

EDA (EXPLORATORY DATA ANALYSIS) AND DATA CLEANING

Grupo 6

Álvaro González González
Juan Antonio Maldonado Hervás
Sofia Volpe
Sol Etcheverry

30 de diciembre de 2022

Introducción

La limpieza y el preprocesamiento de los datos son unos pasos esenciales en el data science, además de ser los pasos que más tiempo ocupan en el proceso de creación de un modelo. Los datos sin procesar son muchas veces desordenados e incompletos y necesitan ser limpiados y transformados antes de ser utilizados para un análisis o un modelo. Esto incluye tareas como el manejo de los valores nulos, eliminar duplicados, corregir errores y escalar algunas de sus características.

En este proyecto nos enfocaremos en la limpieza y el preprocesamiento de un dataset para que se pueda luego utilizar para entrenar un modelo. Empezamos importando las librerías necesarias y cargando los datos. Luego, realizaremos algunas exploraciones iniciales, para tener una idea de la estructura del dataset utilizado. Después identificamos y abordamos todos los datos que tendrán que ser limpiados o transformados. Esta fase consiste en la aplicación de varias técnicas y herramientas como la imputación, la codificación y el escalamiento.

Al final de este proyecto, tendremos un dataset limpio y preprocesado, listo para ser utilizado dentro de un modelo. Gracias a este proceso, adquirimos una comprensión más profunda de la importancia de la limpieza y del preprocesamiento de los datos, y de cómo aplicar estas técnicas en un contexto real.

EDA (Exploratory Data Analysis) and Data Cleaning

El objetivo del siguiente trabajo es enumerar, describir y explicar el por qué de cada decisión tomada en el pre-procesamiento de datos con el objeto final de estimar el precio de los vehículos.


Pasos a seguir:

1. Importación de librerías

- *pandas* para manipular el DataFrame
- *numpy* para cálculo numérico y análisis de datos
- *pickle*
- *matplotlib.pyplot* para visualizaciones
- *seaborn* también para visualizaciones
- *sklearn.preprocessing* por sus algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad.

2. Carga de datos en un DataFrame

3. Exploración del conjunto de datos



- .shape()
- .info()
- .describe()
- .head() para visualizar la tabla

Con estas funciones podemos tener un pantallazo del dataset: cantidad de variables, tipos y, de las variables numéricas, podemos ver datos descriptivos que inicialmente nos puede mostrar qué variables están fuera de rango.

Tipo	Variable	Observaciones
bool	gps	
float64	km	
	potencia	
	precio	target
object	aire_acondicionado	True/False
	alerta_lim_velocidad	True/False
	asientos_traseros_plegables	True/False
	bluetooth	True/False
	camara_trasera	True/False
	color	ver correlación con P
	elevallunas_electrico	True/False
	fecha_registro	modificarlas
	fecha_venta	modificarlas
		borrar por ser mismo valor en todas las obs
	marca	
	modelo	alta cardinalidad - reducir
	tipo_coche	ver correlación con P
	tipo_gasolina	ver correlación con P
	volante_regulable	True/False

Vemos que no hay líneas duplicadas (.duplicated()) aunque si datos nulos en las variables.

Total Observaciones 4843

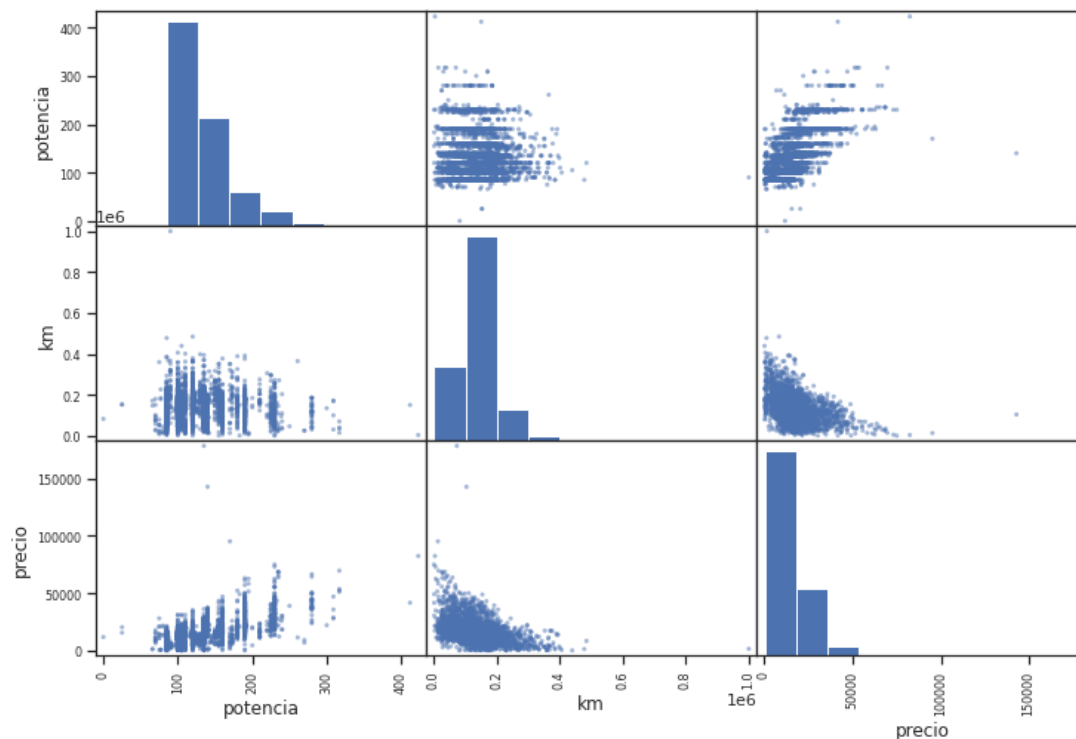
Variable	NA
marca	2
modelo	3
km	2
potencia	1
fecha_registro	1
tipo_gasolina	5
color	12
tipo_coche	9
volante_regulable	4
aire_acondicionado	2
camara_trasera	2
asientos_traseros_plegables	4
elevallunas_electrico	2
bluetooth	4
gps	0
alerta_lim_velocidad	2
precio	6
fecha_venta	1
Total NA	62

1,28%

Observación: si bien podríamos eliminarlos por la baja proporción sobre el total de observaciones, en la mayoría de los casos vamos a trabajarlas.

Previo a comenzar a limpiar el dataset, vemos en crudo, las potenciales correlaciones de las actuales variables numéricas.

Con **scatter_matrix** vemos inicialmente correlaciones. Esto nos sirve para comenzar a ver atributos que puedan predecir el precio de los vehículos.



```
precio    1.000000
potencia   0.639254
gps        -0.005227
km         -0.410189
```

A simple vista podemos ver que potencia y km podrían ser buenos atributos para predecir el target.

4. Limpieza del Dataset

- Borramos la variable Marca, porque solo tiene un valor.
- Al tener fechas (registro y venta), separamos en día, mes, año, para ver el impacto. Día se elimina. De esta manera el formato object de fecha pasa a float para mes y float para año, y finalmente se eliminan las fechas originales.
- Separamos las variables según su categoría, para limpiar los Na a nivel grupo.
- Contamos con 3 listas iniciales de variables y sobre ellas aplicamos transformaciones para predecir los NA.

Variables numéricas

Vamos a aplicar **IterativeImputer**.

Entrena un modelo de regresión por cada variable numérica para predecir los NA, en función de las otras variables. Luego entrena el modelo en la data actualizada y repite el proceso varias veces, mejorando el modelo y los valores de reemplazos en cada iteración. Por default el n de iteraciones =10, de modo que por cada NA hay como máximo 10 (max-iter) predicciones y la iteración termina cuando la diferencia de la predicción de la iteración anterior y la actual es < a un valor dado que por default es 1e-1.

$$\text{Criterio de stop} = \frac{\max(\text{abs}(X_t - X_{t-1}))}{\max(\text{abs}(X_{\text{valores conocidos}}))} = \text{tol} = 1e-1$$

Variables categóricas

Vamos a aplicar **SimpleImputer**. El beneficio es que almacena, por ejemplo, la media (depende lo que se ponga en el parámetro 'strategy') de cada variable. Esto permite que se imputen missing values no solo al training set (esto no lo vimos, pero deberíamos separar inicialmente un training set de un test set) sino también al test set.

Para este caso, aplicamos la moda para predecir los valores nulos de las variables.

```
imputer = SimpleImputer(strategy="most_frequent")
```

Target

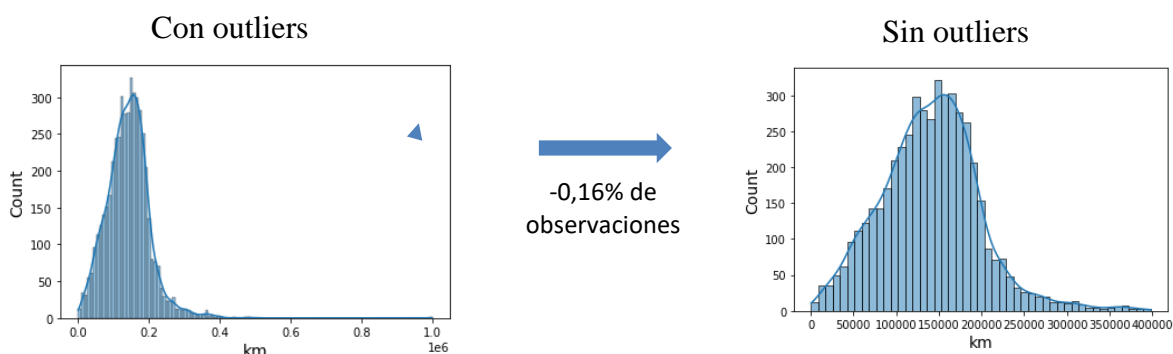
Borramos los null para mayor simpleza.

Hasta este punto tenemos un data set sin duplicados y sin nulos.

E. Vamos a analizar los outliers para que no sesguen el modelo.

OUTLIERS NUMERICOS

KM: tenemos km<0 que decidimos borrarlos, y km>400.000 que también los hemos borrado. De esta forma tenemos una distribución cercana a una distribución normal.



POTENCIA: tenemos potencia < 60 (que es lo mínimo para un coche urbano) que decidimos borrarlo, y > 300 que también los hemos borrado.
La distribución no cambia demasiado.

FECHAS: borramos datos donde la fecha de venta es anterior a la fecha de registro. Eliminamos la variable mes (tanto de registro, como de venta) y también borramos la variable fecha venta dado que el dataset abarca solo ventas del año 2017.

Adicionalmente, tenemos años de venta que están en string, con lo cual no se unen con el integer del mismo año (2011). Entonces pasamos toda la columna a float y después a int.

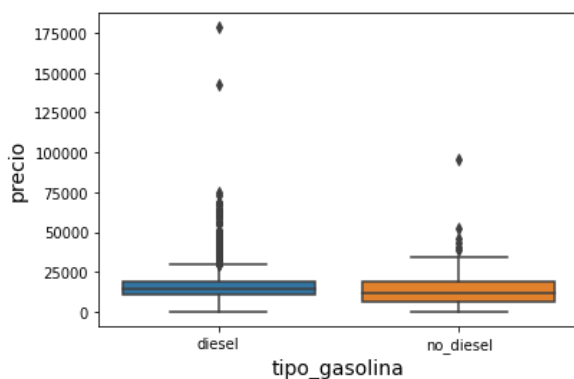
Para FECHA_REGISTRO_AÑO (inicialmente separamos cada fecha en día/mes/año, y nos quedamos con año como una potencial variable que explique el precio de los coches) borramos aquellos registros con frecuencia < 4.

VARIABLES CATEGORICAS

MODELO: Al tener tantos modelos, cuando se haga el OneHot van a haber muchas columnas por la alta cardinalidad de la variable. Por lo tanto, aplicamos **BinaryEncoder**. Esto creará 6 grupos de modelos con números binarios. Elimaremos en la copia siguiente la variable 'modelo', para dejar sólo las dummies de cada grupo.

modelo	modelo_ 0	modelo_ 1	modelo_ 2	modelo_ 3	modelo_ 4	modelo_ 5	modelo_ 6
118	0	0	0	0	0	0	0
320	0	0	0	0	0	0	1
420	0	0	0	0	0	0	1
425	0	0	0	0	0	1	0
335	0	0	0	0	0	1	0

TIPO_GASOLINA: El 96% de las observaciones corresponde a diesel, con lo cual se elimina la variable. El precio casi no varía por tipo de gasolina.



```
df_pricing_08['diesel'].value_counts()

True      4436
False     184
Name: diesel, dtype: int64
```

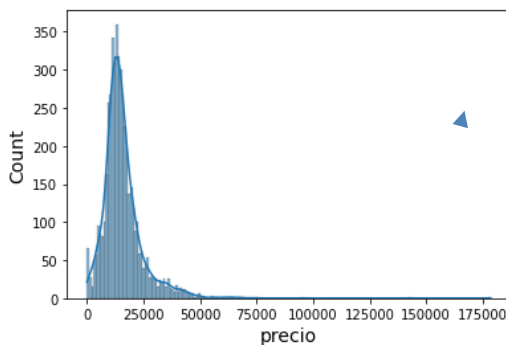
COLOR Y TIPO COCHE: para color, reducimos un poco la cardinalidad porque había colores con pocas observaciones, entonces las agrupamos en categoría "otro_color". Tanto Color, como tipo coche pasaran por one hot para pasar a columnas binarias.

ANÁLISIS DE TARGET

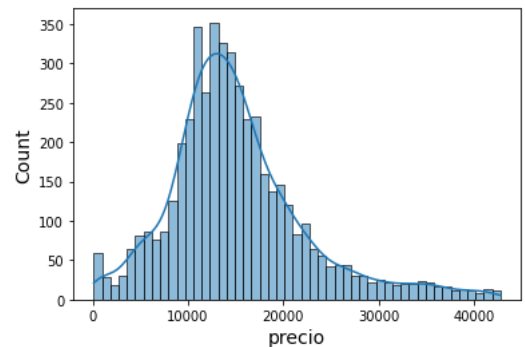
Dado que hay outliers, vamos a acotar los valores con el z – score que es el número de desviaciones estándar de la media de un punto. El punto de corte puede ser desde -3 desviaciones estándar hasta 3 desviaciones estándar.

$$z_{score} = \left(\frac{i - media}{std} \right)$$

Aplicando el z score, cambia ampliamente la distribución del target



-1,60% de
observaciones

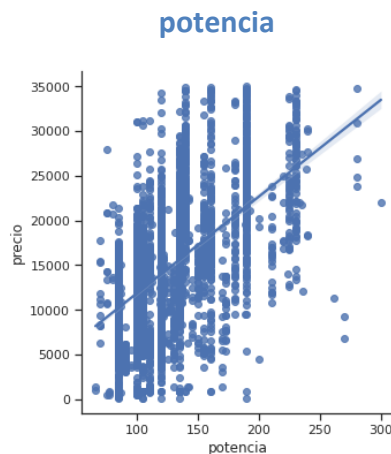
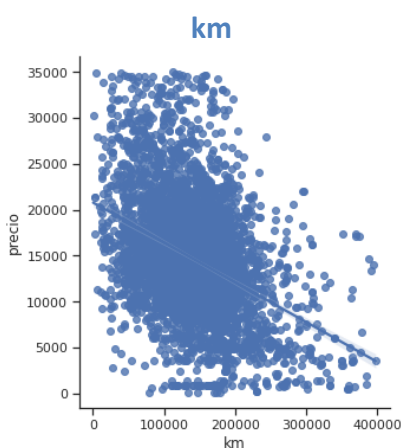


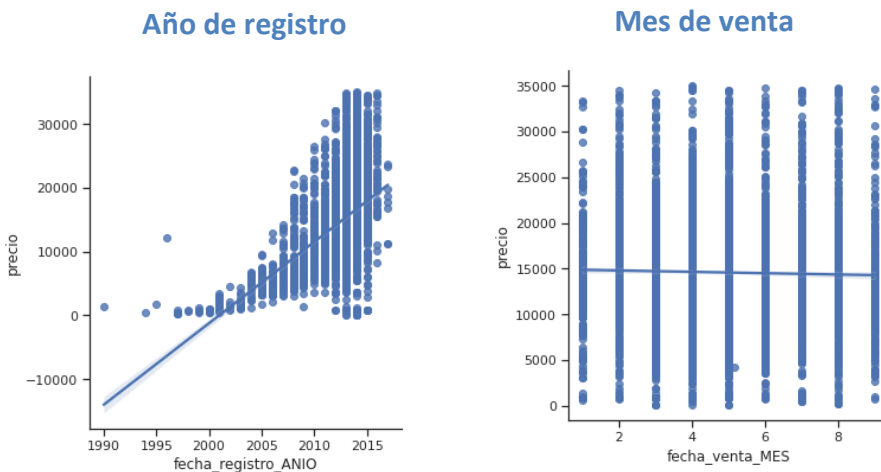
Adicionalmente, eliminamos el 2,5% mas bajo del target para acotar las puntas con menos observaciones. El z-score no lo hace porque para esa franja el coeficiente no es <-3.

El siguiente paso es **verificar si la variable del target tiene una distribución normal**. Si no la tuvieses deberíamos de aplicar una transformación a los datos, como puede ser, por ejemplo, la logarítmica en el caso de que la distribución tenga un sesgo a la derecha.

Como hemos visto en el paso anterior, esto no será necesario. Con la normalización de los datos, a través del uso del valor de Z-score y eliminando el primero 2.5%, la distribución de los datos se aproxima a una distribución normal.

Analizamos relaciones entre target y variables numéricas más relevantes (los grupos de la variable 'modelo' ahora son dummies, pero para resumir los gráficos, no los presentaremos).

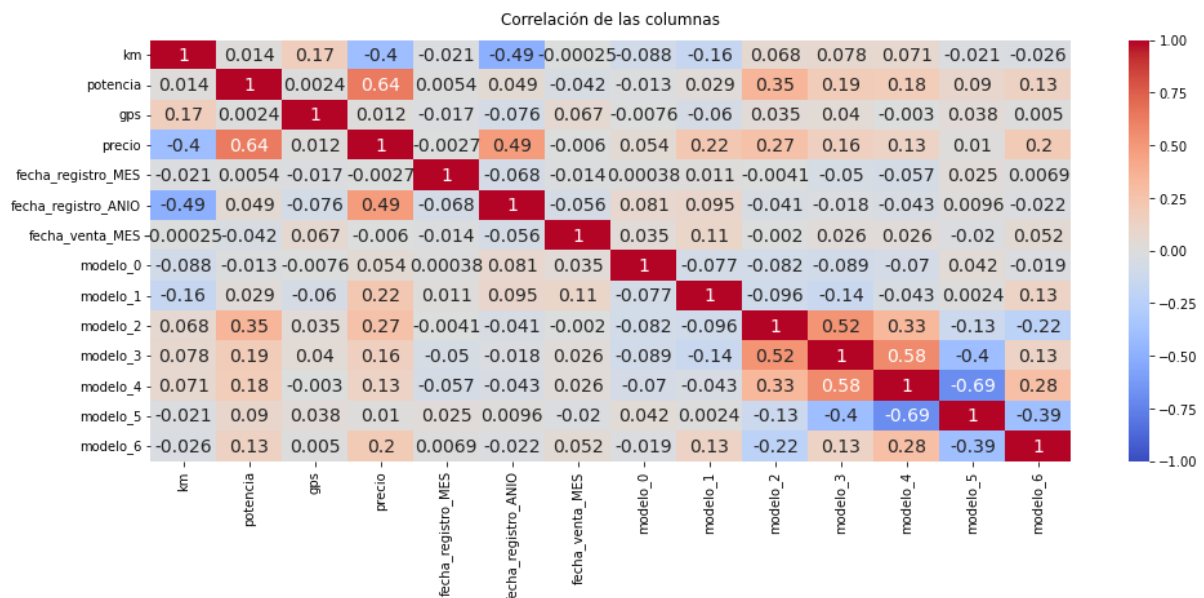




Podemos ver que el Precio tiene una tendencia con una pendiente casi 0 contra las variables 'Mes de venta' y 'Mes de registro', con lo cual se procede a eliminar dichas variables del modelo.

Correlación entre las columnas: Como primera paso observamos la existencia de alguna correlación entre las variables.

Para obtener una mejor visualización de los datos los insertamos dentro de una tabla con un gradiente de color que nos permita detectar las variables fuertemente relacionadas.



Normalización de las variables numéricas (km , potencia, año de registro del coche): para reescalar de modo que varíen entre 0 y 1.

La fórmula por detrás es la siguiente:

$$X_{nuevo} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Para este proceso utilizamos la función de MinMaxScaler de Sklearn.

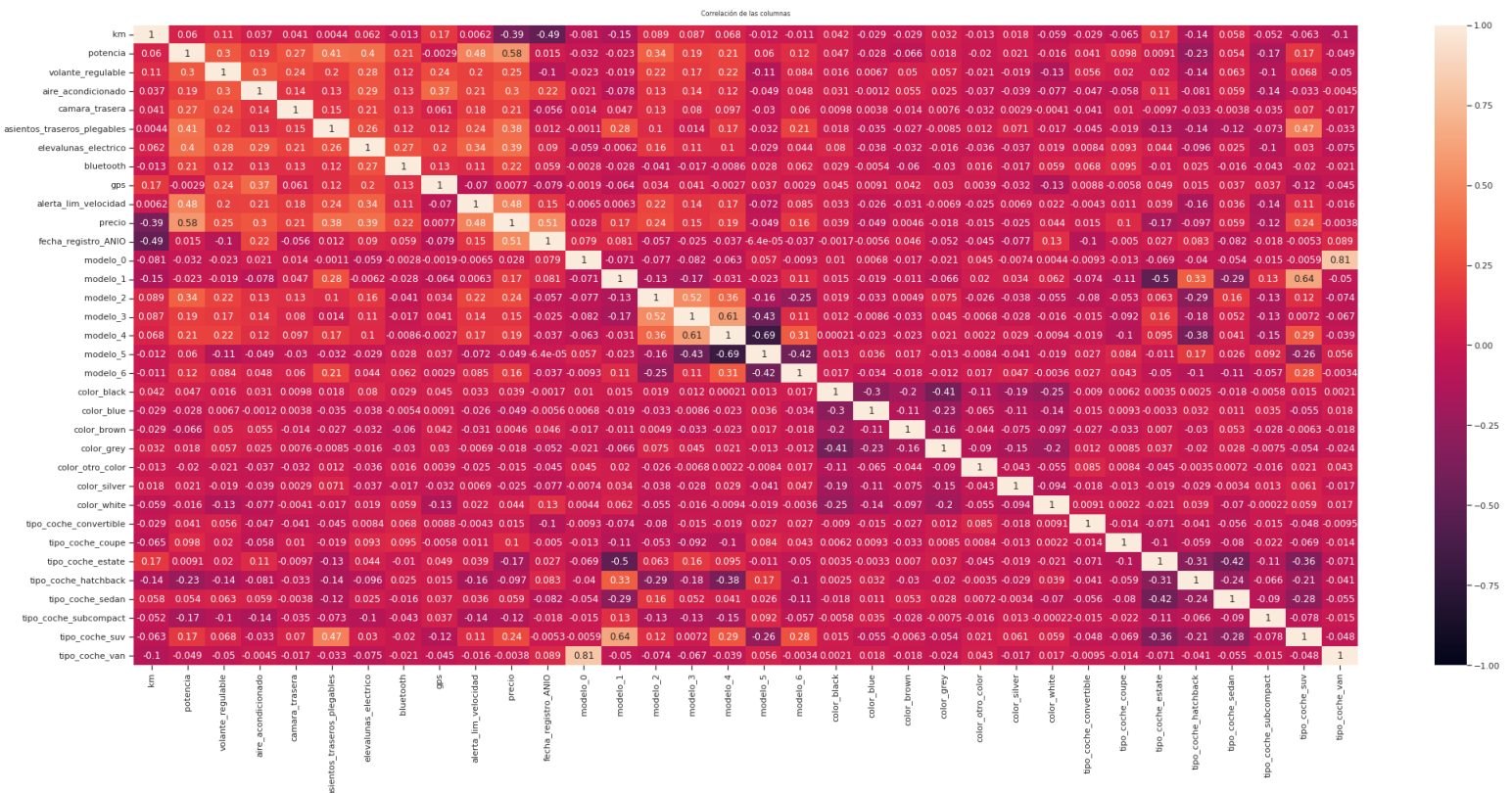
Tratamiento de las variables booleanas: Tratándose de variables binarias, simplemente transformamos las variables en tipo integral, para que los True sean iguales a 1 y los False a 0.

Tratamiento de las variables categóricas: En este caso, utilizamos la función de Pandas get_dummies, que permite eliminar la primera de las columnas generadas para cada característica codificada para evitar la denominada colinealidad.

Ya con todos los datos transformados y normalizados, analizamos la correlatividad para decidir si eliminamos variables que estén correlacionadas entre si y afecten el modelo.

Comprobación de posibles correlaciones entre las columnas de los datos preprocesados

Sacamos de nuevo la correlación de las columnas, esta vez mucho más numerosas. El mapa de calor es una forma visual muy útil para para conocer las variables y sus relaciones



Para poder tener una visión más rápida de todos los valores que más se acercan al 1 y al -1 podemos ordenar nuestra serie de correlaciones y mirar las primeras 10 filas para obtener las correlaciones que más se acercan al número negativo y las últimas 10 para ver si estas se acercan al número positivo.

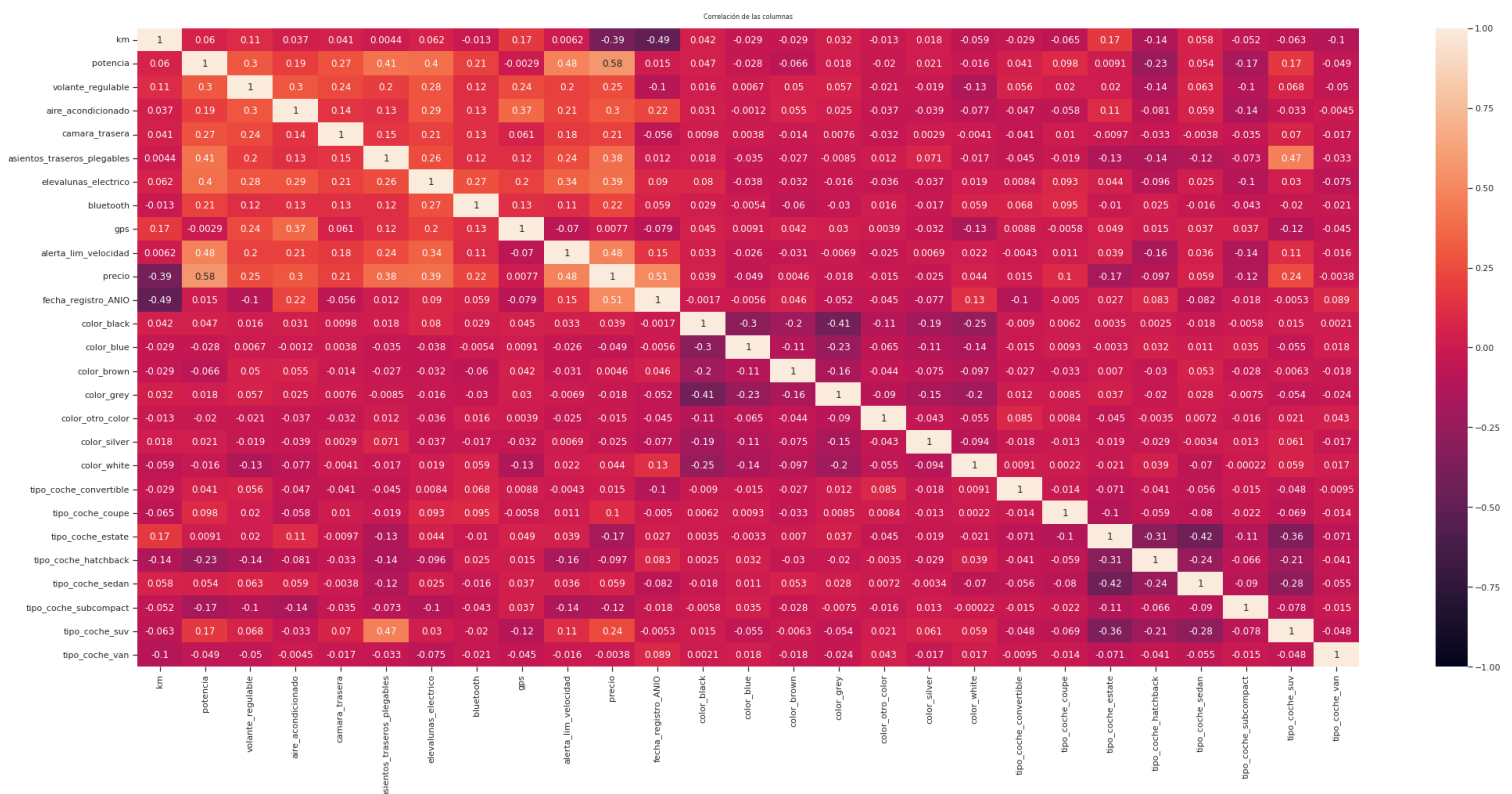
```

modelo_5          modelo_4 -0.694943
tipo_coche_estate modelo_1 -0.504571
fecha_registro_ANIO km      -0.487868
modelo_5          modelo_3 -0.431151
modelo_6          modelo_5 -0.422426

...
modelo_3          modelo_2  0.518647
precio            potencia  0.576930
modelo_4          modelo_3  0.607601
tipo_coche_suv    modelo_1  0.642878
tipo_coche_van    modelo_0  0.808011
Length: 561, dtype: float64

```

Se puede observar que las variables de 'modelo' tienen cierta correlación entre si (como también con los tipos de coches), pudiendo indicar multicolinealidad, es decir, que básicamente ofrecen la misma información. Adicionalmente tienen una corr cercana a 0 contra precio, implicando que no Con lo cual decidimos eliminar las variables de modelo.



Aquí se puede observar cómo es la relación del Precio con el resto de las variables.

precio			
precio	1.000000	tipo_coche_sedan	0.059129
potencia	0.576930	color_white	0.044266
fecha_registro_ANIO	0.509059	color_black	0.039279
alerta_lim_velocidad	0.483975	tipo_coche_convertible	0.014968
elevallunas_electrico	0.391332	gps	0.007736
asientos_traseros_plegables	0.383051	color_brown	0.004570
aire_acondicionado	0.296376	tipo_coche_van	-0.003813
volante_regulable	0.252173	color_otro_color	-0.014817
tipo_coche_suv	0.239949	color_grey	-0.017814
bluetooth	0.221278	color_silver	-0.024710
camara_trasera	0.205571	color_blue	-0.049133
tipo_coche_coupe	0.099885	tipo_coche_hatchback	-0.096893
		tipo_coche_subcompact	-0.119988
		tipo_coche_estate	-0.173283
		km	-0.393325