UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTD DE ESTUDIOS ESTADÍSTICOS

Curso 2024/2025

TRABAJO DE FIN DE MÁSTER

Máster de Big Data, Data Science e Inteligencia Artificial.

Ainhoa Díaz Cabrera

Madrid, 18 de septiembre de 2025.

ÍNDICE

INTRODUCCIÓN.

En este trabajo se desarrollará un estudio sobre el turismo en las Islas Canarias. Durante el último año el aumento del turismo en Canarias ha provocado en los residentes un descontento generalizado que ha llevado a manifestaciones. Esto es debido a que el crecimiento de los viajeros ha hecho que cada vez haya más apartamentos turísticos y menos residencias convencionales, suponiendo así una subida de los precios de los alquileres. También supone un aumento de los precios generales de los servicios, un aumento de la cantidad de estos e incluso en ciertos casos un declive de los hábitats naturales.

Por ello aquí vamos a realizar un estudio para comprobar si esto es cierto, si ha habido esa creciente cantidad de viajeros de la que se habla y si eso ha afectado a los gastos y la cantidad de servicios.

Para realizar el estudio tenemos 4 datasets obtenidos en el INE. Uno que indica los viajeros y pernoctaciones en el territorio canario, durante todos los meses desde enero del 2000 hasta mayo de 2025. El otro es lo mismo, pero para los apartamentos turísticos. El siguiente es el total de los gastos de los viajeros. Y el último es del total en los servicios, teniendo en cuenta el índice de servicios.

DEPURACIÓN DE DATOS.

En primer lugar, antes de realizar el estudio hay que realizar una depuración de los datos. Para ello comenzamos leyendo los distintos archivos.

```
#Conjunto de datos que indican los turistas que se han hospedado en hoteles.
hoteles = pd.read_csv('hoteles.csv', encoding='ISO-8859-1', sep=';')

#Conjunto de datos que indican los turistas que se han quedado en apartamentos turísticos.
apar_turisticos = pd.read_csv('apartamentos_turisticos.csv', encoding = 'ISO-8859-1', sep = ';')

#Conjunto de datos que indican los gastos distintos gastos, el total, el medio por persona, etc.
gastos = pd.read_csv('gastos.csv', encoding = 'ISO-8859-1', sep=';')

#Conjunto de datos con información sobre los tipos de servicios.
servicios = pd.read_csv('servicios.csv', encoding = 'ISO-8859-1', sep=';')
```

Lo primero es observar si se han cargado los tipos de los datos adecuadamente.

```
HOTELES
 Totales Territoriales               object
Comunidades y Ciudades Autónomas     object
Provincias                          float64
Viajeros y pernoctaciones            object
Residencia: Nivel 1                  object
Residencia: Nivel 2                  object
Periodo                              object
Total                                object
dtype: object
```

```
APARTAMENTOS
 Totales Territoriales               object
Comunidades y Ciudades Autónomas     object
Viajeros y pernoctaciones            object
Residencia: Nivel 1                  object
Residencia: Nivel 2                  object
Periodo                              object
Total                                object
dtype: object
```

```
GASTOS
 Comunidades  autónomas                object
Gastos y duración media de los viajes  object
Tipo de dato                           object
Periodo                                object
Total                                  object
dtype: object
```

```
SERVICIOS
 Comunidades y Ciudades Autónomas    object
Índice y Tasas                       object
Periodo                              object
Total                                object
dtype: object
```

En general hay que cambiar los periodos a datetime64 y los totales a int o float en el caso que corresponda.

```
#En primer lugar, en el conjunto de hoteles, hay que poner la fecha de forma correcta.
#También debemos transformar el total en un entero.
hoteles['Total'] = hoteles['Total'].astype(str).str.replace('.','').astype(int)
hoteles['Periodo'] = hoteles['Periodo'].str.replace('M', '-').str.strip()
hoteles['Periodo'] = pd.to_datetime(hoteles['Periodo'], format = '%Y-%m')
```

Pata el resto de los datasets se procede de forma análoga.

A continuación, al analizar los conjuntos de datos podemos observar que hay varias de las columnas que no aportan información por distintos motivos, por ejemplo, en el conjunto de datos de hoteles, una de las columnas es comunidades y ciudades autónomas, en este caso, hemos reducido solo a las islas canarias que son las deseadas para estudiar, por tanto, todo es Canarias. Y así, eliminaremos las columnas no necesarias en cada conjunto de datos.

```
hoteles = hoteles.drop(['Totales Territoriales', 'Comunidades y Ciudades Autónomas','Provincias', 'Viajeros y pernoctaciones',
                        'Residencia: Nivel 1', 'Residencia: Nivel 2'], axis=1)
```

```
apar_turisticos = apar_turisticos.drop(['Totales Territoriales', 'Comunidades y Ciudades Autónomas', 'Viajeros y pernoctaciones',
                                        'Residencia: Nivel 1', 'Residencia: Nivel 2'], axis=1)
```

```
gastos = gastos.drop(['Comunidades  autónomas', 'Gastos y duración media de los viajes',
                      'Tipo de dato'], axis=1)
```

```
servicios = servicios.drop(['Comunidades y Ciudades Autónomas','Índice y Tasas'], axis=1)
```

Lo siguiente a tener en cuenta, es que hay varios 0 en los conjuntos de datos, esto es debido a que son valores perdidos por ello vamos a reemplazarlos por nan.

```
hoteles['Total'] = hoteles['Total'].replace(0,np.nan)
apar_turisticos['Total'] = apar_turisticos['Total'].replace(0,np.nan)
gastos['Total'] = gastos['Total'].replace('Total').replace(0, np.nan)
servicios['Total'] = servicios['Total'].replace('Total').replace(0, np.nan)
```

Ahora comprobamos cuantos hay en cada caso.

```
print(hoteles.isna().sum(), '\n')
print(apar_turisticos.isna().sum(), '\n')
print(gastos.isna().sum(), '\n')
print(servicios.isna().sum(),'\n')
```

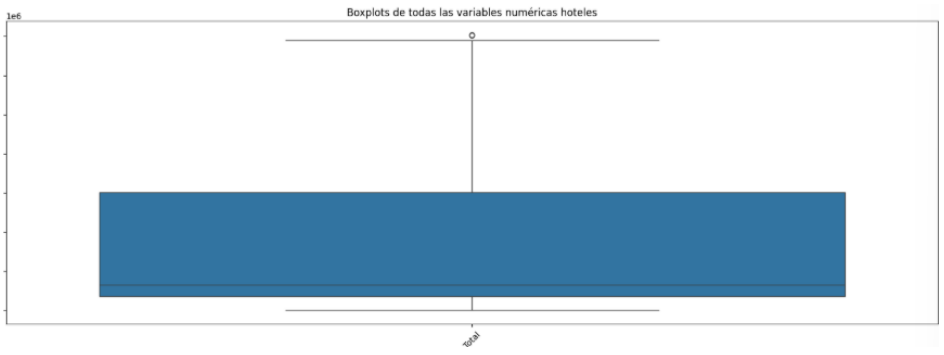
```
Periodo    0
Total      6
dtype: int64

Periodo     0
Total      10
dtype: int64

Periodo    0
Total      2
dtype: int64

Periodo    0
Total      0
dtype: int64
```

Y para tratarlos lo que haremos será imputarlos usando la media.

```
#Imputamos usando la media
media_Total_hot = hoteles['Total'].mean()
hoteles['Total'] = hoteles['Total'].fillna(media_Total_hot)

media_Total_apar = apar_turisticos['Total'].mean()
apar_turisticos['Total'] = apar_turisticos['Total'].fillna(media_Total_apar)

media_Total_gastos = gastos['Total'].mean()
gastos['Total'] = gastos['Total'].fillna(media_Total_gastos)
```

Comprobaremos ahora mediante boxplots, los outliers.

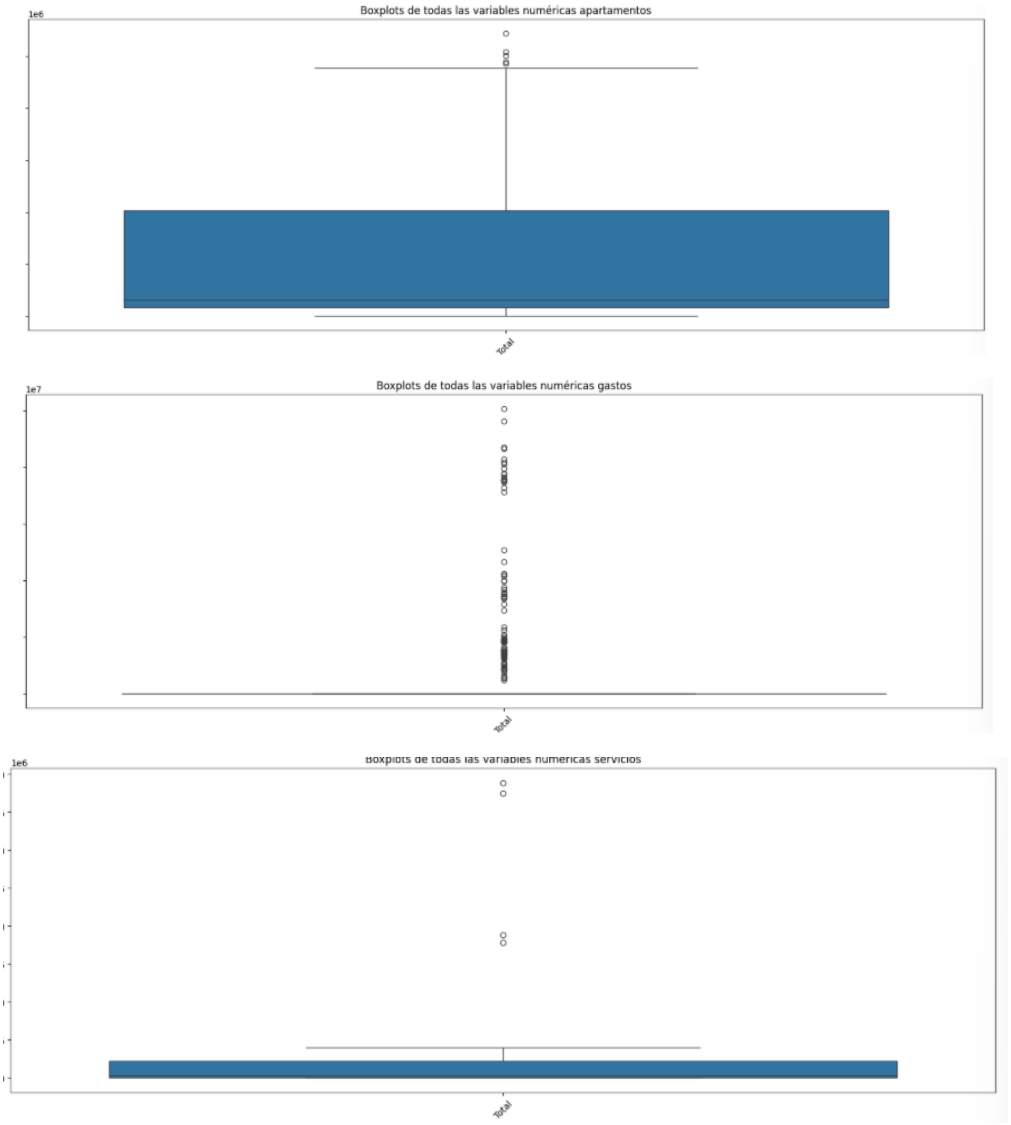
Para los hoteles tenemos.



Y se imputarán los outliers.

```python
#Imputamos los outliers
for col in numericas_hot.columns:
    Q1 = hoteles[col].quantile(0.25)
    Q3 = hoteles[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    media = hoteles[col].mean()
    hoteles[col] = np.where(
        (hoteles[col] < lower_bound) | (hoteles[col] > upper_bound),
        media,
        hoteles[col]
    )
```

Esto lo realizamos para todo, a continuación, pondré los boxplots de los apartamentos, los gastos y los servicios.



Otra cosa útil es agrupar los totales por periodo, ya que lo que nos interesa realmente es el total de viajeros en general, nos da igual si por ejemplo son residentes españoles o extranjeros y las demás diferenciaciones que se realizan en los conjuntos de datos.

```
#Vamos a sumar los totales por periodo.
# Hoteles
hoteles_unid = hoteles.groupby('Periodo', as_index=False)['Total'].sum()
hoteles_unid = hoteles_unid.rename(columns={'Total': 'Total_hoteles'})

# Apartamentos
apar_unid = apar_turisticos.groupby('Periodo', as_index=False)['Total'].sum()
apar_unid = apar_unid.rename(columns={'Total': 'Total_apartamentos'})

#Gastos y servicios no hace falta porque solo hay un valor total por periodo.
```

Ahora que hemos realizado esto podemos unir en un único conjunto de datos tanto el total de turistas hospedados en hoteles como en apartamentos turísticos, puesto que ambos conjuntos de datos los tenemos por periodo de meses desde enero del año 2000 hasta mayo de 2025.

```
#De ahora en adelante para facilidad de uso, vamos a juntar los ficheros de hoteles y apartamentos.
#Esto tiene sentido puesto que tienen los mismos periodos y además aportan la misma información, la de turistas.
hot_apar = pd.merge(hoteles_unid, apar_unid, on = 'Periodo', how = 'outer')
```

Ahora que tenemos los dos conjuntos de datos unidos en uno solo podemos crear algunas variables que nos aporten información interesante. Crearemos las siguientes, por un lado, el número total de alojamientos, que es el número total de viajeros realmente. Con esto podremos tener el porcentaje de los viajeros en hoteles y el porcentaje en los apartamentos y también vamos a crear dos variables que indiquen los cambios mensuales de los hoteles y de los apartamentos.

```
#Creamos el total de alojamientos que se han usado
hot_apar['Total_alojamientos'] = hot_apar['Total_hoteles'] + hot_apar['Total_apartamentos']

#Porcentaje que representan los hoteles y los apartamentos respecto al total.
hot_apar['Pct_hoteles'] = hot_apar['Total_hoteles'] / hot_apar['Total_alojamientos']*100
hot_apar['Pct_apar'] = hot_apar['Total_apartamentos'] / hot_apar['Total_alojamientos']*100

#Vaciación mensual
hot_apar['Cambio_mensual_hot'] = hot_apar['Total_hoteles'].pct_change() * 100
hot_apar['Cambio_mensual_apar'] = hot_apar['Total_apartamentos'].pct_change() * 100
```

Ahora también podemos añadir los gastos. El dataset de los gastos nos proporciona información desde 2015, por lo que al unirlos solo obtendremos la información de este año en adelante.

```
hot_apar_gastos = pd.merge(hot_apar, gastos, on = 'Periodo', how = 'inner')
#Renombramos columna de total de gastos
hot_apar_gastos = hot_apar_gastos.rename(columns={'Total': 'Total_gastos'})
```

Este conjunto de datos será útil de cara al estudio de las relaciones entre los viajeros y los gastos. También pensando en ello, creamos un conjunto de datos de los viajeros con el índice de servicios proporcionado. En este caso los datos empezarán desde 2025.

```
hot_apar_serv = pd.merge(hot_apar, servicios, on = 'Periodo', how = 'inner')
#Renombramos columna de servicios
hot_apar_serv = hot_apar_serv.rename(columns={'Total': 'Total_servicios'})
```

Y, por último, puede ser útil tener toda la información junta.

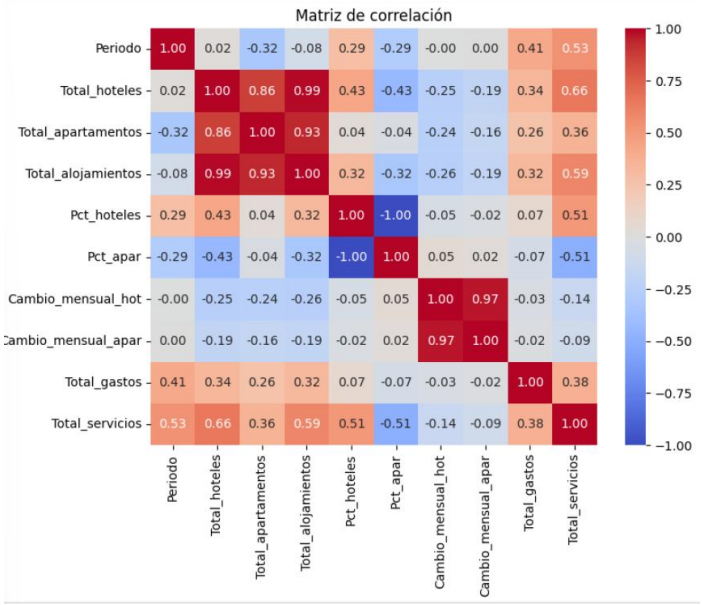
```
#También vamos a crear un conjunto con todos los datos que puede ser útil.
hot_apar_gast_serv = pd.merge(hot_apar_gastos, servicios, on = 'Periodo', how = 'inner')
#Renombramos columna de servicios
hot_apar_gast_serv = hot_apar_gast_serv.rename(columns={'Total': 'Total_servicios'})
```

Para enriquecer nuestros conjuntos de datos de cara al estudio y creación de los modelos es interesante crear algunas variables temporales, además de lags y medias móviles.

```
hot_apar['Mes'] = hot_apar['Periodo'].dt.month
hot_apar['Trimestre'] = hot_apar['Periodo'].dt.quarter

hot_apar['lag1'] = hot_apar['Total_alojamientos'].shift(1)
hot_apar['lag3'] = hot_apar['Total_alojamientos'].shift(3)
hot_apar['rolling3'] = hot_apar['Total_alojamientos'].rolling(3).mean()
```

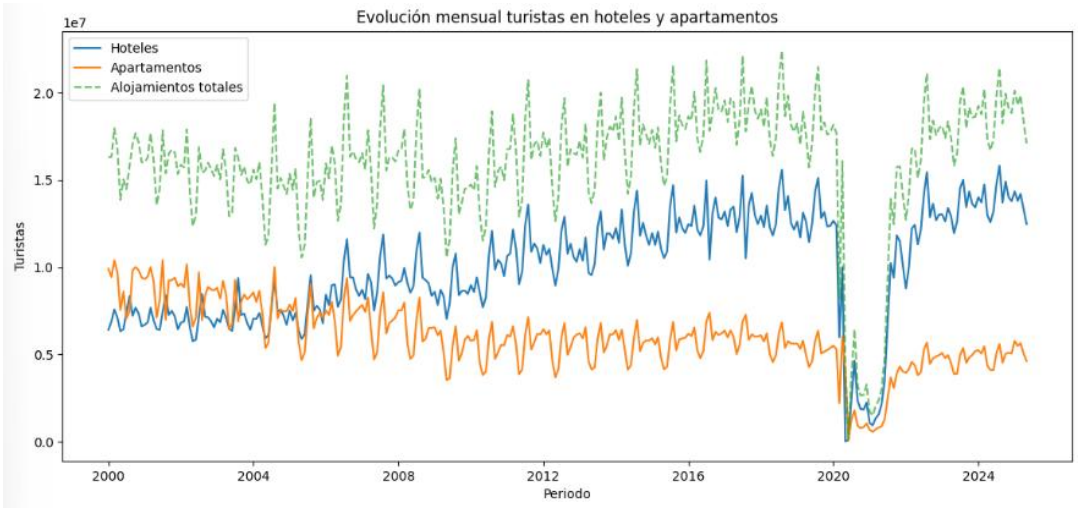
ANÁLISIS DESCRIPTIVO. VISUALIZACIONES, RELACIONES Y TENDENCIAS TEMPORALES.

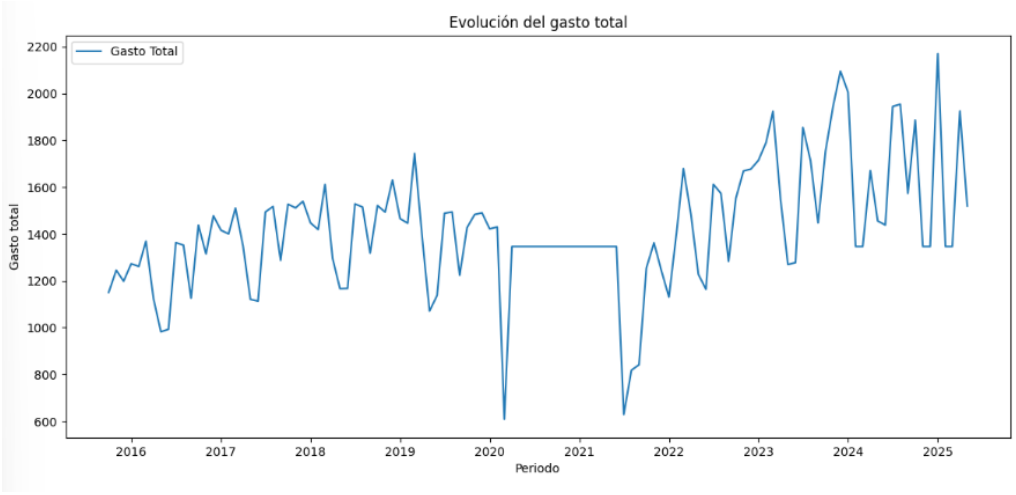
Usando el conjunto de datos que tiene toda la información, podemos crear la matriz de correlaciones.

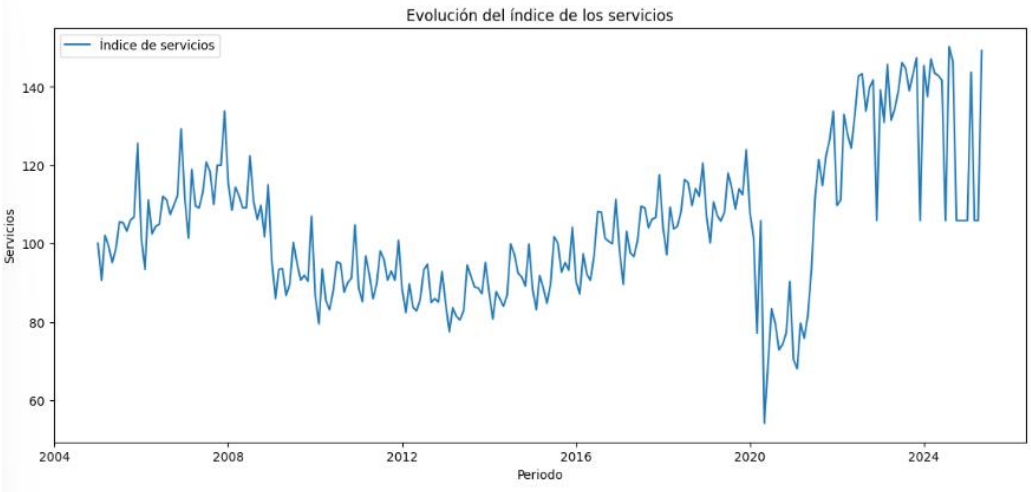


Claramente el total de los hoteles y el total de los apartamentos está fuertemente correlacionado con el total de los alojamientos, debido a que esta última se obtiene juntando las otras dos. También podemos ver que el porcentaje de los hoteles y el de los apartamentos está inversamente relacionada, es decir cuando una aumenta la otra disminuye y viceversa. Se observa cambio como los cambios mensuales de los hoteles y los apartamentos están muy relacionados, esto es debido a que ambos siguen casi la misma tendencia mensual, lo que tiene sentido si tenemos en cuenta que los viajeros crecen y decrecen en hoteles y apartamentos en los mismos meses. Es interesante destacar que el total de los hoteles y de los alojamientos tienen una relación respectiva con el total de servicios de 0.66 y 0.59, por lo que podemos prever de esto que a más turistas tanto en hoteles como en apartamentos mayor demanda de los servicios. Y otra cosa a tener en cuenta es que no observamos mucha relación entre el total de alojamientos y el de los gastos, por lo que nos puede indicar que el aumento de los turistas no siempre implica proporcionalmente más gasto.

Después de observar las relaciones podemos visualizar la evolución mensual de los turistas, gastos y servicios.



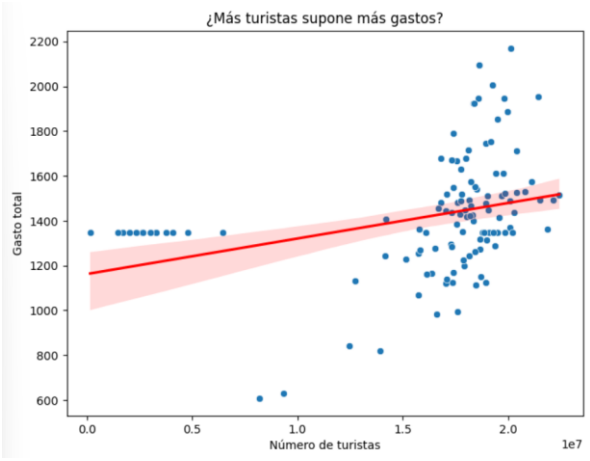




Podemos comprobar que hay una tendencia a lo largo de los años en los meses que hay más turistas, que coincide con los que hay más servicios. También está relacionado con los gastos, pero es menos clara la tendencia. En las tres gráficas está claro que todo tuvo en decrecimiento en la época del COVID como era de esperar. También es importante destacar que los servicios estos últimos años son más que antes de 2020 y que tanto los hoteles como los apartamentos están volviendo a crecer cada año en número de viajeros.

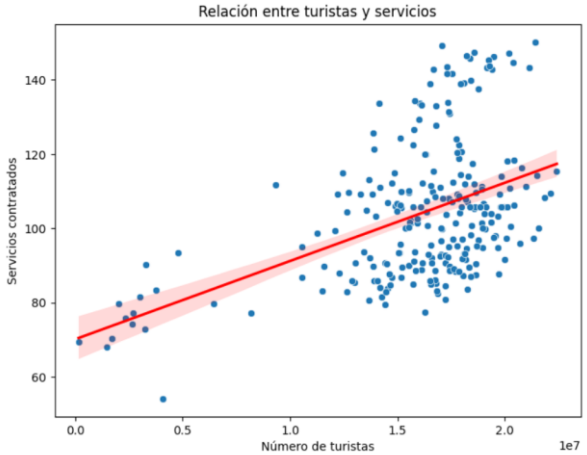
Con la matriz de correlación ya vimos un poco las relaciones que existían entre las variables pero, realmente, ¿más turistas suponen más gastos? Y también, ¿tener más turistas implica directamente que haya más servicios?

Respondiendo a la primera pregunta tenemos este gráfico:



Analizando esto podemos, tal y como habíamos previsto con la matriz de correlaciones, determinar que, aunque existe cierta relación no es directamente proporcional. Lo observamos en que, la recta roja muestra una tendencia positiva, lo que nos dice que cuanto más número de turistas más gastos, pero la pendiente es muy suave, lo que nos dice que el crecimiento del gasto no es proporcional al aumento de turistas. Vemos una correlación moderada debido a que los puntos está bastante dispersos alrededor de la recta. Cabe destacar que hay valores atípicos, probablemente debido a la época de la pandemia, que fue un periodo especial. Con todo esto llegamos a la conclusión de que, aunque el gasto depende de los turistas, también depende de otros factores y por ello podríamos concluir que aumentar los turistas no nos asegura un incremento equivalente de los ingresos en los negocios (es decir, gastos de los turistas).

Ahora veamos la gráfica de relación para responder a la segunda pregunta que nos habíamos planteado.



En este caso si podemos ver como la pendiente de la línea roja es bastante más pronunciada, por lo que el aumento de los turistas aumenta proporcionalmente

el número de servicios. Aquí la correlación es mucho más fuerte tal y como habíamos previsto anteriormente con la matriz de correlación. También, al observar los puntos, podemos comprobar que hay cierta variabilidad, pero mucho más moderada, hay menos dispersión que en el anterior gráfico, en este la tendencia está más clara. Con todo esto, nos indica que los servicios sí dependen directamente del volumen de viajeros. Esto puede llegar a ser un problema, puesto que como vimos la tendencia del turismo va en aumento cada año, por lo que podría significar una saturación de los servicios que estarían sometidos en algunas épocas a una presión significativa, de hecho, esta es una de las quejas que los canarios tienen, puesto que los servicios que acostumbran a usar, como por ejemplo, el transporte público, se satura mucho con la llegada de los turistas y la población residente acaba por no poder utilizarlos debido a la gran masa de viajeros.

MODELIZACIÓN PREDICTIVA.

A continuación, procederemos a realizar los modelos de machine learning. Comenzamos con preparar las variables que necesitamos para entrenar los modelos.

```
#Primero debemos convertir variables temporales en características útiles para nuestros modelos.
#Para ello realizamos una copia de nuestro dataset combinado con todos.
df_combinado = hot_apar_gast_serv.copy()
#Extraemos año, mes que será en función de lo que se predecirá el número de turistas y los gastos totales.
df_combinado['Año'] = df_combinado['Periodo'].dt.year
df_combinado['Mes'] = df_combinado['Periodo'].dt.month

#Determinamos las variables predictoras, debido a que la suma de hoteles y apartamentos da los alojamientos
#no podemos usarlas para predecir
X_turis = df_combinado[['Año', 'Mes', 'Total_servicios']]
#Variables que queremos predecir
y_turistas = df_combinado['Total_alojamientos']

#Determinamos las variables predictoras
X_gas = df_combinado[['Año', 'Mes', 'Total_hoteles', 'Total_apartamentos', 'Total_servicios']]
#Variable a predecir
y_gastos = df_combinado['Total_gastos']
```

La idea es predecir el número de turistas y el gasto total en función de diferentes factores. Comenzamos realizando una regresión lineal. Este es un modelo que es fácil de interpretar aun no captura las relaciones no lineales. Lo he elegido como primer modelo por ser sencillo, interpretable y que permite establecer la base de rendimiento. Veamos que obtenemos.

```
#Realizamos las predicciones del modelo
y_pred_turisLin = regresionLineal_turis.predict(X_test_turis)

#Calculamos las métricas:
mae_turis_lin = mean_absolute_error(y_test_turis, y_pred_turisLin)
rmse_turis_lin = np.sqrt(mean_squared_error(y_test_turis, y_pred_turisLin))
r2_turis_lin = r2_score(y_test_turis, y_pred_turisLin)

print('Regresión lineal')
print('Mae', mae_turis_lin, 'Rmse', rmse_turis_lin, 'R2', r2_turis_lin)
```

```
Regresión lineal
Mae 5750148.416280508 Rmse 8520574.237438295 R2 -47.594518130182735
```

Obtenemos en el caso de predicción del número de turistas, que la regresión lineal no es un modelo adecuado para predecir el número de turistas. R2 es negativo, lo que indica que la predicción es peor que si calculáramos simplemente la media, por lo que esto nos indica que las relaciones entre las variables explicativas y el turismo son demasiado complejas para capturarlas correctamente con un modelo lineal simple. Además, tenemos errores muy altos (MAE y RMSE), lo que nos implica que se equivoca la predicción en millones de turistas.

Ahora veamos qué pasa con las predicciones de los gastos.

```python
#Realizamos las predicciones del modelo
y_pred_gasLin = regresionLineal_gas.predict(X_test_gas)

#Calculamos las métricas:
mae_gas_lin = mean_absolute_error(y_test_gas, y_pred_gasLin)
rmse_gas_lin = np.sqrt(mean_squared_error(y_test_gas, y_pred_gasLin))
r2_gas_lin = r2_score(y_test_gas, y_pred_gasLin)

print('Regresión lineal')
print('Mae', mae_gas_lin, 'Rmse', rmse_gas_lin, 'R2', r2_gas_lin)
```

```
Regresión lineal
Mae 248.415554447997 Rmse 279.1283413644523 R2 0.010332612265888774
```

Obtenemos que el error absoluto y el cuadrático medio son bastante altos en comparación con el rango típico de gasto. Y también tenemos que el r2 = 0,01 significa que apenas se explica un 1% la variabilidad de los datos. Todo esto lo que nos indica es que la relación entre las variables explicativas y el gasto no es lineal ni fuerte, cosa que ya hemos ido comentando a lo largo del trabajo.

Puesto que la regresión lineal no nos ha servido, cosa que tiene sentido porque no captura las relaciones no lineales, y hemos estado viendo que, aunque hay relaciones en las variables, quizás no sean tan proporcionales como se puede pensar a priori, vamos a probar a usar el modelo de predicción GridSearch con Random Forest. En este caso, elegimos este modelo porque nos permite capturar relaciones que sean no lineales y más complejas, lo cual no se consigue con la regresión lineal. Comencemos con la predicción del número de turistas.

```python
#El siguiente será el modelo gridsearch con random forest
#Definimos la rejilla de parámetros a probar
param_grid={'n_estimators':[100,200,300],
            'max_depth':[5,10,None],
            'min_samples_split':[2,5],
            'min_samples_leaf':[1,2]}
#GridSearchCV con random forest
tscv = TimeSeriesSplit(n_splits=5)
grid_turisRf = GridSearchCV(RandomForestRegressor(random_state=42),
                        param_grid,
                        cv= tscv,
                        scoring= 'neg_mean_absolute_error',
                        n_jobs=-1)
#Entrenamos el gridSearch
grid_turisRf.fit(X_train_turis, y_train_turis)
#Elegimos mejor modelo.
best_turis_rf = grid_turisRf.best_estimator_
#Predicciones
y_pred_turisRf = best_turis_rf.predict(X_test_turis)

mae_turis_rf = mean_absolute_error(y_test_turis, y_pred_turisRf)
rmse_turis_rf = np.sqrt(mean_squared_error(y_test_turis, y_pred_turisRf))
r2_turis_rf = r2_score(y_test_turis, y_pred_turisRf)
```

Obtenemos los valores:

```
Random Forest Optimizado
Mae 1894721.1370496545 Rmse 2360971.0360368486 R2 -2.7310482820289446
```

Mejoramos con respecto a la regresión lineal, obtenemos mucha más precisión, puesto que los errores son mucho menores. Aunque, por otro lado, seguimos teniendo un R2 negativo, lo que nos indica que el modelo sigue siendo peor que una predicción trivial de la media, esto nos indica que el modelo no logra capturar adecuadamente los patrones en los datos por lo que sería arriesgado basar una estrategia en estas predicciones. Todo esto lo más probable es que sea debido

a que es un conjunto de datos con una estacionalidad fuerte, el turismo se mide por patrones cíclicos, en verano siempre hay más turistas y en invierno baja, volviendo a subir algo en navidad, y así. Esto no es captado adecuadamente por el modelo de predicción Random Forest.

Comprobemos ahora las predicciones para los gastos.

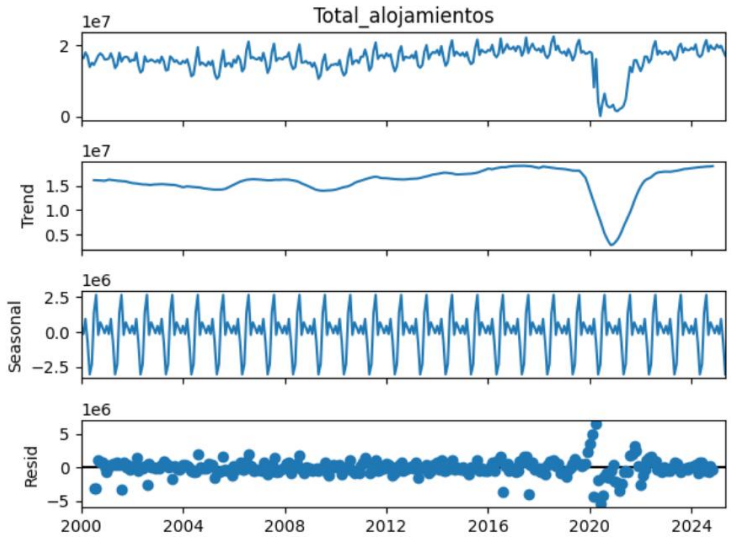
```
#El siguiente será el modelo Random Forest
#Definimos la rejilla de parámetros a probar
param_grid={'n_estimators':[100,200,300],
            'max_depth':[5,10,None],
            'min_samples_split':[2,5],
            'min_samples_leaf':[1,2]}
#GridSearchCV con random forest
tscv_gas = TimeSeriesSplit(n_splits=5)
grid_gasRf = GridSearchCV(RandomForestRegressor(random_state=42),
                      param_grid,
                      cv= tscv_gas,
                      scoring= 'neg_mean_absolute_error',
                      n_jobs=-1)
#Entrenamos el gridSearch
grid_gasRf.fit(X_train_gas, y_train_gas)
#Elegimos mejor modelo.
best_gas_rf = grid_gasRf.best_estimator_
#Predicciones
y_pred_gasRf = best_gas_rf.predict(X_test_gas)

mae_gas_rf = mean_absolute_error(y_test_gas, y_pred_gasRf)
rmse_gas_rf = np.sqrt(mean_squared_error(y_test_gas, y_pred_gasRf))
r2_gas_rf = r2_score(y_test_gas, y_pred_gasRf)
```

Obtenemos los siguientes resultados:

```
Random Forest Optimizado
Mae 257.36330570650676 Rmse 302.78666983685565 R2 -0.16454129236487125
```

En este caso, para predecir los gastos obtenemos peores resultados con el Random Forest. Los errores aumentan y ahora tenemos un R2 negativo, no se logra captar la variabilidad real de los datos. Todo esto nos sugiere que los gastos tienen una estructura más lineal con las variables predictoras que tenemos en nuestro conjunto de datos. Sin embargo, estas no son suficientes para predecir adecuadamente los gastos realizados por los turistas, lo que nos implica, como habíamos visto anteriormente, que, aunque está relacionado los gastos con el número de turistas, hay muchos más factores (que en estos conjuntos de datos no se tienen en cuenta) que influyen en los gastos realizados.

Por último, como la serie de los turistas es estacional vamos a realizar un SARIMA. Esto lo haremos debido a que este conjunto de datos presenta un comportamiento estacional y dependiente del tiempo, lo cual suele estar bien capturado por este modelo. En primer lugar, mostramos la descomposición estacional.



En la segunda gráfica que pertenece a la tendencia, observamos como va habiendo un crecimiento suave hasta 2019, donde por efecto del COVID hay una caída abrupta, que posteriormente se recupera. En cuanto a la estacionalidad (seasonal), el patrón es claro anualmente, los picos y las caídas se repiten de manera constante cada 12 meses. En cuanto a los residuales, sin tener en cuenta los años del COVID, hay bastante estabilidad.

Ahora veamos el modelo SARIMA.

```
modelo_auto = auto_arima(train_turisAR,
                         start_p=0, start_q=0, max_p=3, max_q=3,
                         start_P=0, start_Q=0, max_P=2, max_Q=2,
                         seasonal=True, m=12, d=None, D=None, trace=True,
                         error_action='ignore', suppress_warnings=True,
                         stepwise=True)
print(modelo_auto.summary())
```

Tenemos los resultados:

```
Best model:  ARIMA(0,1,2)(1,0,1)[12]
Total fit time: 33.442 seconds
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                  244
Model:             SARIMAX(0, 1, 2)x(1, 0, [1], 12)   Log Likelihood               -3805.904
Date:                              Fri, 05 Sep 2025   AIC                           7621.808
Time:                                      13:13:48   BIC                           7639.274
Sample:                                  01-01-2000   HQIC                          7628.843
                                       - 04-01-2020
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.5269      0.052    -10.052      0.000      -0.630      -0.424
ma.L2         -0.1711      0.059     -2.891      0.004      -0.287      -0.055
ar.S.L12       0.8801      0.056     15.805      0.000       0.771       0.989
ma.S.L12      -0.5955      0.099     -6.026      0.000      -0.789      -0.402
sigma2      3.414e+12   1.99e-14   1.72e+26      0.000    3.41e+12    3.41e+12
===================================================================================
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):              1670.08
Prob(Q):                              0.83   Prob(JB):                         0.00
Heteroskedasticity (H):               0.94   Skew:                            -1.27
Prob(H) (two-sided):                  0.78   Kurtosis:                        15.59
===================================================================================
```

Y por último las métricas de evaluación que obtenemos:

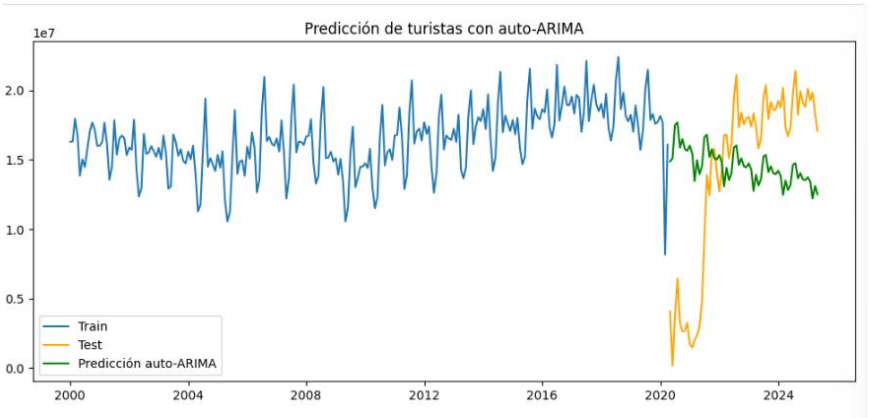
```
#Evaluamos
mae_turisAR = mean_absolute_error(test_turisAR, pred_turis)
rmse_turisAR = math.sqrt(mean_squared_error(test_turisAR, pred_turis))
```

```
ARIMA
Mae 5978628.315547884
Rmse 7176963.709107422
```

Obtenemos valores bastante altos, nuestro modelo capta la tendencia y variabilidad, pero aun así hay errores bastante grandes. Esto puede ser debido al impacto del COVID que rompe la serie y no puede preverse con el modelo SARIMA, que es estadístico. En contextos más estables, el modelo tendría un desempecho más fiable.

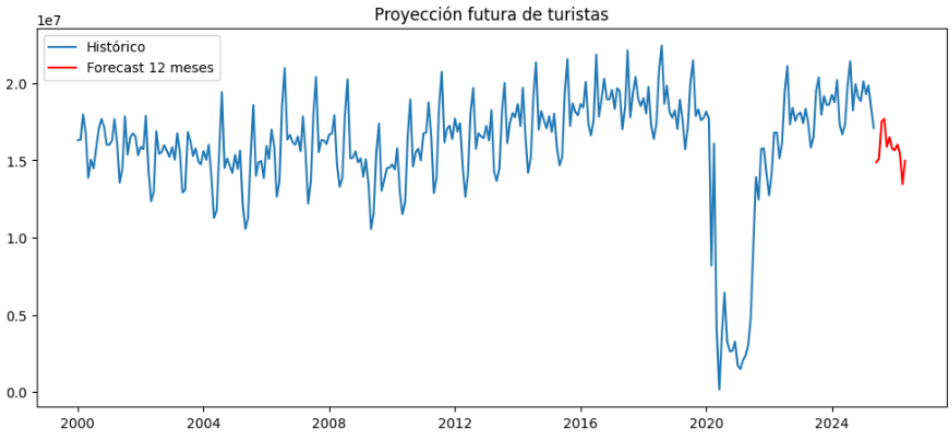
Con esto podemos concluir, que el problema no es el modelo en sí, sino los datos y los choques externos que son impredicibles.

Veamos lo descrito anteriormente mediante gráficos.



Aquí se observa claramente como el modelo no capta esta caída abrupta en 2020, como habíamos comentado anteriormente. El modelo predice que va a haber una caída leve cada año, siguiendo la tendencia anterior. Falla frente a eventos disruptivos como lo que se vivió con el COVID.

A continuación, veamos la predicción futura de los turistas.



El forecast nos proyecta una ligera tendencia a la baja, cuando realmente está en aumento, con la caída abrupta y los picos muy altos pre-COVID puede que el modelo no refleje adecuadamente la tendencia futura.

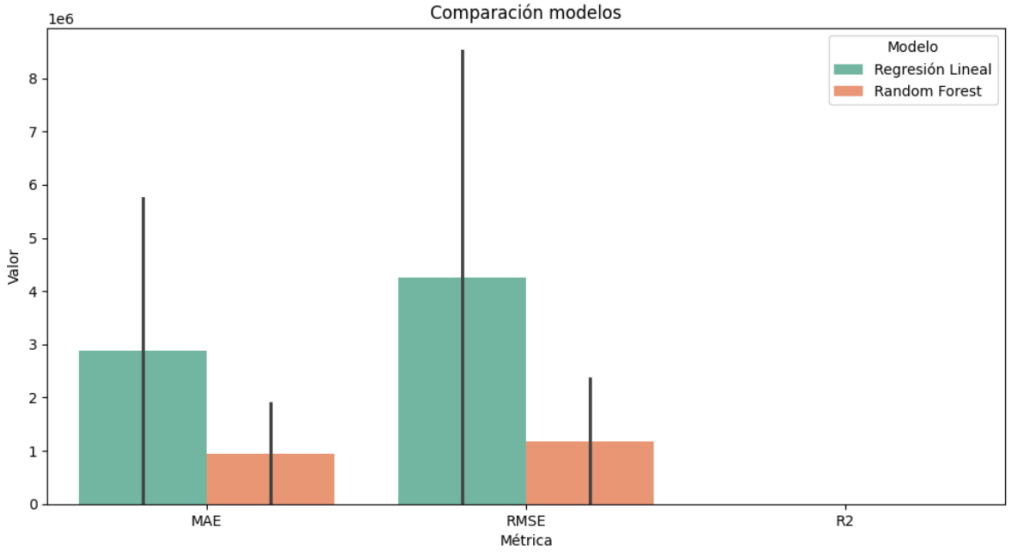
Para el conjunto de datos de los gastos no realizamos el SARIMA debido a que no es tan estacional y no nos predeciría bien.

Para hacernos una idea de los valores de los modelos anteriores tenemos la siguiente tabla y gráfico de barras.

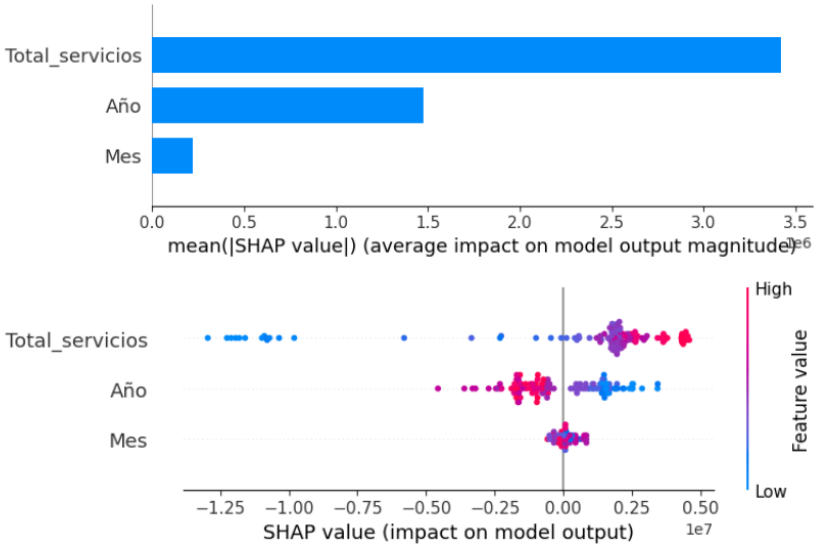
```
resultados = pd.DataFrame({ 'Modelo': ['Regresión Lineal', 'Random Forest', 'Regresión Lineal', 'Random Forest'],
                            'Variable': ['Turistas', 'Turistas', 'Gastos', 'Gastos'],
                            'MAE': [mae_turis_lin, mae_turis_rf, mae_gas_lin, mae_gas_rf],
                            'RMSE':[rmse_turis_lin, rmse_turis_rf, rmse_gas_lin, rmse_gas_rf],
                            'R2': [r2_turis_lin, r2_turis_rf, r2_gas_lin, r2_gas_rf]})
print(resultados)
```

	Modelo	Variable	MAE	RMSE	R2
0	Regresión Lineal	Turistas	5.750148e+06	8.520574e+06	-47.594518
1	Random Forest	Turistas	1.894721e+06	2.360971e+06	-2.731048
2	Regresión Lineal	Gastos	2.484156e+02	2.791283e+02	0.010333
3	Random Forest	Gastos	2.573633e+02	3.027867e+02	-0.164541

```
#Y ahora comparamos con un gráfico de barras.
result = resultados.melt(id_vars=['Modelo', 'Variable'], value_vars=['MAE', 'RMSE', 'R2'],
                         var_name= 'Métrica', value_name='Valor')
plt.figure(figsize=(12,6))
sns.barplot(data= result, x='Métrica', y='Valor', hue='Modelo', dodge=True, palette='Set2')
plt.title('Comparación modelos')
plt.xlabel('Métrica')
plt.ylabel('Valor')
plt.legend(title='Modelo')
plt.show()
```

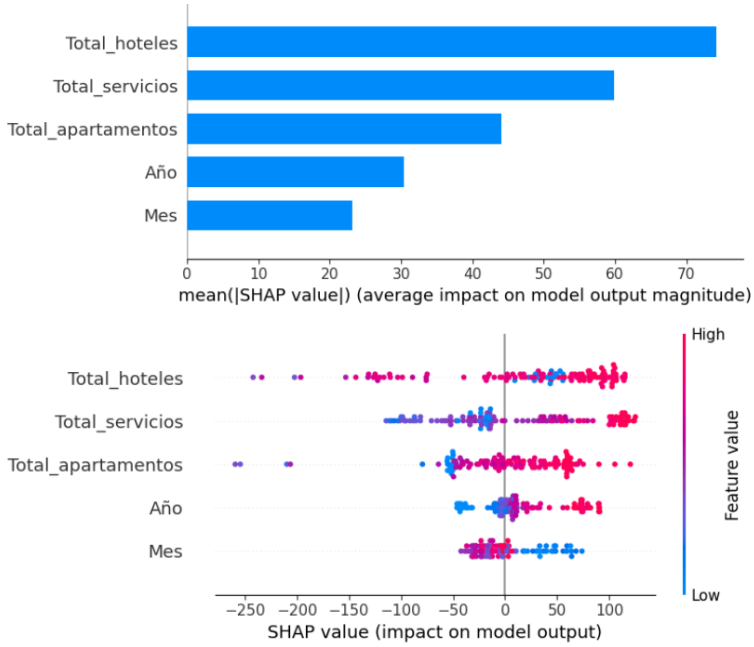


Por último, hacemos un estudio de la interpretabilidad de las variables, cuales influyen más en el modelo de Random Forest. Comenzamos con el modelo de turistas obteniendo los siguientes gráficos.



La variable más influyente en la predicción del número de turistas es Total_servicios, lo cual tiene sentido puesto que hemos ido viendo que hay una relación entre ambos. El año también parece tener un peso relevante, que ayuda a captar la tendencia creciente o decreciente a largo plazo. En cuanto a la variable Mes, no se capta tanto la estacionalidad mensual, lo cual tiene sentido, puesto que nuestro conjunto de datos tiene estacionalidad anual.

Ahora veamos las variables influyentes en el modelo de Random Forest para los gastos.



En este caso tanto el número de turistas alojados en hoteles es la variable más influyente en el modelo, y el índice de los servicios también es bastante importante. Las variables temporales tienen un peso más bajo, lo que nos confirma que depende de factores como la cantidad y el tipo de turistas más que de la época del año, los gastos tienen otros factores no estudiados en este trabajo.

CONCLUSIÓN.

A lo largo del trabajo hemos analizado las relaciones entre el turismo y los servicios y gastos, para ver si influye. También hemos realizado modelos de predicción.

En conclusión, el turismo y el índice de los servicios están claramente relacionados, cuando sube la cantidad de turistas, también lo hacen los servicios. Esto puede suponer un riesgo a la larga, debido a la saturación de los recursos y la presión que pueden tener las infraestructuras, de hecho, es una de las quejas de los residentes canarios, debido a que, por ejemplo, afecta de cara al transporte público, que se ven saturados y no pueden ser utilizados por la población autóctona.

En cuanto al turismo y los gastos, la relación es más débil. El aumento de los viajeros no implica directamente el aumento de los gastos, depende también de otros factores que no se estudian aquí.

Por último, otra queja de los canarios y que se quería estudiar era el aumento de los turistas, cada vez más y cada vez menos viviendas convencionales. Aunque no nos hemos centrado directamente en ello, si que se puede ver en las gráficas como la tendencia del turismo es en crecimiento, además de que los apartamentos para turistas, aunque tuvieron un decrecimiento desde 2007 aproximadamente hasta antes del COVID, se puede ver cómo está volviendo a aumentar los turistas en apartamentos de este estilo, lo que implica que también haya cada vez más, puesto que es un servicio y está directamente relacionado y esto a su vez hace que haya menos vivienda convencional.

Por lo que podemos determinar, que efectivamente, lo que reclaman los canarios, de saturación de los servicios y problemas con la cantidad de turistas y el crecimiento de la vivienda vacacional es cierto y la tendencia nos dice que va en aumento.

Es bueno destacar que este estudio tiene ciertas limitaciones, sobre todo de cara a predecir los gastos como hemos visto, hay otros factores que influyen, como, por ejemplo, el tipo de turista, y habría que realizar un estudio mucho más exhaustivo para poder predecir adecuadamente este punto.

BIBLIOGRAFÍA.

- Conjuntos de datos obtenidos en el INE.
- https://shap.readthedocs.io/en/latest/
- Apuntes del máster sobre machine learning y minería de datos.