

# Análisis de ocupación turística de residentes y extranjeros en España

## Descripción del proyecto

### Objetivo

Desarrollar un modelo integral de machine learning que permita predecir la ocupación turística en alojamientos de España, segmentar el mercado turístico por duración de estancia y analizar las diferencias entre residentes y extranjeros, tomando en cuenta la variabilidad geográfica a lo largo de los meses entre 2018 y 2022, según los datos obtenidos a través del Instituto Nacional de Estadística en España.

### Contexto

Este proyecto proporcionará una herramienta avanzada para comprender y anticipar patrones de ocupación turística en diversas regiones de España. La combinación de análisis predictivo, segmentación de mercado y comparación entre diferentes grupos de visitantes permitirá facilitar la toma de decisiones estratégicas y la gestión de recursos en el sector turístico.

### Impacto

La implementación exitosa de este modelo de machine learning supondrá un cambio significativo en la gestión y planificación del turismo en España. La predicción precisa de la ocupación permitirá una asignación más eficiente de recursos, promoviendo un desarrollo sostenible y maximizando el impacto económico en cada región. El proyecto también facilitará la identificación de oportunidades de mercado, mejorando la competitividad de las empresas turísticas al adaptar sus ofertas a las preferencias específicas de diferentes segmentos de visitantes, ya sean residentes o extranjeros.

### Alcance

El alcance del proyecto abarca:

- Recopilación y limpieza de un conjunto de datos históricos de ocupación turística en alojamientos, desglosados por región, duración de estancia y origen del visitante.
- Análisis exploratorio de los datos para identificar patrones, relaciones y tendencias en la ocupación turística.
- Desarrollo de modelos predictivos para estimar la ocupación turística en función de diversas variables, como región visitada, la duración de estancia y origen del visitante.

- Segmentación del mercado turístico para identificar perfiles de visitantes y analizar diferencias entre residentes y extranjeros.
- Validación y ajuste de los modelos utilizando técnicas de validación cruzada y optimización de hiperparámetros.
- Implementación de una interfaz o plataforma que permita a las autoridades turísticas y empresas del sector acceder a las predicciones en tiempo real.
- Documentación y capacitación para los usuarios finales sobre la interpretación y aplicación efectiva de los resultados.
- Plan para el mantenimiento y actualización periódica del modelo para adaptarse a cambios estacionales y eventos extraordinarios.

## Métricas del negocio a considerar

- Precisión de predicción de ocupación: Evaluación de la capacidad del modelo para predecir con precisión la ocupación turística real.
- Segmentación efectiva del mercado: Medición de la capacidad del modelo para identificar y diferenciar entre diferentes perfiles de visitantes.
- Impacto en la planificación de recursos: Evaluación de cómo el uso del modelo afecta la asignación eficiente de recursos en función de la demanda turística prevista.

## Stakeholders

Los stakeholders de este proyecto incluyen representantes de las autoridades turísticas a nivel regional y nacional, empresas del sector turístico, agencias de viaje, y organizaciones vinculadas al desarrollo económico y sostenibilidad en España.

## Obtención y limpieza de datos

```
In [ ]: import requests
import pandas as pd
import numpy as np
```

### Descarga del dataset

Se utilizará un dataset público proporcionado por el Instituto Nacional de Estadística de España disponible en [datos.gob.es](https://www.ine.es/jaxiT3/files/t/csv_bdsc/48423.csv).

```
In [ ]: # URL del archivo CSV
URL = "https://www.ine.es/jaxiT3/files/t/csv_bdsc/48423.csv"

# Ruta específica donde se desea guardar el archivo CSV descargado
PATH = "../data/raw/numero_alojamientos_turisticos_y_noches_ocupados_por_residen
```

```
In [ ]: # Realizamos la solicitud HTTP para descargar el archivo
r = requests.get(URL)
```

```

# Verificamos que la solicitud fue exitosa (código de estado 200)
if r.status_code == 200:
    # Guardamos la información descargada en el archivo local
    with open(PATH, 'wb') as f:
        f.write(r.content)
    print(f"Archivo descargado exitosamente en: {PATH}")
else:
    print(f"Error al descargar el archivo. Código de estado: {r.status_code}")

```

Archivo descargado exitosamente en: ../data/raw/numero\_alojamientos\_turisticos\_y\_noches\_ocupados\_por\_residencia\_del\_viajero\_spain.csv

## Carga del dataset

Cargamos nuestro dataset dentro de un dataframe de `pandas` que nos permite manipular, limpiar y visualizar fácilmente nuestros datos.

```
In [ ]: data = pd.read_csv(PATH, delimiter=';')
```

Utilizamos el método `data.info` para conocer el tipo de dato de cada columna y la cantidad de valores no nulos.

```
In [ ]: data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12420 entries, 0 to 12419
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Concepto turístico      12420 non-null   object 
 1   Total Nacional          12420 non-null   object 
 2   Comunidades y Ciudades Autónomas 11799 non-null   object 
 3   Residencia del viajero     12420 non-null   object 
 4   Periodo                  12420 non-null   object 
 5   Total                     11907 non-null   object 
dtypes: object(6)
memory usage: 582.3+ KB

```

```
In [ ]: data.head()
```

	Concepto turístico	Total Nacional	Comunidades y Ciudades Autónomas	Residencia del viajero	Periodo	Total
0	Número de alojamientos turísticos ocupados	Total Nacional	NaN	Total	2023M09	1.019.000
1	Número de alojamientos turísticos ocupados	Total Nacional	NaN	Total	2023M08	1.287.000
2	Número de alojamientos turísticos ocupados	Total Nacional	NaN	Total	2023M07	1.164.000
3	Número de alojamientos turísticos ocupados	Total Nacional	NaN	Total	2023M06	775.000
4	Número de alojamientos turísticos ocupados	Total Nacional	NaN	Total	2023M05	691.000

Al ver la información del dataset nos damos cuenta que la columna `Total` debería ser un valor de tipo numérico en vez de tipo objeto. Esto sucede porque `pandas` interpreta los valores como string, ya que hay valores faltantes no codificados como `NaN` y, además, se está usando el punto como separador de miles y la coma como separador decimal, por lo que debemos reemplazar dichos caracteres y convertir los datos al tipo correspondiente.

Adicionalmente, reemplazaremos los valores nulos de la columna `Comunidades y Ciudades Autónomas` por la cadena de caracteres `"Total Nacional"` ya que se refiere a los datos obtenidos en el conjunto de todas las Comunidades y Ciudades Autónomas.

```
In [ ]: # Reemplazamos valores faltantes codificados con puntos por NaN
data = data.replace({'..':np.nan}).replace('.':np.nan)

# Eliminamos los puntos como separadores de miles y reemplazamos Las comas por p
data['Total'] = data['Total'].str.replace('.', '').str.replace(',', '.').astype('float')

# Reemplazamos Los valores nulos de La columna "Comunidades y Ciudades Autónomas"
data['Comunidades y Ciudades Autónomas'] = data['Comunidades y Ciudades Autónoma
```

Además, convertimos el tipo de dato de la columna `Periodo` a `datetime` para mejor manipulación.

```
In [ ]: # Convertimos la columna "Periodo" a tipo datetime para trabajar con fechas
data['Periodo'] = pd.to_datetime(data['Periodo'], format='%Y%m')
```

```
# Extraemos el año y mes de la columna "Periodo" y creamos nuevas columnas
data['Año'] = data['Periodo'].dt.year
data['Mes'] = data['Periodo'].dt.month

# Eliminamos la columna "Periodo" ya que no la necesitaremos más
data = data.drop('Periodo', axis=1)
```

In [ ]: data

	Concepto turístico	Total Nacional	Comunidades y Ciudades Autónomas	Residencia del viajero	Total	Año	Mes
0	Número de alojamientos turísticos ocupados	Total Nacional	Total Nacional	Total	1019000.0	2023	9
1	Número de alojamientos turísticos ocupados	Total Nacional	Total Nacional	Total	1287000.0	2023	8
2	Número de alojamientos turísticos ocupados	Total Nacional	Total Nacional	Total	1164000.0	2023	7
3	Número de alojamientos turísticos ocupados	Total Nacional	Total Nacional	Total	775000.0	2023	6
4	Número de alojamientos turísticos ocupados	Total Nacional	Total Nacional	Total	691000.0	2023	5
...	...	...	...	...	...	...	...
12415	Estancia media	Total Nacional	19 Melilla	Residentes en el Extranjero	NaN	2018	5
12416	Estancia media	Total Nacional	19 Melilla	Residentes en el Extranjero	NaN	2018	4
12417	Estancia media	Total Nacional	19 Melilla	Residentes en el Extranjero	NaN	2018	3
12418	Estancia media	Total Nacional	19 Melilla	Residentes en el Extranjero	NaN	2018	2
12419	Estancia media	Total Nacional	19 Melilla	Residentes en el Extranjero	NaN	2018	1

12420 rows × 7 columns

```
In [ ]: data.nunique()
```

```
Concepto turístico      3
Total Nacional          1
Comunidades y Ciudades Autónomas  20
Residencia del viajero    3
Total                     823
Año                      6
Mes                       12
dtype: int64
```

Eliminamos la columna `Total Nacional` ya que sólo tiene un valor único que no aporta utilidad al estudio.

```
In [ ]: data.drop('Total Nacional', axis=1, inplace=True)
```

Ahora, queremos llenar los valores nulos de la columna `Total` con el promedio del respectivo mes.

```
# Definimos las columnas para agrupar
group_columns = ['Concepto turístico', 'Comunidades y Ciudades Autónomas', 'Resi

# Agrupamos los datos en base a las columnas definidas y calculamos el promedio
average_totals = data.groupby(group_columns)['Total'].mean()

# Fusionamos el DataFrame original con el DataFrame de promedios
data = pd.merge(data, average_totals, on=group_columns, suffixes=('', '_average')

# Llenamos los valores nulos en la columna "Total" con el valor promedio correspondiente
data['Total'] = data['Total'].fillna(data['Total_average'])

# Eliminamos la columna de promedios ya que no es necesaria
data.drop('Total_average', axis=1, inplace=True)
```

Verificamos los nulos de las columnas que nos quedaron.

```
In [ ]: data.isnull().sum()
```

```
Concepto turístico      0
Comunidades y Ciudades Autónomas  0
Residencia del viajero    0
Total                     2702
Año                      0
Mes                       0
dtype: int64
```

Eliminamos las filas con valores nulos.

```
In [ ]: data = data.dropna().reset_index(drop=True)
```

```
In [ ]: data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9718 entries, 0 to 9717
Data columns (total 6 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Concepto turístico    9718 non-null   object  
 1   Comunidades y Ciudades Autónomas 9718 non-null   object  
 2   Residencia del viajero      9718 non-null   object  
 3   Total                      9718 non-null   float64 
 4   Año                        9718 non-null   int32   
 5   Mes                        9718 non-null   int32  
dtypes: float64(1), int32(2), object(3)
memory usage: 379.7+ KB

```

```
In [ ]: data.to_csv('../data/interim/data_cleaned.csv')
```

## Exploración y visualización de datos

Cargamos las librerías que utilizaremos para la manipulación y visualización de los datos.

```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [ ]: # Cargamos el dataset Limpio
data_cleaned = pd.read_csv("../data/interim/data_cleaned.csv", index_col=0)
```

```
In [ ]: data_cleaned.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Index: 9718 entries, 0 to 9717
Data columns (total 6 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Concepto turístico    9718 non-null   object  
 1   Comunidades y Ciudades Autónomas 9718 non-null   object  
 2   Residencia del viajero      9718 non-null   object  
 3   Total                      9718 non-null   float64 
 4   Año                        9718 non-null   int64   
 5   Mes                        9718 non-null   int64  
dtypes: float64(1), int64(2), object(3)
memory usage: 531.5+ KB

```

Reordenamos la tabla de datos para mayor legibilidad y manejo de la misma.

```

In [ ]: data_pivot = data_cleaned.pivot_table(index=['Comunidades y Ciudades Autónomas'],
                                                columns='Concepto turístico',
                                                values='Total').reset_index()

# Renombramos las columnas para mayor claridad
data_pivot.columns.name = None # Eliminar el nombre de la columna índice
data_pivot = data_pivot.rename(columns={
    'Número de alojamientos turísticos ocupados': 'Alojamientos Ocupados',
    'Número de noches ocupadas': 'Noches Ocupadas',
    'Estancia media': 'Estancia Media'
})

```

Consultamos los valores nulos

```
In [ ]: data_pivot.isnull().sum()
```

```
Comunidades y Ciudades Autónomas      0
Residencia del viajero                 0
Mes                                     0
Año                                     0
Estancia Media                         301
Alojamientos Ocupados                  301
Noches Ocupadas                        0
dtype: int64
```

Eliminamos los valores nulos

```
In [ ]: data_pivot = data_pivot.dropna().reset_index(drop=True)
```

```
In [ ]: data_pivot.describe()
```

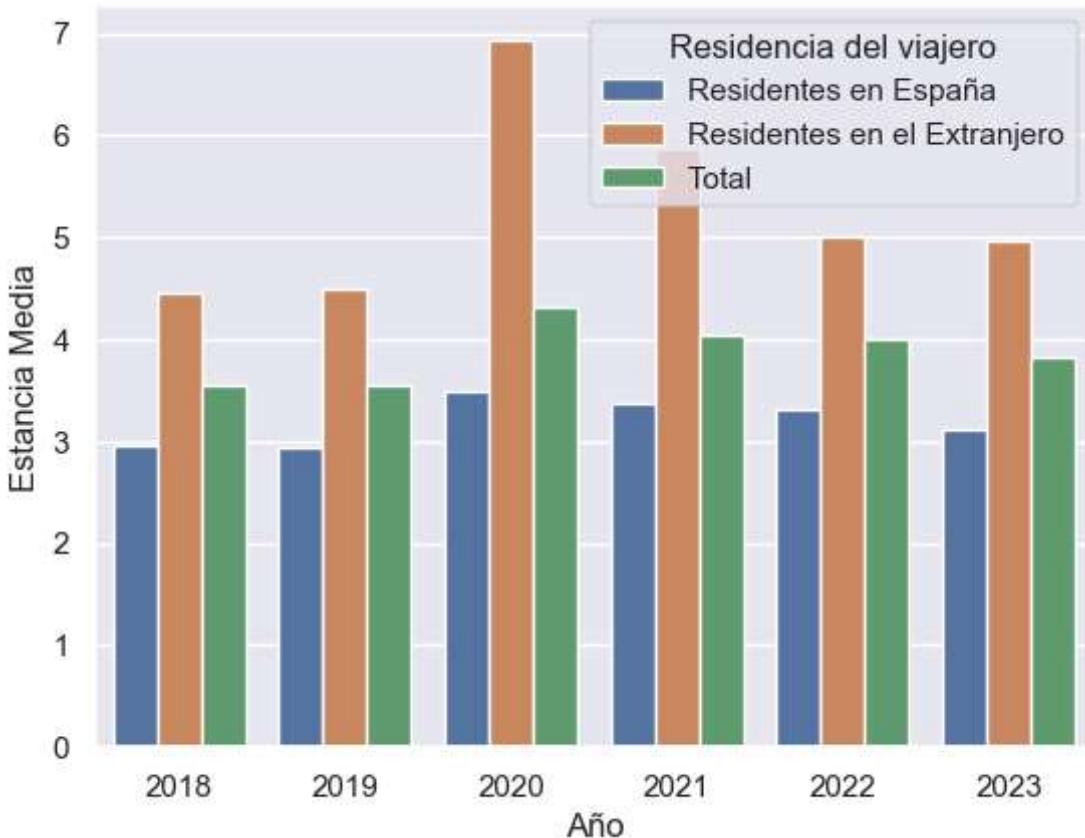
	Mes	Año	Estancia Media	Alojamientos Ocupados	Noches Ocupadas
<b>count</b>	3139.000000	3139.000000	3139.000000	3.139000e+03	3.139000e+03
<b>mean</b>	6.341510	2020.291813	3.979356	4.645014e+04	2.105814e+05
<b>std</b>	3.387917	1.651303	1.658770	1.111718e+05	5.243361e+05
<b>min</b>	1.000000	2018.000000	1.500000	1.000000e+03	2.000000e+03
<b>25%</b>	3.000000	2019.000000	2.800000	4.000000e+03	1.100000e+04
<b>50%</b>	6.000000	2020.000000	3.500000	1.000000e+04	3.300000e+04
<b>75%</b>	9.000000	2022.000000	4.700000	3.900000e+04	1.830000e+05
<b>max</b>	12.000000	2023.000000	22.000000	1.287000e+06	6.558000e+06

```
In [ ]: data_pivot.to_csv('../data/interim/data_eda.csv')
```

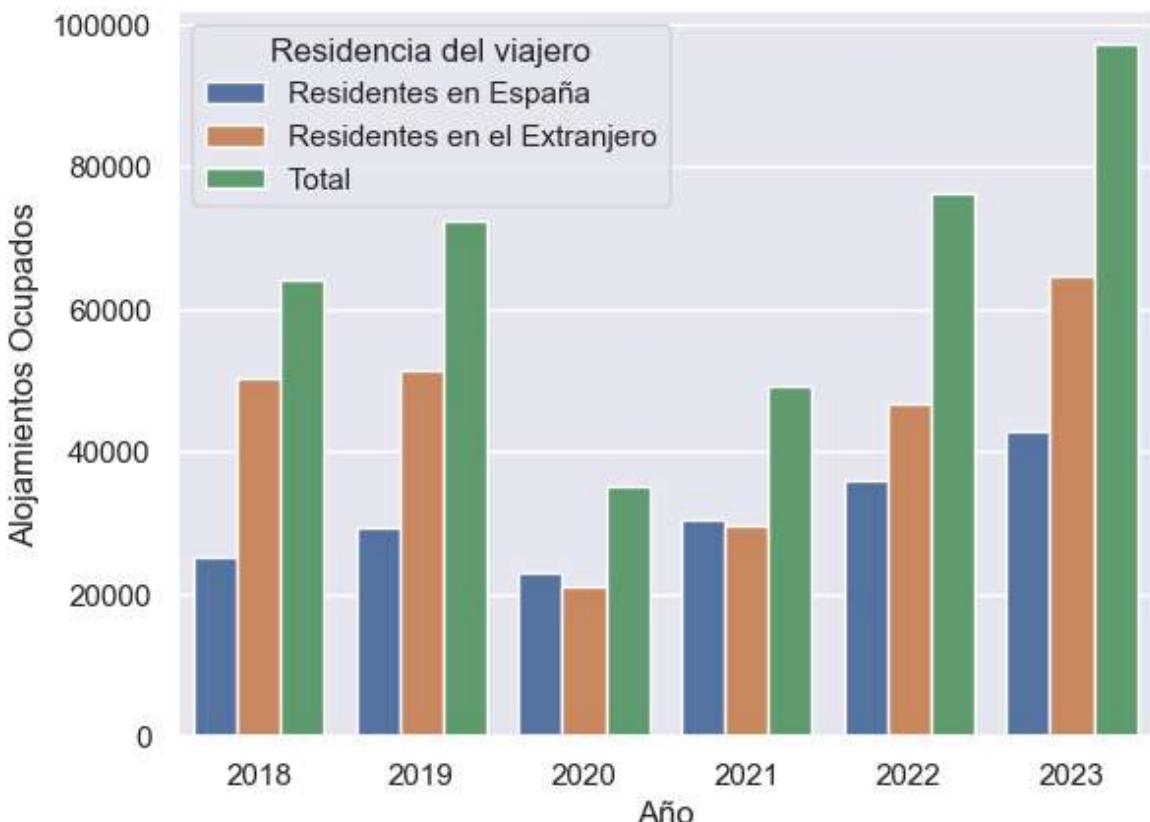
## Gráficos de datos

```
In [ ]: sns.set_theme(style='darkgrid')
```

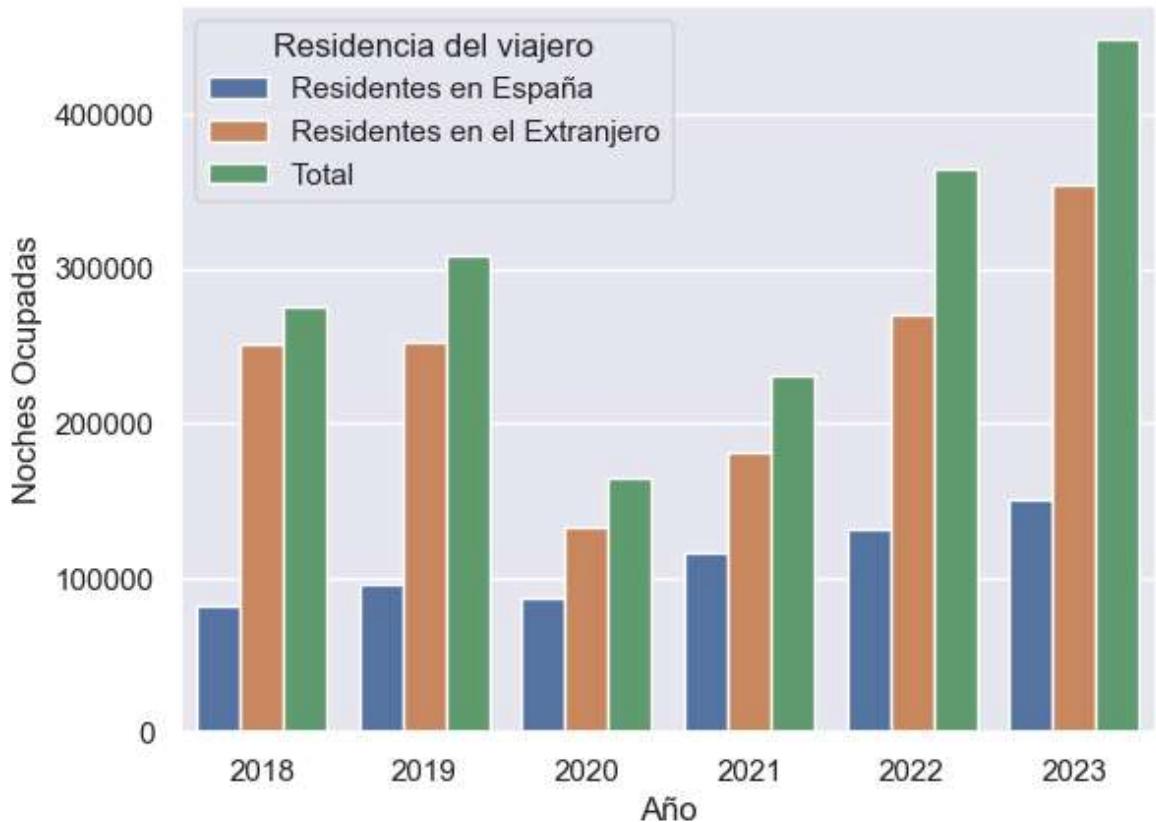
```
In [ ]: sns.barplot(data=data_pivot, x='Año', y='Estancia Media', hue='Residencia del vi
plt.show()
```



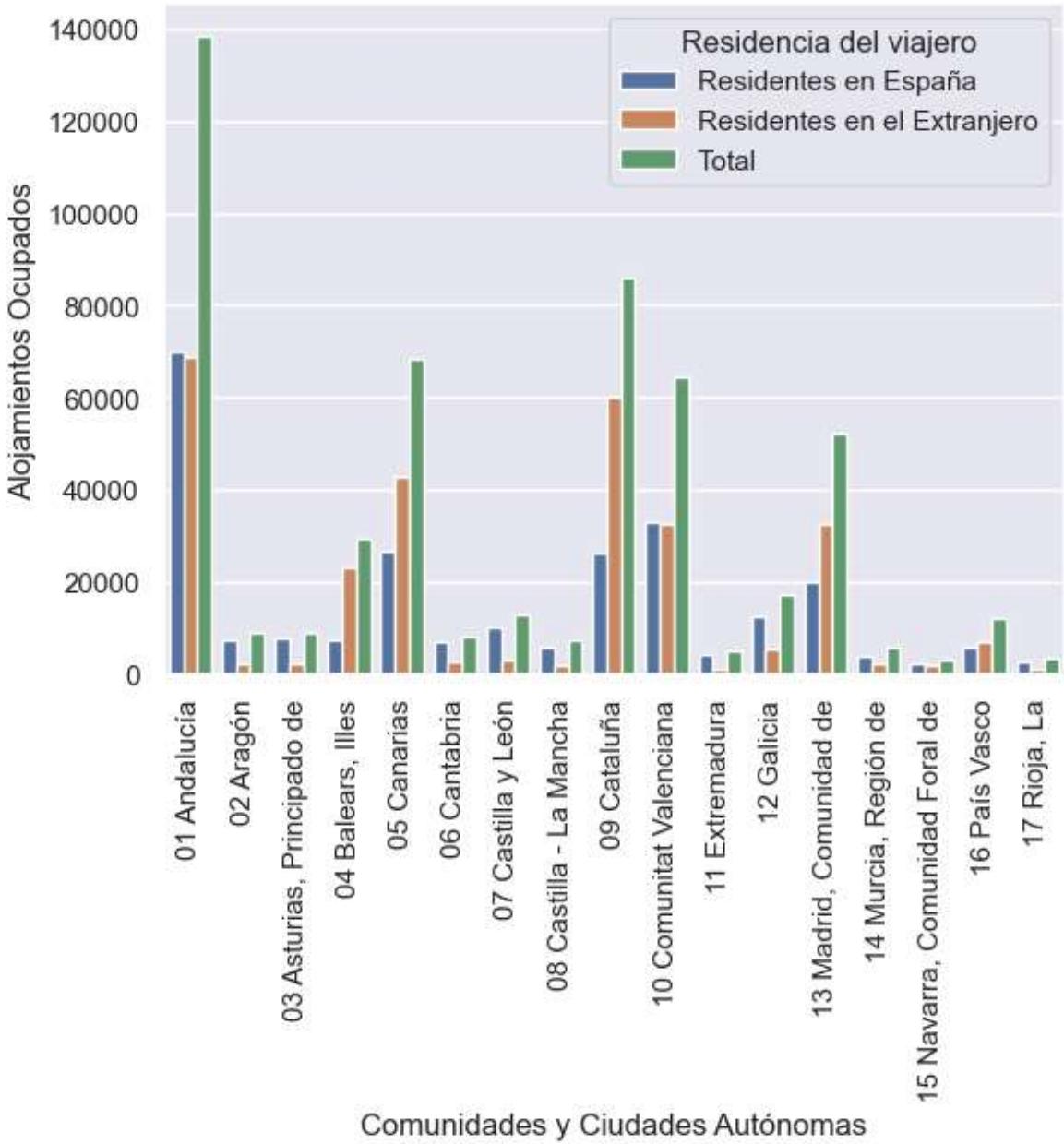
```
In [ ]: sns.barplot(data=data_pivot, x='Año', y='Alojamientos Ocupados', hue='Residencia del viajero')
plt.show()
```



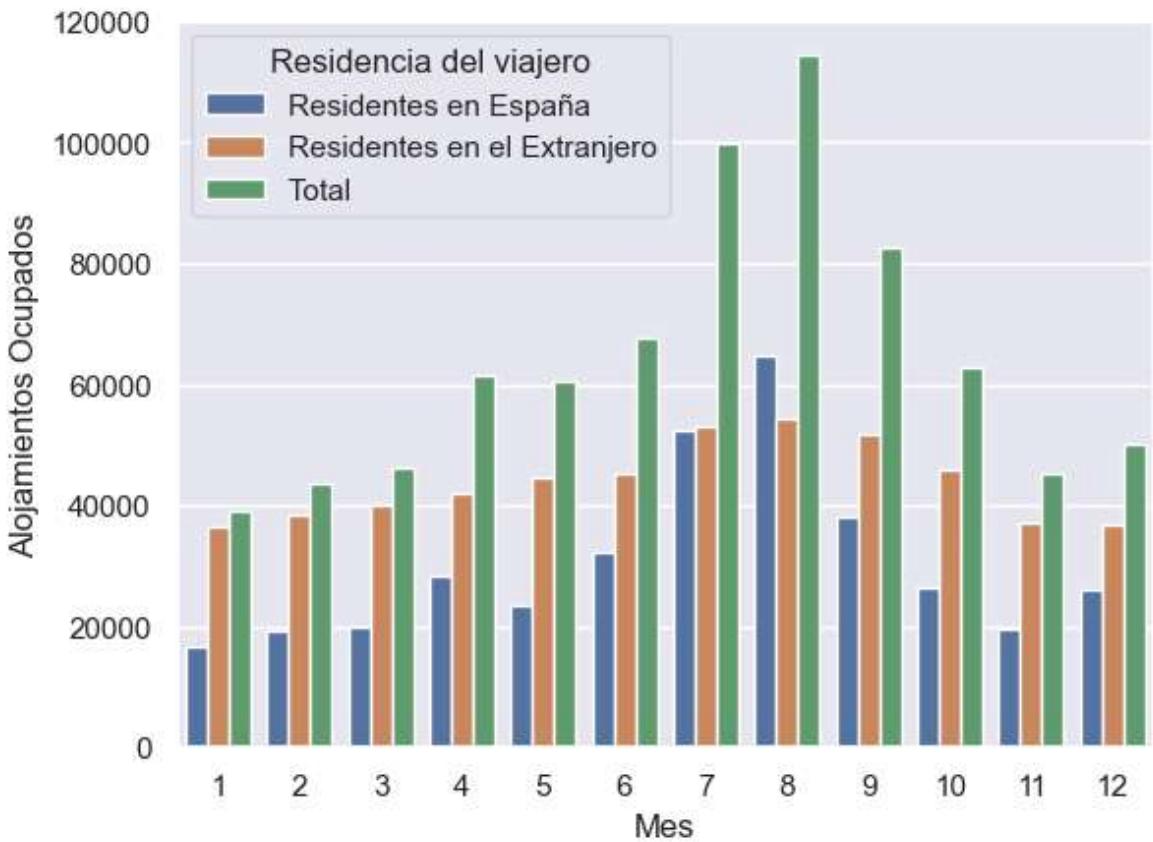
```
In [ ]: sns.barplot(data=data_pivot, x='Año', y='Noches Ocupadas', hue='Residencia del viajero')
plt.show()
```



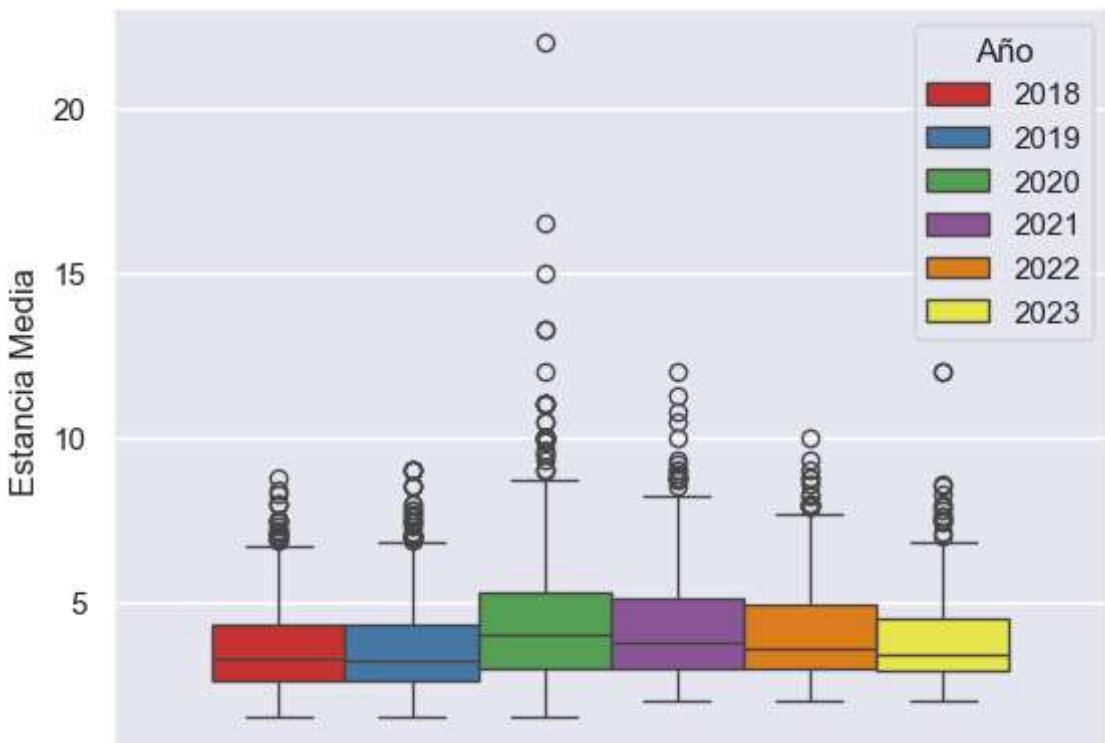
```
In [ ]: sns.barplot(data=data_pivot.query(`Comunidades y Ciudades Autónomas` !='Total Na
plt.xticks(rotation=90)
plt.show()
```



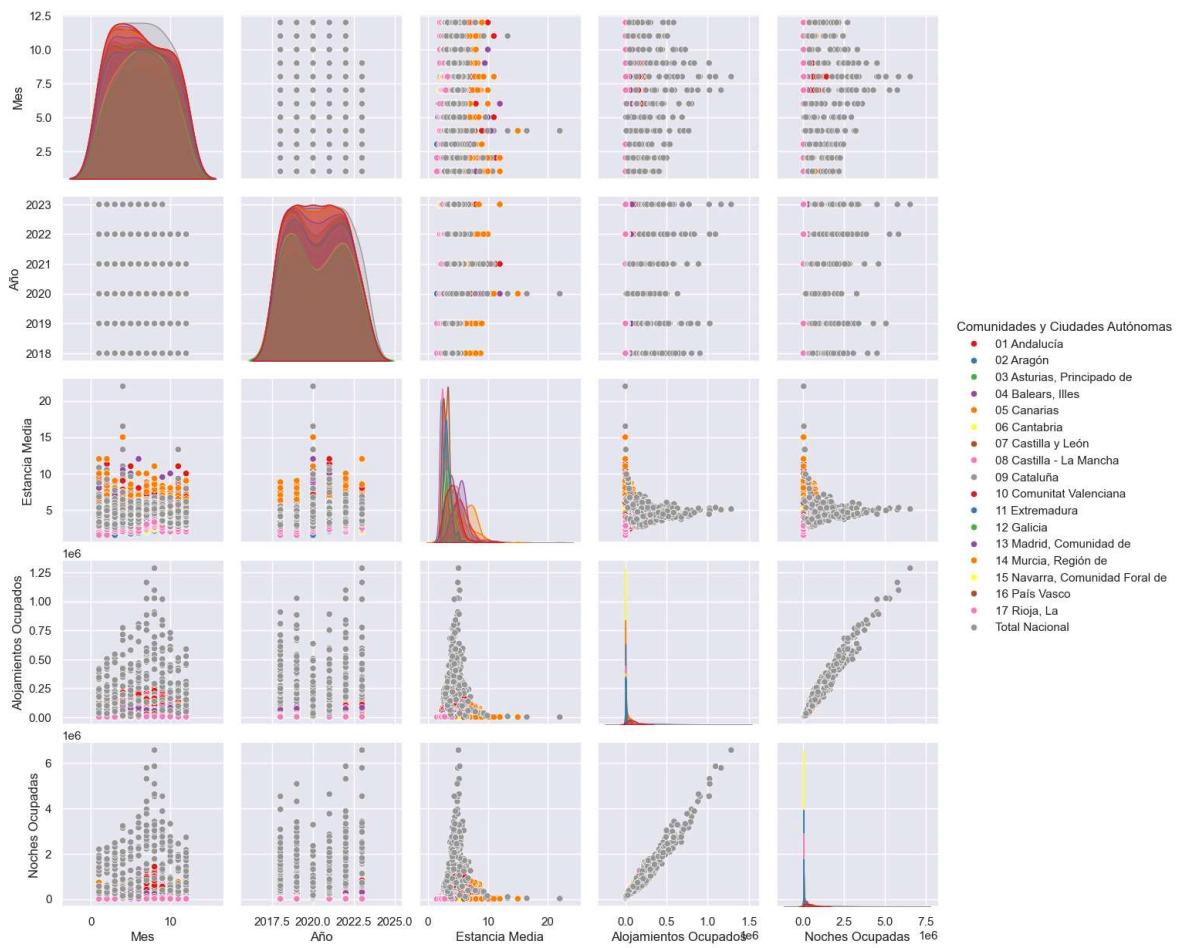
```
In [ ]: sns.barplot(data_pivot, x='Mes', y='Alojamientos Ocupados', hue='Residencia del viajero')
plt.show()
```



```
In [ ]: sns.boxplot(hue='Año', y='Estancia Media', data=data_pivot, palette='Set1')
plt.show()
```



```
In [ ]: sns.pairplot(data_pivot, hue='Comunidades y Ciudades Autónomas', palette='Set1')
plt.show()
```



## Feature Engineering

```
In [ ]: import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

```
In [ ]: data_fe = pd.read_csv('../data/interim/data_eda.csv', index_col=0)
```

```
In [ ]: data_fe.head()
```

	Comunidades y Ciudades Autónomas	Residencia del viajero	Mes	Año	Estancia Media	Alojamientos Ocupados	Noches Ocupadas
0	01 Andalucía	Residentes en España	1	2018	2.4	34000.0	80000.0
1	01 Andalucía	Residentes en España	1	2019	2.5	40000.0	98000.0
2	01 Andalucía	Residentes en España	1	2020	2.5	53000.0	135000.0
3	01 Andalucía	Residentes en España	1	2021	2.7	21000.0	56000.0
4	01 Andalucía	Residentes en España	1	2022	3.0	52000.0	155000.0

In [ ]: `data_fe.info()`

```
<class 'pandas.core.frame.DataFrame'>
Index: 3139 entries, 0 to 3138
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Comunidades y Ciudades Autónomas    3139 non-null   object 
 1   Residencia del viajero            3139 non-null   object 
 2   Mes                            3139 non-null   int64  
 3   Año                            3139 non-null   int64  
 4   Estancia Media                 3139 non-null   float64
 5   Alojamientos Ocupados          3139 non-null   float64
 6   Noches Ocupadas                3139 non-null   float64
dtypes: float64(3), int64(2), object(2)
memory usage: 196.2+ KB
```

## Estandarización

La estandarización es una técnica común en el preprocesamiento de datos utilizada para transformar características (variables) numéricas en una escala común.

Esto es útil porque algoritmos de aprendizaje automático funcionan mejor cuando las características están en una escala similar y cercana a una distribución normal.

En este caso usaremos el StandardScaler de la librería Scikit learn.

In [ ]: `std_scaler = StandardScaler()  
num_vars = data_fe.select_dtypes(include=['float64'])  
df_num_vars_std = std_scaler.fit_transform(num_vars)  
df_num_vars_std = pd.DataFrame(df_num_vars_std, columns=num_vars.columns, index=df_num_vars_std.index)`

	<b>Estancia Media</b>	<b>Alojamientos Ocupados</b>	<b>Noches Ocupadas</b>
<b>0</b>	-0.952277	-0.112008	-0.249081
<b>1</b>	-0.891982	-0.058029	-0.214746
<b>2</b>	-0.891982	0.058926	-0.144170
<b>3</b>	-0.771391	-0.228963	-0.294861
<b>4</b>	-0.590505	0.049929	-0.106020

```
In [ ]: data_std = data_fe.copy()
data_std.update(df_num_vars_std)
data_std.head()
```

	<b>Comunidades y Ciudades Autónomas</b>	<b>Residencia del viajero</b>	<b>Mes</b>	<b>Año</b>	<b>Estancia Media</b>	<b>Alojamientos Ocupados</b>	<b>Noches Ocupadas</b>
<b>0</b>	01 Andalucía	Residentes en España	1	2018	-0.952277	-0.112008	-0.249081
<b>1</b>	01 Andalucía	Residentes en España	1	2019	-0.891982	-0.058029	-0.214746
<b>2</b>	01 Andalucía	Residentes en España	1	2020	-0.891982	0.058926	-0.144170
<b>3</b>	01 Andalucía	Residentes en España	1	2021	-0.771391	-0.228963	-0.294861
<b>4</b>	01 Andalucía	Residentes en España	1	2022	-0.590505	0.049929	-0.106020

## Encoding de variables categóricas

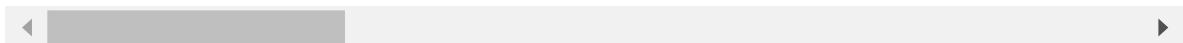
Procederemos a convertir las variables categóricas en un formato específico o codificación para su representación y almacenamiento en un sistema de manera que se puedan interpretar, procesar o transmitir de manera adecuada.

Para esto utilizaremos la técnica One Hot Encoding mediante la librería Scikit learn.

```
In [ ]: oh_model = OneHotEncoder()
cat_vars = data_std.select_dtypes(include=['object'])
df_oh_model = oh_model.fit_transform(cat_vars)
df_oh_model = pd.DataFrame(df_oh_model.toarray(), columns=oh_model.get_feature_n
df_oh_model.head()
```

	Comunidades y Ciudades Autónomas_01 Andalucía	Comunidades y Ciudades Autónomas_02 Aragón	Comunidades y Ciudades Autónomas_03 Asturias, Principado de	Comunidades y Ciudades Autónomas_04 Balears, Illes	Comunidades y Ciudades Autónomas_05 Canarias	Comu y C Autón Ci
0	1.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0

5 rows × 21 columns



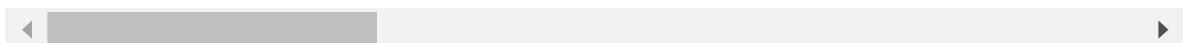
Concatenamos el dataframe original con el dataframe con las columnas codificadas.

Mantener las columnas originales y codificadas podría introducir redundancia y complicar el análisis. Las columnas codificadas ya contienen la información de las columnas originales de manera binaria, por lo tanto, procederemos a eliminar las columnas originales.

```
In [ ]: data_encoding = pd.concat([data_std, df_oh_model], axis=1)
data_encoding = data_encoding.drop(cat_vars, axis=1)
data_encoding.head()
```

	Mes	Año	Estancia Media	Alojamientos Ocupados	Noches Ocupadas	Comunidades y Ciudades Autónomas_01 Andalucía	Comunidades y Ciudades Autónomas_02 Aragón	Comu y C Autón Princ
0	1	2018	-0.952277	-0.112008	-0.249081	1.0	0.0	
1	1	2019	-0.891982	-0.058029	-0.214746	1.0	0.0	
2	1	2020	-0.891982	0.058926	-0.144170	1.0	0.0	
3	1	2021	-0.771391	-0.228963	-0.294861	1.0	0.0	
4	1	2022	-0.590505	0.049929	-0.106020	1.0	0.0	

5 rows × 26 columns



```
In [ ]: data_encoding.info()
```

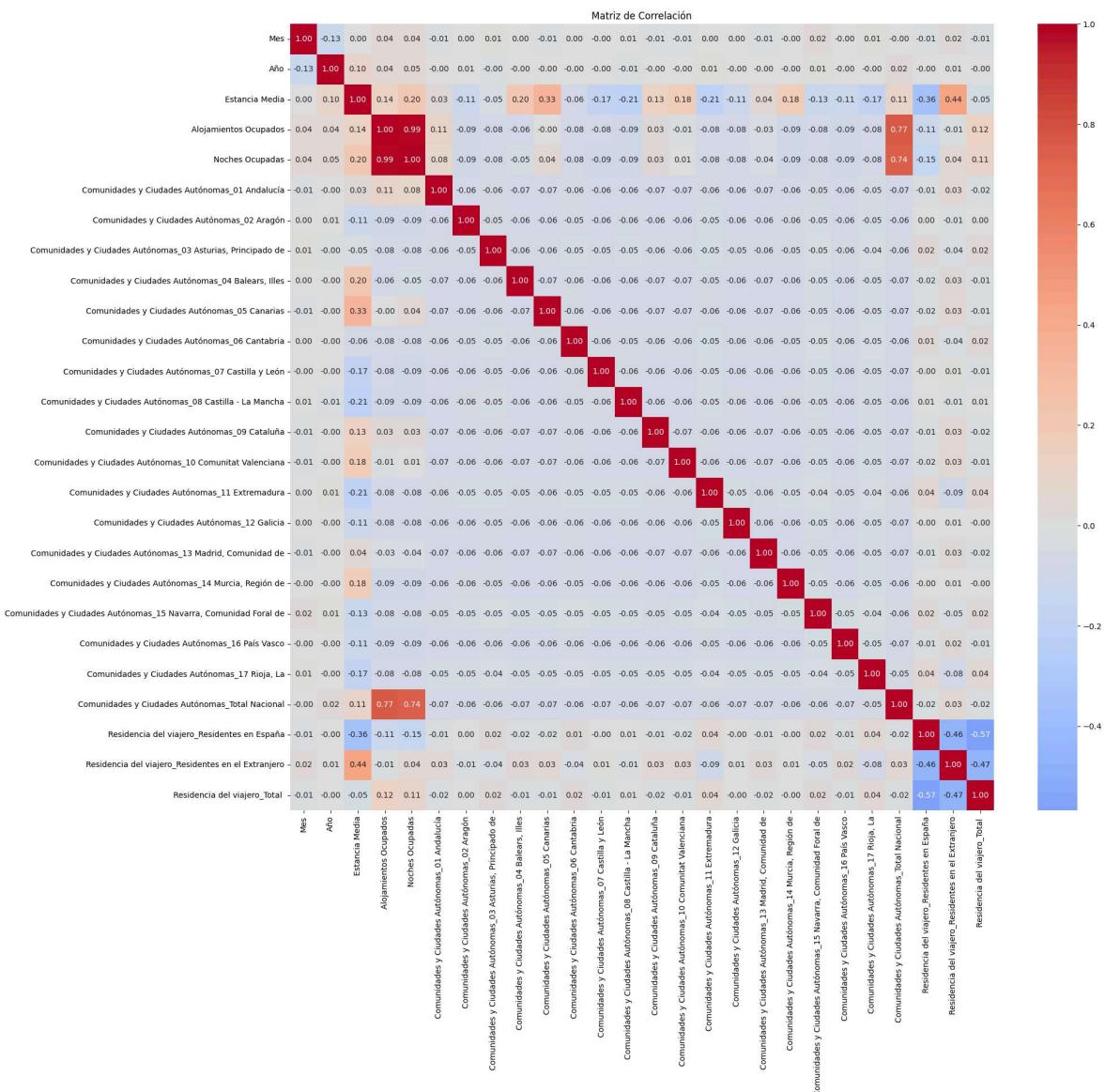
```
<class 'pandas.core.frame.DataFrame'>
Index: 3139 entries, 0 to 3138
Data columns (total 26 columns):
 #   Column           Non-Null Co
unit Dtype            -----
---  -----
0   Mes              3139 non-nu
11  int64
1   Año              3139 non-nu
11  int64
2   Estancia Media  3139 non-nu
11  float64
3   Alojamientos Ocupados 3139 non-nu
11  float64
4   Noches Ocupadas 3139 non-nu
11  float64
5   Comunidades y Ciudades Autónomas_01 Andalucía 3139 non-nu
11  float64
6   Comunidades y Ciudades Autónomas_02 Aragón 3139 non-nu
11  float64
7   Comunidades y Ciudades Autónomas_03 Asturias, Principado de 3139 non-nu
11  float64
8   Comunidades y Ciudades Autónomas_04 Balears, Illes 3139 non-nu
11  float64
9   Comunidades y Ciudades Autónomas_05 Canarias 3139 non-nu
11  float64
10  Comunidades y Ciudades Autónomas_06 Cantabria 3139 non-nu
11  float64
11  Comunidades y Ciudades Autónomas_07 Castilla y León 3139 non-nu
11  float64
12  Comunidades y Ciudades Autónomas_08 Castilla - La Mancha 3139 non-nu
11  float64
13  Comunidades y Ciudades Autónomas_09 Cataluña 3139 non-nu
11  float64
14  Comunidades y Ciudades Autónomas_10 Comunitat Valenciana 3139 non-nu
11  float64
15  Comunidades y Ciudades Autónomas_11 Extremadura 3139 non-nu
11  float64
16  Comunidades y Ciudades Autónomas_12 Galicia 3139 non-nu
11  float64
17  Comunidades y Ciudades Autónomas_13 Madrid, Comunidad de 3139 non-nu
11  float64
18  Comunidades y Ciudades Autónomas_14 Murcia, Región de 3139 non-nu
11  float64
19  Comunidades y Ciudades Autónomas_15 Navarra, Comunidad Foral de 3139 non-nu
11  float64
20  Comunidades y Ciudades Autónomas_16 País Vasco 3139 non-nu
11  float64
21  Comunidades y Ciudades Autónomas_17 Rioja, La 3139 non-nu
11  float64
22  Comunidades y Ciudades Autónomas_Total Nacional 3139 non-nu
11  float64
23  Residencia del viajero_Residentes en España 3139 non-nu
11  float64
24  Residencia del viajero_Residentes en el Extranjero 3139 non-nu
11  float64
25  Residencia del viajero_Total 3139 non-nu
11  float64
```

```
dtypes: float64(24), int64(2)
memory usage: 662.1 KB
```

## Matriz de Correlación

En esta matriz, cada fila y columna representan una variable, y los elementos de la matriz muestran las correlaciones entre las variables. Las correlaciones indican la fuerza y la dirección de la relación entre las variables, lo que puede ayudar a comprender cómo están relacionadas y si existe alguna dependencia.

```
In [ ]: correlation_matrix = data_encoding.corr()
plt.figure(figsize=(20, 18))
plt.title("Matriz de Correlación")
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', center=0)
plt.show()
```



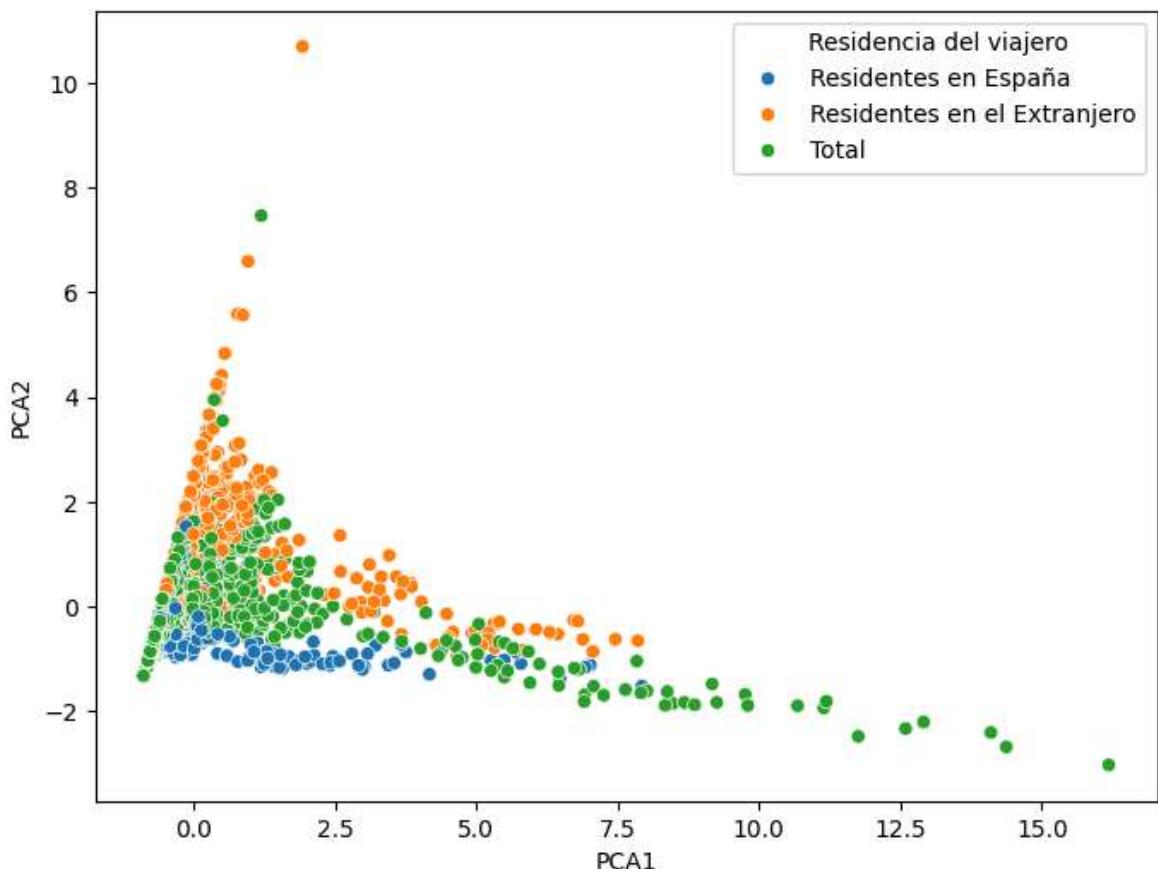
## PCA

```
In [ ]: pca_model = PCA(n_components=2)
df_pca = pd.DataFrame(pca_model.fit_transform(data_std.iloc[:, -3:]))
df_pca
```

	0	1
<b>0</b>	-0.464517	-0.873314
<b>1</b>	-0.390122	-0.829405
<b>2</b>	-0.261115	-0.860899
<b>3</b>	-0.535456	-0.668973
<b>4</b>	-0.172701	-0.570683
...	...	...
<b>3134</b>	4.680291	-1.027458
<b>3135</b>	5.551166	-1.233626
<b>3136</b>	1.411552	0.118796
<b>3137</b>	5.024056	-0.913720
<b>3138</b>	6.731811	-1.181912

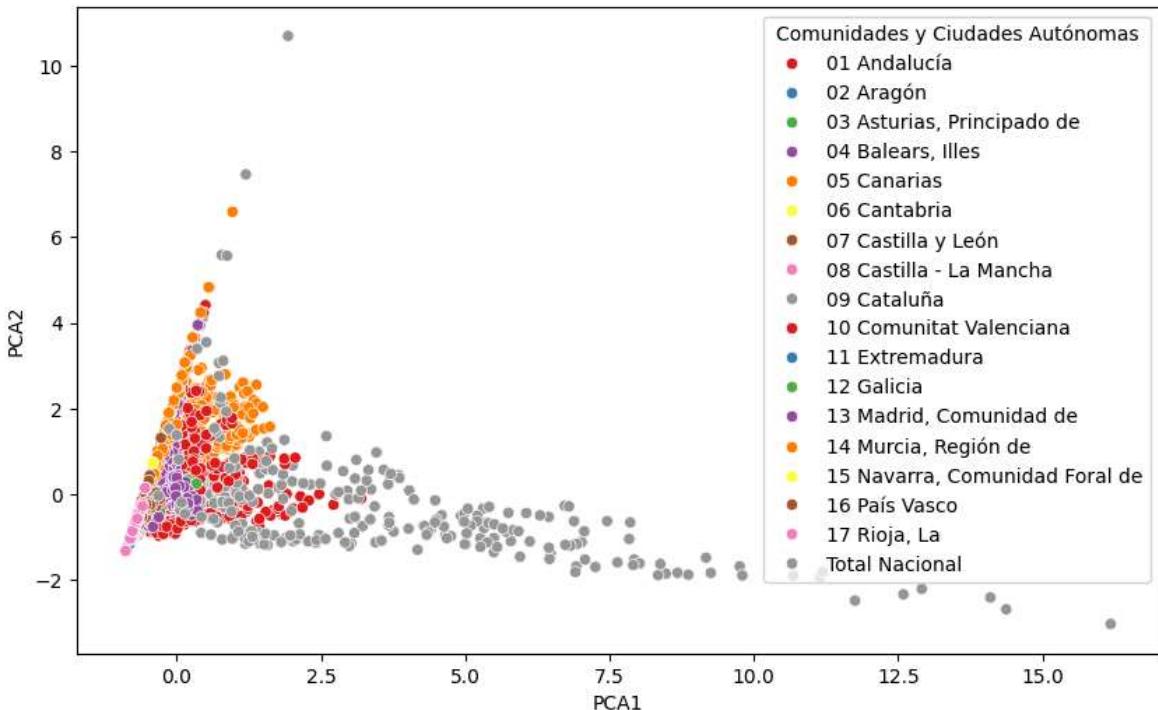
3139 rows × 2 columns

```
In [ ]: plt.figure(figsize=(8, 6))
sns.scatterplot(x=df_pca[0], y=df_pca[1], hue=data_std['Residencia del viajero'])
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()
```



```
In [ ]: plt.figure(figsize=(10, 6))
sns.scatterplot(x=df_pca[0], y=df_pca[1], hue=data_std['Comunidades y Ciudades A'])
```

```
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()
```



```
In [ ]: data_encoding.to_csv('../data/processed/data_processed.csv', index=False)
```

## Modelos de predicción

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

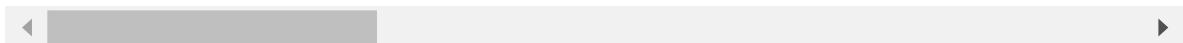
Cargamos el dataset procesado

```
In [ ]: df_models = pd.read_csv('../data/processed/data_processed.csv')
```

```
In [ ]: df_models.head()
```

Mes	Año	Estancia Media	Alojamientos Ocupados	Noches Ocupadas	Comunidades y Ciudades Autónomas_01 Andalucía	Comunidades y Ciudades Autónomas_02 Aragón	Comuni y Autón Princ
0	1	2018	-0.952277	-0.112008	-0.249081	1.0	0.0
1	1	2019	-0.891982	-0.058029	-0.214746	1.0	0.0
2	1	2020	-0.891982	0.058926	-0.144170	1.0	0.0
3	1	2021	-0.771391	-0.228963	-0.294861	1.0	0.0
4	1	2022	-0.590505	0.049929	-0.106020	1.0	0.0

5 rows × 26 columns

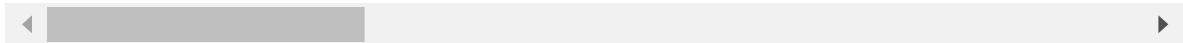


## Separación de los datos para entrenamiento y prueba de los modelos

```
In [ ]: target_columns = ['Estancia Media', 'Alojamientos Ocupados', 'Noches Ocupadas']
X = df_models.drop(target_columns, axis=1)
X.head()
```

Mes	Año	Comunidades y Ciudades Autónomas_01 Andalucía	Comunidades y Ciudades Autónomas_02 Aragón	Comunidades y Ciudades Autónomas_03 Asturias, Principado de	Comunidades y Ciudades Autónomas_04 Balears, Illes	Comunida y Ciuda Autónoma: Cana
0	1	2018	1.0	0.0	0.0	0.0
1	1	2019	1.0	0.0	0.0	0.0
2	1	2020	1.0	0.0	0.0	0.0
3	1	2021	1.0	0.0	0.0	0.0
4	1	2022	1.0	0.0	0.0	0.0

5 rows × 23 columns



```
In [ ]: y = df_models[target_columns]
y.head()
```

	Estancia Media	Alojamientos Ocupados	Noches Ocupadas
--	----------------	-----------------------	-----------------

0	-0.952277	-0.112008	-0.249081
1	-0.891982	-0.058029	-0.214746
2	-0.891982	0.058926	-0.144170
3	-0.771391	-0.228963	-0.294861
4	-0.590505	0.049929	-0.106020

```
In [ ]: # Separando los datos en train y test
train, test = train_test_split(df_models, test_size=0.25)

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
X_train.shape, X_test.shape

((2511, 23), (628, 23))
```

Guardamos los datasets de train y set

```
In [ ]: datasets = {
    'X_train': X_train,
    'X_test': X_test,
    'y_train': y_train,
    'y_test': y_test
}

for name, data in datasets.items():
    data.to_csv(f'../data/processed/{name}.csv', index=False)
```

## Regresión Lineal Múltiple

```
In [ ]: mlr_model = LinearRegression()

mlr_model.fit(X_train, y_train)

predictions_mlr_model = mlr_model.predict(X_test)
```

Raíz del error cuadrático medio del modelo de regresión lineal

```
In [ ]: rmse_mlr = np.sqrt(mean_squared_error(y_test, predictions_mlr_model))

print("RMSE:", rmse_mlr)
```

RMSE: 0.5455297443761002

Coeficiente de determinación del modelo de regresión lineal

```
In [ ]: r2_mlr = r2_score(y_test, predictions_mlr_model)

print("R2 Score:", r2_mlr)
```

R2 Score: 0.6388974376538233

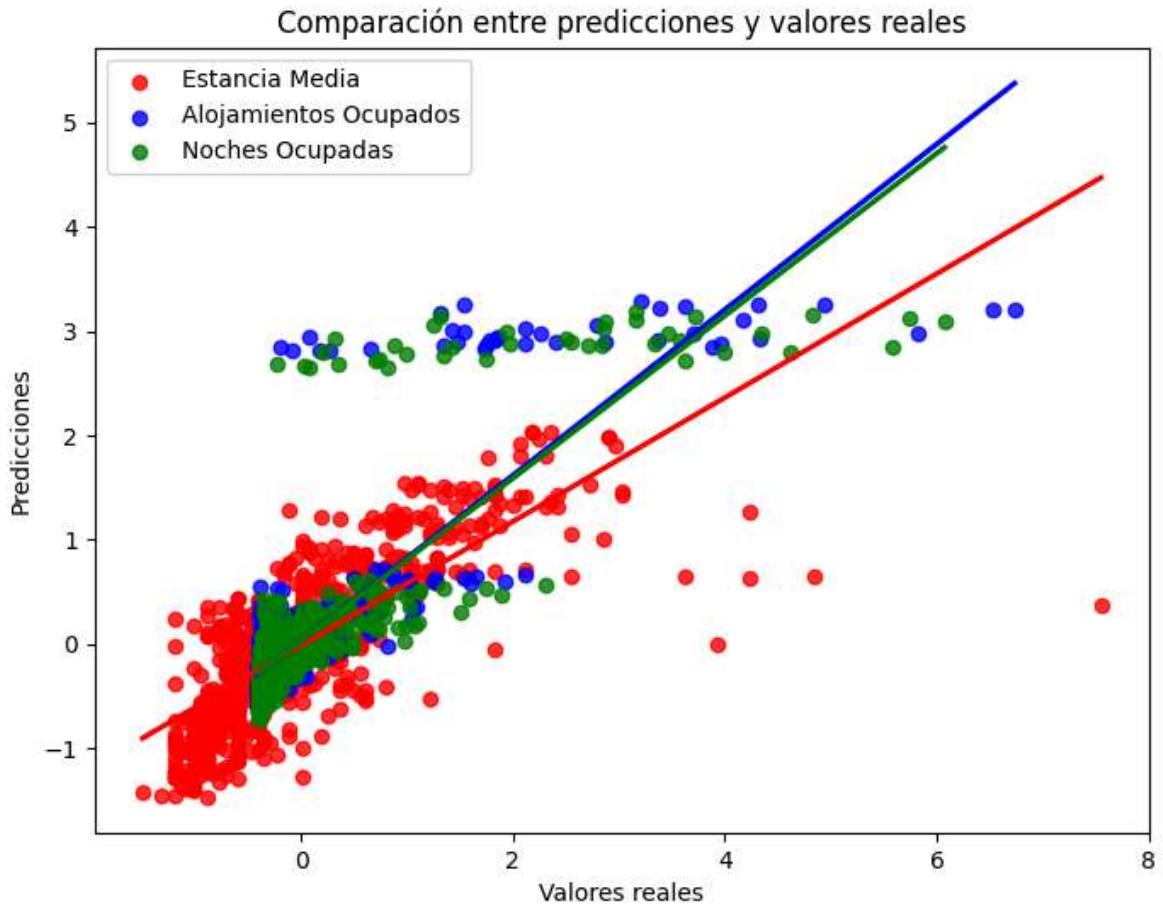
Creamos gráficos de regresión lineal para cada variable de respuesta

```
In [ ]: plt.figure(figsize=(8, 6))

sns.regplot(x=y_test['Estancia Media'], y=predictions_mlr_model[:,0], ci=None, c
sns.regplot(x=y_test['Alojamientos Ocupados'], y=predictions_mlr_model[:,1], ci=
sns.regplot(x=y_test['Noches Ocupadas'], y=predictions_mlr_model[:,2], ci=None,

plt.title('Comparación entre predicciones y valores reales')
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.legend()

plt.show()
```



## Random Forest

```
In [ ]: rfr_model = RandomForestRegressor(n_estimators=100, random_state=42)

rfr_model.fit(X_train, y_train)

predictions_rfr_model = rfr_model.predict(X_test)
```

### RMSE y R2 score del modelo Random Forest

```
In [ ]: rmse_rfr = np.sqrt(mean_squared_error(y_test, predictions_rfr_model))
r2_rfr = r2_score(y_test, predictions_rfr_model)

print("RMSE:", rmse_rfr)
print("R2:", r2_rfr)
```

RMSE: 0.3168377336270024

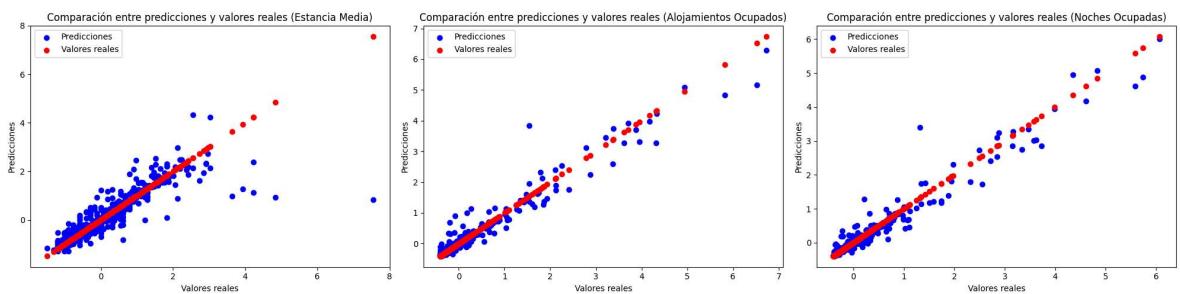
R2: 0.8923571179980216

Graficamos las predicciones vs los valores reales para cada variable de respuesta

```
In [ ]: fig, axs = plt.subplots(1, 3, figsize=(20, 5))

for i, variable in enumerate(['Estancia Media', 'Alojamientos Ocupados', 'Noches
    axs[i].scatter(y_test[variable], predictions_rfr_model[:, i], color='blue',
    axs[i].scatter(y_test[variable], y_test[variable], color='red', label='Valo
    axs[i].set_xlabel('Valores reales')
    axs[i].set_ylabel('Predicciones')
    axs[i].set_title(f'Comparación entre predicciones y valores reales ({variabl
    axs[i].legend()

plt.tight_layout()
plt.show()
```



Los resultados nos permiten concluir que el modelo de Random Forest es más preciso.