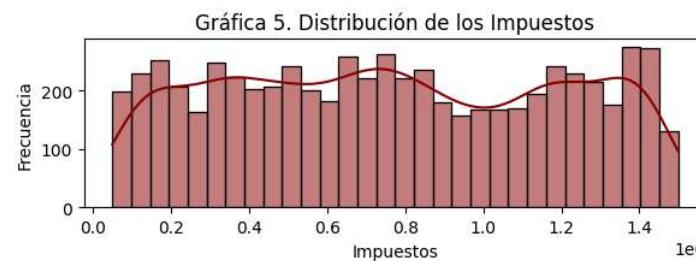
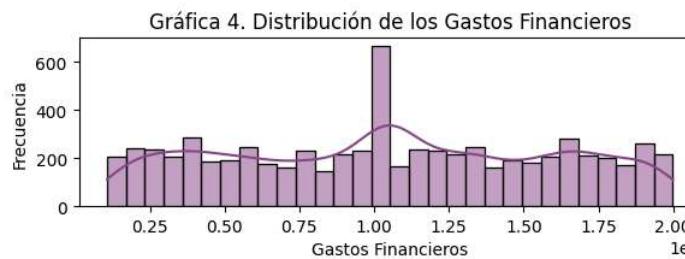
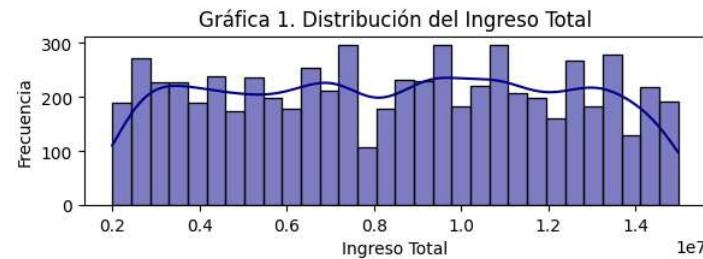
# Distribución de los Dividendos
```

```
sns.histplot(df['Dividendos'], bins=30, color="#8B8B00", kde=True, ax=axes[6])
axes[6].set_title('Gráfica 7. Distribución de los Dividendos')
axes[6].set_xlabel('Dividendos')
axes[6].set_ylabel('Frecuencia')

# Distribución del Margen de Utilidad
sns.histplot(df['Margen_Utilidad'], bins=30, color="#006400", kde=True, ax=axes[7])
axes[7].set_title('Gráfica 8. Distribución del Margen de Utilidad')
axes[7].set_xlabel('Margen de Utilidad')
axes[7].set_ylabel('Frecuencia')

# Ajustar el layout para evitar superposición de títulos y etiquetas
plt.tight_layout()

# Mostrar las gráficas
plt.show()
```



Como se puede observar en las gráficas de las variables que componen el Estado de Resultados, la distribución que tienen es normal, por lo que nuestras medidas de tendencia normal son estadísticamente significativas y no se requiere alguna normalización o transformación.

```
In [70]: # Visualización y Distribución de Las demás razones financieras (Histogramas)
# Configurar el estilo de las gráficas
fig, axes = plt.subplots(3, 2, figsize=(12, 9))

# Aplanar el arreglo de ejes para indexarlos fácilmente
axes = axes.ravel()

# Distribución de Los Activos
sns.histplot(df['Activos'], bins=30, color="#88A513", kde=True, ax=axes[0])
axes[0].set_title('Gráfica 1. Distribución de los Activos')
axes[0].set_xlabel('Activos')
```

```
axes[0].set_ylabel('Frecuencia')

# Distribución de los Pasivos
sns.histplot(df['Pasivos'], bins=30, color="#2F4F4F", kde=True, ax=axes[1])
axes[1].set_title('Gráfica 2. Distribución de los Pasivos')
axes[1].set_xlabel('Pasivos')
axes[1].set_ylabel('Frecuencia')

# Distribución del Patrimonio Neto
sns.histplot(df['Patrimonio_Neto'], bins=30, color="#B22222", kde=True, ax=axes[2])
axes[2].set_title('Gráfica 3. Distribución del Patrimonio Neto')
axes[2].set_xlabel('Patrimonio Neto')
axes[2].set_ylabel('Frecuencia')

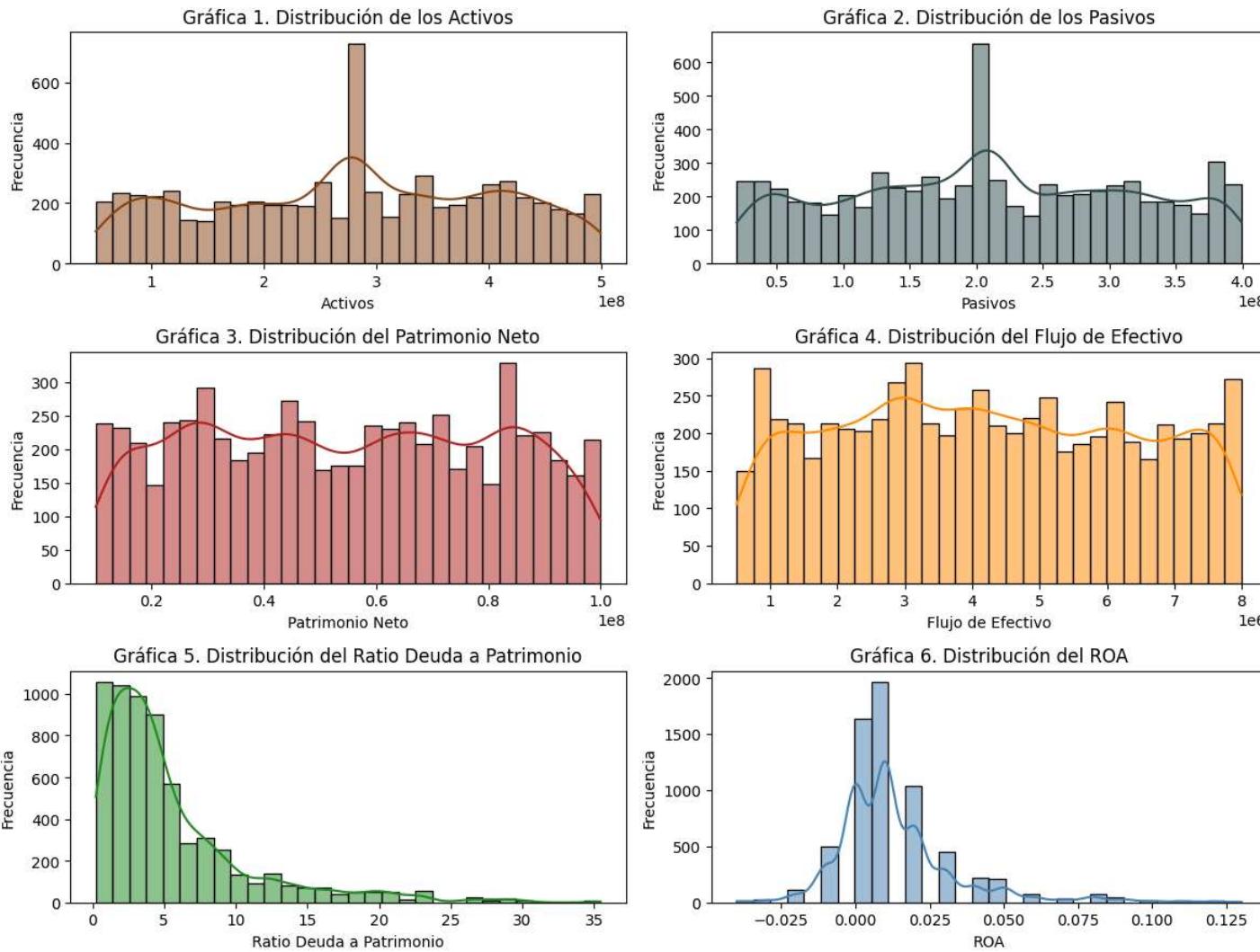
# Distribución del Flujo de Efectivo
sns.histplot(df['Flujo_Efectivo'], bins=30, color="#FF8C00", kde=True, ax=axes[3])
axes[3].set_title('Gráfica 4. Distribución del Flujo de Efectivo')
axes[3].set_xlabel('Flujo de Efectivo')
axes[3].set_ylabel('Frecuencia')

# Distribución del Ratio Deuda a Patrimonio
sns.histplot(df['Ratio_deuda_patrimonio'], bins=30, color="#228B22", kde=True, ax=axes[4])
axes[4].set_title('Gráfica 5. Distribución del Ratio Deuda a Patrimonio')
axes[4].set_xlabel('Ratio Deuda a Patrimonio')
axes[4].set_ylabel('Frecuencia')

# Distribución del ROA
sns.histplot(df['ROA'], bins=30, color="#4682B4", kde=True, ax=axes[5])
axes[5].set_title('Gráfica 6. Distribución del ROA')
axes[5].set_xlabel('ROA')
axes[5].set_ylabel('Frecuencia')

# Ajustar el layout para evitar superposición de títulos y etiquetas
plt.tight_layout()

# Mostrar las gráficas
plt.show()
```



A excepción de las variable Ratio Deuda a Patrimonio, las demás razones financieras mantienen una distribución normal, por lo que nuestras medidas de tendencia normal son estadísticamente significativas y no se requiere alguna normalización o transformación.

```
In [71]: # Visualización y Distribución de variables que componen el Estado de Resultados (Boxplots)
# Configurar el estilo de las gráficas
fig, axes = plt.subplots(4, 2, figsize=(12, 9))

# Aplanar el arreglo de ejes para indexarlos fácilmente
axes = axes.ravel()

# Distribución del Ingreso Total
sns.boxplot(x=df['Ingreso_Total'], color="lightskyblue", ax=axes[0])
axes[0].set_title('Gráfica 1. Distribución del Ingreso Total')
axes[0].set_xlabel('Ingreso Total')
```

```
axes[0].set_ylabel('Valores')

# Distribución de los Gastos Operativos
sns.boxplot(x=df['Gastos_Operativos'], color="#21609F", ax=axes[1])
axes[1].set_title('Gráfica 2. Distribución de los Gastos Operativos')
axes[1].set_xlabel('Gastos Operativos')
axes[1].set_ylabel('Valores')

# Distribución del Ingreso Operativo
sns.boxplot(x=df['Ingreso_Operativo'], color="#6C7B8B", ax=axes[2])
axes[2].set_title('Gráfica 3. Distribución del Ingreso Operativo')
axes[2].set_xlabel('Ingreso Operativo')
axes[2].set_ylabel('Valores')

# Distribución de los Gastos Financieros
sns.boxplot(x=df['Gastos_Financieros'], color="#8B4789", ax=axes[3])
axes[3].set_title('Gráfica 4. Distribución de los Gastos Financieros')
axes[3].set_xlabel('Gastos Financieros')
axes[3].set_ylabel('Valores')

# Distribución de los Impuestos
sns.boxplot(x=df['Impuestos'], color="#8B0000", ax=axes[4])
axes[4].set_title('Gráfica 5. Distribución de los Impuestos')
axes[4].set_xlabel('Impuestos')
axes[4].set_ylabel('Valores')

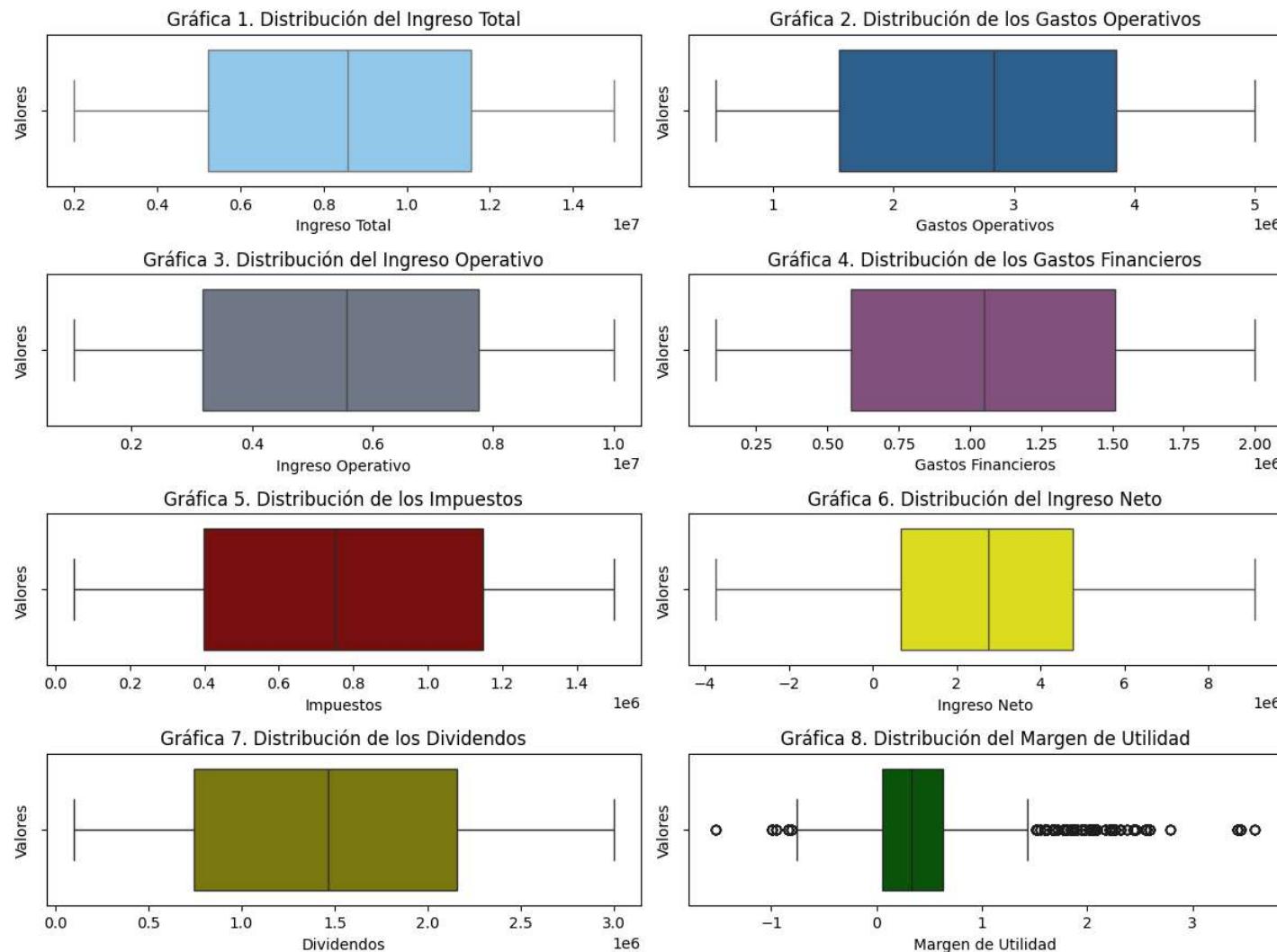
# Distribución del Ingreso Neto
sns.boxplot(x=df['Ingreso_Neto'], color="yellow", ax=axes[5])
axes[5].set_title('Gráfica 6. Distribución del Ingreso Neto')
axes[5].set_xlabel('Ingreso Neto')
axes[5].set_ylabel('Valores')

# Distribución de los Dividendos
sns.boxplot(x=df['Dividendos'], color="#8B8B00", ax=axes[6])
axes[6].set_title('Gráfica 7. Distribución de los Dividendos')
axes[6].set_xlabel('Dividendos')
axes[6].set_ylabel('Valores')

# Distribución del Margen de Utilidad
sns.boxplot(x=df['Margen_Utilidad'], color="#006400", ax=axes[7])
axes[7].set_title('Gráfica 8. Distribución del Margen de Utilidad')
axes[7].set_xlabel('Margen de Utilidad')
axes[7].set_ylabel('Valores')

# Ajustar el layout para evitar superposición de títulos y etiquetas
plt.tight_layout()

# Mostrar las gráficas
plt.show()
```



Como se puede observar, a excepción del Margen de Utilidad, las demás variables no presentan valores extremos y el 50% de sus datos se concentran cerca de la mediana. El caso del Margen de Utilidad es importante analizar el caso por separado.

```
In [72]: # Visualización y Distribución de Las demás razones financieras (Boxplots)
# Configurar el estilo de las gráficas
fig, axes = plt.subplots(3, 2, figsize=(12, 9))

# Aplanar el arreglo de ejes para indexarlos fácilmente
axes = axes.ravel()

# Distribución de Los Activos
sns.boxplot(x=df['Activos'], color="#8B4513", ax=axes[0])
axes[0].set_title('Gráfica 1. Distribución de los Activos')

# Distribución de Los Pasivos
```

```
sns.boxplot(x=df['Pasivos'], color="#2F4F4F", ax=axes[1])
axes[1].set_title('Gráfica 2. Distribución de los Pasivos')

# Distribución del Patrimonio Neto
sns.boxplot(x=df['Patrimonio_Neto'], color="#B22222", ax=axes[2])
axes[2].set_title('Gráfica 3. Distribución del Patrimonio Neto')

# Distribución del Flujo de Efectivo
sns.boxplot(x=df['Flujo_Efectivo'], color="#FF8C00", ax=axes[3])
axes[3].set_title('Gráfica 4. Distribución del Flujo de Efectivo')

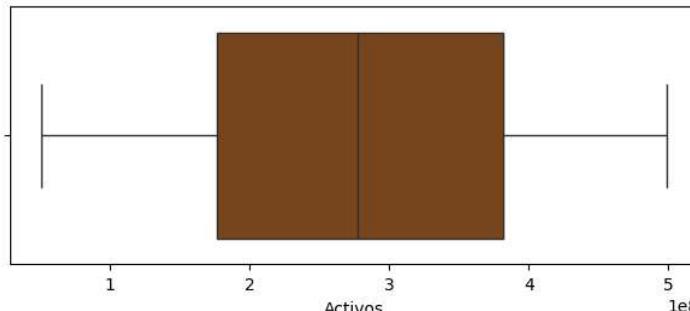
# Distribución del Ratio Deuda a Patrimonio
sns.boxplot(x=df['Ratio_deuda_patrimonio'], color="#228B22", ax=axes[4])
axes[4].set_title('Gráfica 5. Distribución del Ratio Deuda a Patrimonio')

# Distribución del ROA
sns.boxplot(x=df['ROA'], color="#4682B4", ax=axes[5])
axes[5].set_title('Gráfica 6. Distribución del ROA')

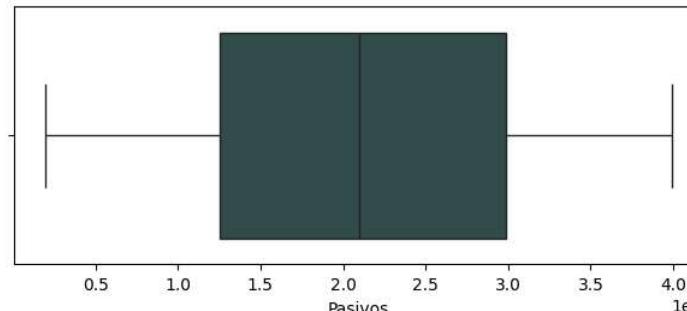
# Ajustar el Layout para evitar superposición de títulos y etiquetas
plt.tight_layout()

# Mostrar las gráficas
plt.show()
```

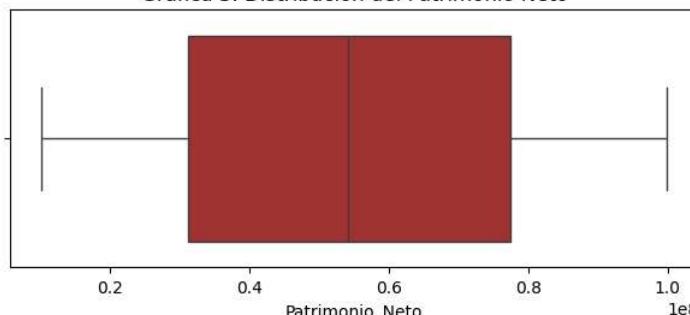
Gráfica 1. Distribución de los Activos



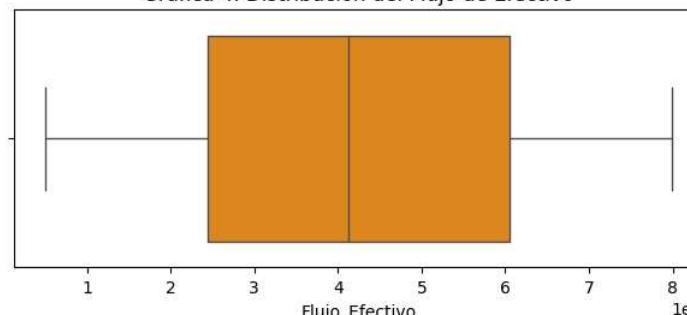
Gráfica 2. Distribución de los Pasivos



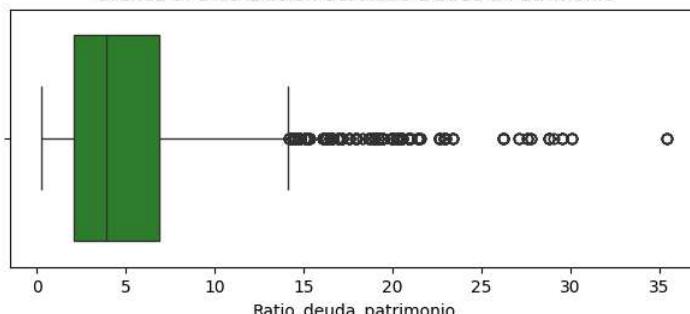
Gráfica 3. Distribución del Patrimonio Neto



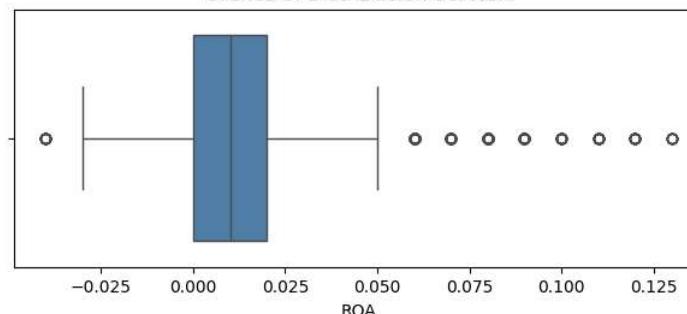
Gráfica 4. Distribución del Flujo de Efectivo



Gráfica 5. Distribución del Ratio Deuda a Patrimonio



Gráfica 6. Distribución del ROA



Lo que las gráfica de boxplot nos revelan es que la distribución del Ratio Deuda a Patrimonio y del ROA presentan valores extremos y el 50% de sus datos no se concentran cerca de la mediana.

3. Correlación entre Variables

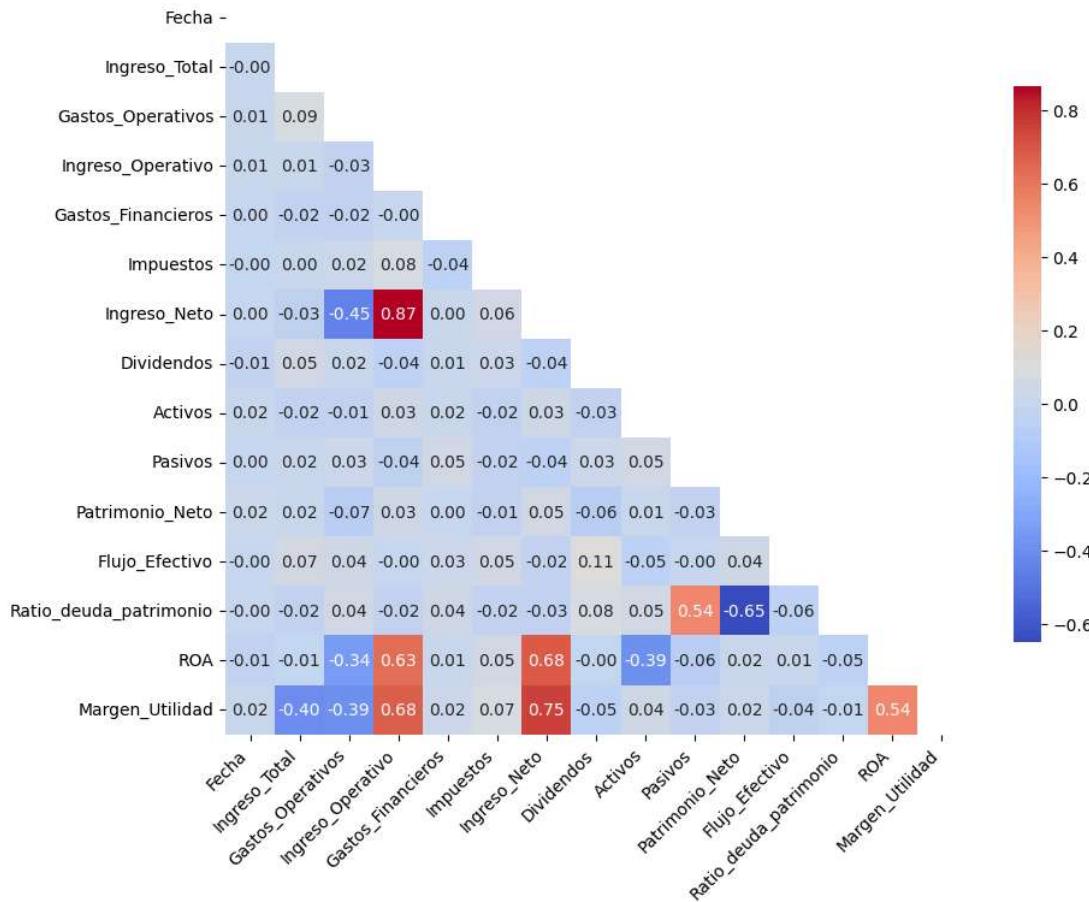
```
In [73]: # Realizamos un análisis de correlación entre las variables numéricas del dataset
correlation_matrix = df.corr()
print(correlation_matrix.to_markdown())
```

		Fecha	Ingreso_Total	Gastos_Operativos	Ingreso_Operativo	Gastos_Financieros	Impuestos	Ingreso_Neto	Dividendos	Activos	Pasivos	Patrimonio_Neto
Flujo_Efectivo	Ratio_deuda_patrimonio	ROA	Margen_Utilidad									
-0.00290792	-0.0027369	-0.0100498	0.0159546	0.00505795	0.00710418	0.00184033	-0.00479577	0.00213168	-0.00995705	0.0154109	0.00148196	0.0215986
0.0673709	-0.0175916	-0.00892799	-0.400541	0.0867347	0.00796596	-0.0225339	0.00450434	-0.0346589	0.0531352	-0.0244087	0.0156665	0.0199727
0.0437675	0.00505795	0.0867347	1	-0.341659	-0.0272533	-0.0175785	0.020345	-0.446563	0.0158804	-0.0121693	0.0266922	-0.0690481
0.0443318	-0.341659	-0.388809	-0.0272533	1	-0.00279745	0.0811811	0.865326	-0.0427804	0.0322438	-0.0367822	0.0324543	
-0.000728285	-0.0225458	0.625227	0.675656	-0.0225339	-0.0175785	-0.00279745	1	-0.0435155	0.00293386	0.00660629	0.019099	0.050304
0.0318415	0.00184033	0.0232774	0.00879523	0.0232774	0.00450434	0.020345	0.0811811	-0.0435155	0.0581534	0.0267689	-0.0226413	-0.0151364
0.0476212	-0.00479577	0.00450434	-0.0171915	0.0466957	0.0747995	0.020345	-0.0435155	1	-0.0398268	0.0339637	-0.0395311	-0.00989173
0.0443318	-0.0346589	-0.446563	-0.029671	0.68357	0.74766	0.865326	0.00293386	0.0581534	1	-0.0398268	0.0484389	
-0.0164899	0.00213168	-0.0531352	-0.00995705	0.0531352	0.0158804	-0.0427804	0.00660629	0.0267689	-0.0398268	1	-0.0308016	0.0282511
0.113006	0.0751387	-0.00469888	-0.0476407	-0.04644672	-0.0289529	-0.0427804	0.00660629	0.0267689	1	-0.0308016	-0.0649056	
0.0522236	0.0154109	-0.0244087	-0.0121693	-0.0244087	-0.0121693	0.0322438	0.019099	-0.0226413	0.0339637	-0.0308016	1	0.047279
0.0548679	-0.385306	0.0425539	-0.00148196	0.0156665	0.0266922	-0.0367822	0.050304	-0.0151364	-0.0395311	0.0282511	1	-0.0287744
0.00155067	0.537995	-0.0644672	-0.029671	-0.0644672	-0.0289529	-0.0289529	0.050304	-0.0151364	-0.0395311	0.0282511	1	
0.0215986	0.0215986	0.0199727	-0.647682	0.0151557	0.023203	-0.0690481	0.0324543	0.00147284	-0.00989173	0.0484389	-0.0649056	0.00913711
0.0390116	-0.647682	0.0151557	0.023203	-0.0690481	0.0437675	-0.000278285	0.0318415	0.0476212	-0.0164899	0.113006	-0.0522236	-0.00155067
1	-0.0550966	0.0102304	-0.0027369	-0.0175916	-0.0443318	-0.0225458	0.0370926	-0.0171915	-0.029671	0.0751387	0.0548679	0.537995
-0.0550966	1	-0.0495421	-0.0100498	-0.0495421	-0.00517838	-0.000278285	0.00879523	0.0466957	0.68357	-0.00469888	-0.385306	-0.0644672
0.0102304	-0.0495421	1	-0.0100498	-0.0495421	-0.341659	0.625227	0.00879523	0.0466957	0.68357	-0.00469888	-0.385306	0.0151557
Margen_Utilidad	0.0159546	-0.400541	-0.00517838	0.0159546	-0.388809	0.675656	0.0232774	0.0747995	0.74766	-0.0476407	0.0425539	-0.0289529
-0.0364115	0.54	1	0.54	0.54	1							0.023203

```
In [74]: # Grafica de Matriz de Correlación de las variables numéricas
# Crear la máscara para el triángulo superior.
# np.ones_Like(correlation_matrix) crea una matriz del mismo tamaño llena de unos.
# np.triu() crea una matriz triangular superior a partir de ella.
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

# Graficamos la matriz de correlación
plt.figure(figsize=(10, 8))
plt.title('Matriz de Correlación entre Variables Numéricas')
sns.heatmap(correlation_matrix,
            mask=mask,
            annot=True,
            fmt=".2f",
            cmap='coolwarm',
            square=True,
            cbar_kws={"shrink": 0.75})
plt.xticks(rotation=45, ha="right") # Rotar las etiquetas del eje x para mejor visibilidad
plt.show()
```

Matriz de Correlación entre Variables Numéricas



Interpretación:

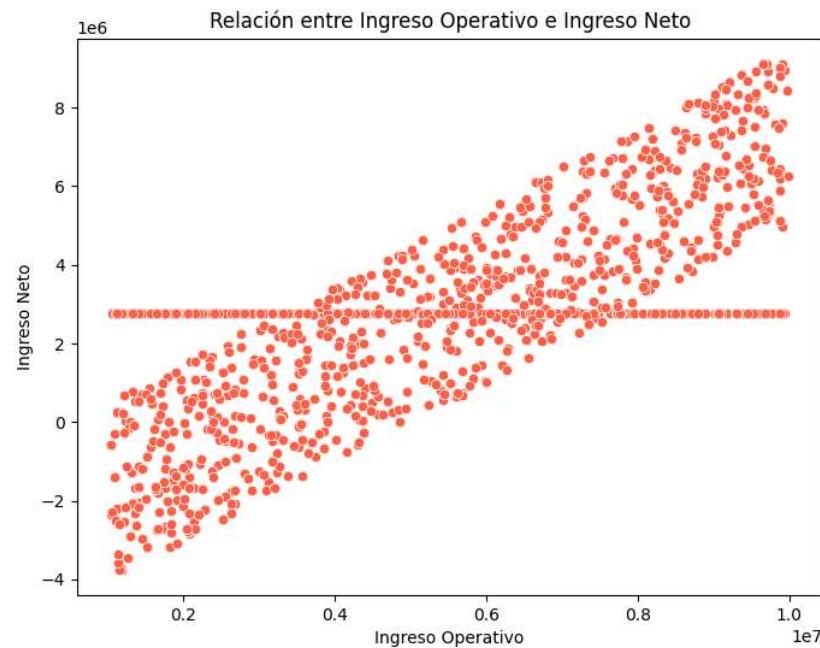
Se presentan los siguientes hallazgos para algunas variables:

- 1. Ingreso Neto e Ingreso Operativo:** Presentan un índice de correlación relativamente alto de signo positivo. Es decir, por cada aumento en el ingreso operativo, el ingreso neto aumenta en la misma proporción.
- 2. Ratio Deuda Patrimonio y Patrimonio Neto:** Presentan un índice de correlación casi alto de signo negativo. Es decir, por cada aumento en el patrimonio neto, el ratio deuda patrimonio disminuye en la misma proporción y viceversa.
- 3. ROA e Ingreso Neto:** Presentan un índice de correlación relativamente alto de signo positivo. Es decir, por cada aumento en el ROA, el ingreso neto aumenta en la misma proporción.
- 4. Margen de Utilidad e Ingreso Operativo:** Presentan un índice de correlación relativamente alto de signo positivo. Es decir, por cada aumento en el ingreso operativo, el margen de utilidad aumenta en la misma proporción.
- 5. Margen de Utilidad e Ingreso Neto:** Presentan un índice de correlación relativamente alto de signo positivo. Es decir, por cada aumento en el ingreso neto, el margen de utilidad aumenta en la misma proporción.

Para las demás variables no hay evidencia suficiente de que exista una relación lineal.

```
In [75]: # Análisis de dos variables específicas: Ingreso Neto vs Ingreso Operativo
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Ingreso_Operativo', y='Ingreso_Neto', data=df, color="#FF6347")
```

```
plt.title('Relación entre Ingreso Operativo e Ingreso Neto')
plt.xlabel('Ingreso Operativo')
plt.ylabel('Ingreso Neto')
plt.show()
```

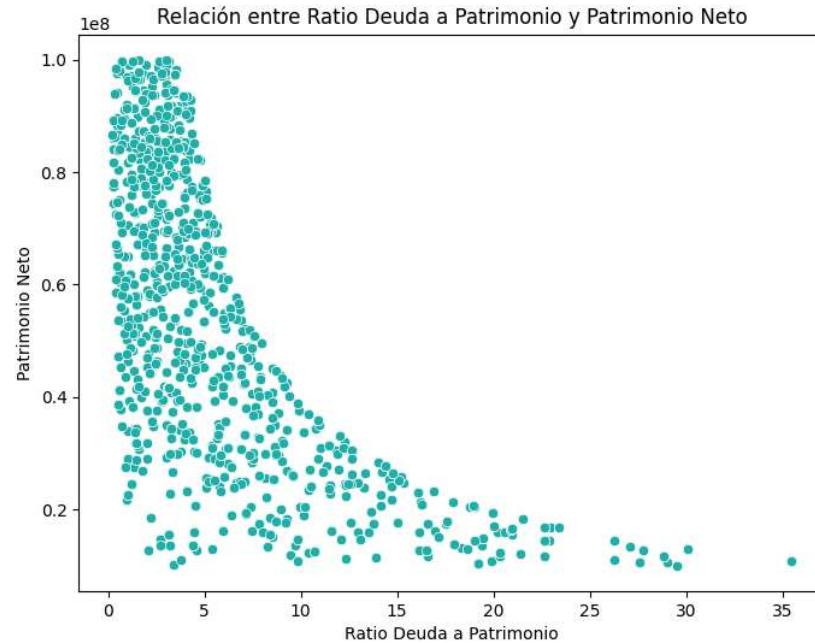


Interpretación:

La relación entre ****Ingreso Operativo**** e ***Ingreso Neto*** es positiva y fuerte, lo que indica que a medida que el Ingreso Operativo aumenta, el Ingreso Neto también tiende a aumentar.

Esto se podría explicar debido a que el Ingreso Neto es una función directa del Ingreso Operativo, después de considerar otros gastos e impuestos.

```
In [76]: # Análisis de dos variables específicas: Ratio Deuda a Patrimonio vs Patrimonio Neto
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Ratio_deuda_patrimonio', y='Patrimonio_Neto', data=df, color="#20B2AA")
plt.title('Relación entre Ratio Deuda a Patrimonio y Patrimonio Neto')
plt.xlabel('Ratio Deuda a Patrimonio')
plt.ylabel('Patrimonio Neto')
plt.show()
```



Interpretación:

La relación entre *Ratio Deuda a Patrimonio* y *Patrimonio Neto* parece ser negativa y débil. Esto sugiere que a medida que el *Ratio Deuda a Patrimonio* aumenta, el *Patrimonio Neto* tiende a disminuir ligeramente, pero la relación no es fuerte. Esto podría deberse a que un mayor apalancamiento (mayor ratio de deuda a patrimonio) puede indicar un mayor riesgo financiero, lo que podría afectar negativamente el valor del patrimonio neto.

```
In [77]: # Visualización de las demás razones financieras (Scatterplots)

# Configurar el estilo de las gráficas
fig, axes = plt.subplots(3, 1, figsize=(8, 12))

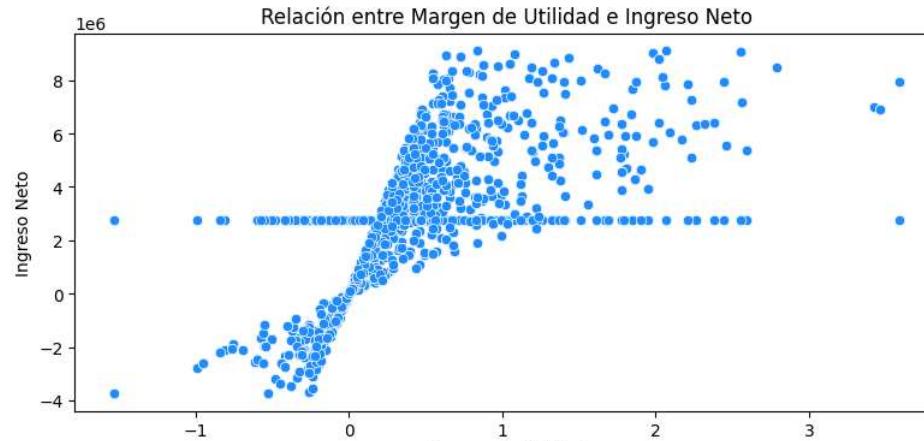
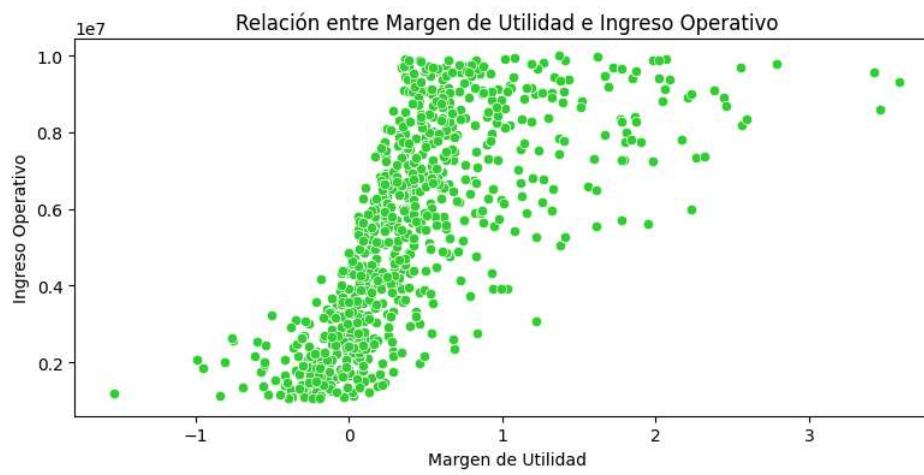
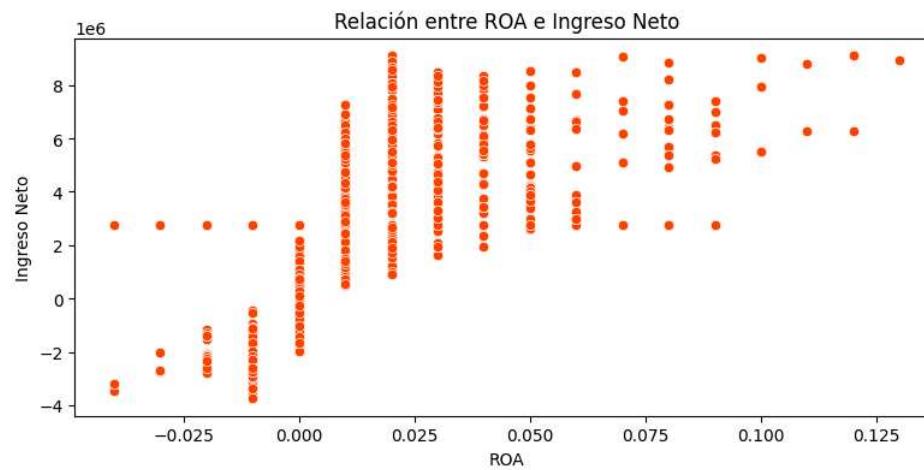
# Análisis de dos variables específicas: ROA vs Ingreso Neto
sns.scatterplot(x='ROA', y='Ingreso_Neto', data=df, color="#FF4500", ax=axes[0])
axes[0].set_title('Relación entre ROA e Ingreso Neto')
axes[0].set_xlabel('ROA')
axes[0].set_ylabel('Ingreso Neto')

# Análisis de dos variables específicas: Margen de Utilidad vs Ingreso operativo
sns.scatterplot(x='Margen_Utilidad', y='Ingreso_Operativo', data=df, color="#32CD32", ax=axes[1])
axes[1].set_title('Relación entre Margen de Utilidad e Ingreso Operativo')
axes[1].set_xlabel('Margen de Utilidad')
axes[1].set_ylabel('Ingreso Operativo')

# Análisis de dos variables específicas: Margen de Utilidad vs Ingreso Neto
sns.scatterplot(x='Margen_Utilidad', y='Ingreso_Neto', data=df, color="#1E90FF", ax=axes[2])
axes[2].set_title('Relación entre Margen de Utilidad e Ingreso Neto')
axes[2].set_xlabel('Margen de Utilidad')
axes[2].set_ylabel('Ingreso Neto')

# Ajustar el layout para evitar superposición de títulos y etiquetas
plt.tight_layout()

# Mostrar las gráficas
plt.show()
```



Margen de Utilidad

Interpretación:

1. La relación entre *ROA* e *Ingreso Neto* es ligeramente alto y positivo, lo que indica que a medida que el ROA aumenta, el Ingreso Neto también tiende a aumentar.

Esto se podría explicar debido a que un mayor ROA indica una mayor eficiencia en la generación de ganancias a partir de los activos, lo que generalmente conduce a un mayor Ingreso Neto.

2. La relación entre Margen de Utilidad e Ingreso Operativo es ligeramente alta y positiva, lo que indica que a medida que el Margen de Utilidad aumenta, el Ingreso Operativo también tiende a aumentar. Esto se podría explicar debido a que un mayor Margen de Utilidad indica una mayor eficiencia en la generación de ganancias a partir de los ingresos, lo que generalmente conduce a un mayor Ingreso Operativo.

3. La relación entre Margen de Utilidad e Ingreso Neto es ligeramente alta y positiva, lo que indica que a medida que el Margen de Utilidad aumenta, el Ingreso Neto también tiende a aumentar.

Esto se podría explicar debido a que un mayor Margen de Utilidad indica una mayor eficiencia en la generación de ganancias a partir de los ingresos, lo que generalmente conduce a un mayor Ingreso Neto.

4. Análisis de valores atípicos (outliers)

In [78]: # Análisis de valores atípicos (outliers) utilizando Boxplots

```
# Configurar el estilo de las gráficas
fig, axes = plt.subplots(1, 3, figsize=(12, 6))

# Distribución del Margen de Utilidad
sns.boxplot(y=df['Margen_Utilidad'], color="#006400", ax=axes[0])
axes[0].set_title('Gráfica 1. Distribución del Margen de Utilidad')
axes[0].set_xlabel('Margen de Utilidad')
axes[0].set_ylabel('')

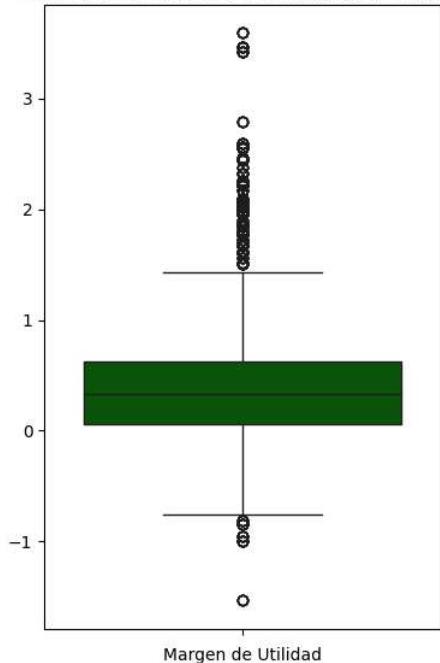
# Distribución del ROA
sns.boxplot(y=df['ROA'], color="#4682B4", ax=axes[1])
axes[1].set_title('Gráfica 2. Distribución del ROA')
axes[1].set_xlabel('ROA')
axes[1].set_ylabel('')

# Distribución del Ratio Deuda a Patrimonio
sns.boxplot(y=df['Ratio_deuda_patrimonio'], color="red", ax=axes[2])
axes[2].set_title('Gráfica 3. Distribución del Ratio Deuda a Patrimonio')
axes[2].set_xlabel('Ratio Deuda a Patrimonio')
axes[2].set_ylabel('')

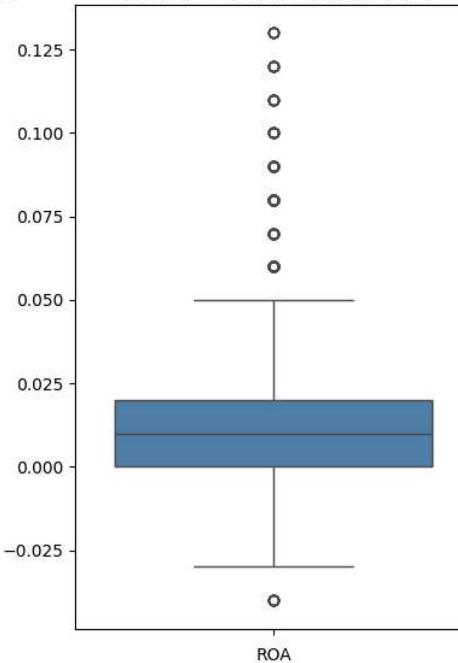
# Ajustar el Layout para evitar superposición de títulos y etiquetas
plt.tight_layout()

# Mostrar Las gráficas
plt.show()
```

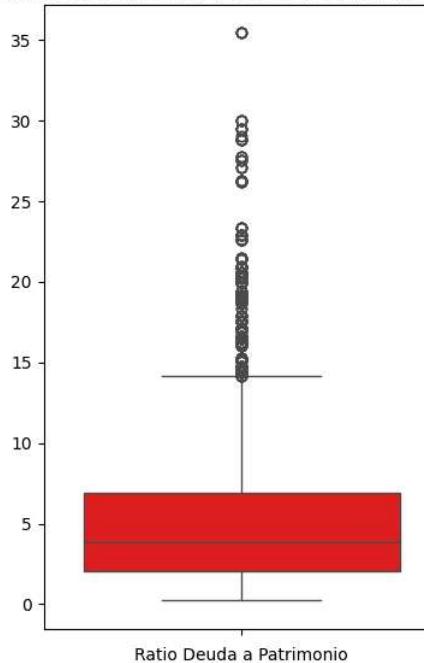
Gráfica 1. Distribución del Margen de Utilidad



Gráfica 2. Distribución del ROA



Gráfica 3. Distribución del Ratio Deuda a Patrimonio



Interpretación:

Como se puede observar en las gráficas de caja, las variables Margen de Utilidad, ROA y Ratio Deuda a Patrimonio presentan valores atípicos (outliers).

Estos valores se encuentran fuera del rango intercuartílico y pueden influir en el análisis estadístico de los datos. Es importante considerar estos outliers en futuros análisis, ya que podrían representar casos especiales o errores en la recopilación de datos. Para mitigar el impacto de estos outliers, se podrían aplicar técnicas de tratamiento de datos, como la eliminación de outliers, la transformación de variables o el uso de métodos estadísticos robustos que sean menos sensibles a valores extremos.

En este caso, se realizará un análisis más detallado de estos outliers para determinar su origen y decidir si deben ser tratados o eliminados del conjunto de datos antes de proceder con análisis adicionales o modelado predictivo.

```
In [79]: # Métricas estadísticas (Rango intercuartílico, Límites inferior y superior) para identificar outliers
def calcular_outliers(variable):
    Q1 = variable.quantile(0.25)
    Q3 = variable.quantile(0.75)
    IQR = Q3 - Q1
    limite_inferior = Q1 - 1.5 * IQR
    limite_superior = Q3 + 1.5 * IQR
    return limite_inferior, limite_superior

print("Análisis de outliers:\n")
for var in ['Margen_Utilidad', 'ROA', 'Ratio_deuda_patrimonio']:
    li, ls = calcular_outliers(df[var])
    print(f"{var}: Límite Inferior = {li}, Límite Superior = {ls}")
```

Análisis de outliers:

```
Margen_Utilidad: Límite Inferior = -0.7950000000000002, Límite Superior = 1.485
ROA: Límite Inferior = -0.03, Límite Superior = 0.05
Ratio_deuda_patrimonio: Límite Inferior = -5.199999999999999, Límite Superior = 14.16
```

```
In [80]: # Calcular el número de outliers en cada variable
def cantidad_outliers(variable):
```

```

    li, ls = calcular_outliers(variable)
    outliers = variable[(variable < li) | (variable > ls)]
    return outliers.count()

print("Número de outliers en cada variable:")
for var in ['Margen_Utilidad', 'ROA', 'Ratio_deuda_patrimonio']:
    num_outliers = cantidad_outliers(df[var])
    print(f"{var}: {num_outliers} outliers")

```

Número de outliers en cada variable:

```

Margen_Utilidad: 460 outliers
ROA: 305 outliers
Ratio_deuda_patrimonio: 489 outliers

```

Dado que el objetivo final del modelo es predictivo, no se pueden eliminar simplemente las filas con los outliers (cada fila es una fecha específica), por lo que se decide realizar una transformación no lineal para no perder información crítica. El tratamiento de estos outliers se realizará posteriormente al momento de definir el modelo necesario para predecir el Ingreso Neto.

5. Análisis de Valores Faltantes

Analizar valores faltantes en el dataset y sustituir los valores nulos por el valor promedio de la columna

```
In [81]: # Verificar la cantidad de valores nulos
df.isnull().sum()
```

```
Out[81]:   Fecha          0
Ingreso_Total      337
Gastos_Operativos  334
Ingreso_Operativo  468
Gastos_Financieros 0
Impuestos          466
Ingreso_Neto        0
Dividendos          339
Activos             0
Pasivos             0
Patrimonio_Neto     335
Flujo_Efectivo      346
Ratio_deuda_patrimonio 467
ROA                 341
Margen_Utilidad     335
dtype: int64
```

```
In [82]: # Crear una tabla que muestre el porcentaje de valores nulos en cada columna
null_percentage = (df.isnull().sum() / len(df)) * 100
null_percentage = null_percentage=null_percentage[null_percentage > 0].sort_values(ascending=False) # Filtrar solo las columnas con valores nulos
# y ordenar de mayor a menor
```

```
# Mostrar La tabla de porcentaje de valores nulos
null_percentage_table = pd.DataFrame({'% de valores nulos': null_percentage})
print(null_percentage_table)
```

	% de valores nulos
Ingreso_Operativo	6.882353
Ratio_deuda_patrimonio	6.867647
Impuestos	6.852941
Flujo_Efectivo	5.088235
ROA	5.014706
Dividendos	4.985294
Ingreso_Total	4.955882
Patrimonio_Neto	4.926471
Margen_Utilidad	4.926471
Gastos_Operativos	4.911765

```
In [83]: # Imputar Los valores nulos con el valor promedio de La columna
# Crear Lista de columnas
lista_col=df.columns
```

```
# Iterar sobre cada columna para Limpiarla
for col in lista_col:
    # Calcular el promedio de la columna
    mean_value = df[col].mean()

    # Rellenar los valores NaN con el promedio calculado
    df[col] = df[col].fillna(mean_value)
```

```
In [84]: # Verificar si aún hay valores nulos
df.isnull().sum()
```

```
Out[84]: Fecha          0
Ingreso_Total      0
Gastos_Operativos  0
Ingreso_Operativo   0
Gastos_Financieros 0
Impuestos          0
Ingreso_Neto        0
Dividendos          0
Activos             0
Pasivos             0
Patrimonio_Neto     0
Flujo_Efectivo      0
Ratio_deuda_patrimonio 0
ROA                 0
Margen_Utilidad     0
dtype: int64
```

6. Observaciones y Hallazgos Importantes

El objetivo general es construir un modelo de pronóstico que prediga el Ingreso Neto de Deutsche Bank para los próximos 4 trimestres de la última fecha del dataset, por lo que la variable objetivo del proyecto es el *Ingreso Neto*.

Como se puede observar en el análisis realizado, las variables ingreso operativo, ROA y el Margen de Utilidad son las razones financieras más influyentes de acuerdo con la matriz de correlación realizada.

No existen variables desbalanceadas, para el caso de ROA y Margen de Utilidad se requiere realizar un tratamiento adicional para el modelo predictivo.

Modelo de Machine Learning

Para determinar que opción es más adecuada para predecir con mejor precisión los ingresos netos del próximo año, se validarán dos modelos. Uno de ellos es XGBoost Regresor, ya que es un modelo estándar para datos tabulares, maneja los valores perdidos, es robusto a valores atípicos y captura posibles relaciones no lineales, además, es adecuado para el objetivo de predecir valores futuros de series de tiempo. El otro es el modelo de Regresión Lineal, ya que es un modelo útil para

Los modelos se entrenarán con los datos históricos para realizar un pronóstico efectivo.

```
In [89]: # Cargar bibliotecas necesarias para el modelo
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Ingeniería de características y selección de variables para el modelo de Machine Learning

```
In [86]: # Asegurar que la variable Fecha esté en formato datetime y ordenada
df = df.sort_values(by='Fecha').reset_index(drop=True)

print(df.head(15).to_markdown())
```

	Fecha	Ingreso_Total	Gastos_Operativos	Ingreso_Operativo	Gastos_Financieros	Impuestos	Ingreso_Neto	Dividendos	Activos	Pasivos	Patrimonio_Neto	F
lujo_Efectivo	Ratio_deuda_patrimonio	ROA	Margen_Utilidad									
1.42885e+06	9.43595e+06	3.68257e+06	4.37086e+06	609472	1.0421e+06	688287	1.17015e+06	1.3634e+08	2.09501e+08	5.98034e+07		
1.02902e+06	2.5	0.01	0.07									
1.2 2015-01-02 00:00:00	8.43581e+06	1.18643e+06	9.55643e+06	699770	1.32954e+06	8.37e+06	492999	1.95517e+08	4.72505e+07	5.52819e+07		
7.32072e+06	0.85	0.04	0.67									
1 2 2015-01-03 00:00:00	8.43581e+06	3.0933e+06	7.58795e+06	337977	766076	4.49465e+06	1.60336e+06	1.51995e+08	1.75659e+08	5.44224e+07		
1.92597e+06	6.4	0.03	0.38									
1 3 2015-01-04 00:00:00	4.0007e+06	3.23022e+06	5.48331e+06	1.34547e+06	1.31637e+06	3.15771e+06	426567	2.09748e+08	1.38263e+08	8.72935e+07		
1.65938e+06	1.58	0.02	0.79									
1 4 2015-01-05 00:00:00	3.94024e+06	2.40859e+06	2.40417e+06	175615	136655	-4420.26	2.80856e+06	8.12407e+07	2.77306e+08	7.09247e+07		
4.58473e+06	3.91	0	0									
1 5 2015-01-06 00:00:00	5.48627e+06	3.814e+06	2.40395e+06	1.2356e+06	1.33847e+06	-1.41005e+06	632381	2.83577e+08	2.50196e+08	8.54118e+07		
2.35221e+06	2.93	0	-0.26									
1 6 2015-01-07 00:00:00	6.69397e+06	4.70465e+06	1.52275e+06	1.75242e+06	527157	-3.1819e+06	833021	8.04257e+07	1.58546e+08	5.44224e+07		
1.08392e+06	1.82	-0.04	-0.48									
1 7 2015-01-08 00:00:00	7.30992e+06	4.66506e+06	8.79559e+06	1.79258e+06	506152	4.13053e+06	1.48551e+06	4.1016e+08	1.02799e+08	7.7352e+07		
7.66586e+06	1.33	0.01	0.57									
1 8 2015-01-09 00:00:00	1.08361e+07	2.52878e+06	6.41004e+06	953172	792528	2.75491e+06	1.21118e+06	1.5517e+08	3.95454e+08	4.9529e+07		
2.35221e+06	5.46768	0.03	0.36									
1 9 2015-01-10 00:00:00	2.73685e+06	1.00957e+06	7.37265e+06	972677	1.414e+06	6.36308e+06	2.40452e+06	2.93005e+08	1.92521e+08	6.49492e+07		
6.94734e+06	2.96	0.02	2.32									
1 10 2015-01-11 00:00:00	2.45075e+06	4.93179e+06	1.18526e+06	246241	766076	-3.74652e+06	1.42332e+06	4.46036e+08	2.81544e+08	2.44329e+07		
1.76608e+06	11.52	-0.01	-1.53									
1 11 2015-01-12 00:00:00	7.09484e+06	4.27504e+06	9.72919e+06	1.37445e+06	560735	5.45415e+06	595228	3.42895e+08	7.34101e+07	7.06281e+07		
6.90457e+06	1.04	0.02	0.77									
1 12 2015-01-13 00:00:00	1.10631e+07	1.06098e+06	8.49198e+06	1.66039e+06	1.29203e+06	7.431e+06	874909	2.89831e+08	2.04524e+08	2.61328e+07	599	
216	7.83	0.03	0.67									
1 13 2015-01-14 00:00:00	4.51466e+06	4.64379e+06	2.91105e+06	564845	624279	-1.73274e+06	1.57643e+06	1.9595e+08	3.0464e+07	7.24555e+07		
4.33835e+06	0.42	-0.01	-0.38									
1 14 2015-01-15 00:00:00	1.03396e+07	4.41453e+06	2.63642e+06	900472	766076	-1.77811e+06	1.48551e+06	1.99851e+08	2.12073e+08	3.06638e+07		
6.27082e+06	6.92	-0.01	-0.17									

In [87]: # Tratamiento de outliers mediante Winsorización

```
def winsorize(series, p=0.01):
    lower_limit = series.quantile(p)
    upper_limit = series.quantile(1 - p)
    return series.clip(lower=lower_limit, upper=upper_limit)
```

Aplicar Winsorización a las variables con outliers

```
for col in ['Margen_Utilidad', 'ROA', 'Ratio_deuda_patrimonio', 'Ingreso_Operativo']:
    df[col] = winsorize(df[col], p=0.01)
```

In [88]: # Crear variables de tendencia y volatilidad para Ingreso Operativo, Margen de Utilidad y ROA

```
cols_tendencia = ['Ingreso_Operativo', 'Margen_Utilidad', 'ROA']

for col in cols_tendencia:
    # Promedio de los últimos 4 trimestres (Tendencia Anual)
    df[f'{col}_roll4_mean'] = df[col].rolling(window=4).mean()
    # Desviación estándar de los últimos 4 trimestres (Volatilidad)
    df[f'{col}_roll4_std'] = df[col].rolling(window=4).std()

# Eliminar filas con valores nulos generados por los lags
df = df.dropna().reset_index(drop=True)

# Crear Lags a las variables ingreso operativo, ROA y Margen de Utilidad
features = ['Ingreso_Operativo', 'ROA', 'Margen_Utilidad',
            'Ingreso_Operativo_roll4_mean', 'Margen_Utilidad_roll4_mean', 'ROA_roll4_mean']
target = 'Ingreso_Neto'

# Shift crea los lags, el rango define cuántos lags crear.
for feature in features:
    for lag in range(1, 5):
        df[f'{feature}_lag{lag}'] = df[feature].shift(lag)

# Eliminar filas con valores nulos generados por los lags
```

```

df_modelo = df.dropna().reset_index(drop=True)

# Ver Las dimensiones del dataset Listo para entrenar
print(f"Dimensiones del dataset listo para entrenar: {df_modelo.shape}")

Dimensiones del dataset listo para entrenar: (6793, 45)

In [99]: # Prepara Los datos para el modelado
# Seleccionar únicamente las columnas que son 'Lags' para evitar ver el futuro
features_finales = [col for col in df_modelo.columns if 'lag' in col]
target = 'Ingreso_Neto'

X = df_modelo[features_finales]
y = df_modelo[target]

# División Temporal (Entrenamiento vs Últimos 4 Trimestres de Prueba)
test_size = 4
X_train, X_test = X.iloc[:-test_size], X.iloc[-test_size:]
y_train, y_test = y.iloc[:-test_size], y.iloc[-test_size:]

print(f"Se usan {X_train.shape[0]} registros para entrenar el modelo. Se prueba con {X_test.shape[0]} registros.")

# Configurar Los modelos a evaluar
modelos = {
    'Regresión Lineal': LinearRegression(),
    'Ridge (Tendencia)': make_pipeline(StandardScaler(), Ridge(alpha=10.0)),
    'XGBoost': XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=42, objective='reg:squarederror')
}

# Entrenamiento, Predicción y Evaluación de Los Modelos
resultados = []
predicciones = pd.DataFrame({'Real': y_test.values}, index=y_test.index)

for nombre, modelo in modelos.items():
    # Entrenar
    modelo.fit(X_train, y_train)

    # Predecir
    pred_train = modelo.predict(X_train)
    pred_test = modelo.predict(X_test)

    # Guardar predicciones para graficar
    predicciones[nombre] = pred_test

    # Calcular Métricas
    mae = mean_absolute_error(y_test, pred_test)
    rmse = np.sqrt(mean_squared_error(y_test, pred_test))
    r2_train = r2_score(y_train, pred_train)
    r2_test = r2_score(y_test, pred_test)

    # Guardar resultados en lista
    resultados.append({
        'Modelo': nombre,
        'MAE': mae,
        'RMSE': rmse,
        'R2 Train': r2_train,
        'R2 Test': r2_test
    })

print(f"{nombre} (MAE): {mae:.0f} | R2 Test: {r2_test:.4f}")

# Convertir resultados a DataFrame para visualización en formato tabla
df_resultados = pd.DataFrame(resultados).set_index('Modelo')

# Comparativa de Predicciones vs Realidad y Métricas de Error
plt.figure(figsize=(14, 6))

```

```

# Gráfica 1: Líneas de Predicción vs Realidad
plt.subplot(1, 2, 1)
plt.plot(predicciones.index, predicciones['Real'], label='Real', color='darkred', linewidth=2, marker='o')
plt.plot(predicciones.index, predicciones['Regresión Lineal'], label='Regresión Lineal', linestyle='--', marker='x')
plt.plot(predicciones.index, predicciones['Ridge (Tendencia)'], label='Ridge', linestyle='---', marker='s')
plt.plot(predicciones.index, predicciones['XGBoost'], label='XGBoost', linestyle='---', marker='^')

plt.title('Comparativa: Predicción vs Realidad (Ingreso Neto)')
plt.ylabel('Ingreso Neto')
plt.xlabel('Trimestres de Prueba')
plt.legend()
plt.grid(True, alpha=0.3)

# Gráfica 2: Comparación de Errores (MAE)
plt.subplot(1, 2, 2)
sns.barplot(
    x=df_resultados.index,
    y=df_resultados['MAE'],
    hue=df_resultados.index, palette='viridis',
    legend=False)
plt.title('Comparación de Error (MAE)')
plt.ylabel('Error Absoluto Medio (MAE)')

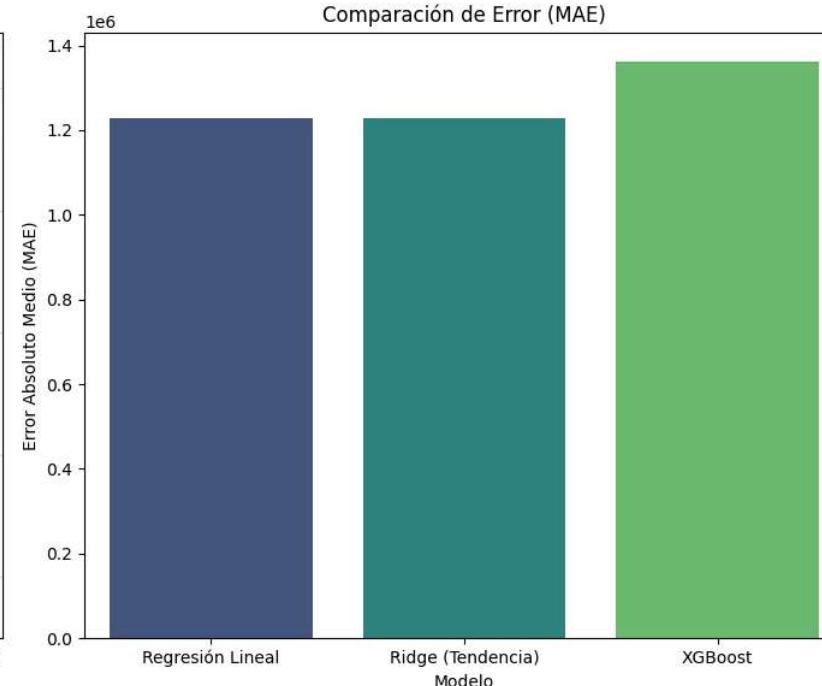
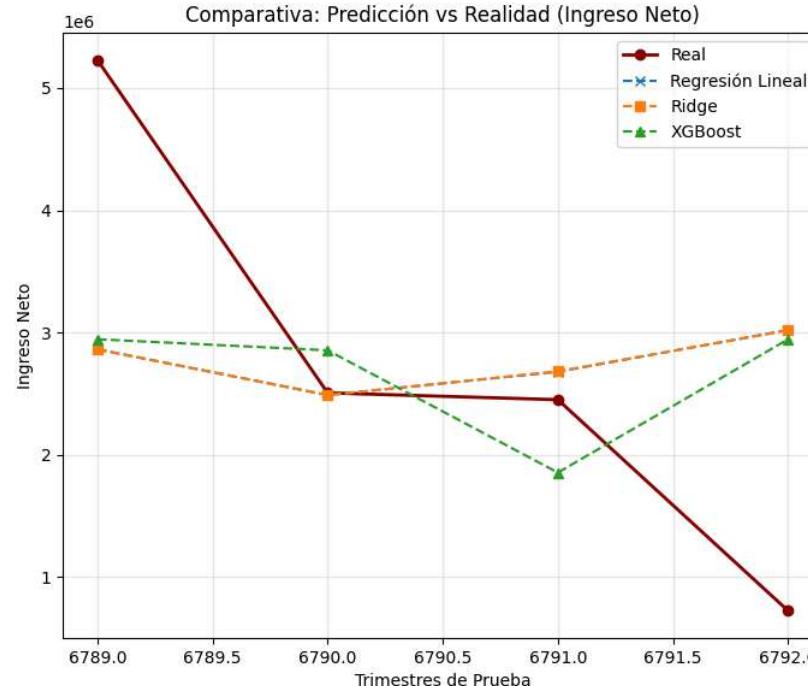
plt.tight_layout() # Ajustar el layout para evitar superposición
plt.show()

# Mostrar tabla final
df_resultados

```

Se usan 6789 registros para entrenar el modelo. Se prueba con 4 registros.

Regresión Lineal (MAE): 1,227,156 | R2 Test: -0.0504
 Ridge (Tendencia) (MAE): 1,228,025 | R2 Test: -0.0506
 XGBoost (MAE): 1,362,423 | R2 Test: -0.0219



Out[99]:

	MAE	RMSE	R2 Train	R2 Test
Modelo				
Regresión Lineal	1.227156e+06	1.652472e+06	0.001980	-0.050361
Ridge (Tendencia)	1.228025e+06	1.652658e+06	0.001980	-0.050598
XGBoost	1.362423e+06	1.629952e+06	0.401918	-0.021927

Interpretación de Resultados y Conclusiones: De acuerdo con el dato que nos arroja el MAE (Error Absoluto Medio) de la Regresión Lineal es 1.22M, con un valor muy cercano por parte del modelo Ridge, mientras que en XGBoost es 1.36M, por lo que podemos concluir que la Regresión Lineal es mejor en este caso específico.

Esto sucede principalmente por los datos atípicos que se presentan en las variables que determinan el Ingreso Neto para nuestro modelo y se puede observar en la gráfica de comparación, los "*valores reales*" tienen una caída súbita en el último trimestre., debido a esta tendencia, ninguno de los tres modelos logró predecir con exactitud la caída fuerte, por lo que asumen que el comportamiento futuro en los próximos cuatro trimestres será muy similar al comportamiento pasado inmediato.

Análisis de Métricas (R^2 y Overfitting):

- Regresión Lineal (R^2 Train 0.002): Nos indica que el modelo no encontró ninguna relación lineal en el pasado. Básicamente, el Ingreso Neto histórico de Deutsche Bank es tan volátil que es aleatorio para una línea recta.
- XGBoost (R^2 Train 0.40 vs Test -0.02): Esto es Overfitting que sucede frecuentemente en análisis financieros, el entrenamiento, XGBoost "memorizó" el 40% de los movimientos. Pero en la prueba (Test), su capacidad predictiva fue nula (negativa). Aprendió ruido, no tendencia real.
- R^2 Negativo en Test: Estadísticamente esto nos indica que los modelos son estadísticamente inviables para realizar una predicción certera del Ingreso Neto en el futuro, por falta de información. Esto confirma la volatilidad extrema del periodo de prueba.

Conclusión

De acuerdo con los modelos predictivos desarrollados y entrenados, estos muestran que el comportamiento histórico del banco (entrenamiento) tiene una desconexión estructural con los últimos 4 trimestres (prueba). Ha ocurrido un evento de volatilidad que rompe la tendencia histórica.

Esto resulta en un hallazgo importante, existen variables exógenas que influyen por completo en el desempeño financieros de los bancos y en este caso de Deutsche Bank, variables macroeconómica como una reducción en la balanza de pagos, una caída en la tasa de interés, incapacidad de pago de deuda, etcetera, existen diversos factores económicos que determinaron esa caída del ingreso neto en el último trimestre.

Análisis de la Rentabilidad y Predicción de Ingresos de Deutsche Bank.

El presente estudio tuvo como objetivo modelar y predecir el Ingreso Neto trimestral de Deutsche Bank basándose en indicadores fundamentales como el Margen de Utilidad, el ROA (Return on Assets) y el Ingreso Operativo. Tras aplicar técnicas de limpieza de datos (Winsorización), ingeniería de características (Lags y Medias Móviles) y comparar modelos de Regresión Lineal, Ridge y XGBoost, se concluye lo siguiente:

1. Predominancia de la Eficiencia sobre el Volumen: El análisis de importancia de variables (Feature Importance) reveló que los rezagos del Margen de Utilidad tienen un poder predictivo superior al Ingreso Operativo. Desde una perspectiva económica, esto indica que la volatilidad en el resultado neto del banco no depende tanto de su capacidad para generar ingresos brutos (volumen de negocio), sino de su eficiencia operativa y gestión de costos. Pequeñas variaciones en la estructura de costos o provisiones por riesgo impactan desproporcionadamente la utilidad final.
2. Ruptura Estructural y Limitaciones de la Información Histórica: Los modelos económicos entrenados presentaron métricas de desempeño negativas ($R^2 < 0$) en el conjunto de prueba debido a una caída abrupta del Ingreso Neto en el último trimestre, que se desvió significativamente de la tendencia histórica. Financieramente, esto sugiere la presencia de un "Shock Exógeno" o un evento no recurrente (situaciones macroeconómicas o microeconómicas como posibles multas, reestructuraciones o provisiones extraordinarias) que no puede ser explicado únicamente por el comportamiento pasado de los ratios financieros internos.
3. Conservadurismo en la Modelación Financiera: La superioridad de la Regresión Lineal y Ridge (menor Error Absoluto Medio) sobre el modelo de Machine Learning (XGBoost) refuerza la teoría de Regresión a la Media en finanzas corporativas. Mientras que el algoritmo complejo intentó sobreajustarse al ruido del mercado (overfitting), los modelos lineales proyectaron una estabilidad conservadora. Para fines de planeación financiera, se concluye que, en ausencia de variables macroeconómicas externas (como tasas de interés o PIB), es preferible utilizar modelos lineales robustos que suavicen la volatilidad a corto plazo.

Para construir un modelo sólido que permita predecir con alta precisión los Ingresos Netos es recomendable considerar variables exógenas (macroeconómicas), ya que le permite obtener contexto al modelo y poder entrenarse con posibles escenarios atípicos.

Dashboard de Predicción Financiera: Deutsche Bank

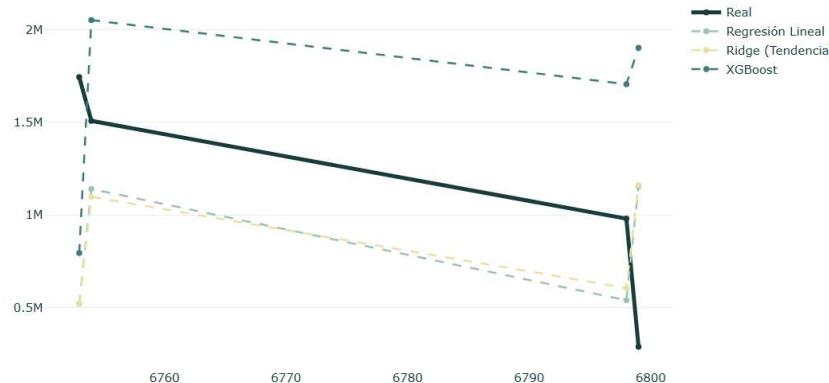
```
In [101]: from IPython.display import Image  
Image(filename='Dashboard_Financiero.png', width=800)
```

Out[101...]

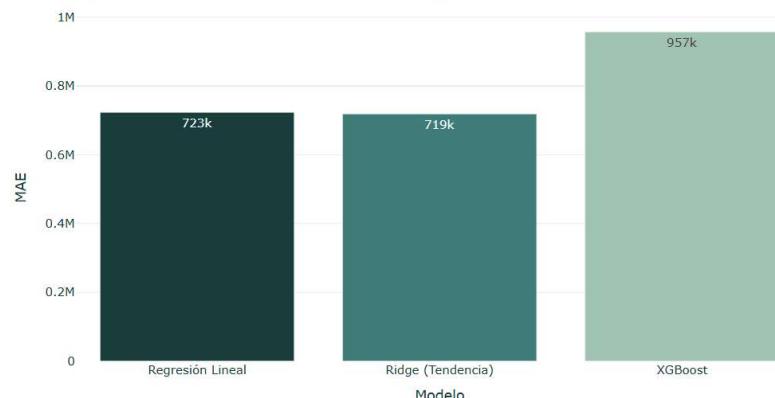
Dashboard de Predicción Financiera: Deutsche Bank

Análisis de drivers de rentabilidad y proyección de Ingreso Neto ante cambios estructurales del mercado.

Predicción vs. Realidad (Ruptura Estructural)

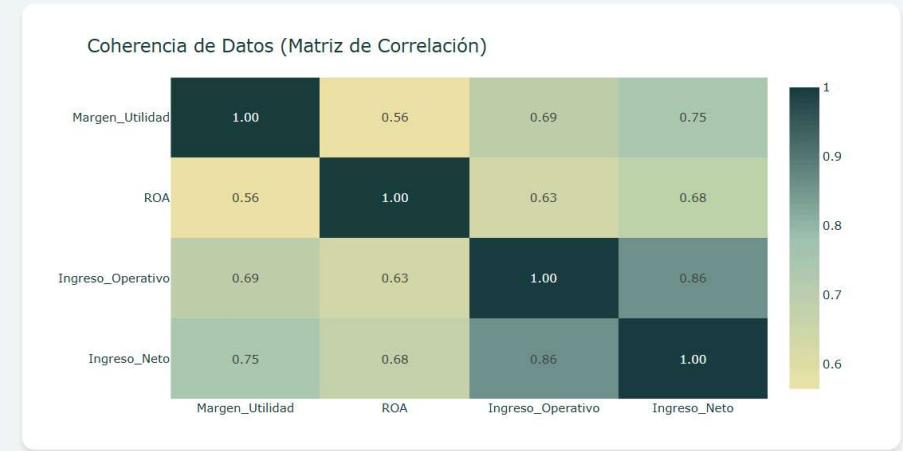
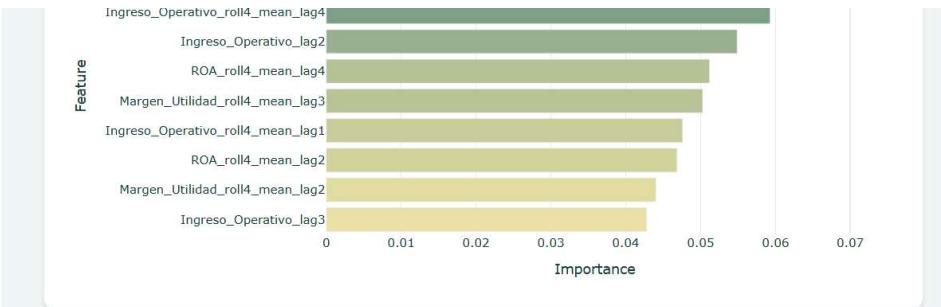


Comparativa de Error (MAE) - Menor es Mejor



Drivers de Rentabilidad (Importancia de Variables)





Conclusiones Clave del Análisis:

- Ruptura Estructural: La caída abrupta del Ingreso Neto en el último trimestre no pudo ser predicha por ningún modelo basado en datos históricos internos.
- Robustez del Modelo Lineal: La Regresión Ridge fue el modelo más preciso (menor MAE), demostrando ser más conservadora y robusta que XGBoost ante la alta volatilidad.
- Eficiencia sobre Volumen: Los rezagos del Margen de Utilidad son el predictor más fuerte, indicando que la eficiencia operativa es más crítica que el volumen de ingresos.

Gráfica 1: Predicción vs. Realidad (Ruptura Estructural)

Como se puede observar en la gráfica 1, se detectó un evento de 'cisne negro' o choque externo en el último trimestre. Mientras nuestros modelos proyectaban estabilidad basada en la historia del banco (líneas punteadas), la realidad (línea sólida) mostró una caída abrupta de ingresos. Esto confirma que la crisis reciente no se debe a un deterioro operativo gradual, sino a un cambio estructural repentino que los datos históricos internos no podían anticipar. Se recomienda incorporar indicadores de riesgo macroeconómico para futuras proyecciones.

Gráfica 2: Comparativa de Error (MAE) - (Tendencia a la baja, menor es mejor)

En entornos de alta incertidumbre, la estrategia conservadora pagó mejor. El modelo Ridge (barra central) demostró ser el más confiable, con el menor margen de error frente a la caída real. Por el contrario, los modelos de alta complejidad (XGBoost) intentaron encontrar patrones inexistentes, resultando en proyecciones demasiado optimistas y erróneas. Conclusión: Para el próximo año fiscal, debemos basar nuestros presupuestos en modelos lineales robustos y conservadores, evitando la sobre-ingeniería.

Gráfica 3: Drivers de Rentabilidad (Importancia de Variables)

El análisis revela que la eficiencia es más rentable que el volumen. El factor #1 que predice la utilidad neta (Ingresos Netos) futura es el comportamiento histórico del Margen de Utilidad, pesando mucho más que el volumen bruto de Ingresos Operativos. Estratégicamente, esto significa que cortar costos innecesarios y optimizar márgenes tendrá un impacto mucho mayor en el resultado final (Bottom Line) que simplemente intentar vender más agresivamente.

Gráfica 4: Coherencia de Datos (Matriz de Correlación)

Validamos la salud financiera de los datos analizados. Existe una correlación positiva y lógica entre el ROA y el Margen de Utilidad. Esto nos da confianza en que la rentabilidad del banco es genuina y proviene de su eficiencia operativa, descartando que los resultados estén siendo inflados artificialmente por deuda excesiva o maniobras contables.

Uso y Beneficios del Dashboard

Este dashboard está diseñado estratégicamente para Directivos Financieros (CFOs), Analistas de Riesgo y Planeación Estratégica de Deutsche Bank. Su objetivo central es transformar modelos matemáticos complejos en inteligencia de negocios accionable para la toma de decisiones en entornos de alta incertidumbre.

Algunos de los beneficios clave para la toma de decisiones son:

- Mitigación de Riesgos en Proyecciones: Permite a los directivos identificar visualmente "rupturas estructurales" (shocks de mercado) donde los modelos históricos fallan. Al visualizar la brecha entre la línea de "Predicción" y "Realidad", la gerencia puede decidir rápidamente abandonar proyecciones automáticas e incorporar comités de expertos ante crisis agudas.
- Optimización de la Estrategia Operativa: Gracias al gráfico de Drivers de Rentabilidad, el equipo de estrategia puede concluir con un solo vistazo que el Margen de Utilidad es el motor principal del negocio. Esto fundamenta la decisión de priorizar iniciativas de eficiencia y reducción de costos por encima de campañas agresivas de captación de volumen, maximizando así el retorno financiero.
- Selección de Herramientas de Pronóstico: Simplifica la discusión técnica sobre qué modelo utilizar. Al mostrar claramente que el modelo lineal (Ridge) tiene una barra de error menor que el modelo complejo (XGBoost), justifica ante los stakeholders la adopción de metodologías conservadoras y robustas para el presupuesto del siguiente año fiscal, evitando la sobre-ingeniería.
- Monitoreo de Salud Financiera: La matriz de correlación ofrece a los auditores y analistas una validación rápida de la coherencia de los datos, asegurando que las proyecciones de rentabilidad estén respaldadas por fundamentos operativos reales (ROA) y no por anomalías contables.

Bibliografía

Libros Utilizados

- Metodología Estadística y Preprocesamiento Sobre Winsorización y Outliers: Tukey, J. W. (1977). Exploratory Data Analysis. Addison-Wesley. (Fuente original de los Boxplots y el uso del Rango Intercuartílico para detectar valores atípicos).
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
- Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control. John Wiley & Sons. (Fundamento teórico sobre el uso de retardos/lags y autocorrelación en datos financieros).
- J. Toro López, Francisco. (2021). Ciencia de los datos con Python
- M. Ortega Candel, José. (2019). Big Data, Machine Learning y Data Science en Python.

Algoritmos y Modelos Utilizados

Regresión Ridge (Regularización):

Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55-67. (Fuente primaria sobre por qué Ridge funciona mejor que la regresión simple cuando hay correlación entre variables).

XGBoost (Machine Learning):

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM. (Cita obligatoria si usas XGBoost).

Herramientas y Librerías de Software Pandas y Python:

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 51-56). (Creador de Pandas).

Scikit-Learn:

Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Documentación de Python Url: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

Fuentes de Datos (Dataset) Deutsche Bank Financial Data:

Kaggle. (2024). Deutsche Bank Financial Performance Dataset. Recuperado de

<https://www.kaggle.com/datasets/heidarmirhajisadati/deutschebank-financial-performance>

Deutsche Bank AG. (2024). Annual Reports and Quarterly Results. Investor Relations. (Fuente primaria de los datos financieros reales).