

▼ **Análisis exploratorio de datos (EDA)**


El objetivo es identificar patrones, relaciones entre las variables y verificar la calidad de los datos para predecir el rendimiento de los en los cultivos en Colombia.

```
1 # Importar librerías y cargar data
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns
```

▼ **1. Entendimiento de las variables:**

- Variable objetivo: Rendimiento (t/ha).
- Variables independientes: Área sembrada (ha), Área cosechada (ha), Producción (t), Año, Departamento, Municipio, Cultivo.


```
1 # Carcar archivo BasePagina
2 data = pd.read_excel('Base agricola 2019 - 2023.xlsx', sheet_name='BasePagina')
3 data.head()
```



	Código Dane departamento	Departamento	Código Dane municipio	Municipio	Desagregación cultivo	Cultivo	Ciclo del cultivo	Grupo cultivo	Subgrupo	Año	Periodo	Área sembrada (ha)	cosi
0	5	Antioquia	5001	Medellín	Aguacate demás variedades	Aguacate	Permanente	Frutales	Demás frutales	2019	2019	24.00	
1	5	Antioquia	5001	Medellín	Aguacate demás variedades	Aguacate	Permanente	Frutales	Demás frutales		2020	8.52	
2	5	Antioquia	5001	Medellín	Aguacate demás variedades	Aguacate	Permanente	Frutales	Demás frutales	2021	2021	8.52	
3	5	Antioquia	5001	Medellín	Aguacate demás variedades	Aguacate	Permanente	Frutales	Demás frutales	2022	2022	17.17	
4	5	Antioquia	5001	Medellín	Aguacate demás variedades	Aguacate	Permanente	Frutales	Demás frutales	2023	2023	14.97	

▼ **2. Verificación de datos nulos y duplicados.**

```
1 # Verificar valores nulos
2 print(data.isnull().sum())
3
4 # Verificar duplicados
5 print(data.duplicated().sum())
6
```



Código Dane departamento	0
Departamento	0
Código Dane municipio	0
Municipio	0
Desagregación cultivo	0
Cultivo	0
Ciclo del cultivo	0
Grupo cultivo	0
Subgrupo	0
Año	0
Periodo	0
Área sembrada (ha)	0
Área cosechada (ha)	0
Producción (t)	0
Rendimiento (t/ha)	0
Nombre científico del cultivo	0

```
Código del cultivo      0
Estado físico del cultivo 0
dtype: int64
0
```

3. Resumen estadístico:

- El objetivo es entender la distribución de los dato

```
1 # Resumen estadístico
2 print(data.describe())
```

```
↗
Código Dane departamento  Código Dane municipio  Área sembrada (ha) \
count      115572.000000      115572.000000      115572.000000
mean         39.401075        39826.710933        233.692231
std          25.036088        25013.085247        1165.780022
min           5.000000         5001.000000         0.000000
25%          17.000000        17174.000000         6.000000
50%          41.000000        41132.000000        20.000000
75%          66.000000        66088.000000        97.000000
max          99.000000        99773.000000       60000.000000

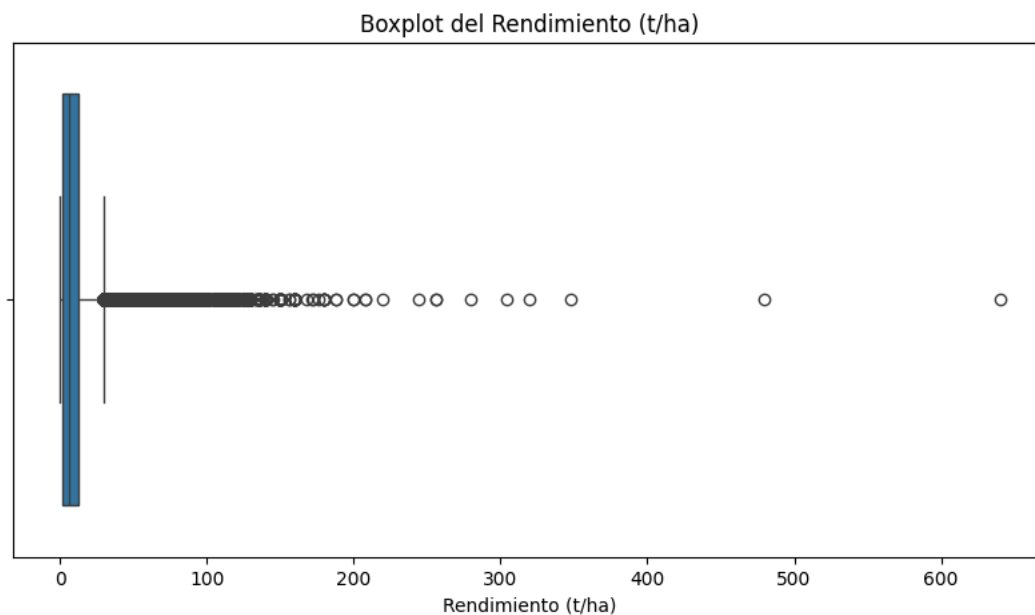
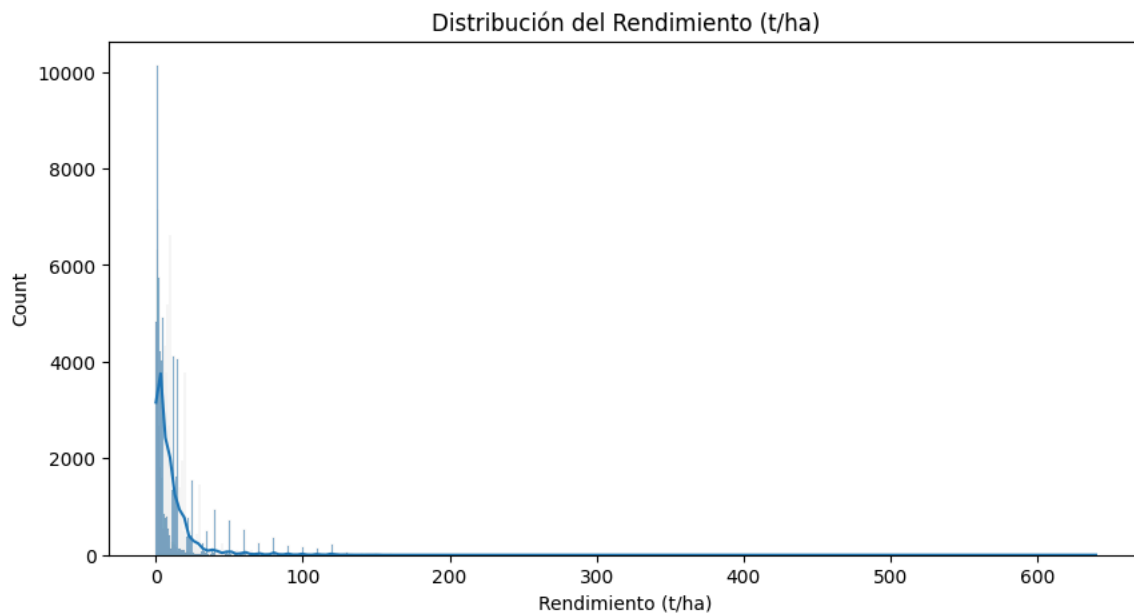
Área cosechada (ha)  Producción (t)  Rendimiento (t/ha) \
count      115572.000000    1.155720e+05    115572.000000
mean         214.304478      3.214482e+03      10.692571
std          1108.192960      4.881239e+04      15.886380
min           0.000000      0.000000e+00      0.000000
25%           5.000000      2.300000e+01      1.900000
50%          18.105000      1.080000e+02      6.000000
75%          85.000000      5.750000e+02     13.000000
max         61000.000000      4.776340e+06     640.000000

Código del cultivo
count      1.155720e+05
mean       1.436985e+06
std        4.854942e+05
min        1.010101e+06
25%        1.051400e+06
50%        1.080400e+06
75%        2.041400e+06
max        2.080300e+06
```

4. Análisis univariado:

- Visualización de la variable objetivo (Rendimiento (t/ha)): Histograma y gráfico de caja (boxplot) para ver la distribución y detectar posibles valores atípicos.

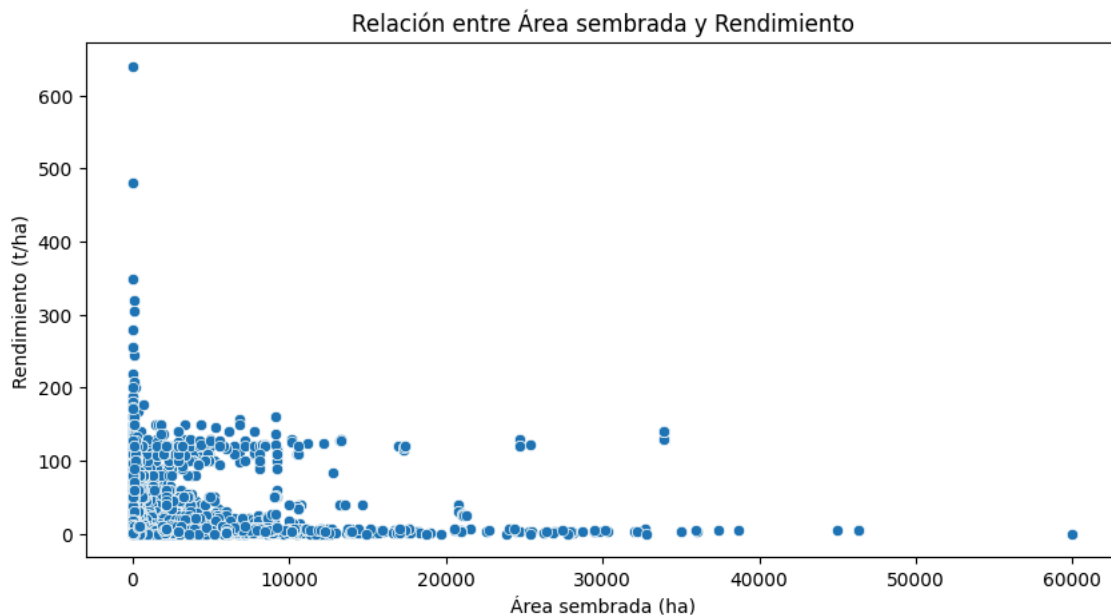
```
1 # Histograma de la variable objetivo
2 plt.figure(figsize=(10, 5))
3 sns.histplot(data['Rendimiento (t/ha)'], kde=True)
4 plt.title('Distribución del Rendimiento (t/ha)')
5 plt.show()
6
7 # Boxplot para detectar outliers
8 plt.figure(figsize=(10, 5))
9 sns.boxplot(x=data['Rendimiento (t/ha)'])
10 plt.title('Boxplot del Rendimiento (t/ha)')
11 plt.show()
```



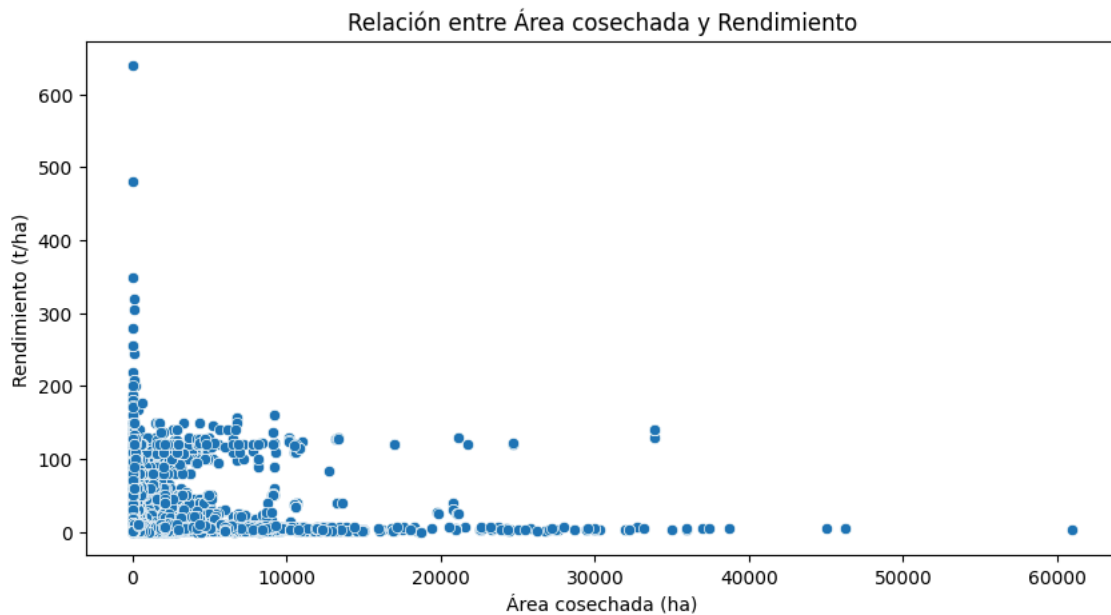
5. Análisis bivariado:

Explora la relación entre las variables independientes y la variable objetivo.

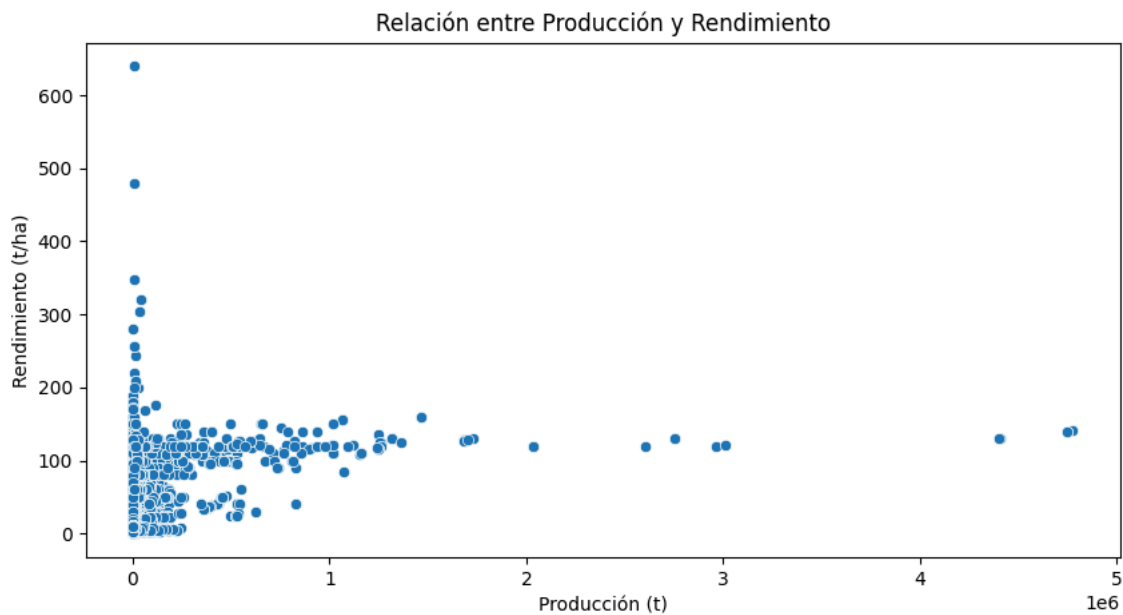
```
1 # Relación entre Área sembrada (ha) y Rendimiento (t/ha)
2 plt.figure(figsize=(10, 5))
3 sns.scatterplot(x='Área sembrada (ha)', y='Rendimiento (t/ha)', data=data)
4 plt.title('Relación entre Área sembrada y Rendimiento')
5 plt.show()
```



```
1 # Relación entre Área cosechada (ha) y Rendimiento (t/ha)
2 plt.figure(figsize=(10, 5))
3 sns.scatterplot(x='Área cosechada (ha)', y='Rendimiento (t/ha)', data=data)
4 plt.title('Relación entre Área cosechada y Rendimiento')
5 plt.show()
```



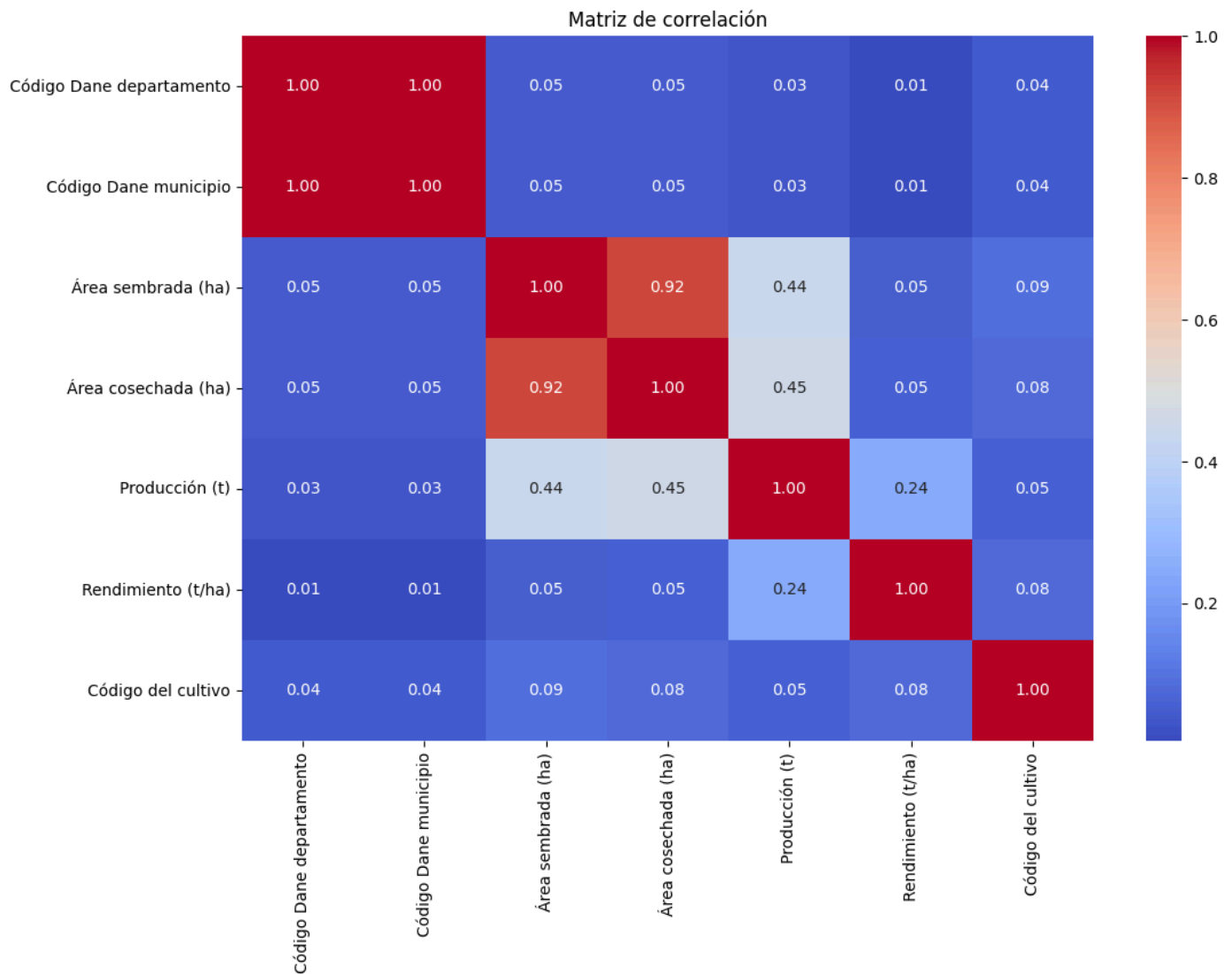
```
1 # Relación entre Producción (t) y Rendimiento (t/ha)
2 plt.figure(figsize=(10, 5))
3 sns.scatterplot(x='Producción (t)', y='Rendimiento (t/ha)', data=data)
4 plt.title('Relación entre Producción y Rendimiento')
5 plt.show()
```



6. Análisis de correlación:

- Matriz de correlación para identificar las relaciones lineales entre las variables numéricas.

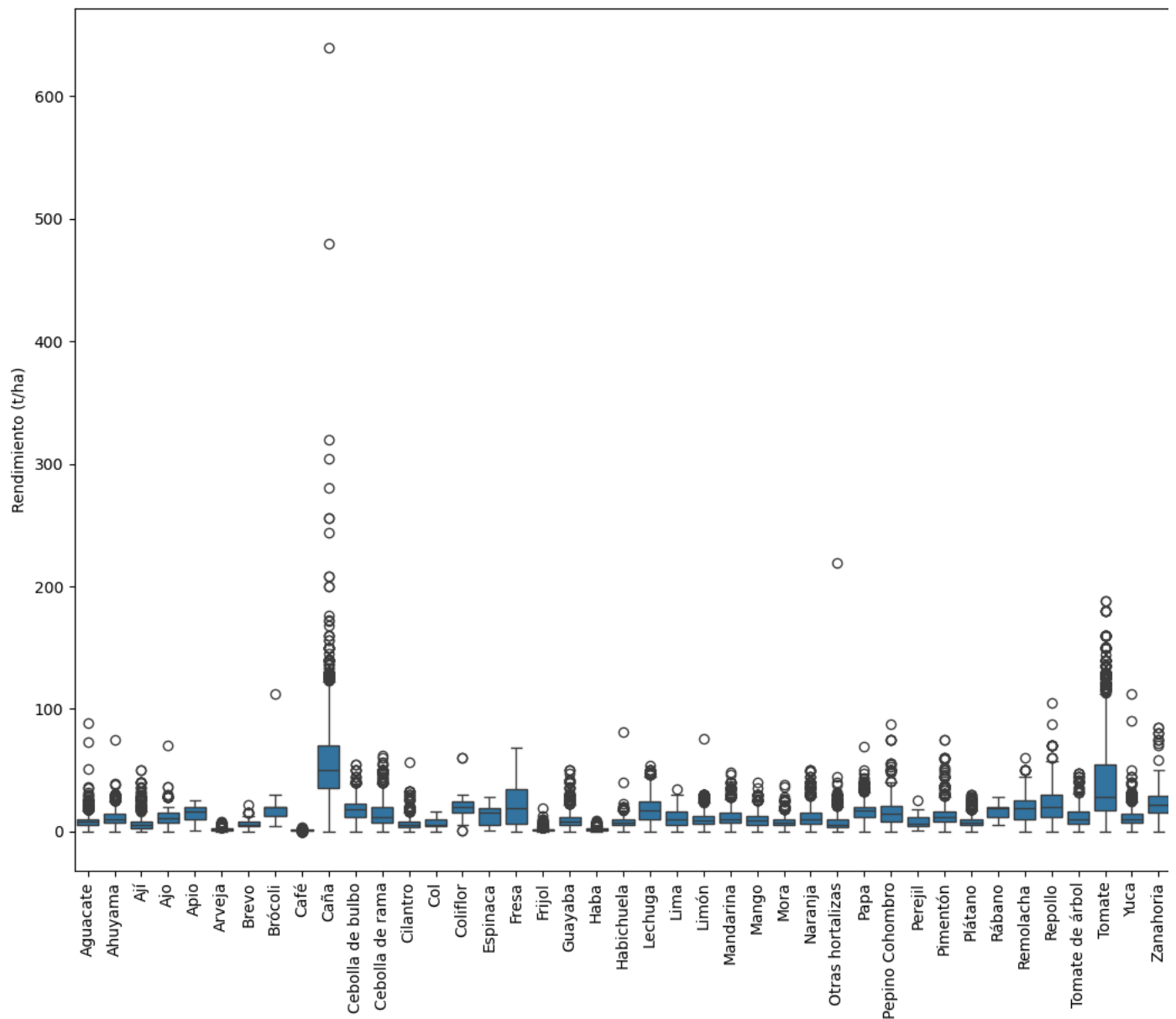
```
1 # Matriz de correlación
2 numerical_data = data.select_dtypes(include=['number'])
3
4 correlation_matrix = numerical_data.corr()
5
6 plt.figure(figsize=(12, 8))
7 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
8 plt.title('Matriz de correlación')
9 plt.show()
```



7. Análisis de categóricas:

Análisis de las variables categóricas para ver cómo afectan el Rendimiento (t/ha).

```
1 # Relación entre Cultivo y Rendimiento (t/ha)
2 plt.figure(figsize=(50, 10))
3 sns.boxplot(x='Cultivo', y='Rendimiento (t/ha)', data=data)
4 plt.xticks(rotation=90)
5 plt.title('Distribución de Rendimiento por Cultivo')
6 plt.show()
```



```

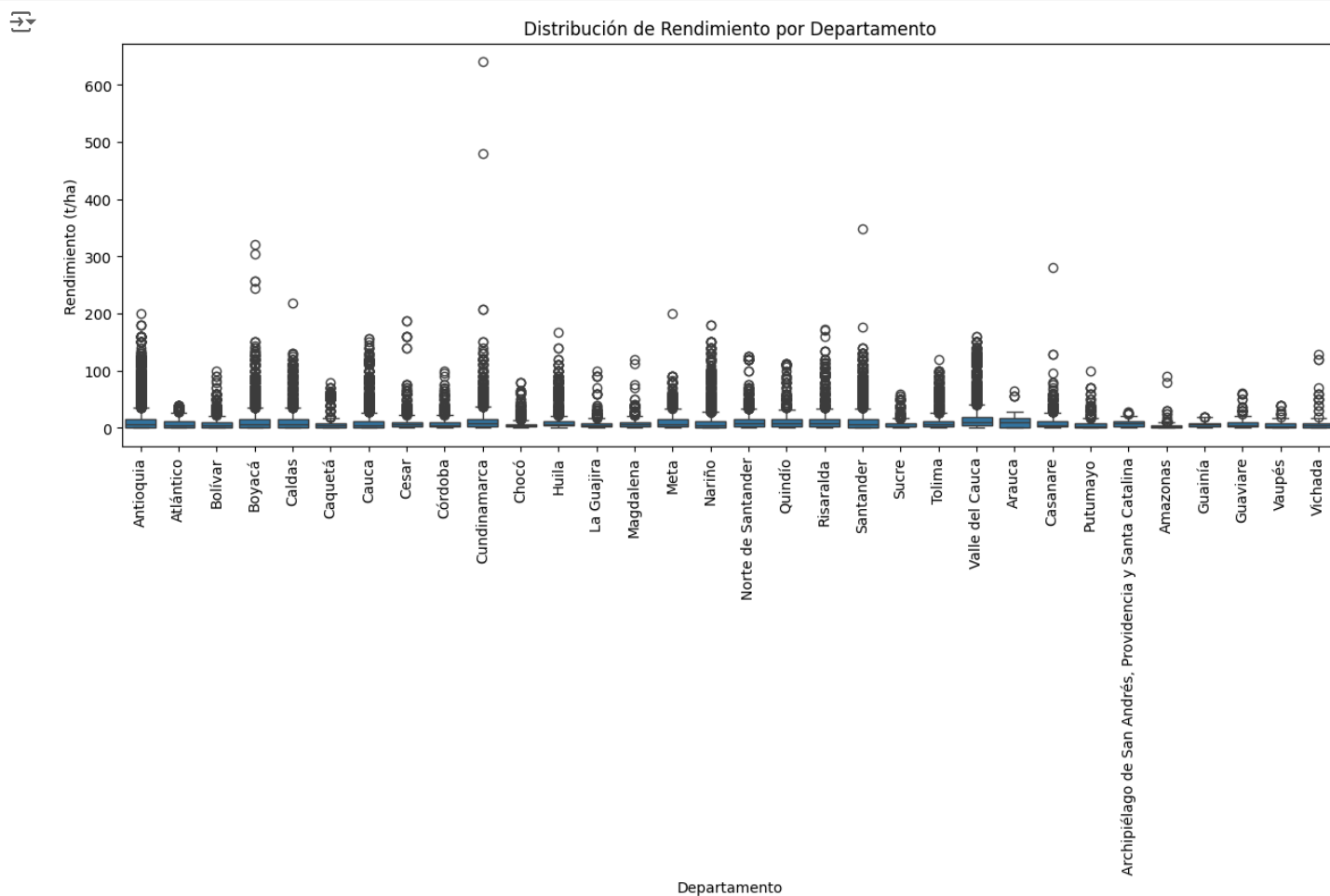
1 # Relación entre Departamento y Rendimiento (t/ha)
2 plt.figure(figsize=(15, 5))
3 sns.boxplot(x='Departamento', y='Rendimiento (t/ha)', data=data)
4 plt.xticks(rotation=90)

```

```

5 plt.title('Distribución de Rendimiento por Departamento')
6 plt.show()

```



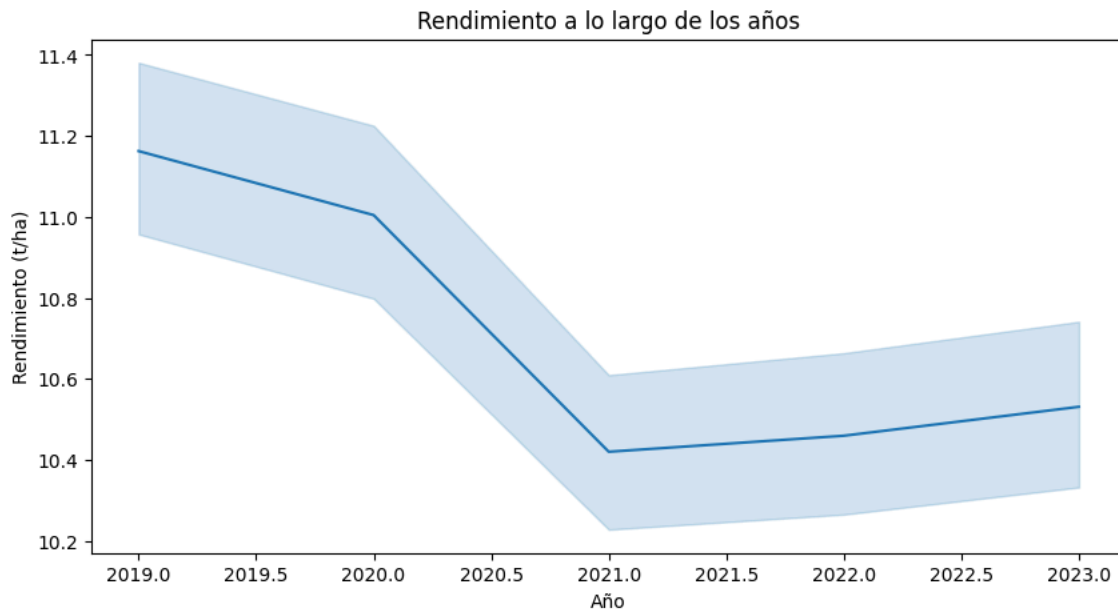
8. Análisis temporal:

- El objetivo es observar cómo ha cambiado el rendimiento a lo largo de los años para detectar tendencias temporales:

```

1 # Relación entre Año y Rendimiento
2
3 data['Año'] = pd.to_numeric(data['Año'], errors='coerce')
4
5 data = data.dropna(subset=['Año'])
6
7 plt.figure(figsize=(10, 5))
8 sns.lineplot(x='Año', y='Rendimiento (t/ha)', data=data)
9 plt.title('Rendimiento a lo largo de los años')
10 plt.show()

```

✓ Preparación para modelado:

✓ 1. Manejo de las variables categóricas:

- Se utiliza la técnicas de One-Hot Encoding para convertir variables categóricas a un formato numérico para que el modelo pueda manejarlas correctamente.

```
1 # One-Hot Encoding de las variables categóricas
2 data_encoded = pd.get_dummies(data, columns=[
3     'Departamento', 'Municipio', 'Cultivo', 'Ciclo del cultivo',
4     'Grupo cultivo', 'Subgrupo', 'Periodo', 'Estado físico del cultivo'
5 ])
```

✓ 2. División de los datos en entrenamiento y prueba:

- Se divide los datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo de manera imparcial. El 20% de los datos se reserva para el conjunto de prueba, mientras que el 80% restante se utiliza para el entrenamiento.

```
1 from sklearn.model_selection import train_test_split
2
3 # Seleccionar las características (X) y la variable objetivo (y)
4 X = data_encoded.drop(['Rendimiento (t/ha)'], axis=1)
5 y = data_encoded['Rendimiento (t/ha)']
6
7 # División del conjunto de datos
8 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 3. Escalado de características:

Se realiza el escalado para mejorar la precisión del modelo.

```
1 # # Escalador
2
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5
6 X = data_encoded.drop(['Rendimiento (t/ha)'], axis=1)
7 y = data_encoded['Rendimiento (t/ha)']
8
9
```

```

10 numeric_cols = X.select_dtypes(include=['number']).columns
11
12 X_numeric = X[numeric_cols]
13
14 X_train, X_test, y_train, y_test = train_test_split(X_numeric, y, test_size=0.2, random_state=42)
15
16 # Inicializar el escalador
17 scaler = StandardScaler()
18
19 # Ajustar el escalador en los datos de entrenamiento numéricos y transformar tanto los datos de entrenamiento como los de prueba
20 X_train_scaled = scaler.fit_transform(X_train)
21 X_test_scaled = scaler.transform(X_test)

```

✓ Selección e implementación del modelo:


✓ 1. Regresión Lineal

- Dado que se busca realizar predicción de rendimiento con los datos agrícolas, se utilizará e modelo sencillo de **Regresión Lineal** para probar el desempeño.

```

1 from sklearn.linear_model import LinearRegression
2 from sklearn.metrics import mean_squared_error, r2_score
3
4 # Inicializar el modelo de regresión lineal
5 linear_model = LinearRegression()
6
7 # Entrenar el modelo con los datos de entrenamiento
8 linear_model.fit(X_train_scaled, y_train)
9
10 # Realizar predicciones con los datos de prueba
11 y_pred = linear_model.predict(X_test_scaled)
12
13 # Evaluar el rendimiento del modelo
14 mse = mean_squared_error(y_test, y_pred)
15 r2 = r2_score(y_test, y_pred)
16
17 print(f"Error cuadrático medio (MSE): {mse}")
18 print(f"Coefficiente de determinación (R²): {r2}")
19

```

 Error cuadrático medio (MSE): 256.9763884878024
 Coeficiente de determinación (R²): 0.06569164547064066

Evaluación del modelo:

- Bajo ajuste del modelo actual: El coeficiente de determinación $R^2 = 0.0657$ $R^2=0.0657$ indica que el modelo solo explica una pequeña fracción (6.57%) de la variabilidad del rendimiento.
- Alta magnitud de error: El error cuadrático medio (MSE) de 256.98 muestra una variación significativa entre las predicciones y los valores reales, lo que sugiere que el modelo comete errores importantes al estimar el rendimiento.
- Limitación del modelo lineal: tiene limitaciones en capturar relaciones no lineales entre las variables y el rendimiento, esta puede ser la razón del bajo desempeño en su capacidad predictiva.

En resumen, el modelo actual necesita ajustarse para mejorar su precisión y capacidad explicativa en la predicción del rendimiento (t/ha).

Posibles mejoras:


- Probar modelos más complejos, como Random Forest o modelos de regresión ajustados con hiperparámetros optimizados,
- Incluir nuevas variables predictoras relevantes que puedan influir en el rendimiento.
- Realizar un análisis de selección de características y explorar interacciones o transformaciones de variables.

✓ 2. Random Forest:

```

1 from sklearn.ensemble import RandomForestRegressor
2
3 # Inicializar el modelo Random Forest
4 rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
5
6 # Entrenar el modelo con los datos de entrenamiento
7 rf_model.fit(X_train_scaled, y_train)
8
9 # Realizar predicciones con los datos de prueba
10 y_pred_rf = rf_model.predict(X_test_scaled)
11
12 # Evaluar el rendimiento del modelo
13 mse_rf = mean_squared_error(y_test, y_pred_rf)
14 r2_rf = r2_score(y_test, y_pred_rf)
15
16 print(f"Error cuadrático medio - Random Forest (MSE): {mse_rf}")
17 print(f"Coeficiente de determinación - Random Forest (R²): {r2_rf}")
18

```

 Error cuadrático medio - Random Forest (MSE): 6.379629368276013
 Coeficiente de determinación - Random Forest (R²): 0.9768051023961524

Evaluación del modelo:

- Error cuadrático medio (MSE): Con un MSE de 6.38, los errores promedio de las predicciones del modelo son considerablemente menores, lo que sugiere que Random Forest está ajustando mucho mejor los datos de rendimiento en comparación con el modelo lineal.
- Coeficiente de determinación (R²): Con un R^2 de 0.9768, el modelo Random Forest explica el 97.68 % de la variabilidad en los datos de rendimiento. Esto indica un ajuste excelente y sugiere que el modelo es capaz de capturar de manera efectiva las relaciones entre las variables y el rendimiento.

✓ Selección del mejor modelo:

Paso 1: Crear un diccionario para almacenar las métricas

```

1 # Crear un diccionario para almacenar las métricas
2 model_metrics = {
3     'Modelo': [],
4     'MSE': [],
5     'R²': []
6 }
7

```

Paso 2: Evaluar cada modelo

```

1 # Agregar métricas de Regresión Lineal al diccionario
2 model_metrics['Modelo'].append('Regresión Lineal')
3 model_metrics['MSE'].append(mse)
4 model_metrics['R²'].append(r2)

```

```

1 # Agregar métricas de Random Forest al diccionario
2 model_metrics['Modelo'].append('Random Forest')
3 model_metrics['MSE'].append(mse_rf)
4 model_metrics['R²'].append(r2_rf)
5


```

Paso 3: Convertir el diccionario a un DataFrame para facilitar la comparación

```

1 # Convertir el diccionario a un DataFrame para visualizar mejor
2 comparison_df = pd.DataFrame(model_metrics)
3 print(comparison_df)

```



	Modelo	MSE	R²
0	Regresión Lineal	256.976388	0.065692
1	Random Forest	6.379629	0.976805