DATA CLEANING & PREPROCESSING

PROJECT BMW

Autores: Aleix Blanch Orduña
Claudia Parals García
Arturo Ramos Rey
Nil Romans i Leon

- ## Introducción

Este trabajo consiste en la limpieza y preprocesado de un dataset de la marca de automóviles BMW. En los siguientes apartados se detallarán los distintos pasos seguidos, desde la limpieza de filas duplicadas y valores nulos, hasta el análisis de variables con el objetivo de entender qué información es relevante a la hora de predecir el precio de venta de estos automóviles.

- ## Lectura y comprensión de los datos

Importamos librerías y el dataset:

Importamos las librerías adecuadas para realizar el data cleaning. También realizamos la lectura del archivo *csv* y lo convertimos a *dataframe.*

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ] #importamos librería
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import LabelEncoder,MinMaxScaler

[ ] df_bmw = pd.read_csv("/content/drive/MyDrive/DATA/Data Engineering/Proyecto_1_Limpieza_Datos/CSV/bmw_pricing_v2.csv",
                         sep=",")
```

Estudiamos la base de datos:

1. Hacemos un ***.head()*** para ver las primeras filas y hacernos una idea de los distintos valores de cada variable.

```
[4] df_bmw.head()
```

	marca	modelo	km	potencia	fecha_registro	tipo_gasolina	color	tipo_coche	volante_regulable	aire_acondicionado	camara_trasera	asientos_traseros_plegables	elevalunas_ele
0	BMW	118	140411.0	100.0	2012-02-01	diesel	black	convertible	True	True	False	False	
1	BMW	M4	13929.0	317.0	2016-04-01	petrol	grey	convertible	True	True	False	False	
2	BMW	320	183297.0	120.0	2012-04-01	diesel	white	convertible	False	False	False	False	
3	BMW	420	128035.0	135.0	2014-07-01	diesel	red	convertible	True	True	False	False	
4	BMW	425	97097.0	160.0	2014-12-01	diesel	silver	convertible	True	True	False	False	

2. Hacemos un ***.info()*** para ver qué tipo de datos tenemos, cuántas columnas contiene el dataset y qué valores almacena. En este paso nos damos cuenta que las variables **tipo_coche, volante_regulable, aire_acondicionado, camara_trasera, asientos_traseros_plegables, elevalunas_electrico, bluetooth** y **alerta_lim_velocidad** son de tipo *String* pero sin embargo sus valores son True o False. En pasos posteriores cambiaremos el tipo de dato a *Boolean.* También nos fijamos en que existen columnas con la **fecha de registro** y la **fecha de venta** del automóvil, las cuales son de tipo *String* y que trabajaremos en ellas más adelante.

```
[8]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4843 entries, 0 to 4842
Data columns (total 18 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   marca                        4841 non-null   object
 1   modelo                       4840 non-null   object
 2   km                           4841 non-null   float64
 3   potencia                     4842 non-null   float64
 4   fecha_registro               4842 non-null   object
 5   tipo_gasolina                4838 non-null   object
 6   color                        4831 non-null   object
 7   tipo_coche                   4834 non-null   object
 8   volante_regulable            4839 non-null   object
 9   aire_acondicionado           4841 non-null   object
 10  camara_trasera               4841 non-null   object
 11  asientos_traseros_plegables  4839 non-null   object
 12  elevalunas_electrico         4841 non-null   object
 13  bluetooth                    4839 non-null   object
 14  gps                          4843 non-null   bool
 15  alerta_lim_velocidad         4841 non-null   object
 16  precio                       4837 non-null   float64
 17  fecha_venta                  4842 non-null   object
dtypes: bool(1), float64(3), object(14)
memory usage: 648.1+ KB
```

3. Hacemos un ***.shape*** para conocer la dimensión del dataset. Se puede observar que el dataset está compuesto por 18 columnas y más de 4800 filas.

4. Hacemos un ***.describe()*** para conocer las estadísticas descriptivas de las variables numéricas del dataset. Podemos darnos cuenta que en la variable **km** hay algún valor negativo. Además, en la variable **potencia** hay valores a 0. Por otro lado, hemos notado que el valor mínimo del **precio** se sitúa en 100, lo cual nos hace pensar que pueda deberse a un error.

En líneas generales, no se recomienda borrar o modificar los valores del target durante el preprocesamiento de un conjunto de datos, ya que el target o la variable objetivo es la variable que queremos predecir en nuestro modelo y es una parte fundamental de nuestro conjunto de datos. Cualquier cambio en los valores del target puede alterar significativamente los resultados de nuestro modelo y afectar la capacidad del modelo para generalizar y hacer predicciones precisas.

```
[9]  df.shape

(4843, 18)
```

```
df.describe()
```

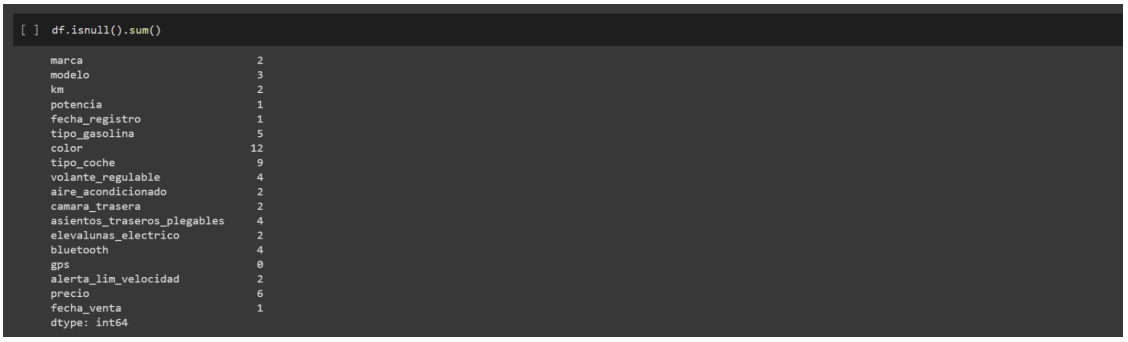
	km	potencia	precio
count	4.841000e+03	4842.000000	4837.000000
mean	1.409593e+05	128.981826	15831.920612
std	6.020853e+04	38.994839	9222.630708
min	-6.400000e+01	0.000000	100.000000
25%	1.028840e+05	100.000000	10800.000000
50%	1.410800e+05	120.000000	14200.000000
75%	1.752170e+05	135.000000	18600.000000
max	1.000376e+06	423.000000	178500.000000

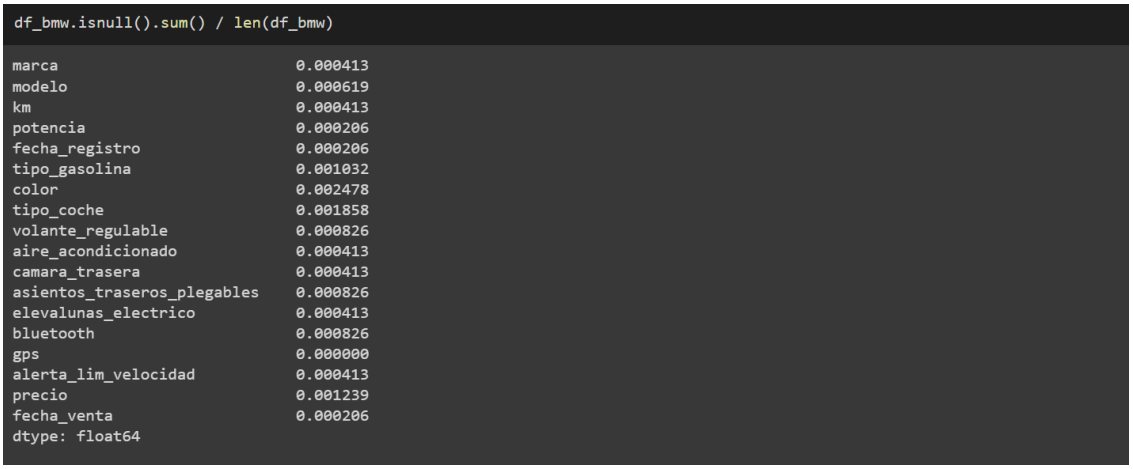
- Limpieza de datos y preparación

Verificar duplicados:
Una vez vistas las características del dataset, el primer paso a realizar es comprobar si el dataset contiene datos duplicados.

En este punto no nos preocupamos porque no hay ningún dato duplicado en el dataset.

Detección de nulos:
Vemos que el dataset no contiene valores duplicados, así que nos centraremos en ver si existen valores nulos. Los analizaremos todos y decidiremos qué hacer con ellos.





En la anterior imagen podemos ver el porcentaje de valores nulos en las columnas respecto al total de filas.

```
[ ] df1.dropna(inplace = True)
```

Decidimos eliminar todos los nulos del dataset ya que es un porcentaje muy pequeño respecto al total. No obstante, otra opción hubiera sido sustituir los nulos de las variables numéricas por las medias o las medianas y en las variables categóricas por ‘otros’ o ‘desconocidos’.

```
[12] df_bmw['marca'].unique()

     array(['BMW', nan], dtype=object)
```

Ya tenemos el dataset limpio de valores nulos, por lo que ya podemos empezar a trabajar con él.

Diferencia de fechas:
Para crear información valiosa a partir de las fechas de registro y venta, creamos una nueva variable con la diferencia entre estas dos fechas. Para ello debemos cambiar el tipo de las columnas de *String* a *datetime*.

```
df1['fecha_venta']

0       2018-01-01
1       2018-02-01
2       2018-02-01
3       2018-02-01
4       2018-04-01
           ...
4837    2018-07-01
4838    2018-08-01
4839    2018-08-01
4840    2018-09-01
4841    2018-09-01
Name: fecha_venta, Length: 4781, dtype: object

[ ] df1['fecha_venta'] = pd.to_datetime(df1['fecha_venta'])

[ ] df1['fecha_registro'] = pd.to_datetime(df1['fecha_registro'])

[ ] df1['tiempo_venta'] = (df1['fecha_venta'] - df1['fecha_registro']) / np.timedelta64(1,'D')

[ ] df1['tiempo_venta'].dtype

    dtype('float64')

[ ] del df1['fecha_venta']

[ ] del df1['fecha_registro']
```

Análisis univariable:

Ahora examinaremos cada variable para ver si se debe limpiar, por ejemplo para unir datos de una variable en una sola. Para ello, separaremos las variables en target, categóricas, numéricas y booleanos.

Previamente, convertimos el tipo de las variables que hemos visto anteriormente con valores True y False de *String* a *Boolean*.

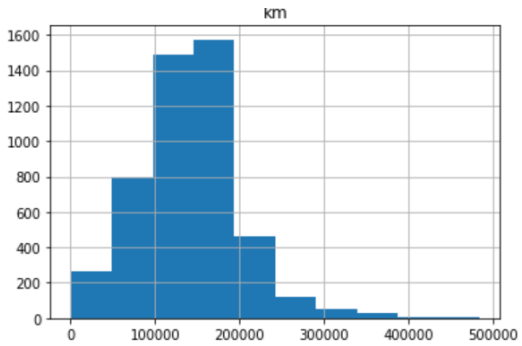
```python
target = ["precio"]
def obtener_lista_variables(dataset):
  lista_numerica=[]
  lista_categorica=[]
  lista_bool=[]

  for i in dataset:
    if i not in target:
      if (dataset[i].dtype.kind == 'f') | (dataset[i].dtype.kind == 'i'):
        lista_numerica.append(i)
      elif dataset[i].dtype.kind == 'O':
        lista_categorica.append(i)
      elif dataset[i].dtype.kind == 'b':
        lista_bool.append(i)
  return lista_numerica, lista_categorica, lista_bool
```

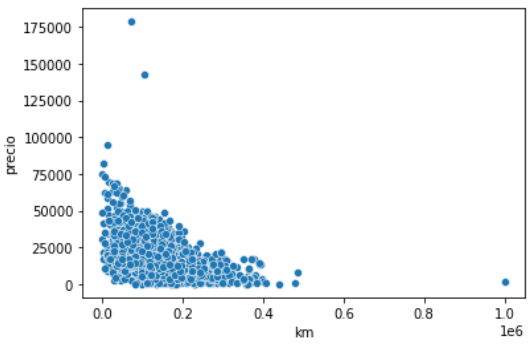
```
[24] l_num, l_cat, l_bool = obtener_lista_variables(df_bmw_sn)

[25] l_num

     ['km', 'potencia', 'tiempo_venta']

[26] l_cat

     ['marca', 'modelo', 'tipo_gasolina', 'color', 'tipo_coche']

[27] l_bool

     ['volante_regulable',
      'aire_acondicionado',
      'camara_trasera',
      'asientos_traseros_plegables',
      'elevalunas_electrico',
      'bluetooth',
      'gps',
      'alerta_lim_velocidad']
```

Una vez catalogadas las variables correctamente, pasaremos a ver si se pueden limpiar por categorías.

Empezamos con las **variables numéricas**:

1. **KM**





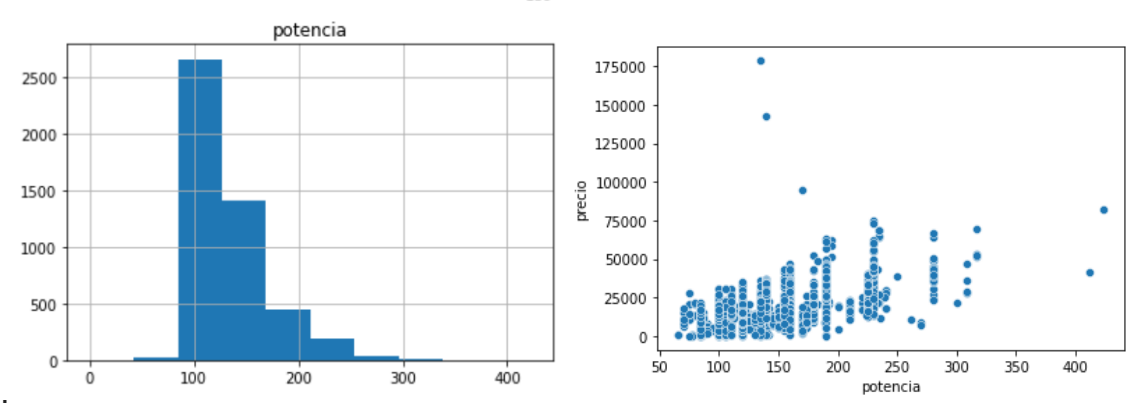
```
[29] df_bmw_sn[(df_bmw_sn['km'] < 0) | (df_bmw_sn['km'] > 500000)]
```

	marca	modelo	km	potencia	tipo_gasolina	color	tipo_coche	volante_regulable	aire_acondicionado	camara_trasera	asientos_traseros_pl
2938	BMW	640 Gran Coupé	-64.0	230.0	diesel	black	sedan	True	True	False	
3732	BMW	118	1000376.0	90.0	diesel	black	subcompact	True	False	False	

```
[30] df_bmw_sn['km'] = np.where((df_bmw_sn['km'] < 0) | (df_bmw_sn['km'] > 500000), df_bmw_sn['km'].mean(), df_bmw_sn['km'])
```

Vemos que en la categoría **km** hay dos valores clasificados como outliers: un negativo (dato imposible dado que el kilometraje debe partir desde cero), y un valor que supera el doble del penúltimo valor al alza, es decir, un outlier máximo. Estos valores los reemplazamos por la media de la columna **km**. Además, vemos que esta variable tiene una correlación negativa con el precio.

2. Potencia



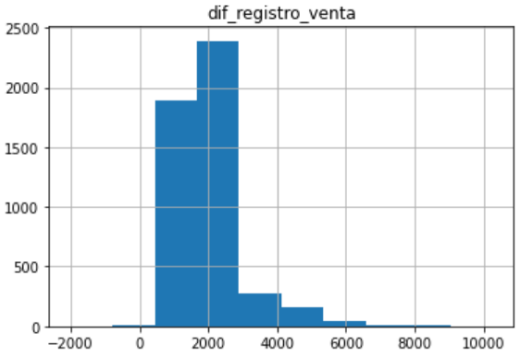
Nos fijamos que, en la variable **potencia**, existen valores mínimos que hacen dudar de su credibilidad. Para eso haremos una búsqueda de datos inferior a 60, que sería la potencia mínima que creemos oportuna para un vehículo de la categoría de una marca como BMW. En este caso vemos tres valores, dos registrados en 25 y uno en 0, valores que eliminaremos para ayudar en la predicción del modelo. Además, vemos que esta variable sí que está correlacionada con el precio.

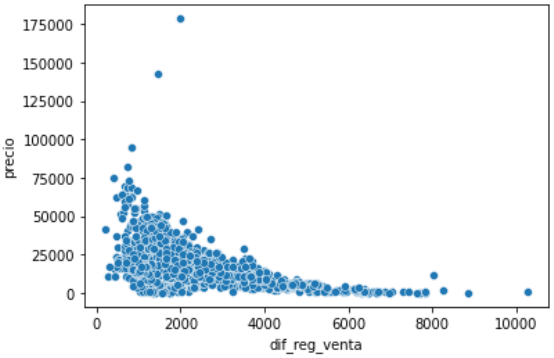
```
df_bmw_sn[df_bmw_sn['potencia'] < 60]
```

	marca	modelo	km	potencia	tipo_gasolina	color	tipo_coche	volante_regulable	aire_acondicionado	camara_trasera	asientos_tra
1796	BMW	i3	152328.0	25.0	hybrid_petrol	black	hatchback	False	True	False	
1925	BMW	i3	152470.0	25.0	hybrid_petrol	black	hatchback	False	True	False	
3765	BMW	X1	81770.0	0.0	diesel	white	suv	False	False	False	

```
[32] df_bmw_sn.drop(df_bmw_sn[df_bmw_sn['potencia'] < 60].index, inplace=True)
```

3. Diferencia fecha registro y venta



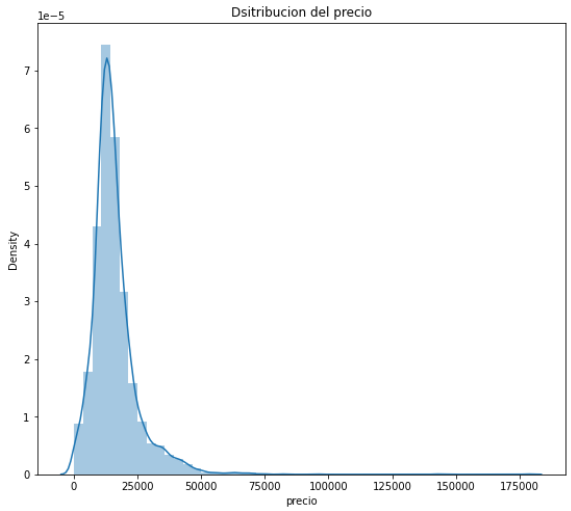


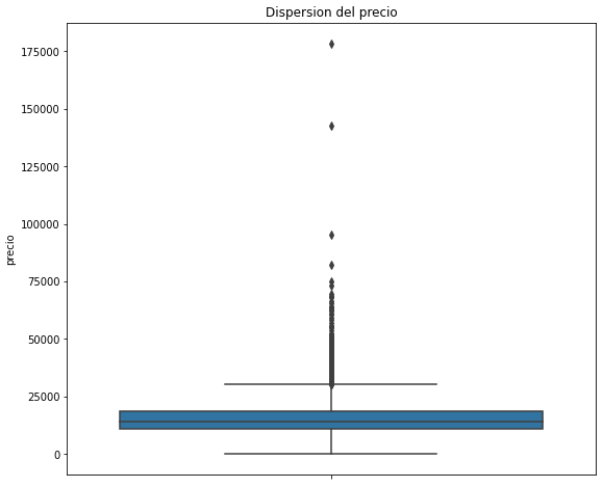

```
df_bmw_sf[df_bmw_sf['dif_reg_venta'] < 0]
```

color	tipo_coche	volante_regulable	aire_acondicionado	camara_trasera	asientos_traseros_plegables	elevalunas_electrico	bluetooth	gps	alerta_lim_velocidad	precio	dif_reg_venta
blue	convertible	True	True	False	False	True	False	True	True	15800.0	-1614
black	coupe	False	True	False	False	True	False	True	False	8200.0	-609
black	estate	False	True	False	False	False	False	True	True	11500.0	-2009
black	estate	True	False	False	False	False	False	True	False	8800.0	-2009

Como podemos ver existen hasta cuatro valores negativos entre la fecha de venta y la fecha de registro del vehículo, incluidos en la nueva variable creada anteriormente. Como no puede resultar más antigua la fecha de venta que la fecha de registro procedemos a eliminar estas filas debido a un error que existe en el dataset. Además, vemos que esta variable tiene una correlación negativa con el precio.

4. Precio





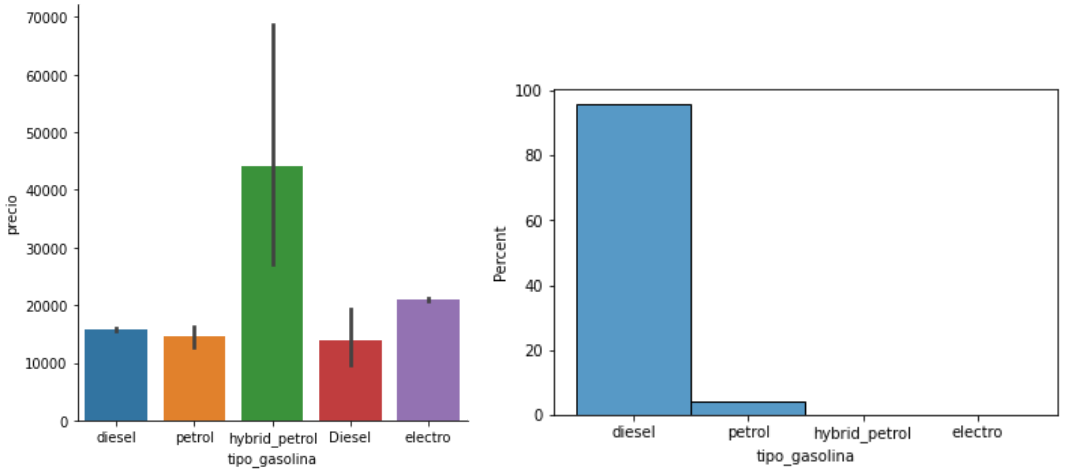
En estos gráficos podemos observar la distribución del precio y su dispersión. El gráfico parece estar sesgado a la derecha, lo que significa que la mayoría de los precios del conjunto de datos son relativamente bajos (alrededor de 15.000). Además, los puntos de datos están muy separados de la media, lo que indica una gran varianza en los precios de los coches.

Variables categóricas:

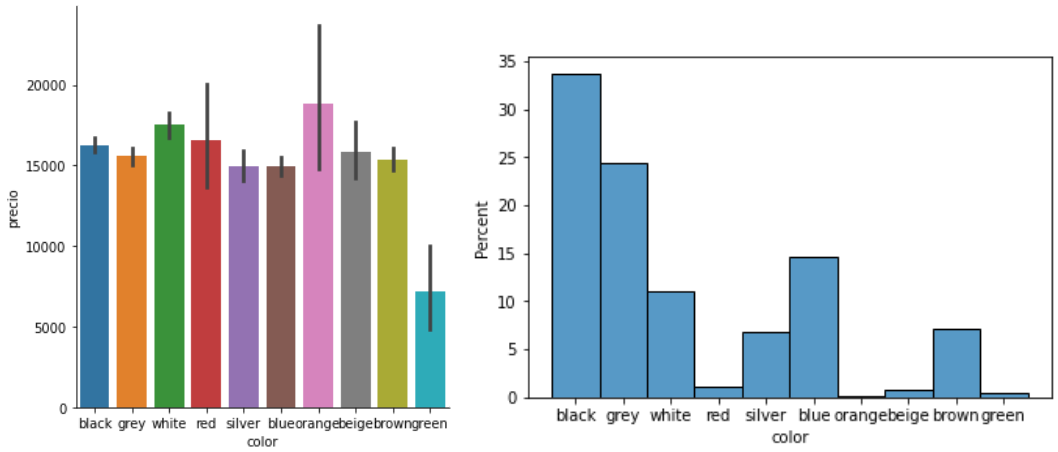
Podemos ver como en la columna **tipo_gasolina** hay dos valores idénticos pero escritos de diferente modo ('Diesel' y 'diesel'), que englobamos en un solo valor para que no exista discrepancia entre estos valores que representan el mismo valor.

```
[167] df_bmw_sn['tipo_gasolina'] = np.where(df_bmw_sn['tipo_gasolina'] == 'Diesel', 'diesel', df_bmw_sn['tipo_gasolina'])
```

Aún así, después de valorarlo de forma gráfica, vemos que el tipo de gasolina diesel hace referencia al 95,85% del total. Por ese motivo, hemos decidido eliminar la columna, ya que no es un factor detonante en la predicción del precio.



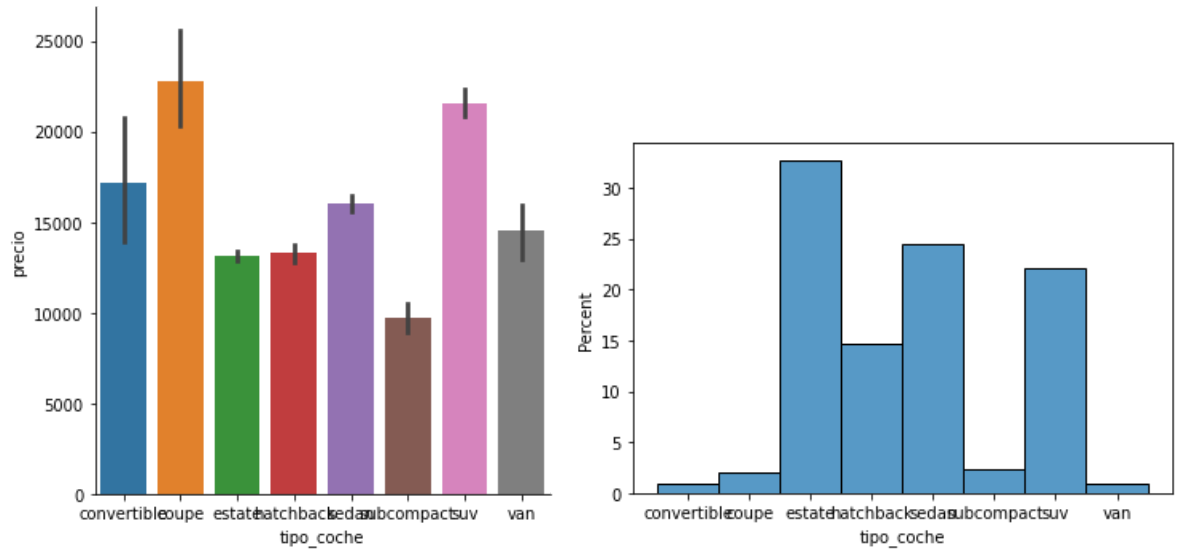
Para la variable de **color** hemos decidido realizar los gráficos antes de tomar decisiones.



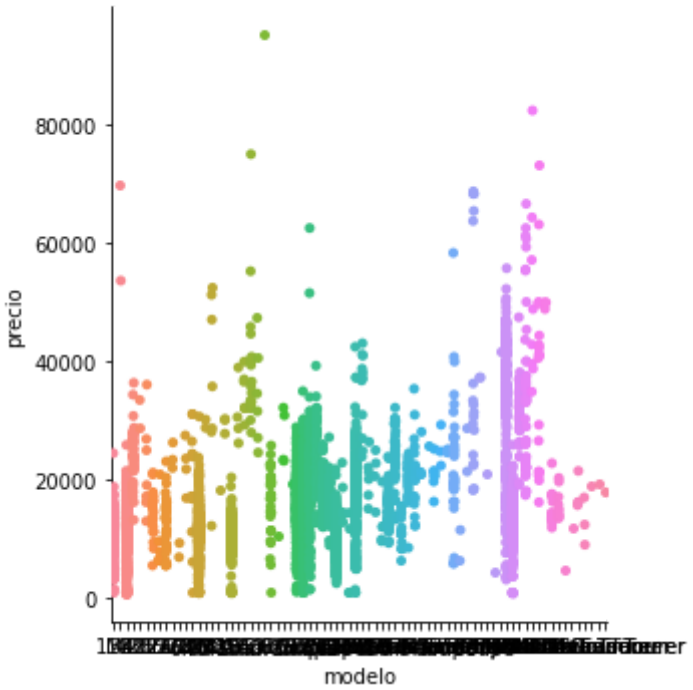
Vemos que no hay mucha diferencia entre colores, así que decidimos poner un umbral hasta 300 para juntar aquellos valores con un porcentaje muy bajo sobre el total.

```
[251] def quitar_por_umbrales(dataset, columna, umbral):
        df3 = pd.DataFrame(dataset[columna].value_counts())
        df3 = df3.reset_index()
        mi_lista = []
        for i in range(len(df3)):
          if df3.iloc[i][columna] <= umbral:
            mi_lista.append(df3.iloc[i]['index'])
        for i in mi_lista:
          dataset[columna] = np.where(dataset[columna] == i, 'otro', dataset[columna])

[252] quitar_por_umbrales(df3, 'color', 300)
```

La variable **tipo_coche** no la tocaremos ya que es un factor relevante a la hora de predecir el precio. Además, el gráfico nos muestra una dispersión entre los diferentes modelos, lo que será bueno para el modelo predictivo.

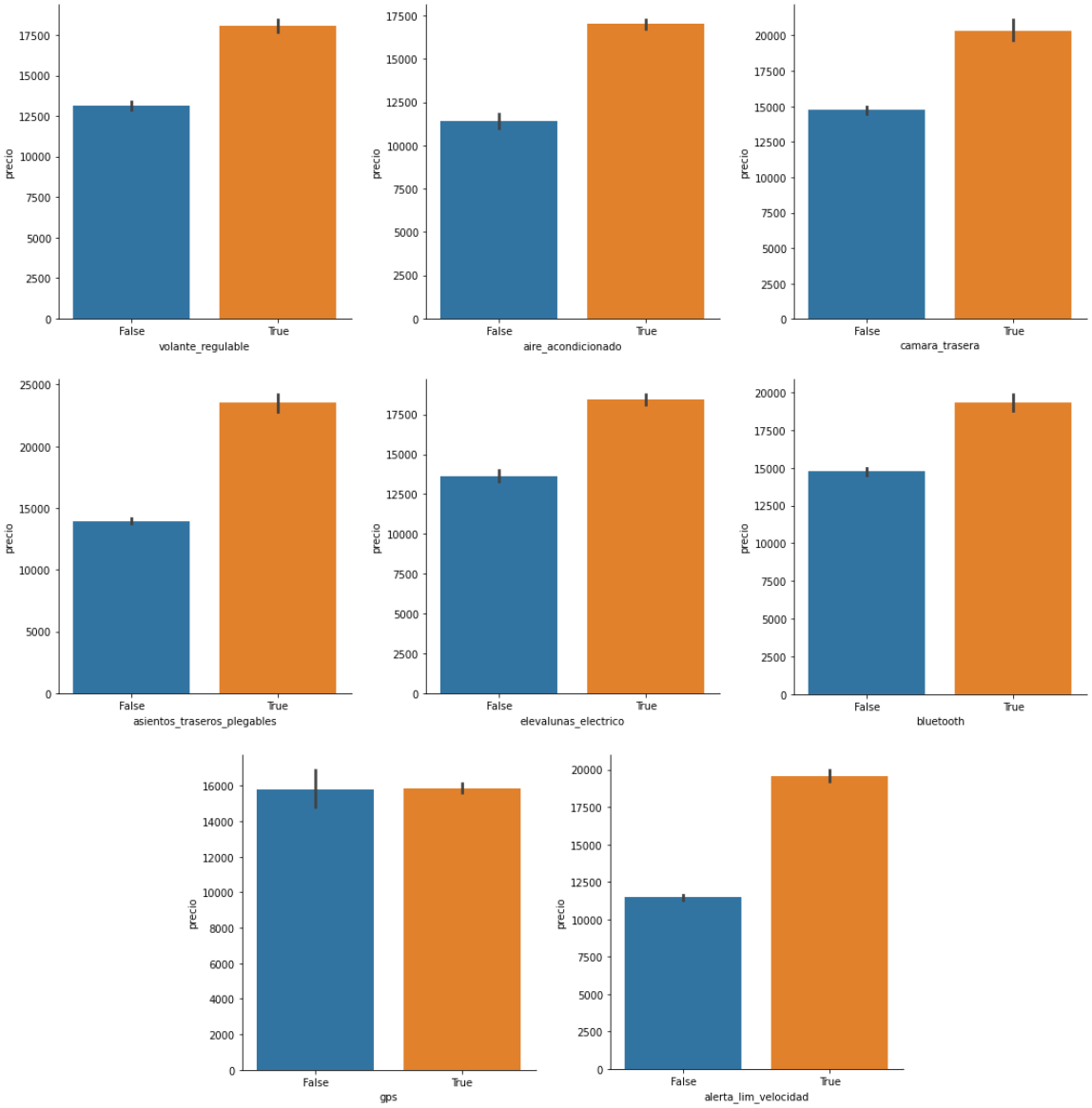


Finalmente, llegamos a la conclusión que las columnas **marca** y **modelo** no aportan información útil para el modelo predictivo. En la primera se puede ver como todos los datos hacen referencia a vehículos de la marca BMW por lo que resulta poco útil mantener dicha columna. En el caso de la variable **modelo** decidimos eliminarla también del modelo predictivo debido a que no tiene una correlación con el precio.

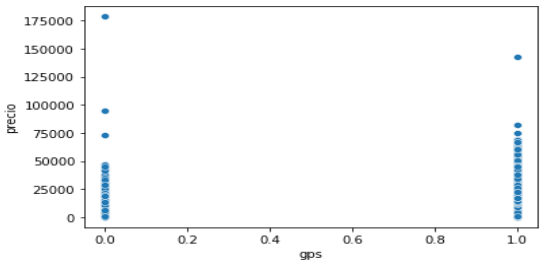


Variables booleanas:

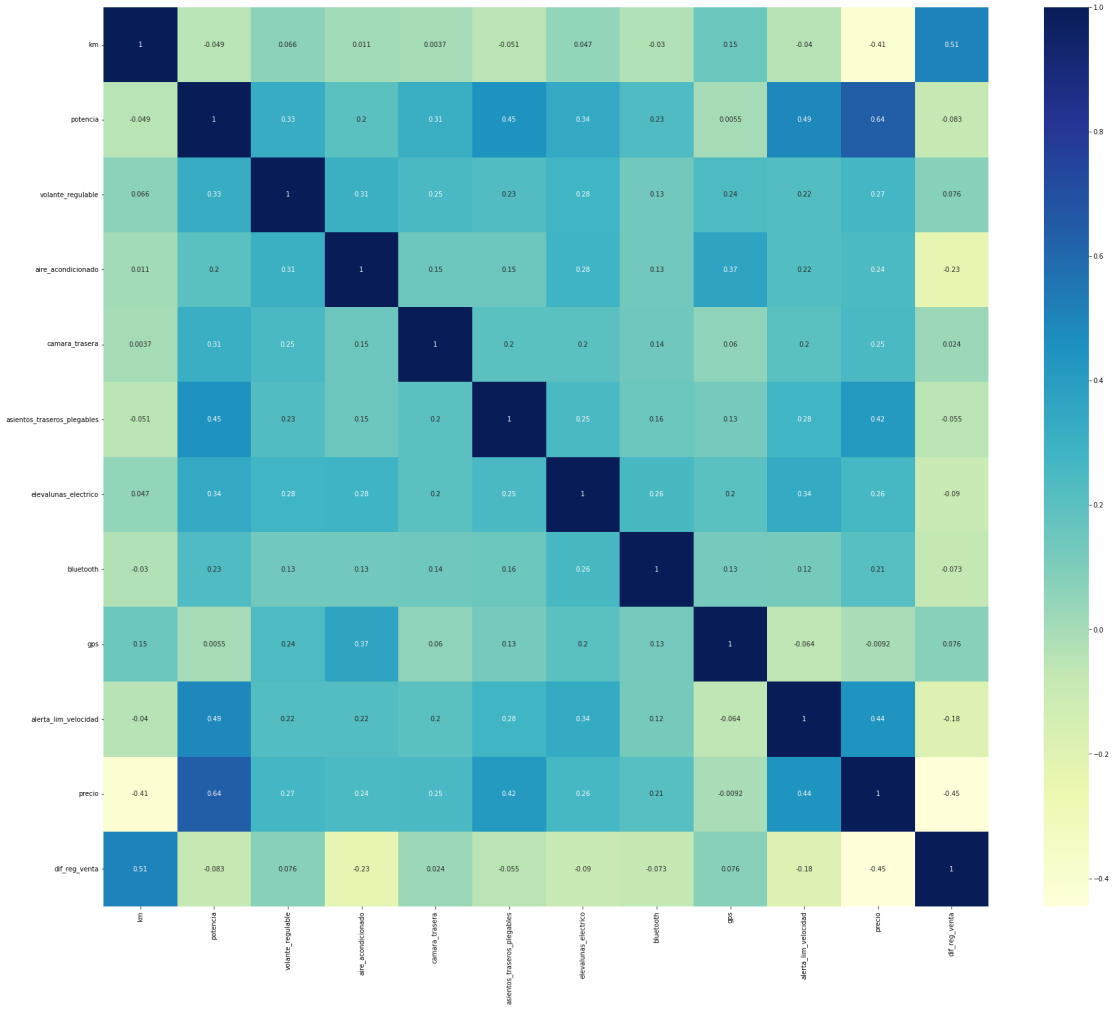
A nivel de variables booleanas, hemos decidido mantenerlas ya que, como se puede observar en las siguientes gráficas, los distintos extras del coche aumentan el precio de venta.



En la variable **gps** puede parecer que tienen el mismo precio medio los coches con gps que los que no lo tiene. Sin embargo, si observamos el *scatterplot*, observamos que existe un outlier que aumenta el precio medio para los coches que no tienen gps, pero en general los coches con gps suelen tener precios más elevados.



Correlación: detectar si hay variables que explican lo mismo



Como se observa en la anterior matriz, no existe ninguna correlación alta entre variables. Por ello, mantenemos todas las variables que tenemos hasta el momento y procederemos a reescalar y codificar sus valores.

- **Reescalado**

Una vez demostrado que no existe ninguna correlación demasiado alta, procedemos al reescalado de variables. Para ello, aplicamos un ***One Hot Encoding*** sobre las variables categóricas y booleanas, y un ***Min Max Scaler*** sobre las variables numéricas.

```
[95] df_bmw_clean = pd.get_dummies(data=df_bmw_clean, columns=l_cat)

[96] df_bmw_clean = pd.get_dummies(data=df_bmw_clean, columns=l_bool)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

for i in l_num:
  df_bmw_scaled[i] = scaler.fit_transform(df_bmw_scaled[i].values.reshape(-1, 1))
```

Tras este paso, obtenemos una correlación negativa sobre las variables relacionadas con los 'extras' de los coches (variables booleanas), por lo que eliminamos las variables False de cada una de ellas.

```
[99] df_bmw_clean.drop(['volante_regulable_False', 'aire_acondicionado_False',
                        'camara_trasera_False', 'asientos_traseros_plegables_False',
                        'elevalunas_electrico_False', 'bluetooth_False', 'gps_False',
                        'alerta_lim_velocidad_False'], axis=1, inplace=True)
```

- **Variables relevantes**

Tras haber realizado todo el análisis, y haber decidido qué variables nos resultan relevantes a la hora de entrenar el modelo, imprimimos por pantalla las columnas resultantes de todo este proceso. El siguiente paso sería separar las columnas por *features* y *target* y entrenar nuestro modelo.

```
[97] df_bmw_final.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4774 entries, 0 to 4773
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   km                              4774 non-null   float64
 1   potencia                        4774 non-null   float64
 2   precio                          4774 non-null   float64
 3   tiempo_venta                    4774 non-null   float64
 4   color_black                     4774 non-null   uint8
 5   color_blue                      4774 non-null   uint8
 6   color_brown                     4774 non-null   uint8
 7   color_grey                      4774 non-null   uint8
 8   color_otro                      4774 non-null   uint8
 9   color_silver                    4774 non-null   uint8
 10  color_white                     4774 non-null   uint8
 11  tipo_coche_convertible          4774 non-null   uint8
 12  tipo_coche_coupe                4774 non-null   uint8
 13  tipo_coche_estate               4774 non-null   uint8
 14  tipo_coche_hatchback            4774 non-null   uint8
 15  tipo_coche_sedan                4774 non-null   uint8
 16  tipo_coche_subcompact           4774 non-null   uint8
 17  tipo_coche_suv                  4774 non-null   uint8
 18  tipo_coche_van                  4774 non-null   uint8
 19  volante_regulable_True          4774 non-null   uint8
 20  aire_acondicionado_True         4774 non-null   uint8
 21  camara_trasera_True             4774 non-null   uint8
 22  asientos_traseros_plegables_True 4774 non-null  uint8
 23  elevalunas_electrico_True       4774 non-null   uint8
 24  bluetooth_True                  4774 non-null   uint8
 25  gps_True                        4774 non-null   uint8
 26  alerta_lim_velocidad_True       4774 non-null   uint8
dtypes: float64(4), uint8(23)
memory usage: 256.5 KB
```

```
[98] df_bmw_final.head()
```

	km	potencia	precio	tiempo_venta	color_black	color_blue	color_brown	color_grey	color_otro	color_silver	...	tipo_coche_suv	tipo_coche_van	volante_regulable_True
0	0.289039	0.095238	11300.0	0.193767	1	0	0	0	0	0	...	0	0	1
1	0.027787	0.703081	69700.0	0.045405	0	0	0	1	0	0	...	0	0	1
2	0.377621	0.151261	10200.0	0.190879	0	0	0	0	0	0	...	0	0	0
3	0.263476	0.193277	25100.0	0.109131	0	0	0	0	1	0	...	0	0	1
4	0.199573	0.263305	33400.0	0.099771	0	0	0	0	0	1	...	0	0	1

5 rows × 27 columns