

**P A D D O C K**



*FAST CAR VALUATION*

¿QUÉ ES **P A D D O C K** ?



**Paddock** es una aplicación que permite mediante el uso de machine learning predecir un precio para un coche dado y compararlo con el precio de otros similares del mercado de segunda mano.

# DESARROLLO

# DESARROLLO

- Obtención de datos
- Limpieza de datos
- Visualización
- Machine Learning
- Einstein (Chatbot)
- Desarrollo web

 Tecnologías usadas Desarrollo

Lenguajes



PYTHON

DART

Entornos

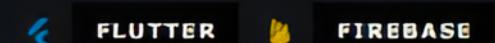


COLAB

VISUALSTUDIO

ANDROIDSTUDIO

FrontEnd



FLUTTER

FIREBASE

BackEnd



GOOGLE-CLOUD

DOCKER

FLASK

MONGODB

Diseño



FIGMA

Control de versiones



GITHUB

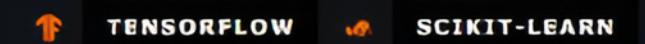
NOTION

 Análisis

NUMPY

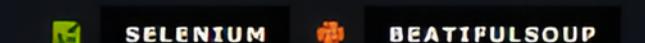
PANDAS

PLOTLY

 Machine Learning

TENSORFLOW

SCIKIT-LEARN

 Scraping

SELENIUM

BEAUTIFULSOUP

 Documentación

CHATGPT

CHATBING

STACKOVERFLOW

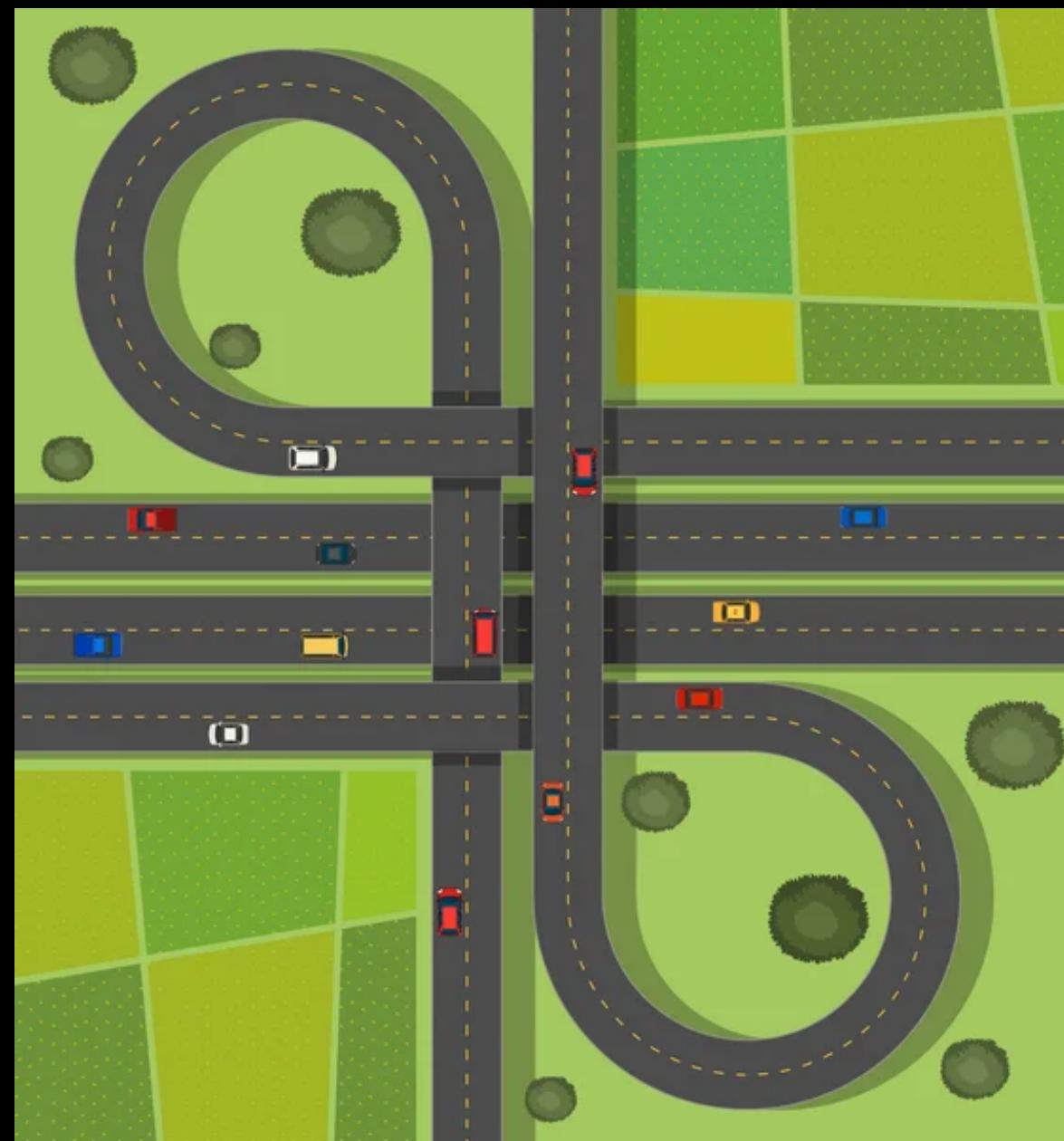
# OBTENCIÓN DE LOS DATOS

**SCRAPING**

# Localizando los objetivos



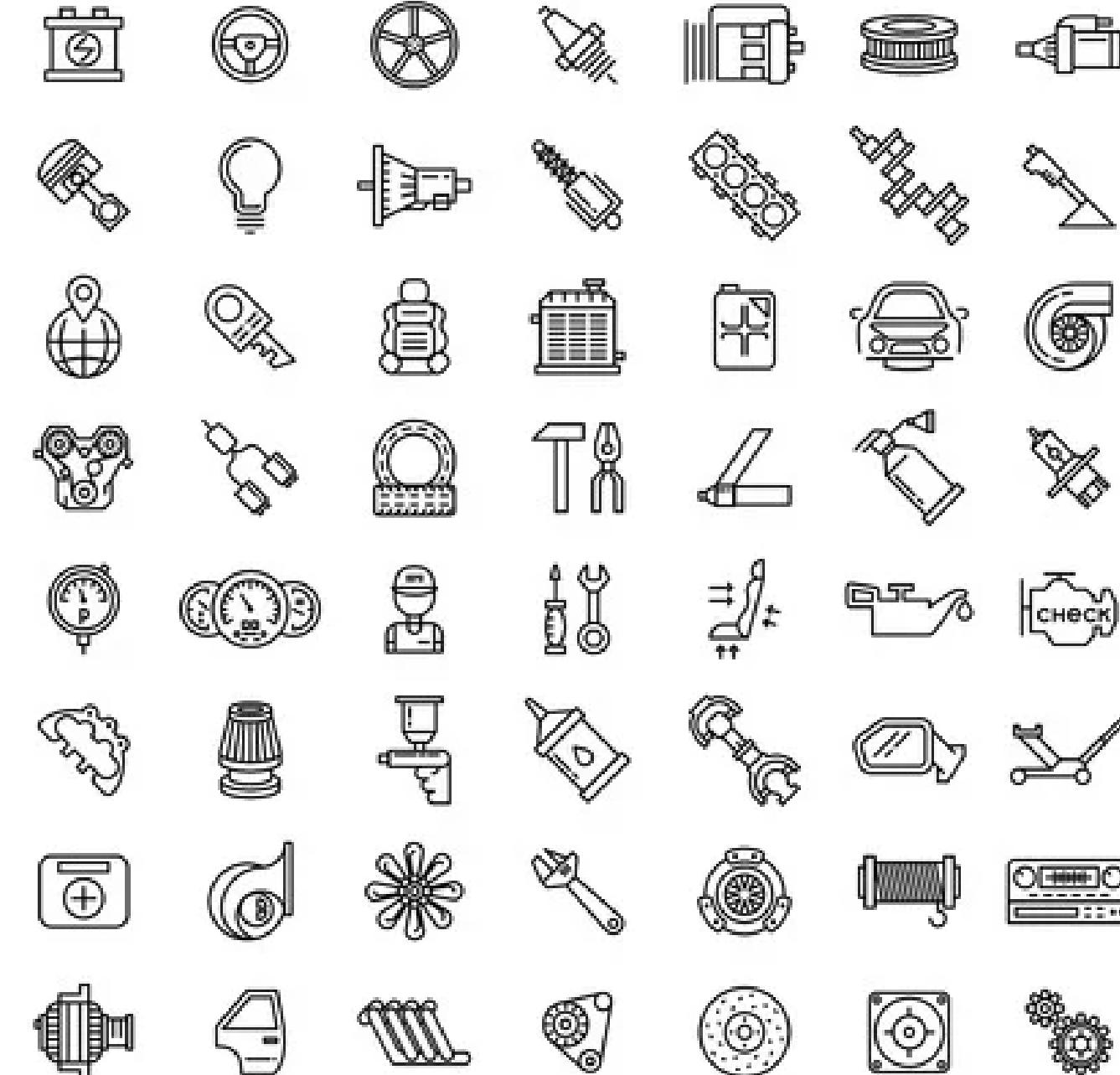
# Multiprocesamiento



# Cazando los datos



# Creación del dataframe



# LIMPIEZA DE LOS DATOS

## PREPROCESADO

## Información general

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179502 entries, 0 to 179501
Data columns (total 35 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0    179502 non-null   int64  
 1   localidad    179502 non-null   object  
 2   marca        179502 non-null   object  
 3   modelo       179502 non-null   object  
 4   año          179502 non-null   float64 
 5   motor (CV)   179502 non-null   float64 
 6   motor (KW)   179502 non-null   float64 
 7   kilometros   179502 non-null   float64 
 8   combustible  179502 non-null   object  
 9   puertas      179502 non-null   float64 
 10  cambio        179502 non-null   object  
 11  emisiones    179502 non-null   float64 
 12  color         179502 non-null   object  
 13  garantía    179502 non-null   object  
 14  vendedor     179502 non-null   object  
 15  precio        179502 non-null   float64 
 16  maletero      179502 non-null   float64 
 17  longitud      179502 non-null   float64 
 18  altura        179502 non-null   float64 
 19  anchura       179502 non-null   float64 
 20  plazas        179502 non-null   float64 
 21  deposito      179502 non-null   float64 
 22  peso max      179502 non-null   float64 
 23  carroceria   179502 non-null   object  
 24  vel. maxima   179502 non-null   float64 
 25  c. mixto      179502 non-null   float64 
 26  c. urbano     179502 non-null   float64 
 27  extraurbano   179502 non-null   float64 
 28  0-100 km/h    179502 non-null   float64 
 29  cilindrada    179502 non-null   float64 
 30  cilindros     179502 non-null   float64 
 31  transmisión   179502 non-null   object  
 32  par maximo    179502 non-null   float64 
 33  marchas       179502 non-null   float64 
 34  traccion      179502 non-null   object  
dtypes: float64(23), int64(1), object(11)
memory usage: 47.9+ MB
```

## Nulos

```

cars.isnull().sum()

      Unnamed: 0      0
      localidad      0
      marca         0
      modelo         0
      año           0
      motor (CV)    0
      motor (KW)    0
      kilometros    0
      combustible   0
      puertas        0
      cambio          0
      emisiones      0
      color          0
      garantía      0
      vendedor       0
      precio          0
      maletero       0
      longitud       0
      altura          0
      anchura         0
      plazas          0
      deposito       0
      peso max        0
      carroceria     0
      vel. maxima    0
      c. mixto        0
      c. urbano       0
      extraurbano    0
      0-100 km/h     0
      cilindrada     0
      cilindros      0
      transmisión    0
      par maximo     0
      marchas         0
      traccion        0
      dtype: int64
  
```

## Información general

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179502 entries, 0 to 179501
Data columns (total 35 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0    179502 non-null   int64  
 1   localidad    179502 non-null   object  
 2   marca        179502 non-null   object  
 3   modelo        179502 non-null   object  
 4   año          179502 non-null   float64 
 5   motor (CV)   179502 non-null   float64 
 6   motor (KW)   179502 non-null   float64 
 7   kilometros   179502 non-null   float64 
 8   combustible  179502 non-null   object  
 9   puertas      179502 non-null   float64 
 10  cambio        179502 non-null   object  
 11  emisiones    179502 non-null   float64 
 12  color         179502 non-null   object  
 13  garantía    179502 non-null   object  
 14  vendedor     179502 non-null   object  
 15  precio        179502 non-null   float64 
 16  maletero      179502 non-null   float64 
 17  longitud      179502 non-null   float64 
 18  altura         179502 non-null   float64 
 19  anchura        179502 non-null   float64 
 20  plazas         179502 non-null   float64 
 21  deposito      179502 non-null   float64 
 22  peso max       179502 non-null   float64 
 23  carroceria   179502 non-null   object  
 24  vel. maxima   179502 non-null   float64 
 25  c. mixto       179502 non-null   float64 
 26  c. urbano      179502 non-null   float64 
 27  extraurbano   179502 non-null   float64 
 28  0-100 km/h    179502 non-null   float64 
 29  cilindrada    179502 non-null   float64 
 30  cilindros      179502 non-null   float64 
 31  transmisión   179502 non-null   object  
 32  par maximo    179502 non-null   float64 
 33  marchas        179502 non-null   float64 
 34  traccion       179502 non-null   object  
dtypes: float64(23), int64(1), object(11)
memory usage: 47.9+ MB
  
```

## Datos de interés

### Información general

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179502 entries, 0 to 179501
Data columns (total 35 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0    179502 non-null   int64  
 1   localidad    179502 non-null   object  
 2   marca        179502 non-null   object  
 3   modelo       179502 non-null   object  
 4   año          179502 non-null   float64 
 5   motor (CV)  179502 non-null   float64 
 6   motor (KW)  179502 non-null   float64 
 7   kilometros  179502 non-null   float64 
 8   combustible 179502 non-null   object  
 9   puertas      179502 non-null   float64 
 10  cambio        179502 non-null   object  
 11  emisiones    179502 non-null   float64 
 12  color         179502 non-null   object  
 13  garantía    179502 non-null   object  
 14  vendedor     179502 non-null   object  
 15  precio        179502 non-null   float64 
 16  maletero      179502 non-null   float64 
 17  longitud     179502 non-null   float64 
 18  altura        179502 non-null   float64 
 19  anchura       179502 non-null   float64 
 20  plazas        179502 non-null   float64 
 21  deposito     179502 non-null   float64 
 22  peso max     179502 non-null   float64 
 23  carroceria   179502 non-null   object  
 24  vel. maxima  179502 non-null   float64 
 25  c. mixto     179502 non-null   float64 
 26  c. urbano    179502 non-null   float64 
 27  extraurbano  179502 non-null   float64 
 28  0-100 km/h   179502 non-null   float64 
 29  cilindrada   179502 non-null   float64 
 30  cilindros    179502 non-null   float64 
 31  transmisión  179502 non-null   object  
 32  par maximo   179502 non-null   float64 
 33  marchas      179502 non-null   float64 
 34  traccion     179502 non-null   object  
dtypes: float64(23), int64(1), object(11)
memory usage: 47.9+ MB
```

### Nulos

```
cars.isnull().sum()
```

	count
Unnamed: 0	179502
año	179502
motor (CV)	179502
motor (KW)	179502
kilometros	179502
puertas	179502
emisiones	179502
precio	179502
maletero	179502
longitud	179502
altura	179502
anchura	179502
plazas	179502
deposito	179502
peso max	179502
carroceria	179502
vel. maxima	179502
c. mixto	179502
c. urbano	179502
extraurbano	179502
0-100 km/h	179502
cilindrada	179502
cilindros	179502
transmisión	179502
par maximo	179502
marchas	179502
traccion	179502
dtype: int64	0

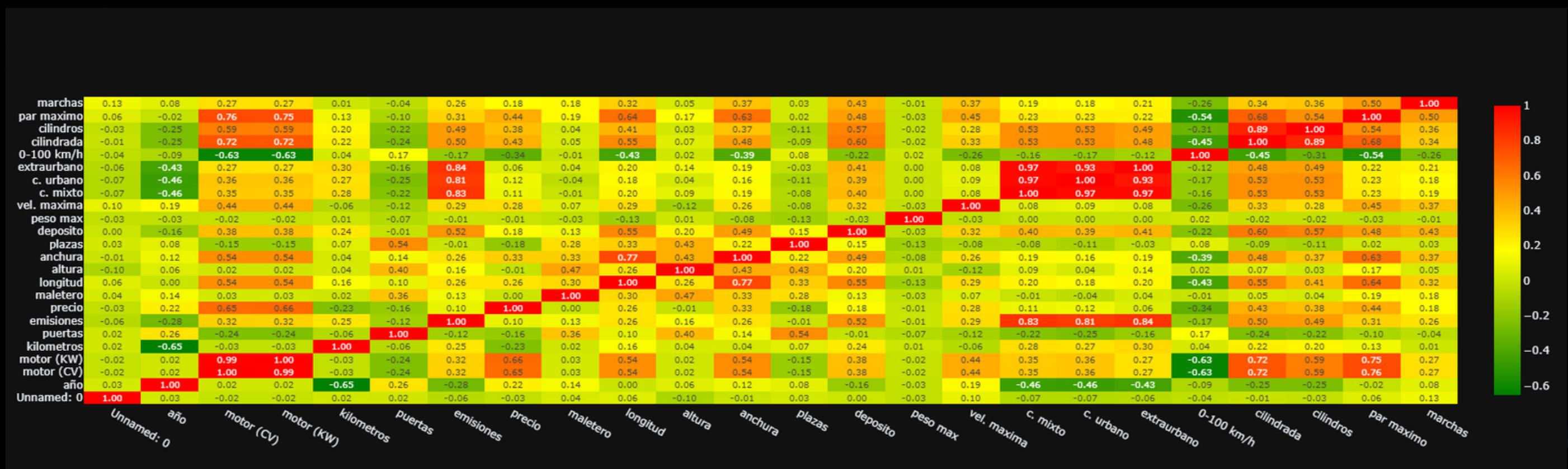
	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	179502.0	4456.412987	3645.833011	0.0	1347.00	3565.50	6802.750	15547.0
año	179502.0	2016.842046	3.841619	1993.0	2015.00	2018.00	2019.000	2023.0
motor (CV)	179502.0	147.338215	74.697602	1.0	110.00	130.00	150.000	887.0
motor (KW)	179502.0	108.396413	55.132019	9.0	81.00	96.00	110.000	817.0
kilometros	179502.0	81811.256900	56117.663986	0.0	40000.00	74498.00	114000.000	2890000.0
puertas	179502.0	4.708109	0.735979	0.0	5.00	5.00	5.000	5.0
emisiones	179502.0	114.994368	48.185070	0.0	103.00	115.00	134.000	495.0
precio	179502.0	23926.384798	32034.218607	1.0	13850.00	18990.00	26450.000	1799700.0
maletero	179502.0	1007.807924	582.970727	0.0	470.00	1122.00	1478.000	6000.0
longitud	179502.0	438.542675	32.171825	224.5	420.40	438.00	462.400	616.5
altura	179502.0	153.986220	11.083085	103.6	145.50	151.00	162.300	223.5
anchura	179502.0	181.004077	7.393938	123.7	177.00	180.70	184.200	224.4
plazas	179502.0	4.965181	0.585251	2.0	5.00	5.00	5.000	9.0
deposito	179502.0	52.278442	14.298763	0.0	45.00	51.00	60.000	121.0
peso max	179502.0	2.638252	25.286031	0.0	1.75	1.92	2.125	975.0
vel. maxima	179502.0	190.620322	43.072073	0.0	180.00	192.00	210.000	350.0
c. mixto	179502.0	4.739973	2.034619	0.0	4.10	4.70	5.500	22.0
c. urbano	179502.0	5.752468	2.805384	0.0	4.70	5.70	6.700	31.0
extraurbano	179502.0	4.087472	1.647663	0.0	3.60	4.10	4.800	17.0
0-100 km/h	179502.0	9.868147	2.725705	0.0	8.60	10.20	11.400	62.0
cilindrada	179502.0	1697.397221	638.232397	0.0	1368.00	1598.00	1995.000	8300.0
cilindros	179502.0	3.921750	0.974608	0.0	4.00	4.00	4.000	12.0
par maximo	179502.0	277.381892	121.722770	0.0	200.00	260.00	340.000	973.0
marchas	179502.0	5.934558	1.482954	0.0	5.00	6.00	6.000	10.0

## Datos de interés

		count	mean	std	min	25%	50%	75%	max
<b>Unnamed: 0</b>	179502.0	4456.412987	3645.833011	0.0	1347.00	3565.50	6802.750	15547.0	
<b>año</b>	179502.0	2016.842046	3.841619	1993.0	2015.00	2018.00	2019.000	2023.0	
<b>motor (CV)</b>	179502.0	147.338215	74.697602	1.0	110.00	130.00	150.000	887.0	
<b>motor (KW)</b>	179502.0	108.396413	55.132019	9.0	81.00	96.00	110.000	817.0	
<b>kilometros</b>	179502.0	81811.256900	56117.663986	0.0	40000.00	74498.00	114000.000	2890000.0	
<b>puertas</b>	179502.0	4.708109	0.735979	0.0	5.00	5.00	5.000	5.0	
<b>emisiones</b>	179502.0	114.994368	48.185070	0.0	103.00	115.00	134.000	495.0	
<b>precio</b>	179502.0	23926.384798	32034.218607	1.0	13850.00	18990.00	26450.000	1799700.0	
<b>maletero</b>	179502.0	1007.807924	582.970727	0.0	470.00	1122.00	1478.000	6000.0	
<b>longitud</b>	179502.0	438.542675	32.171825	224.5	420.40	438.00	462.400	616.5	
<b>altura</b>	179502.0	153.986220	11.083085	103.6	145.50	151.00	162.300	223.5	
<b>anchura</b>	179502.0	181.004077	7.393938	123.7	177.00	180.70	184.200	224.4	
<b>plazas</b>	179502.0	4.965181	0.585251	2.0	5.00	5.00	5.000	9.0	
<b>deposito</b>	179502.0	52.278442	14.298763	0.0	45.00	51.00	60.000	121.0	
<b>peso max</b>	179502.0	2.638252	25.286031	0.0	1.75	1.92	2.125	975.0	
<b>vel. maxima</b>	179502.0	190.620322	43.072073	0.0	180.00	192.00	210.000	350.0	
<b>c. mixto</b>	179502.0	4.739973	2.034619	0.0	4.10	4.70	5.500	22.0	
<b>c. urbano</b>	179502.0	5.752468	2.805384	0.0	4.70	5.70	6.700	31.0	
<b>extraurbano</b>	179502.0	4.087472	1.647663	0.0	3.60	4.10	4.800	17.0	
<b>0-100 km/h</b>	179502.0	9.868147	2.725705	0.0	8.60	10.20	11.400	62.0	
<b>cilindrada</b>	179502.0	1697.397221	638.232397	0.0	1368.00	1598.00	1995.000	8300.0	
<b>cilindros</b>	179502.0	3.921750	0.974608	0.0	4.00	4.00	4.000	12.0	
<b>par maximo</b>	179502.0	277.381892	121.722770	0.0	200.00	260.00	340.000	973.0	
<b>marchas</b>	179502.0	5.934558	1.482954	0.0	5.00	6.00	6.000	10.0	

# Correlaciones

## Correlaciones generales



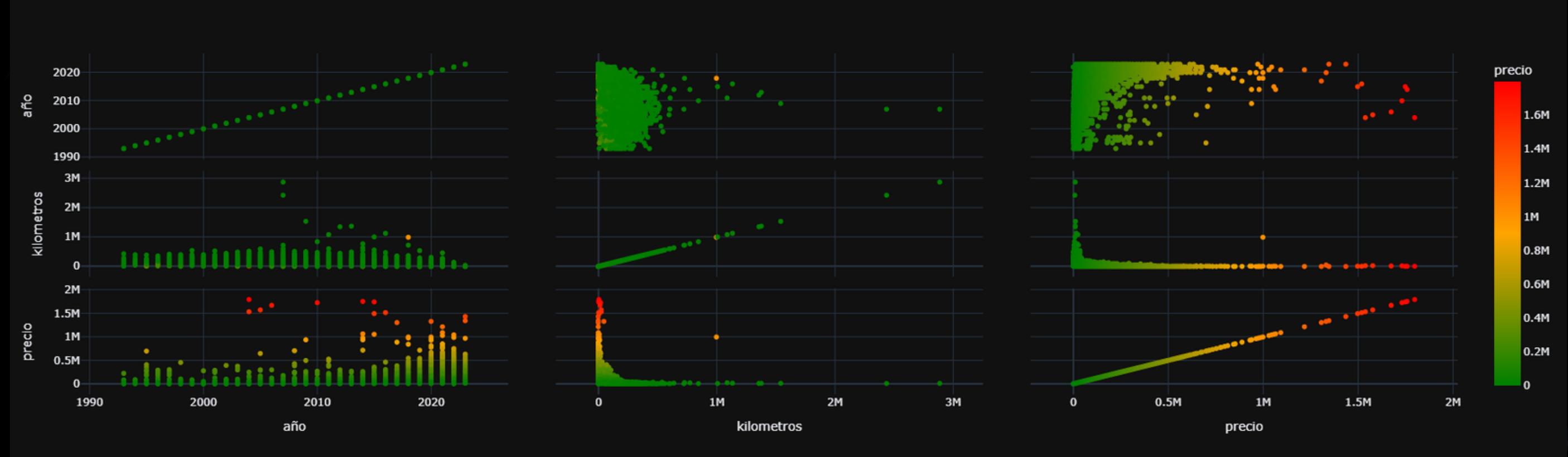
## **Correlaciones generales**



# Correlaciones sobre el precio



## Graficas de correlaciones



## Outlier



# Código



## Eliminación de columnas

```
● ● ●  
1 cars_pre = cars.drop(['Unnamed: 0', 'puertas', 'maletero', 'longitud', 'altura', 'anchura', 'peso max', 'vel. maxima', 'c. mixto', 'c. urbano', 'extraurbano', 'cilindros', 'par maximo', 'color', 'garantia', 'vendedor', 'transmision', 'carroceria', 'traccion'], axis=1)
```

## Función para transformar valores

```
● ● ●  
1 # Función para transformar Los valores categoricos a numeros  
2 def parseador(dataset, marca_data, model_data, combustible_data, cambio_data, locate_car):  
3  
4     for index, model in enumerate(tqdm(marca_data)):  
5         dataset.loc[dataset['marca'] == model, 'marca_id'] = index  
6         car_list_marca.append((index, model))  
7  
8     for index, model in enumerate(tqdm(model_data)):  
9         dataset.loc[dataset['modelo'] == model, 'modelo_id'] = index  
10        car_list_model.append((index, model))  
11  
12    for index, model in enumerate(tqdm(combustible_data)):  
13        dataset.loc[dataset['combustible'] == model, 'combustible_id'] = index  
14        car_list_combustible_data.append((index, model))  
15  
16    for index, model in enumerate(tqdm(cambio_data)):  
17        dataset.loc[dataset['cambio'] == model, 'cambio_id'] = index  
18        car_list_cambio_data.append((index, model))  
19  
20    for index, model in enumerate(tqdm(locate_car)):  
21        dataset.loc[dataset['localidad'] == model, 'localidad_id'] = index  
22        car_list_locate.append((index, model))
```

# Resultados

## Origen

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	179502.0	4456.412987	3645.833011	0.0	1347.00	3565.50	6802.750	15547.0
año	179502.0	2016.842046	3.841619	1993.0	2015.00	2018.00	2019.000	2023.0
motor (CV)	179502.0	147.338215	74.697602	1.0	110.00	130.00	150.000	887.0
motor (KW)	179502.0	108.396413	55.132019	9.0	81.00	96.00	110.000	817.0
kilometros	179502.0	81811.256900	56117.663986	0.0	40000.00	74498.00	114000.000	2890000.0
puertas	179502.0	4.708109	0.735979	0.0	5.00	5.00	5.000	5.0
emisiones	179502.0	114.994368	48.185070	0.0	103.00	115.00	134.000	495.0
precio	179502.0	23926.384798	32034.218607	1.0	13850.00	18990.00	26450.000	1799700.0
maletero	179502.0	1007.807924	582.970727	0.0	470.00	1122.00	1478.000	6000.0
longitud	179502.0	438.542675	32.171825	224.5	420.40	438.00	462.400	616.5
altura	179502.0	153.986220	11.083085	103.6	145.50	151.00	162.300	223.5
anchura	179502.0	181.004077	7.393938	123.7	177.00	180.70	184.200	224.4
plazas	179502.0	4.965181	0.585251	2.0	5.00	5.00	5.000	9.0
deposito	179502.0	52.278442	14.298763	0.0	45.00	51.00	60.000	121.0
peso max	179502.0	2.638252	25.286031	0.0	1.75	1.92	2.125	975.0
vel. maxima	179502.0	190.620322	43.072073	0.0	180.00	192.00	210.000	350.0
c. mixto	179502.0	4.739973	2.034619	0.0	4.10	4.70	5.500	22.0
c. urbano	179502.0	5.752468	2.805384	0.0	4.70	5.70	6.700	31.0
extraurbano	179502.0	4.087472	1.647663	0.0	3.60	4.10	4.800	17.0
0-100 km/h	179502.0	9.868147	2.725705	0.0	8.60	10.20	11.400	62.0
cilindrada	179502.0	1697.397221	638.232397	0.0	1368.00	1598.00	1995.000	8300.0
cilindros	179502.0	3.921750	0.974608	0.0	4.00	4.00	4.000	12.0
par maximo	179502.0	277.381892	121.722770	0.0	200.00	260.00	340.000	973.0
marchas	179502.0	5.934558	1.482954	0.0	5.00	6.00	6.000	10.0

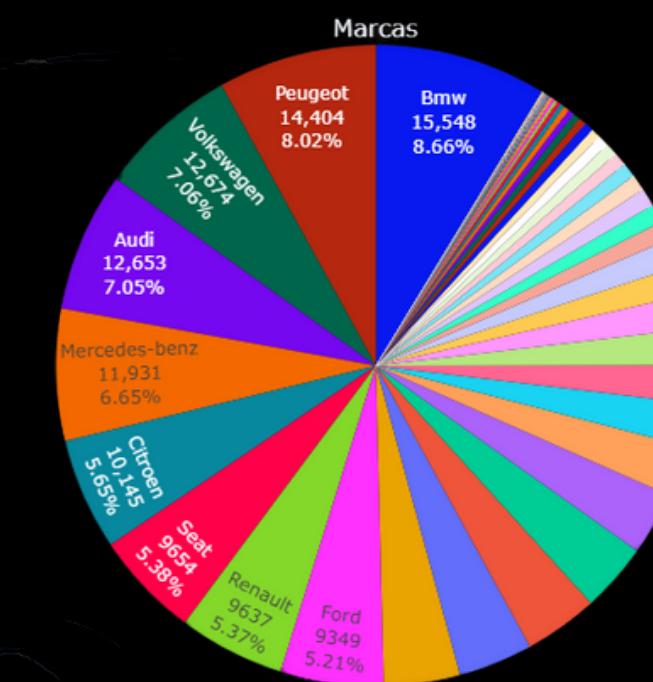
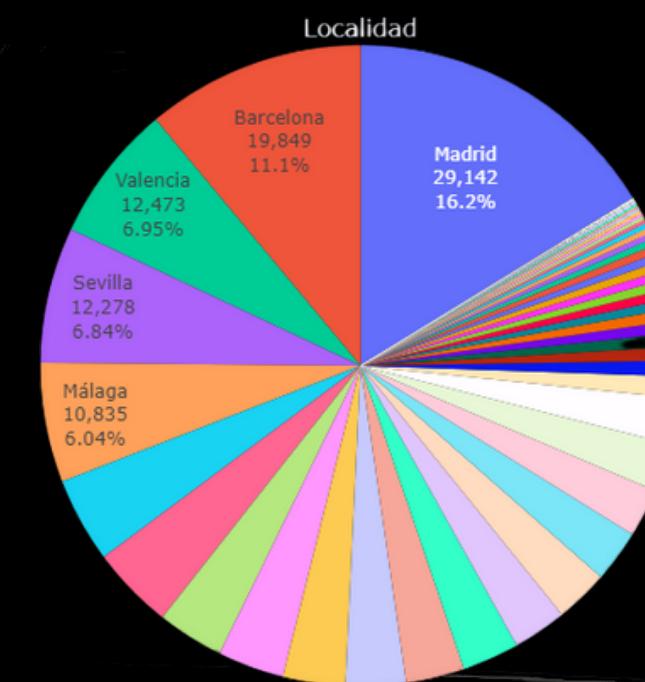
## Origen

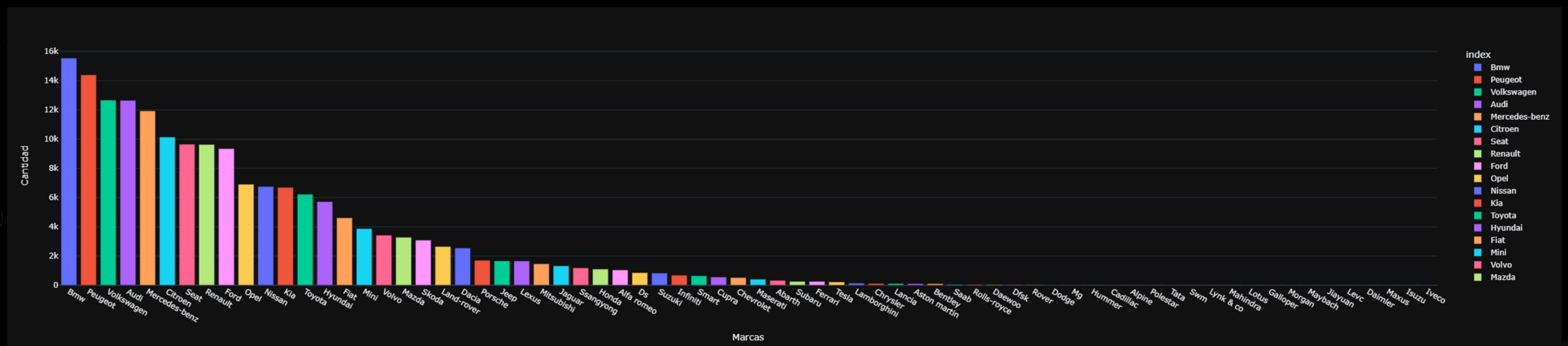
	count	mean	std	min	25%	50%	75%	max
<b>Unnamed: 0</b>	179502.0	4456.412987	3645.833011	0.0	1347.00	3565.50	6802.750	15547.0
año	179502.0	2016.842046	3.841619	1993.0	2015.00	2018.00	2019.000	2023.0
motor (CV)	179502.0	147.338215	74.697602	1.0	110.00	130.00	150.000	887.0
motor (KW)	179502.0	108.396413	55.132019	9.0	81.00	96.00	110.000	817.0
kilometros	179502.0	81811.256900	56117.663986	0.0	40000.00	74498.00	114000.000	2890000.0
puertas	179502.0	4.708109	0.735979	0.0	5.00	5.00	5.000	5.0
emisiones	179502.0	114.994368	48.185070	0.0	103.00	115.00	134.000	495.0
precio	179502.0	23926.384798	32034.218607	1.0	13850.00	18990.00	26450.000	1799700.0
maletero	179502.0	1007.807924	582.970727	0.0	470.00	1122.00	1478.000	6000.0
longitud	179502.0	438.542675	32.171825	224.5	420.40	438.00	462.400	616.5
altura	179502.0	153.986220	11.083085	103.6	145.50	151.00	162.300	223.5
anchura	179502.0	181.004077	7.393938	123.7	177.00	180.70	184.200	224.4
plazas	179502.0	4.965181	0.585251	2.0	5.00	5.00	5.000	9.0
deposito	179502.0	52.278442	14.298763	0.0	45.00	51.00	60.000	121.0
peso max	179502.0	2.638252	25.286031	0.0	1.75	1.92	2.125	975.0
vel. maxima	179502.0	190.620322	43.072073	0.0	180.00	192.00	210.000	350.0
c. mixto	179502.0	4.739973	2.034619	0.0	4.10	4.70	5.500	22.0
c. urbano	179502.0	5.752468	2.805384	0.0	4.70	5.70	6.700	31.0
extraurbano	179502.0	4.087472	1.647663	0.0	3.60	4.10	4.800	17.0
0-100 km/h	179502.0	9.868147	2.725705	0.0	8.60	10.20	11.400	62.0
cilindrada	179502.0	1697.397221	638.232397	0.0	1368.00	1598.00	1995.000	8300.0
cilindros	179502.0	3.921750	0.974608	0.0	4.00	4.00	4.000	12.0
par maximo	179502.0	277.381892	121.722770	0.0	200.00	260.00	340.000	973.0
marchas	179502.0	5.934558	1.482954	0.0	5.00	6.00	6.000	10.0

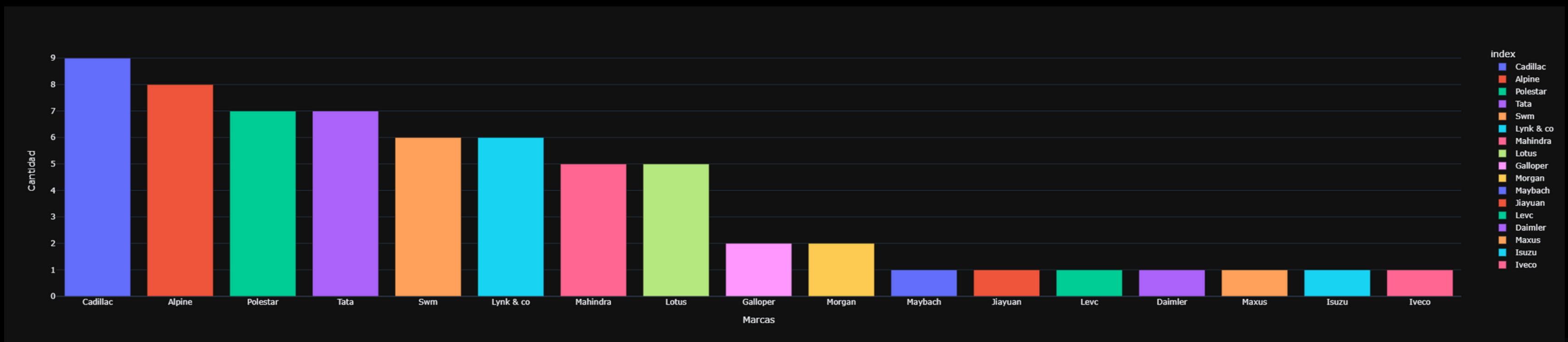
## Resultado

	count	mean	std	min	25%	50%	75%	max
año	179502.0	2016.842046	3.841619	1993.0	2015.0	2018.0	2019.0	2023.0
motor (CV)	179502.0	147.338215	74.697602	1.0	110.0	130.0	150.0	887.0
motor (KW)	179502.0	108.396413	55.132019	9.0	81.0	96.0	110.0	817.0
kilometros	179502.0	81811.256900	56117.663986	0.0	40000.0	74498.0	114000.0	2890000.0
emisiones	179502.0	114.994368	48.185070	0.0	103.0	115.0	134.0	495.0
precio	179502.0	23926.384798	32034.218607	1.0	13850.0	18990.0	26450.0	1799700.0
plazas	179502.0	4.965181	0.585251	2.0	5.0	5.0	5.0	9.0
deposito	179502.0	52.278442	14.298763	0.0	45.0	51.0	60.0	121.0
0-100 km/h	179502.0	9.868147	2.725705	0.0	8.6	10.2	11.4	62.0
cilindrada	179502.0	1697.397221	638.232397	0.0	1368.0	1598.0	1995.0	8300.0
marchas	179502.0	5.934558	1.482954	0.0	5.0	6.0	6.0	10.0
marca_id	179502.0	36.936352	21.835715	0.0	18.0	44.0	54.0	69.0
modelo_id	179502.0	372.703017	214.247216	0.0	186.0	409.0	545.0	718.0
combustible_id	179502.0	1.970446	1.204450	0.0	1.0	1.0	3.0	7.0
cambio_id	179502.0	0.605308	0.488786	0.0	0.0	1.0	1.0	1.0
localidad_id	179502.0	23.404658	14.736913	0.0	8.0	28.0	34.0	49.0

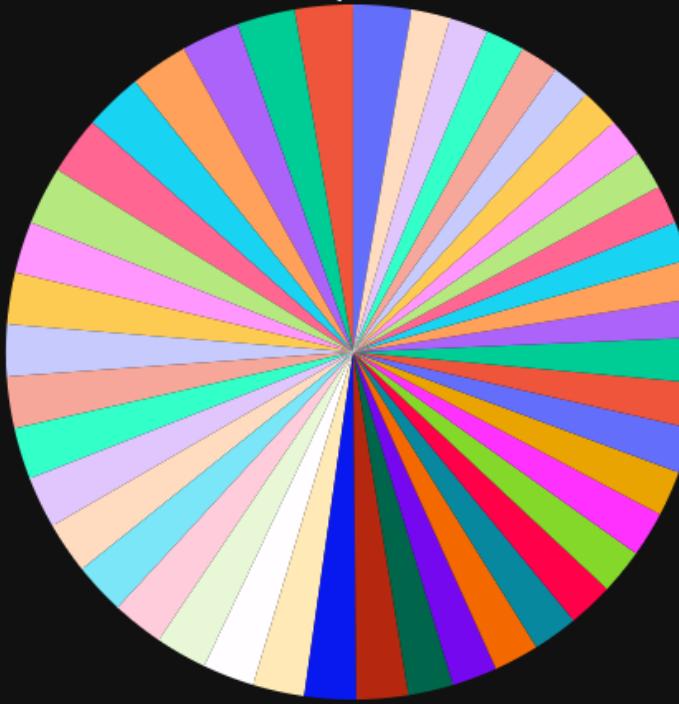
# VISUALIZACIÓN



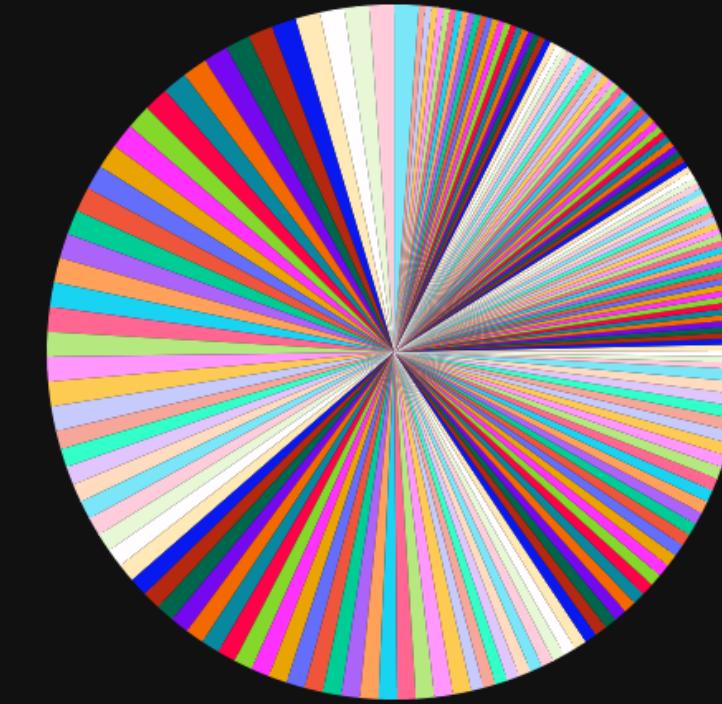




Modelos entre < 10 y > 5. Total coches: 45



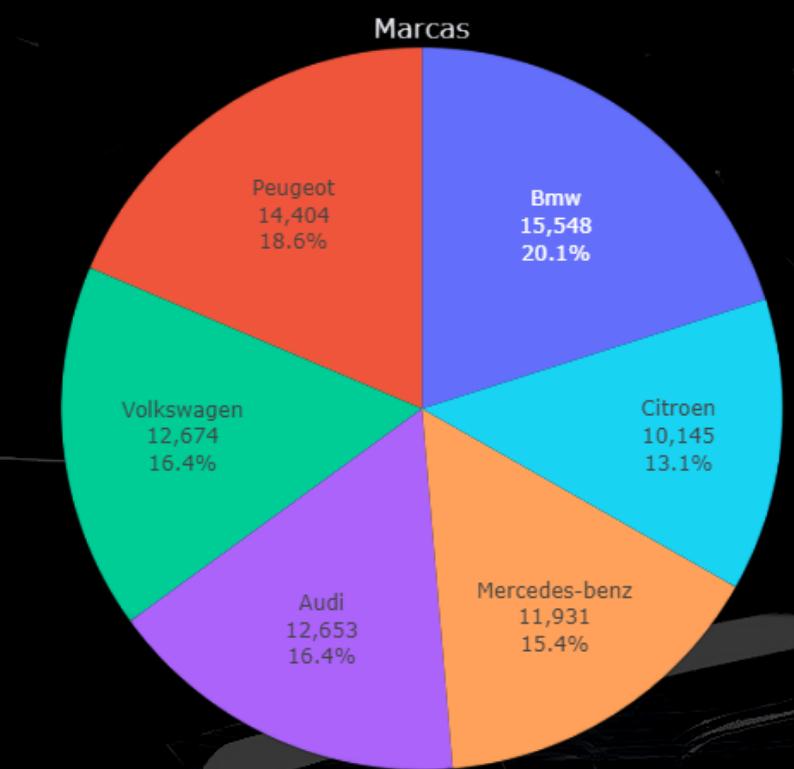
Modelo < 5 de cantidad. Total coches: 177



- Chevrolet Lacetti
- Suzuki Sx4
- Kia Stinger
- Mg Zs suv
- Bmw I4
- Mercedes-benz Slr
- Mercedes-benz Clase sls amg
- Dodge Caliber
- Aston martin Vanquish
- Nissan 370z
- Seat Altea freetrack
- Ferrari Ff
- Renault Mégane scénic
- Alpine A110
- Chevrolet Kalos
- Toyota Camry
- Volvo C70
- Seat Córdoba
- Opel Karl
- Infiniti Qx50
- Rover 75
- Citroen C4 x

# Aplicando filtrado

## Marcas + 10.000 coches



## Explorando los datos precio/año

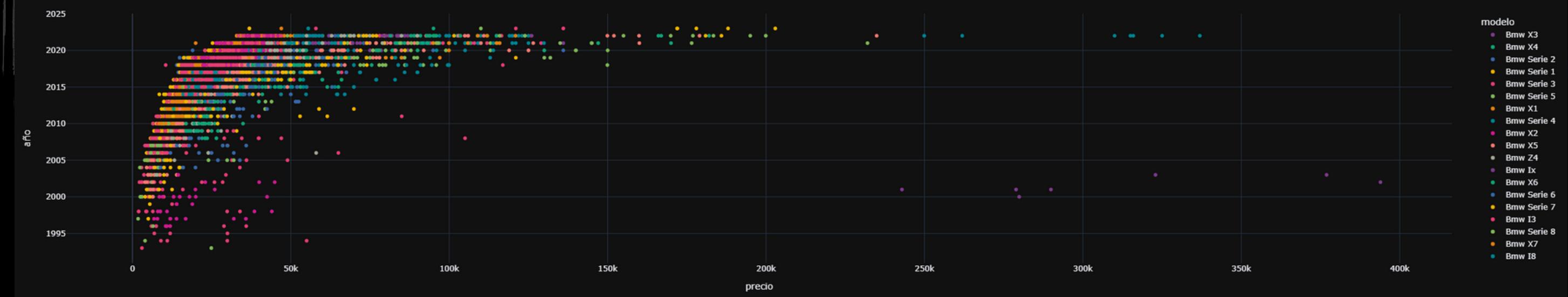
**Peugeot**



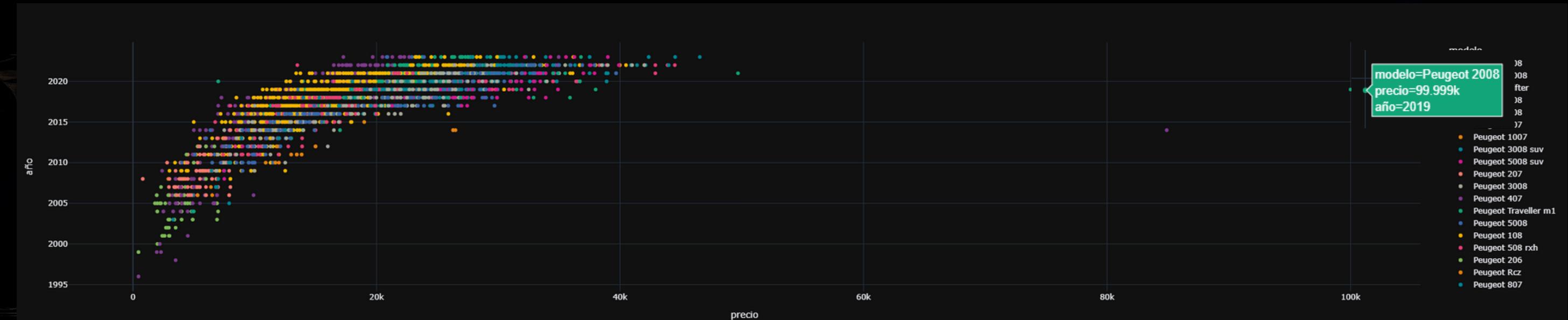
**VW**



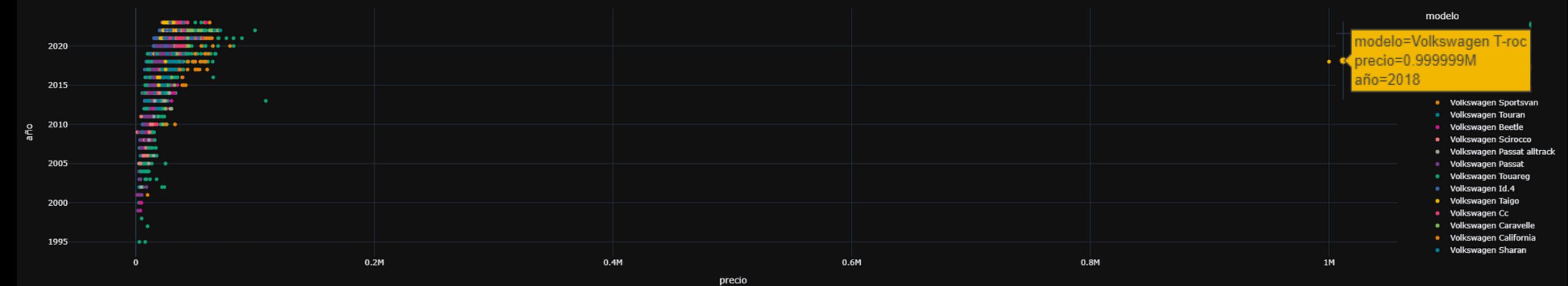
**BMW**



**Peugeot**



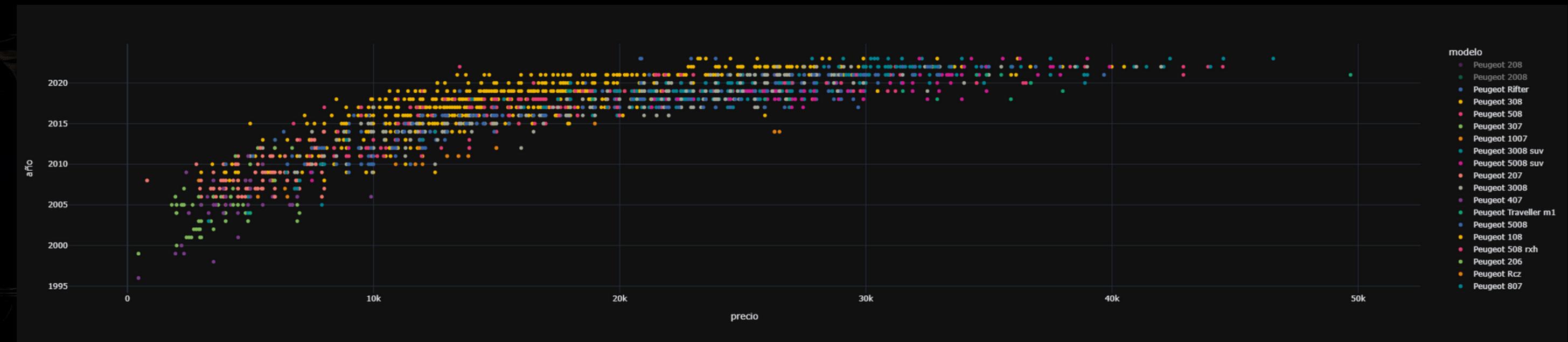
**VW**



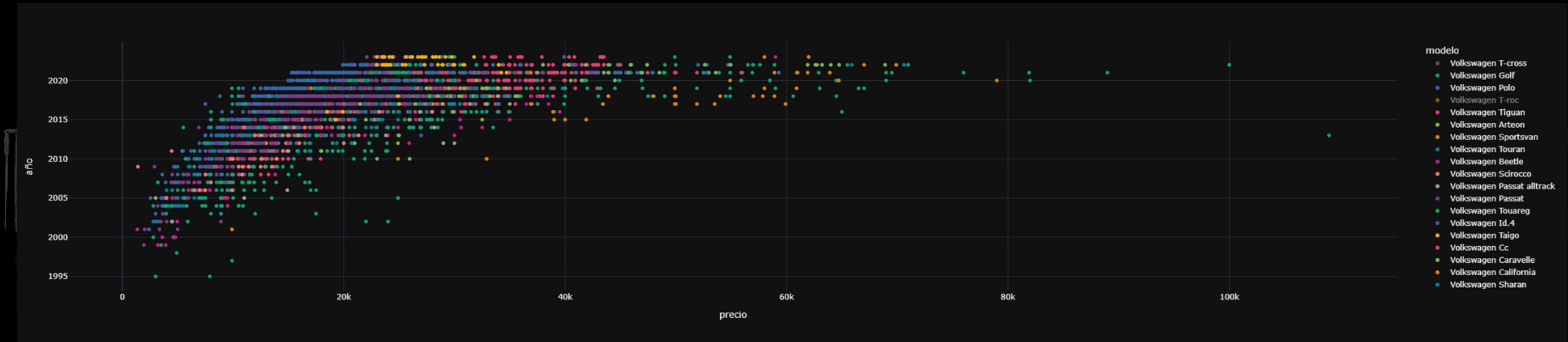
**BMW**



**Peugeot**

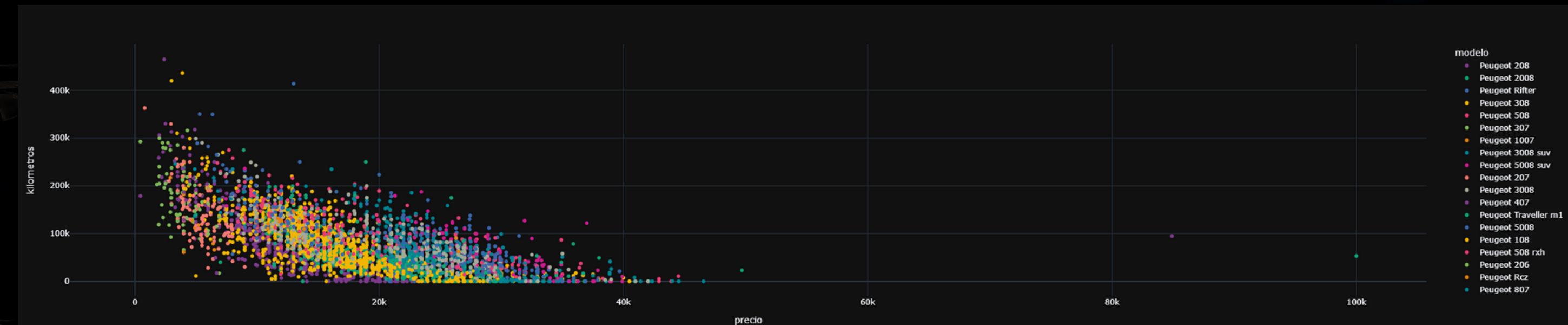


**VW**



# Explorando los datos precio/kilometros

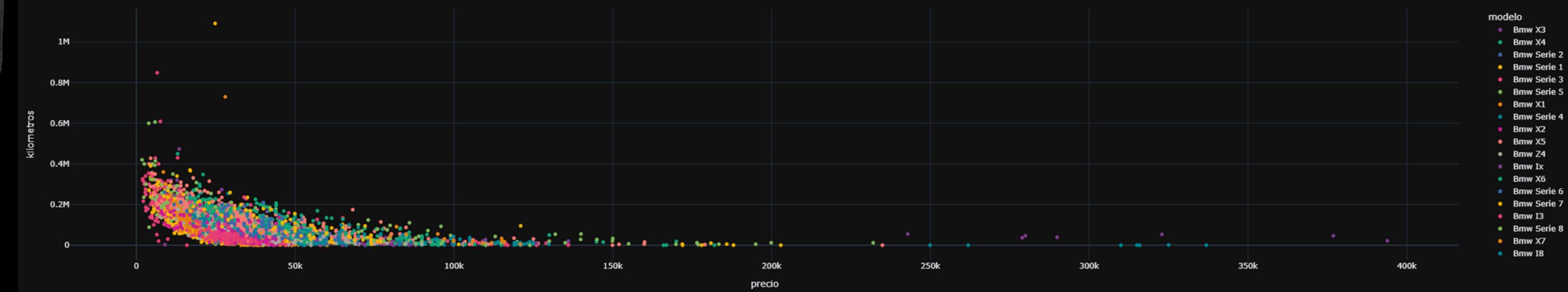
Peugeot



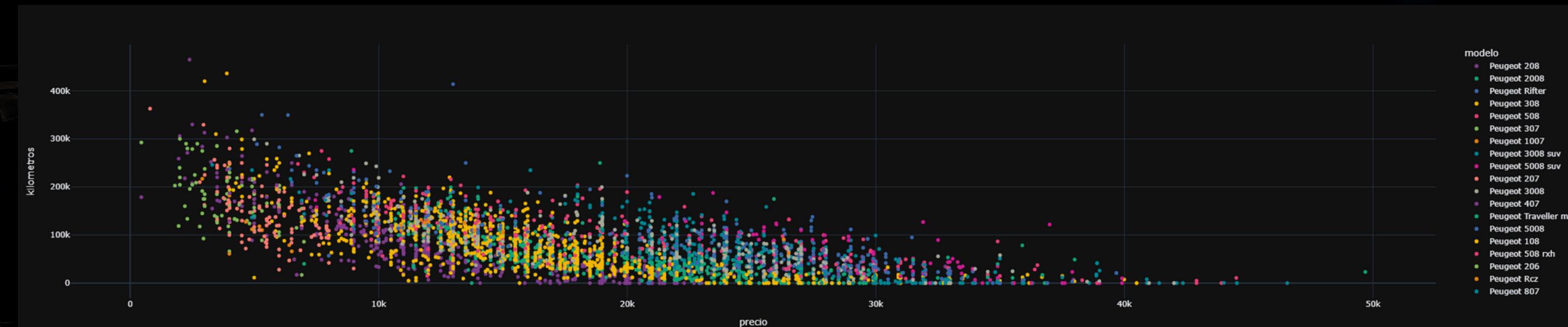
VW



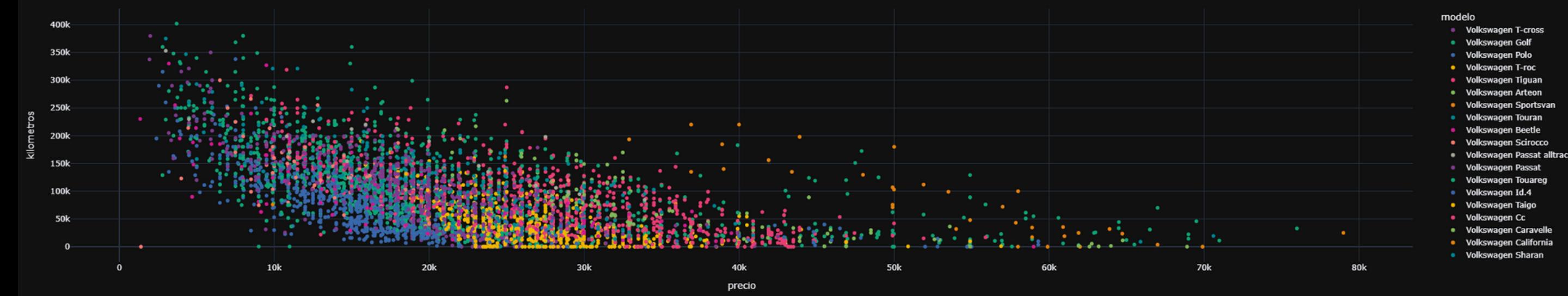
BMW



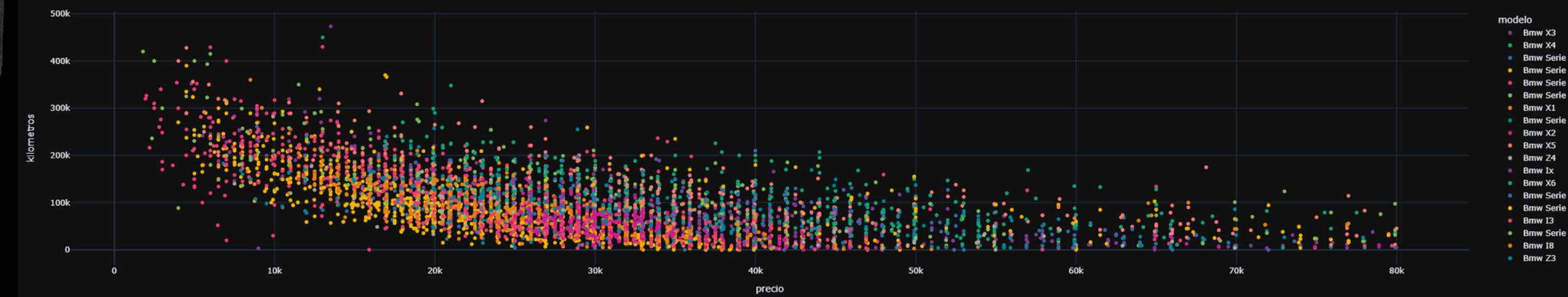
**Peugeot**



**VW**



**BMW**

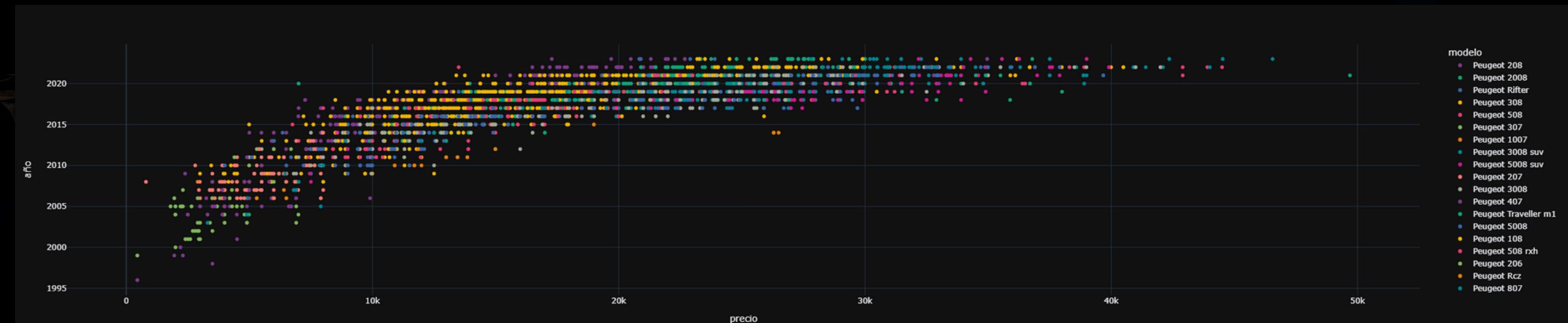


# Explorando los datos precio/año

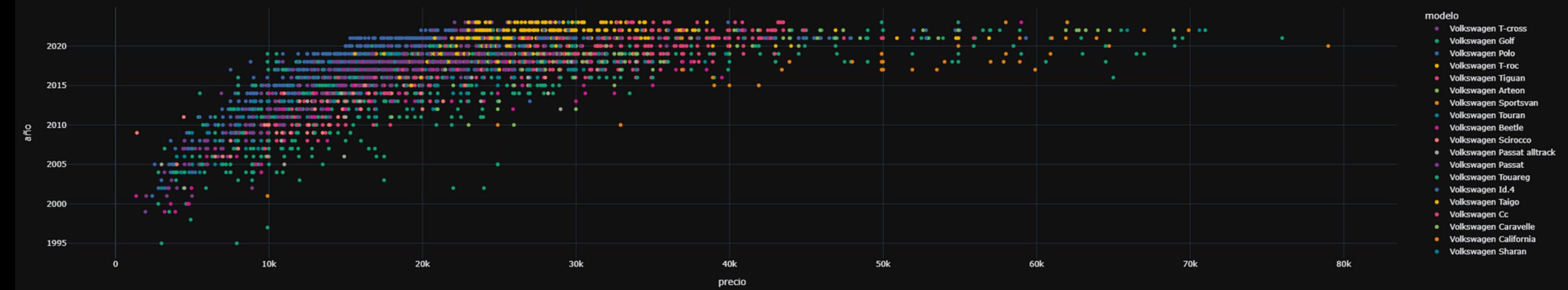
Precio = 80.000€

Kilometros = 500.000km

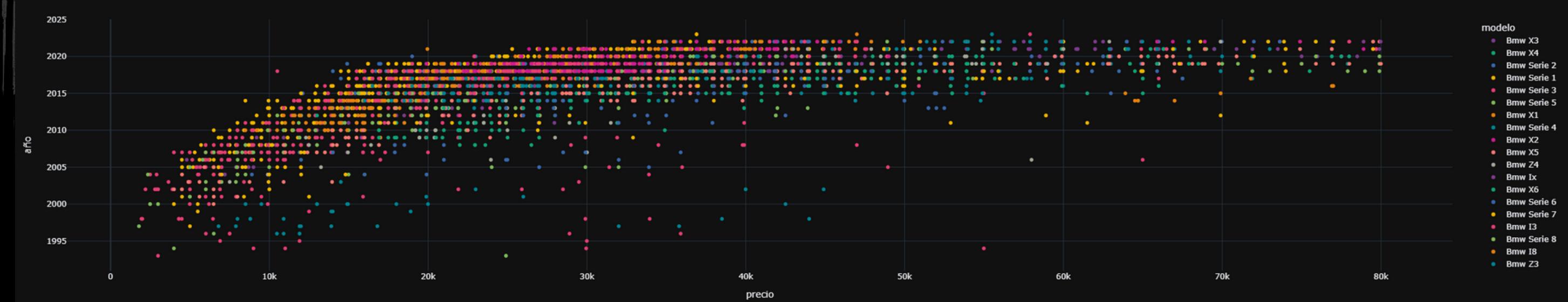
**Peugeot**



**VW**



**BMW**

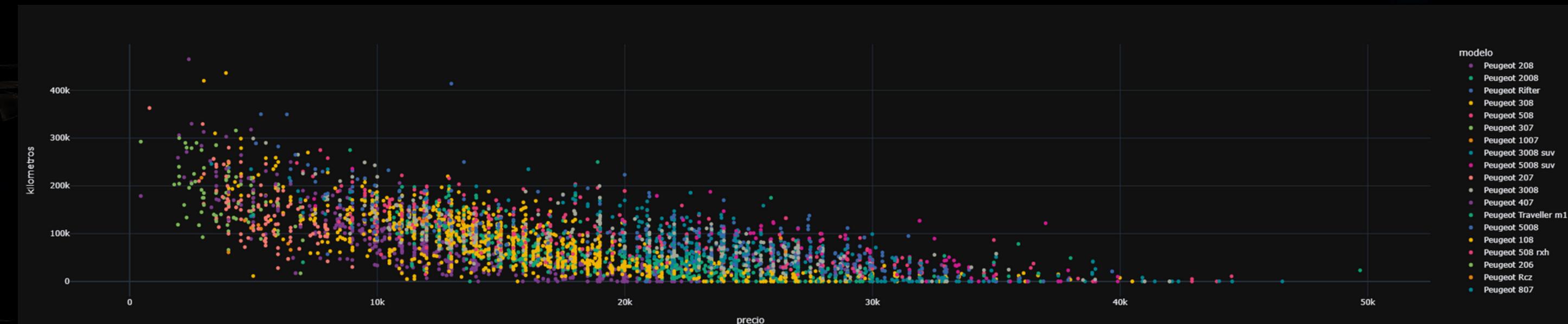


# Explorando los datos precio/kilometros

Precio = 80.000€

Kilometros = 500.000km

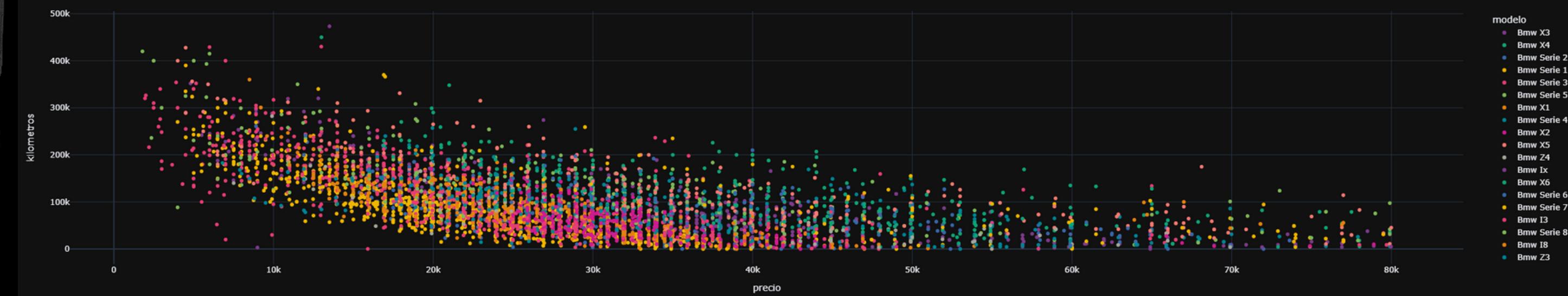
Peugeot



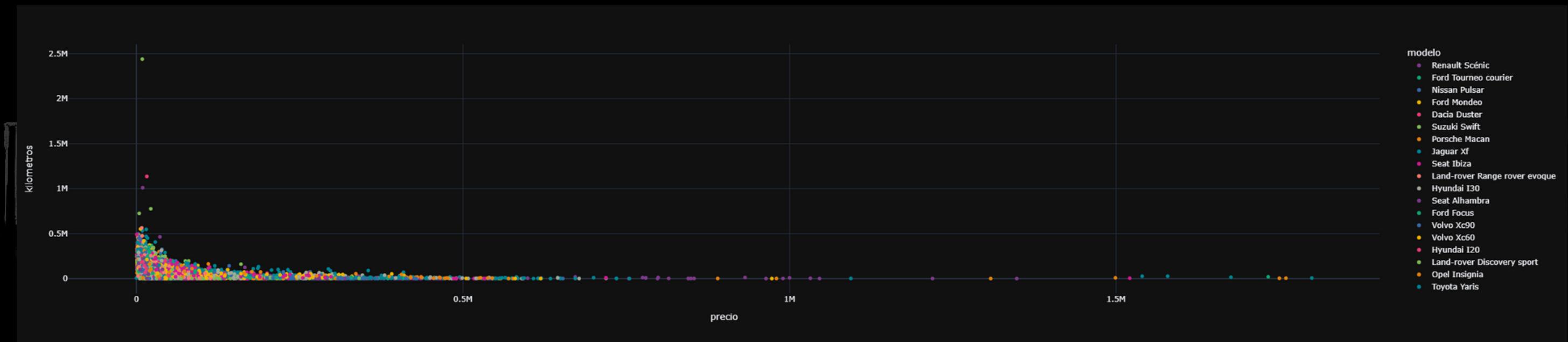
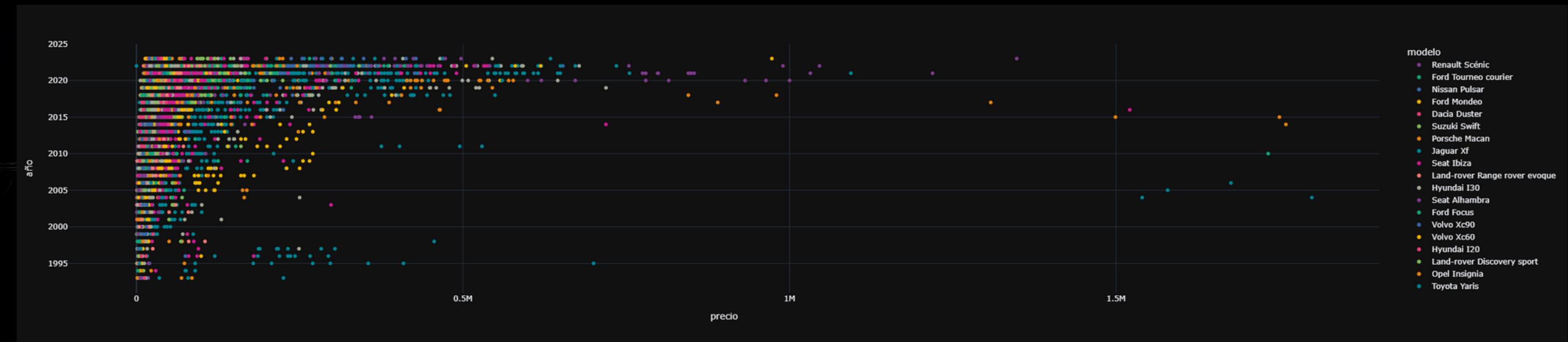
VW



BMW



# Resto de coches



**Resto de coches**

**Precio = 80.000€**

**Kilometros = 500.000km**



# Conclusiones

**Limitar las filas donde los **kilómetros** sea <500.000km**

**Limitar las filas donde el **precio** sea <80.000€**

**Eliminar las **marcas** con menos de 10 coches**

**Eliminar los **modelos** con menos de 5 coches**

**Eliminar **columnas** innecesarias**

# MACHINE LEARNING

## DataFrames

**Completo**

**limit (kilometros/precio)**

**less\_limit(marcas)**

**less\_columns**

**less\_columns\_models(modelos)**

# Clase Prepare

```

...
# Preparador de DataFrames
def prepare_dataframes(self, df_selector, input_dataframe, df_dropping):
    ...
    # Sección de tipo de dataframe a preparar
    if df_selector == 'complete':
        print(f'Mostrando total filas y columnas:{input_dataframe.shape}', '\n')
        # Función que quita columnas
        dropping_data = self.dropping(input_dataframe, df_dropping)
        return dropping_data
    elif df_selector == 'limit':
        # Filtra los líneas del dataframe
        input_dataframe = self.limit(input_dataframe)
        print(f'Mostrando total filas y columnas:{input_dataframe.shape}', '\n')
        # Elimina las columnas del dataframe
        dropping = self.dropping(input_dataframe, df_dropping)
        return dropping
    elif df_selector == 'less_limit':
        # Filtra los líneas del dataframe
        input_dataframe = self.limit(input_dataframe)
        # Filtra las marcas con cantidad de coches inferior a 10
        input_dataframe = self.limit_less(input_dataframe)
        # Elimino las columnas
        dropping_data = self.dropping(input_dataframe, df_dropping)
        return dropping_data
    elif df_selector == 'less_columns':
        # Filtra las líneas del dataframe
        input_dataframe = self.limit(input_dataframe)
        # Filtra las marcas con cantidad de coches inferior a 10
        input_dataframe = self.limit_less(input_dataframe)
        # Elimino las columnas
        dropping_data = self.dropping(input_dataframe, df_dropping)
        # Elimino columnas extras
        input_dataframe = self.reducer(dropping_data, df_dropping)
        print(f'Mostrando reducción del total filas y columnas:{input_dataframe.shape}\n')
        return input_dataframe
    elif df_selector == 'less_models':
        # Filtra los líneas del dataframe
        input_dataframe = self.limit(input_dataframe)
        # Filtra los modelos con cantidad de coches inferior a 5
        input_dataframe = self.limit_less_model(input_dataframe)
        # Elimino las columnas
        dropping_data = self.dropping(input_dataframe, df_dropping)
        # Elimino columnas extras
        input_dataframe = self.reducer(dropping_data, df_dropping)
        print(f'Mostrando reducción del total filas y columnas:{input_dataframe.shape}\n')
        return input_dataframe
    # Métodos para eliminar columnas
    def dropping(self, input_dataframe, df_dropping):
        ...
        if df_dropping == 'tensorflow':
            dropping = input_dataframe.drop(['id_car', 'car_brand_id', 'model_id',
                                             'fuel_id', 'gearbox_id',
                                             'localidad_id'], axis = 1)
            print('Listado de columnas despues del dropping: (dropping.columns)')
            return dropping
        elif df_dropping == 'sklearn':
            dropping = input_dataframe.drop(['id_car', 'locate', 'car_brand', 'model',
                                             'fuel', 'gearbox'], axis = 1)
            print('Listado de columnas despues del dropping: (dropping.columns)')
            return dropping
        # Método para eliminar columnas extras.
        def reducer(self, input_dataframe, df_dropping):
            ...
            if df_dropping == 'tensorflow':
                print('Reduciendo columnas...', '\n')
                dropping = input_dataframe.drop(['locate', 'engine_power',
                                                 'CO2', 'places', 'deposit',
                                                 '0_100km'], axis = 1)
                print('Columnas reducidas', '\n')
                print('Listado de columnas despues del dropping: (dropping.columns)', '\n')
            return dropping
        elif df_dropping == 'sklearn':
            print('Reduciendo columnas...', '\n')
            dropping = input_dataframe.drop(['engine_power', 'CO2', 'places',
                                             'deposit', '0_100km', 'localidad_id'], axis = 1)
            print('Columnas reducidas', '\n')
            print('Listado de columnas despues del dropping: (dropping.columns)', '\n')
        return dropping
    # Métodos para quitar filas
    def limit(self, input_dataframe):
        ...
        input_dataframe = input_dataframe[(input_dataframe['price'] >= 80000) & (input_dataframe['km'] < 500000)]
        print(f'Mostrando el precio y los kilometros máximos:{input_dataframe[['price', 'km']].max()}', '\n')
        return input_dataframe
    # Método para seleccionar las marcas con cantidad de coches inferior a 10
    def limit_less(self, input_dataframe):
        ...
        # Contidad de marcas
        marks = input_dataframe['car_brand'].value_counts()
        ...
        # Guardo las marcas que tienen menos de coches
        cars_ten = marks[marks.values < 10]
        print(f'Coches de marcas con cantidad < 10: {sum(cars_ten.values)}', '\n')
        ...
        # Aplico la mascara anterior al dataframe con ~ delante indica que los valores True son False y viceversa para el isn()
        input_dataframe = input_dataframe[~input_dataframe['car_brand'].isin(cars_ten.index)]
        print(f'Mostrando total filas y columnas:{input_dataframe.shape}', '\n')
        return input_dataframe
    def limit_less_model(self, input_dataframe):
        ...
        # Contidad de modelos
        models = input_dataframe['model'].value_counts()
        ...
        # Guardo las marcas que tienen menos de coches
        model_five = models[model.values < 5]
        print(f'Modelos con cantidad < que 5: {sum(model_five.values)}', '\n')
        ...
        # Aplico la mascara anterior al dataframe con ~ delante indica que los valores True son False y viceversa para el isn()
        input_dataframe = input_dataframe[~input_dataframe['model'].isin(model_five.index)]
        print(f'Mostrando total filas y columnas:{input_dataframe.shape}', '\n')
        return input_dataframe

```

## Metodo preparador

```
1 # Preparador de DataFrames
2 def prepare_dataframes(self, df_selector, input_dataframe, df_dropping):
3
4     # Sección de tipo de dataframe a preparar
5     if df_selector == 'complete':
6         print(f'Mostrando total filas y columnas:{\n{input_dataframe.shape}\n}')
7         # Función que quita columnas
8         dropping_data = self.dropping(input_dataframe, df_dropping)
9
10    return dropping_data
11
12 elif df_selector == 'limit':
13     # Filtra las líneas del dataframe
14     input_dataframe = self.limit(input_dataframe)
15     print(f'Mostrando total filas y columnas:{\n{input_dataframe.shape}\n}')
16     # Elimina las columnas del dataframe
17     dropping = self.dropping(input_dataframe, df_dropping)
18
19    return dropping
20
21 elif df_selector == 'less_limit':
22     # Filtra las líneas del dataframe
23     input_dataframe = self.limit(input_dataframe)
24     # Filtra las marcas con cantidad de coches inferior a 10
25     input_dataframe = self.limit_less(input_dataframe)
26     # Elimino las columnas
27     dropping_data = self.dropping(input_dataframe, df_dropping)
28
29    return dropping_data
30
31 elif df_selector == 'less_columns':
32     # Filtra las líneas del dataframe
33     input_dataframe = self.limit(input_dataframe)
34     # Filtra las marcas con cantidad de coches inferior a 10
35     input_dataframe = self.limit_less(input_dataframe)
36     # Elimino las columnas
37     dropping_data = self.dropping(input_dataframe, df_dropping)
38     # Elimino columnas extras
39     input_dataframe = self.reducer(dropping_data, df_dropping)
40     print(f'Mostrando reducción del total filas y columnas:{\n{input_dataframe.shape}\n}')
41
42    return input_dataframe
43
44 elif df_selector == 'less_columns_models':
45     # Filtra las líneas del dataframe
46     input_dataframe = self.limit(input_dataframe)
47     # Filtra las marcas con cantidad de coches inferior a 10
48     input_dataframe = self.limit_less(input_dataframe)
49     # Filtra los modelos con cantidad de coches inferior a 5
50     input_dataframe = self.limit_less_model(input_dataframe)
51     # Elimino las columnas
52     dropping_data = self.dropping(input_dataframe, df_dropping)
53     # Elimino columnas extras
54     input_dataframe = self.reducer(dropping_data, df_dropping)
55     print(f'Mostrando reducción del total filas y columnas:{\n{input_dataframe.shape}\n}')
56
57    return input_dataframe
```

## Metodos de eliminación

```
68 # Metodos para eliminar columnas
69 def dropping(self, input_dataframe, df_dropping):
70
71     if df_dropping == 'tensorflow':
72         dropping = input_dataframe.drop(['id_car', 'card_brand_id', 'model_id',
73                                         'fuel_id', 'gearbox_id',
74                                         'localidad_id'], axis = 1)
75         print(f'Listado de columnas despues del dropping: {dropping.columns}')
76         return dropping
77
78     elif df_dropping == 'sklearn':
79         dropping = input_dataframe.drop(['id_car', 'locate', 'car_brand', 'model',
80                                         'fuel', 'gearbox'], axis = 1)
81         print(f'Listado de columnas despues del dropping: {dropping.columns}')
82         return dropping
83
84 # Metodo para eliminar columnas extras.
85 def reducir(self, input_dataframe, df_dropping):
86
87     if df_dropping == 'tensorflow':
88         print('Reduciendo columnas...', '\n')
89         dropping = input_dataframe.drop(['locate', 'engine_power',
90                                         'CO2', 'places', 'deposit',
91                                         '0_100km'], axis = 1)
92         print('Columnas reducidas', '\n')
93         print(f'Listado de columnas despues del dropping: {dropping.columns}', '\n')
94
95         return dropping
96
97     elif df_dropping == 'sklearn':
98         print('Reduciendo columnas...', '\n')
99         dropping = input_dataframe.drop(['engine_power','CO2', 'places',
100                                         'deposit','0_100km', 'localidad_id'], axis = 1)
101        print('Columnas reducidas', '\n')
102        print(f'Listado de columnas despues del dropping: {dropping.columns}', '\n')
103
104        return dropping
```

## Metodos de limitación

```
96 # Metodos para quitar filas
97 def limit(self, input_dataframe):
98
99     input_dataframe = input_dataframe[(input_dataframe['price'] <= 80000) & (input_dataframe['km'] < 500000)]
100    print(f'Mostrando el precio y los kilometros maximos:\n{input_dataframe[['price', 'km']].max()}', '\n')
101
102    return input_dataframe
103
104 # Metodo para seleccionar Las marcas con cantidad de coches inferior a 10
105 def limit_less(self, input_dataframe):
106
107    # Cantidad de marcas
108    marks = input_dataframe['car_brand'].value_counts()
109
110    # Guardo Las marcas que tienen menos de coches
111    cars_ten = marks[marks.values < 10]
112    print(f'Coches de marcas con cantidad < que 10: {sum(cars_ten.values)}', '\n')
113
114    # Aplico La mascara anterior al dataframe con ~ delante indico que Los valores True son False y viceversa para el isin()
115    input_dataframe = input_dataframe[~input_dataframe['car_brand'].isin(cars_ten.index)]
116    print(f'Mostrando total filas y columnas:\n{input_dataframe.shape}', '\n')
117
118    return input_dataframe
119
120 def limit_less_model(self, input_dataframe):
121
122    # Cantidad de modelos
123    models = input_dataframe['model'].value_counts()
124
125    # Guardo Las marcas que tienen menos de coches
126    model_five = models[model.values < 5]
127    print(f'Modelos con cantidad < que 5: {sum(model_five.values)}', '\n')
128
129    # Aplico La mascara anterior al dataframe con ~ delante indico que Los valores True son False y viceversa para el isin()
130    input_dataframe = input_dataframe[~input_dataframe['model'].isin(model_five.index)]
131    print(f'Mostrando total filas y columnas:\n{input_dataframe.shape}', '\n')
132
133    return input_dataframe
```

## Creación de los dataframe



```
1 df_selector = ['complete', 'limit', 'less_limit', 'less_columns', 'less_columns_models']
2 df_dropping = ['tensorflow', 'sklearn']
```



```
1 # Bucles para ejecutar la clase y almacenar los resultados en el diccionario del bloque anterior.
2 for i in range(len(df_selector)):
3     for x in range(len(df_dropping)):
4
5         if x == 0:
6             cars_tf = P.prepare_dataframes(df_selector[i], cars, df_dropping[x])
7         elif x == 1:
8             cars_sk = P.prepare_dataframes(df_selector[i], cars, df_dropping[x])
9
10    dict_dataframes['tensorflow'][df_selector[i]] = cars_tf
11    dict_dataframes['sklearn'][df_selector[i]] = cars_sk
```

# Comprobaciones Dataframe TensorFlow

## dict\_dataframes['tensorflow']['less\_columns\_models']

```
less_columns_models tensorflow

Mostrando el precio y los kilometros maximos:
price      80000.0
km        496000.0
dtype: float64

Coches de marcas con cantidad < que 10: 60

Mostrando total filas y columnas:
(176492, 22)

Modelos con cantidad < que 5: 285

Mostrando total filas y columnas:
(176207, 22)

Listado de columnas despues del dropping: Index(['locate', 'car_brand', 'model', 'year', 'horses', 'engine_power', 'km',
       'fuel', 'gearbox', 'CO2', 'price', 'places', 'deposit', '0_100km',
       'displ_engine', 'marches'],
      dtype='object')
Reduciendo columnas...

Columnas reducidas

Listado de columnas despues del dropping: Index(['car_brand', 'model', 'year', 'horses', 'km', 'fuel', 'gearbox',
       'price', 'displ_engine', 'marches'],
      dtype='object')

Mostrando reducción del total filas y columnas:
(176207, 10)
```

# Comprobaciones Dataframe Sklearn

## dict\_dataframes['sklearn']['less\_columns\_models']

```
less_columns_models sklearn

Mostrando el precio y los kilometros maximos:
price      80000.0
km        496000.0
dtype: float64

Coches de marcas con cantidad < que 10: 60

Mostrando total filas y columnas:
(176492, 22)

Modelos con cantidad < que 5: 285

Mostrando total filas y columnas:
(176207, 22)

Listado de columnas despues del dropping: Index(['year', 'horses', 'engine_power', 'km', 'CO2', 'price', 'places',
       'deposit', '0_100km', 'displ_engine', 'marches', 'car_brand_id',
       'model_id', 'fuel_id', 'gearbox_id', 'localidad_id'],
      dtype='object')
Reduciendo columnas...

Columnas reducidas

Listado de columnas despues del dropping: Index(['year', 'horses', 'km', 'price', 'displ_engine', 'marches',
       'car_brand_id', 'model_id', 'fuel_id', 'gearbox_id'],
      dtype='object')

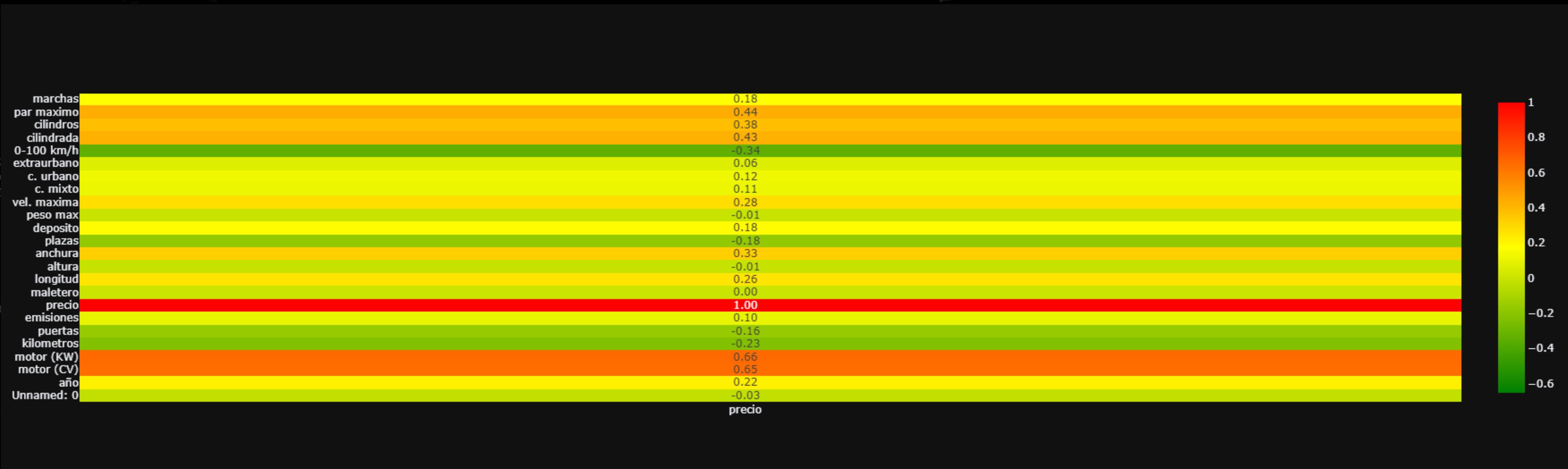
Mostrando reducción del total filas y columnas:
(176207, 10)
```

# Elección del algoritmo

**Arboles de decisión**

# Arboles de decisión

## ¿Por que?



**Modelos TensorFlow**

**GradientBoosted**

**RandomForest**

**Modelos Sklearn**

**RandomForestRegressor**

## DataFrame 'Complete'

```
36/36 [=====] - 1s 14ms/step
Error cuadrático medio: 12093.21
Coeficiente de determinación para 'complete': 0.86
10890.00    12174.65    1284.65
23490.00    21817.99    1672.01
37800.00    35141.57    2658.43
23500.00    22326.94    1173.06
12430.00    14483.66    2053.66
9020.00     11132.07    2112.07
23890.00    26358.06    2468.06
11299.00    12262.75    963.75
31000.00    28375.96    2624.04
53000.00    54085.22    1085.22
```

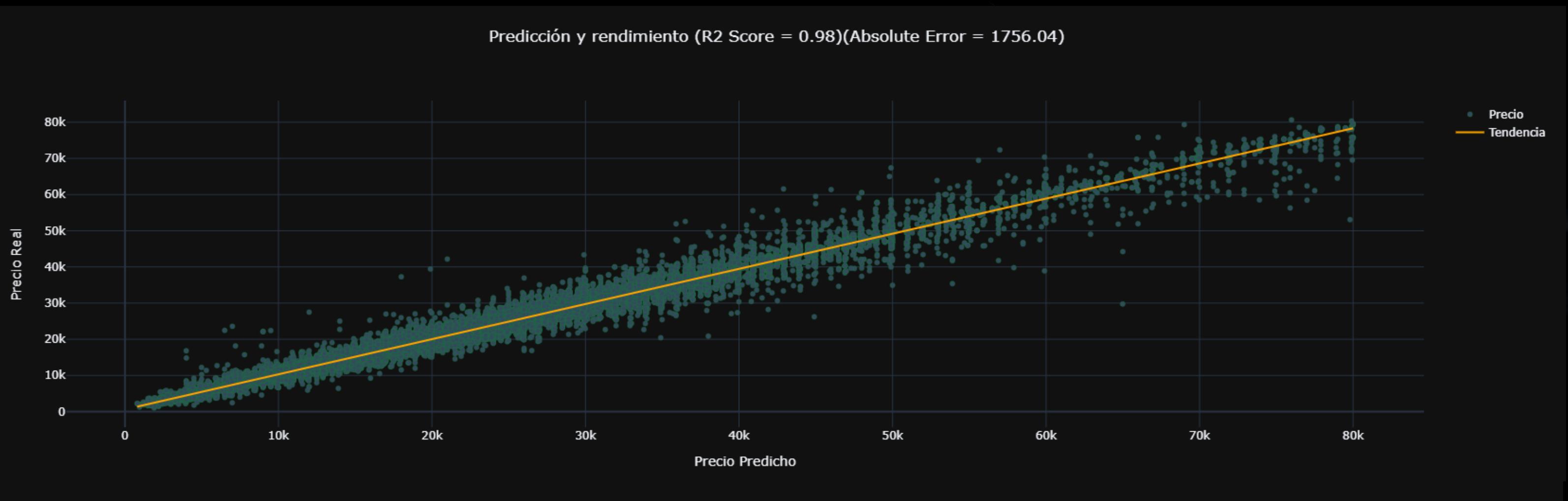
Predicción y rendimiento (R2 Score = 0.86)(Absolute Error = 12093.21)



## DataFrame 'less\_columns\_models'

Error cuadrático medio: 1756.04
Coeficiente de determinación para 'less_columns_models': 0.98
38850.00 40085.57 1235.57
3490.00 2614.31 875.69
34490.00 36892.32 2402.32
41900.00 37487.47 4412.53
18390.00 18178.66 211.34
39000.00 38705.52 294.48
26900.00 27806.16 906.16
20995.00 19156.58 1838.42
21990.00 23512.63 1522.63
28900.00 28593.37 306.63

Predicción y rendimiento (R2 Score = 0.98)(Absolute Error = 1756.04)



# Clase Mlearn

```
1  class Mlearn():
2
3      # Metodo para entrenar un Gradient Boosted
4      def tensorflow_bg(self, input_dataframe, train):
5
6          # Definición del modelo
7          model_car = tfdf.keras.GradientBoostedTreesModel(
8              task=tfdf.keras.Task.REGRESSION,
9              validation_ratio = 0.1,
10             num_trees = 100,
11             max_depth = 10,
12             l1_regularization = 0.01,
13             l2_regularization = 0.01,
14             early_stopping = 'LOSS_INCREASE',
15             loss = "SQUARED_ERROR")
16
17      # Entrenamiento del modelo
18      model_car.fit(train)
19
20      return model_car
21
22      # Metodo para entrenar un Gradient Boosted
23      def tensorflow_rf(self, input_dataframe, train):
24
25          # Definición del modelo
26          model_car = tfdf.keras.RandomForestModel(
27              task=tfdf.keras.Task.REGRESSION,
28              num_trees = 100,
29              max_depth = 10,
30              split_axis = "AXIS_ALIGNED",
31              categorical_algorithm = "CART")
32
33      # Entrenamiento del modelo
34      model_car.fit(train)
35
36      return model_car
```

# Entrenamiento

```
● ● ●  
1 dict_tr_ts = {}  
2  
3 for index, df in enumerate(dict_dataframes['tensorflow']):  
4  
5     data_frame = dict_dataframes['tensorflow'][df]  
6  
7     # Separo los datos de entrenamiento (train) y los de test (test)  
8     train, test = train_test_split(data_frame, test_size=0.2, random_state=42)  
9  
10    # Especificar la columna de la variable de destino (o target)  
11    target = "price"  
12  
13    # Convertir el DataFrame de pandas a un conjunto de datos de TensorFlow  
14    train_cars = tfdf.keras.pd_dataframe_to_tf_dataset(train, label=target, task=tfdf.keras.Task.REGRESSION)  
15    test_cars = tfdf.keras.pd_dataframe_to_tf_dataset(test, label=target, task=tfdf.keras.Task.REGRESSION)  
16  
17    # Añado el test_cars al diccionario para poder usarlo después  
18    dict_tr_ts.update({df:[test, test_cars]})  
19  
20    # Entreno el modelo GradientBoosted  
21    model_car_bg = ML.tensorflow_bg(data_frame, train_cars)  
22  
23    # Guardo el modelo GradientBoosted en Drive  
24    model_car_bg.save(f'/content/drive/MyDrive/Colab Notebooks/Curso_IA_BIGDATA/TFM/export_models/Gredient_Boosted_TensorFlow/model_bg_{df}')  
25  
26    # Entreno el modelo RandomForest  
27    model_car_rf = ML.tensorflow_rf(data_frame, train_cars)  
28  
29    # Guardo el modelo RandomForest en Drive  
30    model_car_rf.save(f'/content/drive/MyDrive/Colab Notebooks/Curso_IA_BIGDATA/TFM/export_models/Random_forest_TensorFlow/model_rf_{df}')
```

# Generador de rendimiento y predicciones

```
● ● ●  
1 def rendimiento_tensorflow(name_model ,input_model, input_test, imput_test_df):  
2     predicted_values = input_model.predict(imput_test_df)  
3     predicted_values = predicted_values.flatten()  
4     real_values = input_test[target].to_numpy()  
5  
6     # Error Cuadrático  
7     print("Error cuadrático medio: %.2f" % mean_squared_error(real_values, predicted_values, squared=False))  
8     # Coeficiente de determinación  
9     print(f"Coeficiente de determinación para '{name_model}': %.2f" % r2_score(real_values, predicted_values))  
10  
11    mse = mean_squared_error(real_values, predicted_values, squared=False)  
12    r2 = r2_score(real_values, predicted_values)  
13  
14    for i in range(10):  
15        real_price = real_values[i]  
16        estim_price = predicted_values[i]  
17        error_abs = abs(real_price - estim_price)  
18        print(f"real_price:{6.2f} estim_price:{12.2f} error_abs:{16.2f}")  
19  
20    grafic_predicction(real_values, predicted_values, r2, mse)
```

# DataFrame 'Complete'

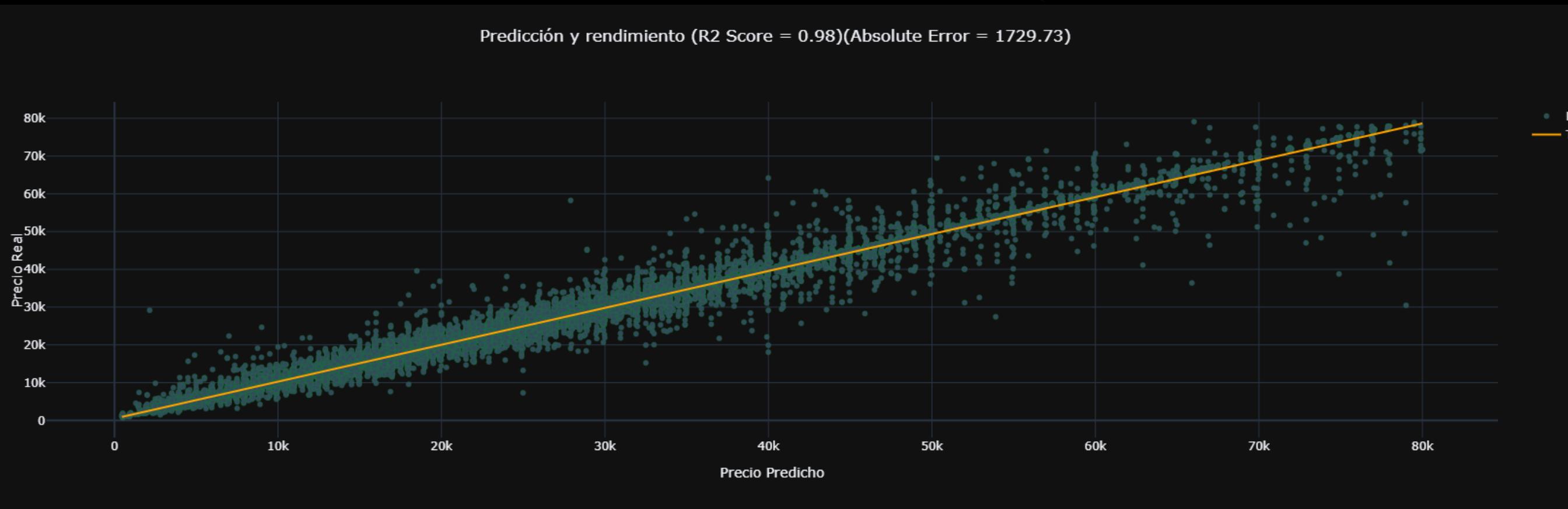
```
Error cuadrático medio: 11500.59
Coeficiente de determinación para 'complete': 0.87
Sacando predicciones
20680.00    20680.00      0.00
31000.00    31198.64    198.64
37990.00    38864.09    874.09
21199.00    21482.06    283.06
14650.00    15479.75    829.75
5500.00     5938.15     438.15
9990.00     9990.00      0.00
19990.00    20041.30    51.30
14835.00    14835.00      0.00
25900.00    26118.90    218.90
```



## DataFrame 'less\_columns\_models'

```
Error cuadrático medio: 1729.73
Coeficiente de determinación para 'less_columns_models': 0.98
Sacando predicciones
30900.00      31581.30      681.30
20930.00      20930.00          0.00
9190.00       9550.60      360.60
15900.00      15900.00          0.00
25990.00      25575.94      414.06
64990.00      70436.91      5446.91
21950.00      21933.30      16.70
17193.00      17193.00          0.00
19990.00      19990.00          0.00
19995.00      19868.51      126.49
```

Predicción y rendimiento (R2 Score = 0.98)(Absolute Error = 1729.73)



# Entrenamiento

```
● ● ●  
1 dict_sklearn = {}  
2  
3 for index, df in enumerate(dict_dataframes['sklearn']):  
4     X = dict_dataframes['sklearn'][df].drop(['price'], axis = 1)  
5     y = dict_dataframes['sklearn'][df]['price']  
6  
7     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)  
8  
9     dict_sklearn.update({df:[X_test, y_test]})  
10  
11 rf_model = RandomForestRegressor().fit(X_train.values, y_train.values)  
12 print(f'Proceso de entrenamiento completado de {df}')  
13  
14 joblib.dump(rf_model, f'/content/drive/MyDrive/Colab Notebooks/Curso_IA_BIGDATA/TFM/export_models/Sklearn_RF/Sklearn_RF_{df}.pkl', compress=9)  
15 print(f'Proceso de exportación de Sklearn_RF_{df} completado')
```

# Generador de rendimiento y predicciones



```
1 def rendimiento_sklearn(name_model ,input_model, input_test, imput_test_df):
2     # Predicción con Las entradas de test
3     predicted_values = input_model.predict(input_test.values)
4     # Saco Los valores reales de test
5     real_values = imput_test_df.values
6
7     # Error Cuadrático
8     print("Error cuadrático medio: %.2f" % mean_squared_error(real_values, predicted_values, squared=False))
9     # Coeficiente de determinación
10    print(f"Coeficiente de determinación para '{name_model}': %.2f" % r2_score(real_values, predicted_values))
11
12    # Saco predicciones
13    print('Sacando predicciones')
14    for i in range(10):
15        real_price = real_values[i]
16        estim_price = predicted_values[i]
17        error_abs = abs(real_price - estim_price)
18        print(f'{real_price:6.2f} {estim_price:12.2f} {error_abs:16.2f}')
19    print('\n')
20
21    # r2_score y mean_squared_error nos permite saber el rendimiento del modelo
22    r2 = r2_score(real_values, predicted_values)
23    mse = mean_squared_error(real_values, predicted_values, squared=False)
24
25    # Dibujo una gráfica con los resultados
26    grafic_predicction(real_values, predicted_values, r2, mse)
```

# Predicción

## Referencia

	year	horses	km	displ_engine	marches	car_brand_id	model_id	fuel_id	gearbox_id
80462	2009.0	150.0	175000.0	1998.0	6.0	57.0	575.0	3.0	1.0
59424	2006.0	175.0	114378.0	1998.0	5.0	57.0	575.0	3.0	1.0
24172	2006.0	150.0	200000.0	1910.0	6.0	57.0	575.0	1.0	1.0
30789	2006.0	175.0	114378.0	1998.0	5.0	57.0	575.0	3.0	1.0
80645	2009.0	180.0	82000.0	1910.0	6.0	57.0	575.0	1.0	1.0
82213	2000.0	150.0	160656.0	1985.0	5.0	57.0	576.0	3.0	1.0
89967	2006.0	175.0	114378.0	1998.0	5.0	57.0	575.0	3.0	1.0
66875	2010.0	180.0	140000.0	1910.0	6.0	57.0	575.0	1.0	1.0

	car_brand	model	year	horses	km	fuel	gearbox	price	displ_engine	marches
173184	Saab	Saab 9-3	2007.0	150.0	270000.0	Diesel	Manual	3500.0	1910.0	6.0
59424	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
24172	Saab	Saab 9-3	2006.0	150.0	200000.0	Diesel	Manual	4600.0	1910.0	6.0
116728	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
148358	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
11764	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
112518	Saab	Saab 9-5	1999.0	150.0	149000.0	Gasolina	Manual	2990.0	1985.0	5.0
113438	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0

## Predicción



```
1 new_car = rf_sk_less_columns.predict([[2005.0, 150.0, 190000.0, 1910.0, 6.0, 57.0, 575.0, 1, 1]])  
2 new_car
```



```
1 new_car = rf_sk_less_columns.predict([[2005.0, 150.0, 190000.0, 1910.0, 6.0, 57.0, 575.0, 1, 1]])
2 new_car
```

array([5637.63])

## Resultado

array([5637.63])

## Referencia

	car_brand	model	year	horses	km	fuel	gearbox	price	displ_engine	marches
173184	Saab	Saab 9-3	2007.0	150.0	270000.0	Diesel	Manual	3500.0	1910.0	6.0
59424	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
24172	Saab	Saab 9-3	2006.0	150.0	200000.0	Diesel	Manual	4600.0	1910.0	6.0
116728	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
148358	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
11764	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0
112518	Saab	Saab 9-5	1999.0	150.0	149000.0	Gasolina	Manual	2990.0	1985.0	5.0
113438	Saab	Saab 9-3	2006.0	175.0	114378.0	Gasolina	Manual	5900.0	1998.0	5.0

# Tabla comparativa

Algoritmos de Machine Learning	MSE	R2_Score	Casos
<b>TensorFlow - GradientBoostedTreesModel</b>			
model_bg_complete	12093.21	0.86	179502
model_bg_limit	1861.69	0.97	176552
model_bg_less_limit	1801.79	0.97	176492
model_bg_less_columns	1827.12	0.97	176492
model_bg_less_columns_models	1756.04	0.98	176207
<b>TensorFlow - RandomForestModel</b>			
model_rf_complete	12003.32	0.86	179502
model_rf_limit	2823.97	0.94	176552
model_rf_less_limit	2813.75	0.94	176492
model_rf_less_columns	2792.48	0.94	176492
model_rf_less_columns_models	2731.08	0.94	176207
<b>SkLearn - RandomForestRegressor</b>			
rf_sk_complete	11500.59	0.87	179502
rf_sk_limit	1753.63	0.98	176552
rf_sk_less_limit	1725.31	0.98	176492
rf_sk_less_columns	1831.87	0.97	176492
rf_sk_less_columns_models	1729.73	0.98	176207

NPL

EINSTEIN (CHATBOT)



# ¿QUE QUEREMOS DE NUESTRO MODELO?

**Sacar el contexto**

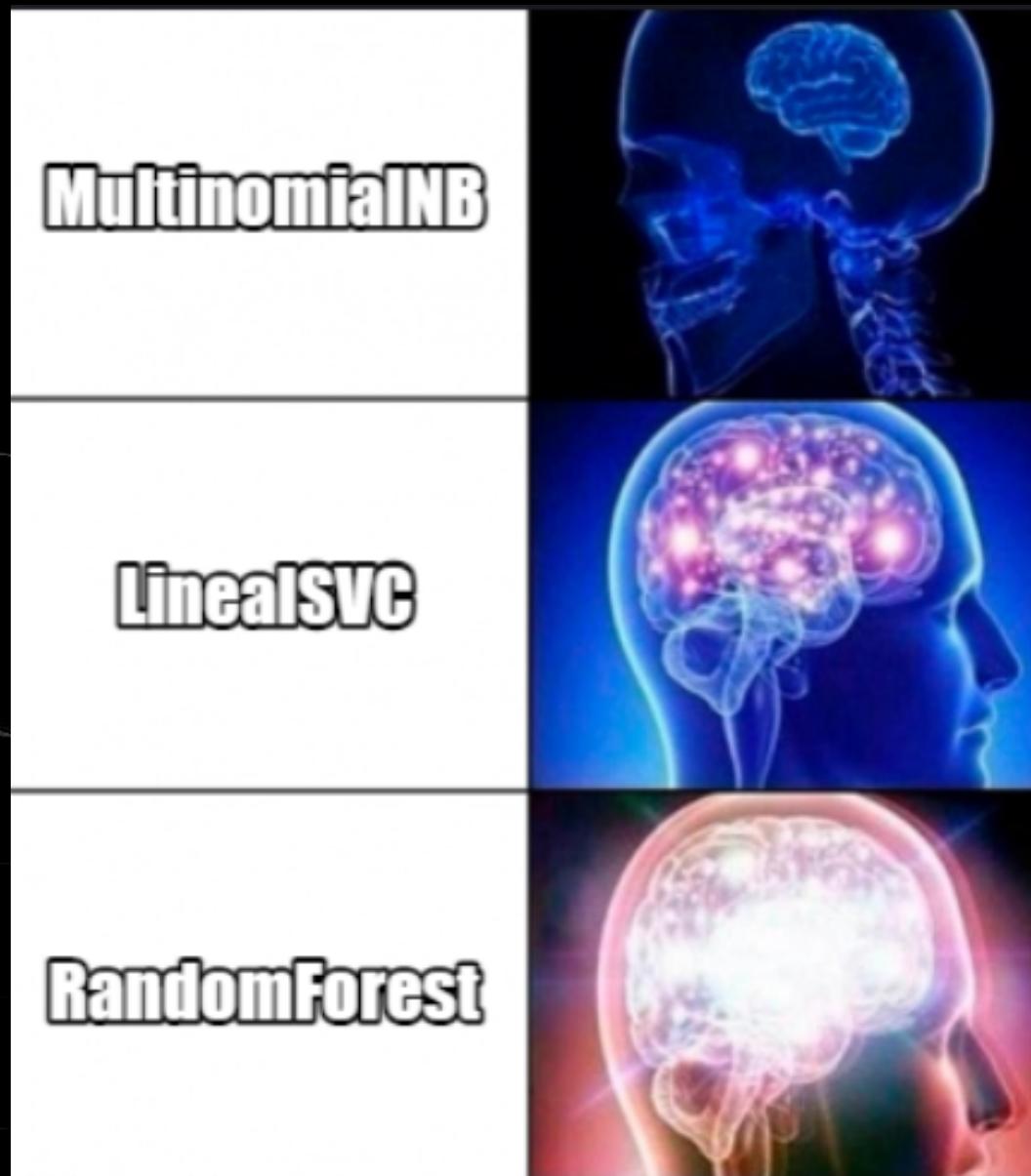


# ¿COMO LO ENTRENAMOS?

**Un modelo NLP cuando recibe  
inputs sin tokenizar**



# ¿MODELOS ESCOGIDOS?





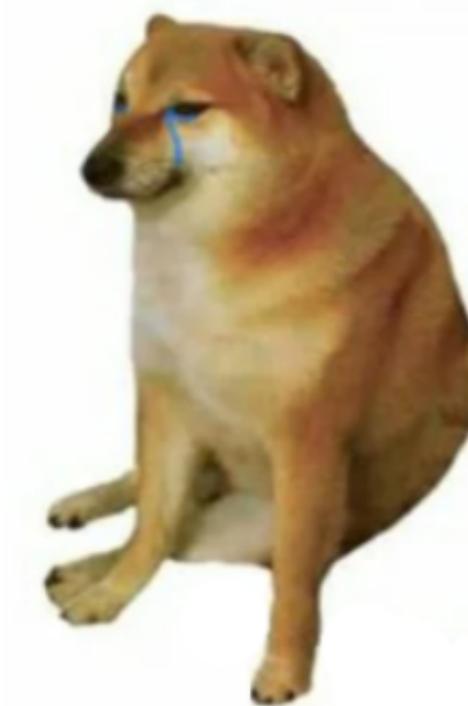
# CONCLUSIONES DE NUESTRO CHATBOT

Eistein cuando su petición está relacionada con coches



Te daré los coches que quieras mi rey,  
aunque hayas puesto una biblia

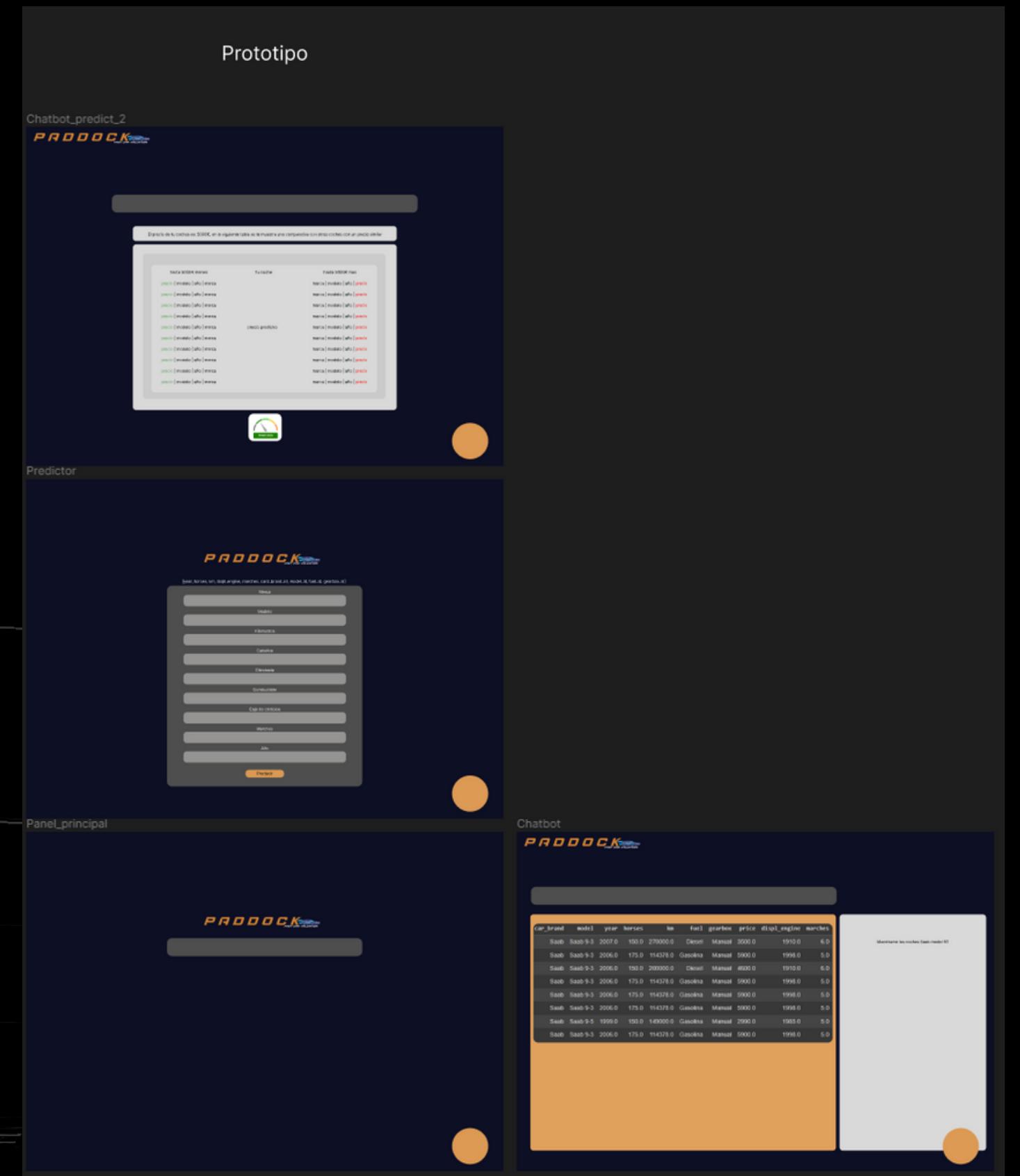
Eistein cuando le hablas de  
política, películas o  
cualquier cosa off topic



Noooooo. No entiendo esos  
mensajes me dan ansiedad

# DESARRALLO WEB





# DEMO

**¿QUE POTENCIAL TIENE NUESTRO  
PROYECTO?**



# MUCHAS GRACIAS



[github.com/Legodark/paddock](https://github.com/Legodark/paddock)