Open in Colab

CODERHOUSE

DATA SCIENCE I - Fundamentos (comision 60935)

- Profesor: Jorge RUIZ
- Tutor: Diego GASCH
- Alumno: Fernando MARGARIT

DATASET: Vehiculos usados en EEUU disponibles para la venta.

OBJETIVO 1: Poder visualmente conocer un precio de venta en caso de contar con un vehiculo usado, considerando Marca, Modelo, Año de Fabricacion, Kilometraje

OBJETIVO 2: Utilizar el dataset para ML y pronosticar su precio de venta

1. Grafico 1: conocer la cantidad de autos disponibles en el mercado por año de fabricacion y el precio promedio
2. Grafico 2: distribucion de precios en cada categoria, vizualizando rangos y outliers
3. Grafico 3: Volumen de vehiculos por las principales Marcas en el mercado de usados
4. Grafico 4: distribucion de precios en cada marca, vizualizando rangos y outliers

Campos del dataset:

vin	Numero de chasis
body_type	Categoria del vehiculo
daysonmarket	Dias en el mercado para la venta
fleet	Fue vehiculo de flota?
frame_damaged	Esta dañado?
fuel_type	Tipo de combustible
has_accidents	Tuvo accidentes registrados?
horsepower	Potencia en caballos de fuerza
isCab	Es taxi?
make_name	Fabricante
maximum_seating	Cantidad de asientos
mileage	Kilometraje
model_name	Nombre del modelo
price	Precio en USD
transmission	Tipo de Transmision (Manual / Aut / etc.)
wheel_system	Tipo de traccion
year	Fecha de Fabricacion

HIPOTESIS

1. La mayor cantidad de vehiculos usados disponibles para la venta tienen una antiguedad mayor a 5 años
2. Los vehiculos de la marca con mayor cantidad de vehiculos ofrecidos en el mercado tienen mas dias promedio en el mercado antes de venderse
3. Los vehiculos de la marca con mayor cantidad de vehiculos ofrecidos en el mercado tienen la mayor dispersion de precios
4. Las categorias con mayor cantidad de vehiculos ofrecidos en el mercado tienen la mayor dispersion de precios
5. Vehiculos con motor a Gasolina son los mas ofrecidos
6. Cuanto mayor es el kilometraje del vehiculo ofrecio, menor su precio
7. Cuanto mas nuevo es el vehiculo usado, mayor es su precio
8. Los vehiculos pueden permanecer mas de 100 dias en promedio en el mercado hasta venderse

Conclusiones al final

```
# Importar librerias
import pandas  as pd
import seaborn as sns
import numpy   as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
# Importar archivo csv (guardado en GitHub)
from google.colab import drive
import os
git  = 'https://raw.githubusercontent.com/fmargarit/CoderHouse_DataScience/main/US%20USED%20CARS%20FOR%20SALES.csv'

df = pd.read_csv(git)
```

```
# conocer la cantidad de filas y columnas del DataFrame
df.shape
```

```
(10000, 17)
```

```
# cantidad de registros con campos NaN
df.isna().sum()
```

```
vin                  0
body_type           36
daysonmarket         0
fleet             3327
frame_damaged     3327
fuel_type          269
has_accidents     3327
horsepower         557
isCab             3327
make_name            0
maximum_seating    517
mileage            308
model_name           0
price                0
transmission       163
wheel_system       465
year                 0
dtype: int64
```

```
# se considera que en los casos de fleet, frame_damaged, has_accidents y isCab el valor NaN corresponde a False

df['fleet'].fillna(False, inplace=True)
df['frame_damaged'].fillna(False, inplace=True)
df['has_accidents'].fillna(False, inplace=True)
df['isCab'].fillna(False, inplace=True)
df.isna().sum()
```

```
vin                  0
body_type           36
```

```
daysonmarket         0
fleet                0
frame_damaged        0
fuel_type          269
has_accidents        0
horsepower         557
isCab                0
make_name            0
maximum_seating    517
mileage            308
model_name           0
price                0
transmission       163
wheel_system       465
year                 0
dtype: int64
```

```
#Borrado de registros NaN

#df['body_type'].dropna(inplace=True)
df.dropna(inplace=True)
```

```
df.shape
```

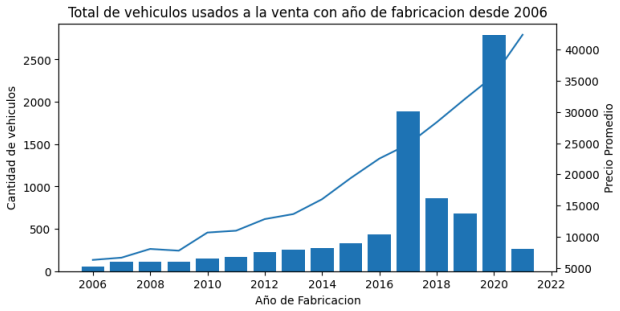
```
(8891, 17)
```

```
df.isna().sum()
```

```
vin               0
body_type         0
daysonmarket      0
fleet             0
frame_damaged     0
fuel_type         0
has_accidents     0
horsepower        0
isCab             0
make_name         0
maximum_seating   0
mileage           0
model_name        0
price             0
transmission      0
wheel_system      0
year              0
dtype: int64
```

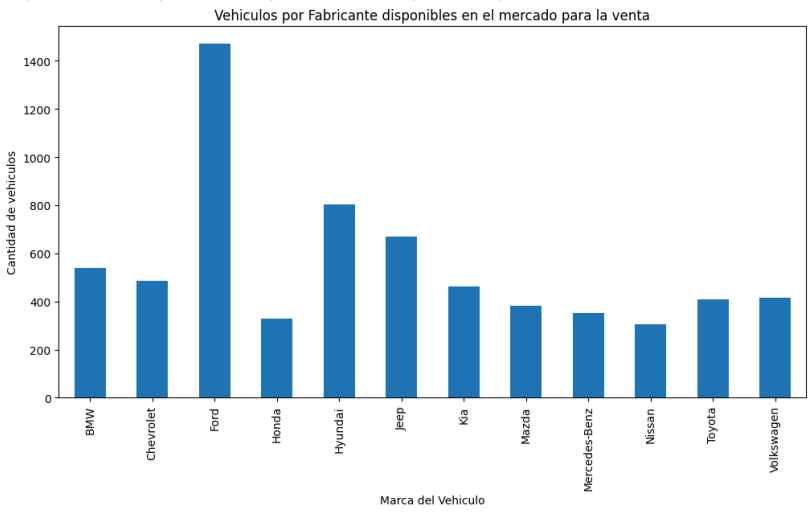
```
# GRAFICO 1
# Total de vehiculos usados a la venta segun su año de fabricacion incluyendo precio promedio
ventas = ventas = df[df['year'] > 2005].groupby('year').agg({'vin':'count', 'price':'mean'})
ventas.columns = ['Cantidad', 'Precio_Promedio']

fig, ax1 = plt.subplots(figsize=(8,4))
ax1.bar(ventas.index, ventas['Cantidad'])
ax1.set_title('Total de vehiculos usados a la venta con año de fabricacion desde 2006')
ax1.set_ylabel('Cantidad de vehiculos')
ax1.set_xlabel('Año de Fabricacion')

ax2 = ax1.twinx()
ax2.plot(ventas.index, ventas['Precio_Promedio'])
ax2.set_ylabel('Precio Promedio')
```

```
Text(0, 0.5, 'Precio Promedio')
```



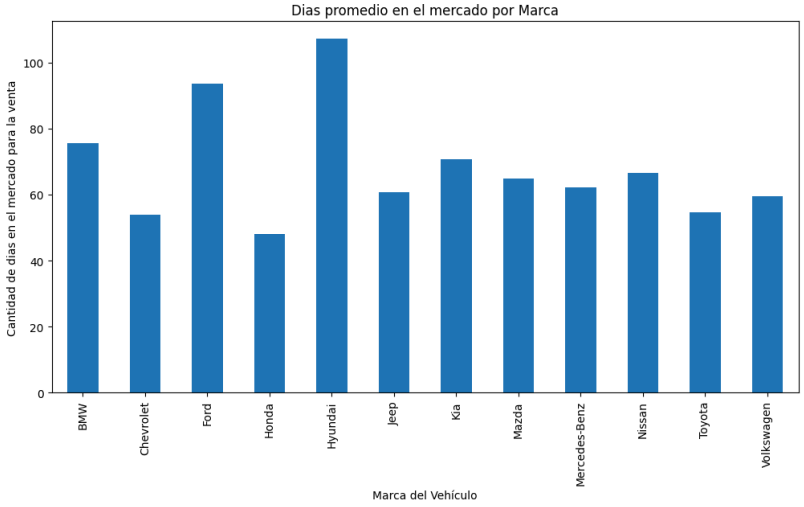

```
# GRAFICO 2
# Total de vehiculos usados a la venta segun categoria
cant = df.groupby(['make_name']).agg({'vin':'count'})
cant = cant[cant['vin'] > 300]

cant['vin'].plot(kind='bar', figsize=(12,6))
plt.xlabel('Marca del Vehiculo')
plt.ylabel('Cantidad de vehiculos')
plt.title('Vehiculos por Fabricante disponibles en el mercado para la venta')
```

```
Text(0.5, 1.0, 'Vehiculos por Fabricante disponibles en el mercado para la venta')
```



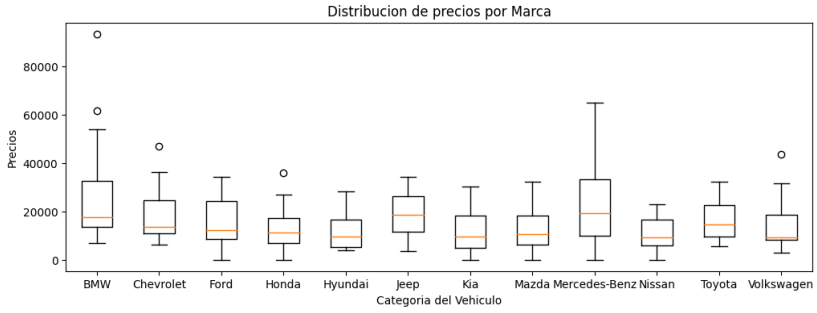
```
# GRAFICO 3
# Dias promedio en el mercado para venta por Marca
dias = df.groupby(['make_name']).agg({'vin':'count','daysonmarket':'mean'})
dias = dias[dias['vin'] > 300]


dias['daysonmarket'].plot(kind='bar', figsize=(12,6))
plt.xlabel('Marca del Vehículo')
plt.ylabel('Cantidad de dias en el mercado para la venta')
plt.title('Dias promedio en el mercado por Marca')
```

```
Text(0.5, 1.0, 'Dias promedio en el mercado por Marca')
```

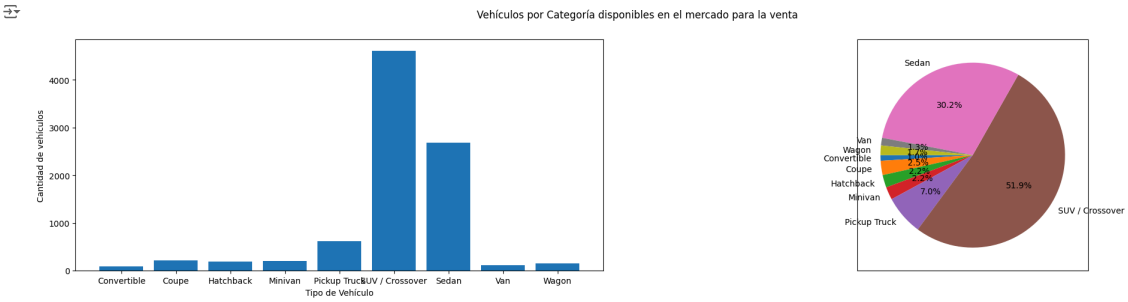



```
# GRAFICO 4
# Precios por marca de vehiculo

# Agrupar por Marca y cantidad
vta_x_marca = df.groupby('make_name')['vin'].count()

# Filtrar las marcas que tienen más de 300 vehículos en venta
vta_flitro = vta_x_marca[vta_x_marca > 300].index

# Filtrar el DataFrame original para incluir solo las marcas filtradas
df_vta = df[df['make_name'].isin(vta_flitro)]

# Crear la tabla dinámica
marca = df_vta[df_vta['year'] > 2005].pivot_table(values='price', index='year', columns='make_name', aggfunc='mean')

# Poner en cero los NaN
marca.fillna(0, inplace=True)

fig, ax = plt.subplots(figsize=(12,4))
ax.boxplot(marca , labels=vta_flitro)
ax.set_title('Distribucion de precios por Marca')
ax.set_ylabel('Precios')
ax.set_xlabel('Categoria del Vehiculo')
```

```
Text(0.5, 0, 'Categoria del Vehiculo')
```

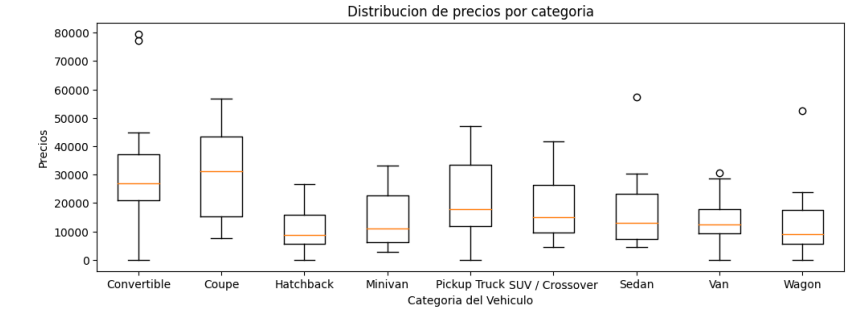



```
# GRAFICO 5
# Total de vehiculos usados a la venta segun categoria
fig, axs = plt.subplots(nrows=1, ncols=2)

cantidad = df.groupby(['body_type']).agg({'vin':'count'})

axs[0].bar(cantidad.index, cantidad['vin'])
axs[0].figure.set_size_inches(40, 5)
axs[0].set_xlabel('Tipo de Vehículo')
axs[0].set_ylabel('Cantidad de vehículos')

axs[1].pie(cantidad['vin'], labels=cantidad.index, autopct='%1.1f%%', startangle=180, )
axs[1].set_frame_on(True)
axs[1].figure.set_size_inches(25, 5)

fig.suptitle("Vehículos por Categoría disponibles en el mercado para la venta")
plt.show()
```

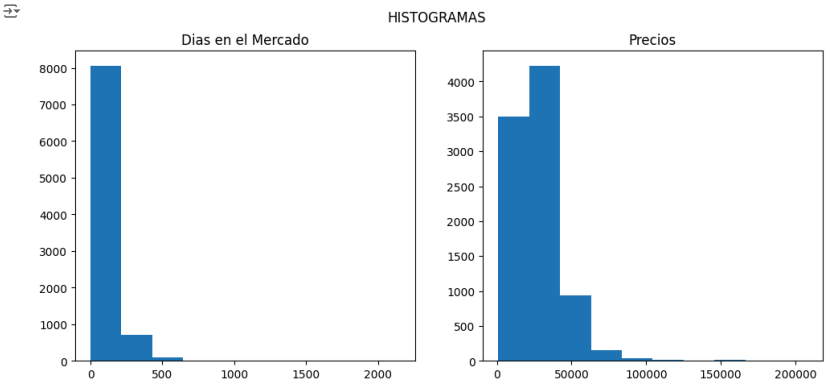



```
# GRAFICO 6
# Precios por categoria de vehiculo
categoria = df[df['year'] > 2005].pivot_table(values='price', index='year', columns='body_type', aggfunc='mean')
categoria.fillna(0, inplace=True)


fig, ax = plt.subplots(figsize=(12,4))
ax.boxplot(categoria, labels=categoria.columns)
ax.set_title('Distribucion de precios por categoria')
ax.set_ylabel('Precios')
ax.set_xlabel('Categoria del Vehiculo')
```

```
Text(0.5, 0, 'Categoria del Vehiculo')
```

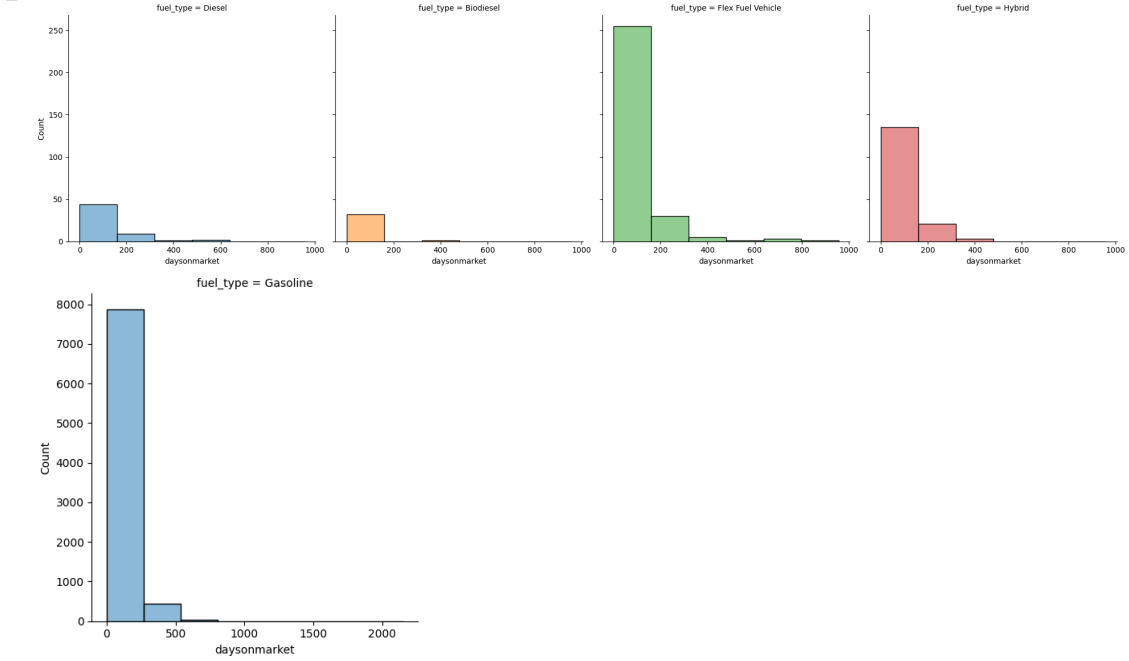



```
# GRAFICO 7
# Histogramas

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12,5))

axs[0].hist(data=df, x='daysonmarket',  bins=10)
axs[0].set_title("Dias en el Mercado")

axs[1].hist(data=df, x='price', bins=10)
axs[1].set_title("Precios")

fig.suptitle("HISTOGRAMAS")
plt.show()
```

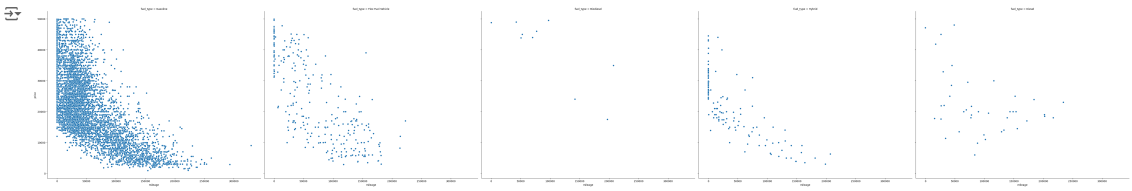



```
# GRAFICO 8
#Histogramas por tipo de Combustible
df_sin_nafta = df[df['fuel_type'] != 'Gasoline']
df_nafta = df[df['fuel_type'] == 'Gasoline']

sns.displot(data=df_sin_nafta, x='daysonmarket', hue='fuel_type', col='fuel_type', legend=False, bins=6)
sns.displot(data=df_nafta, x='daysonmarket', hue='fuel_type', col='fuel_type', legend=False, bins=8)
```

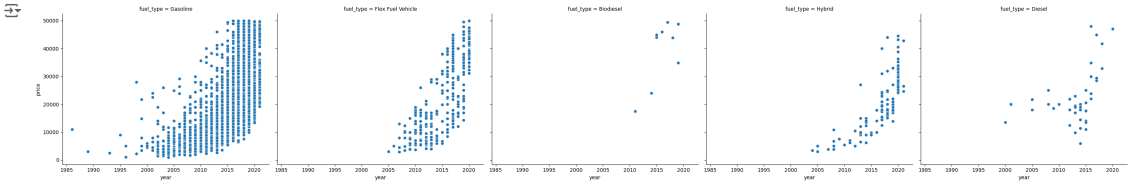
```
<seaborn.axisgrid.FacetGrid at 0x782accb259c0>
```




```
# GRAFICO 9
#SCATTERPLOT por tipo de Combustible - relacion Precio/Kilometraje

dfp = df[df['price'] < 50000]

g = sns.FacetGrid(dfp, col='fuel_type', height=10, aspect=1.2)
g.map(sns.scatterplot, 'mileage', 'price')
plt.show()
```



```
# GRAFICO 10
#SCATTERPLOT por tipo de Combustible - relacion Precio/Fecha Fabricacion

dfp = df[df['price'] < 50000]

g = sns.FacetGrid(dfp, col='fuel_type', height=5, aspect=1.2)
g.map(sns.scatterplot, 'year', 'price')
plt.show()
```



CONCLUSIONES

1. La mayor cantidad de vehiculos usados disponibles para la venta tienen una antiguedad mayor a 5 años

FALSO - la mayor cantidad de vehiculos ofrecidos es posterior a 2017 (Grafico 1)

2. Los vehiculos de la marca con mayor cantidad de vehiculos ofrecidos en el mercado tienen mas dias promedio en el mercado antes de venderse

FALSO - Ford es la marca con mas vehiculos disponibles a la venta pero los vehiculos Hyundai esta mas tiempo para venderse(Grafico 2 y 3)

3. Los vehiculos de la marca con mayor cantidad de vehiculos ofrecidos en el mercado tienen la mayor dispersion de precios

FALSO - Ford es la marca con mas vehiculos disponibles a la venta pero los vehiculos de Mercedes-Benz tienen los precios de venta mas dispersos (Grafico 2 y 4)

4. Las categorias con mayor cantidad de vehiculos ofrecidos en el mercado tienen la mayor dispersion de precios

FALSO - La categoria SUV-Crossover es la mas vendida con 52% del mercado pero no es la que tiene mas dispersion de precios. Convertibles y Coupes, con volumenes inferiores, tienen mas amplitud en sus precios (Grafico 5 y 6)

5. Vehiculos con motor a Gasolina son los mas ofrecidos

VERDADERO (Grafico 8)

6. Cuanto mayor es el kilometraje del vehiculo ofrecio, menor su precio

VERDADERO (Grafico 9)

7. Cuanto mas nuevo es el vehiculo ofrecio, mayor su precio

VERDADERO (Grafico 10)

8. Los vehiculos pueden permanecer mas de 100 dias en promedio en el mercado hasta venderse

VERDADERO (Grafico 8)

FIN - Entrega I

ENTREGA FINAL

OBJETIVO:

Utilizar el modelo de regresion lineal para calcular el valor de mi vehiculo (target) FORD SUV con 17553km y fabricado en el año 2018

```
# Cargar nuevamente el DATASET (guardado en GitHub)
from google.colab import drive
import os
git  = 'https://raw.githubusercontent.com/fmargarit/CoderHouse_DataScience/main/US%20USED%20CARS%20FOR%20SALES.csv'

df = pd.read_csv(git)
```

```
df.shape
```

```
(10000, 17)
```

```
df
```

	vin	body_type	daysonmarket	fleet	frame_damaged	fuel_type	has_accidents	horsepower	isCab	make_name	maximum_seating	mileage	model_name	price	tra
0	ZACNJABB5KPJ92081	SUV / Crossover	522	NaN	NaN	Gasoline	NaN	177.0	NaN	Jeep	5 seats	7.0	Renegade	23141	
1	SALCJ2FX1LH858117	SUV / Crossover	207	NaN	NaN	Gasoline	NaN	246.0	NaN	Land Rover	7 seats	8.0	Discovery Sport	46500	
2	JF1VA2M67G9829723	Sedan	1233	False	False	Gasoline	False	305.0	False	Subaru	5 seats	NaN	WRX STI	46995	
3	SALRR2RV0L2433391	SUV / Crossover	196	NaN	NaN	Gasoline	NaN	340.0	NaN	Land Rover	7 seats	11.0	Discovery	67430	
4	SALCJ2FXXLH862327	SUV / Crossover	137	NaN	NaN	Gasoline	NaN	246.0	NaN	Land Rover	7 seats	7.0	Discovery Sport	48880	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	3VWJX7AT6CM607391	Hatchback	21	False	False	Gasoline	True	170.0	False	Volkswagen	4 seats	124118.0	Beetle	5900	
9996	YV4102RL2M1685225	SUV / Crossover	41	NaN	NaN	Gasoline	NaN	250.0	NaN	Volvo	5 seats	0.0	XC60	55310	
9997	1FMSK8DH9LGC58378	SUV /	54	NaN	NaN	Gasoline	NaN	300.0	NaN	Ford	7 seats	20.0	Explorer	40596	

Next steps: Generate code with `df` View recommended plots

```
#seleccionar los datos necesarios para la regresion
df = df[ ['make_name', 'body_type', 'mileage', 'year', 'price'] ]
```

```
#filtrar por el modelo seleccionado excluyendo outliers de precios
df_modelo = df[ (df['body_type'] == 'SUV / Crossover') & (df['price'] <= 100000) & (df['make_name']== 'Ford')]
```

```
#renombrar nombre de las columnas
df_modelo = df_modelo.rename(columns = {'make_name':'Fabricante','body_type':'Modelo','mileage':'Kilometraje','year':'Año','price':'Precio'})
```

```
df_modelo.isna().sum()
```

```
Fabricante     0
Modelo         0
Kilometraje    5
Año            0
Precio         0
dtype: int64
```

```
# Se elimina los registros NaN por ser poca cantidad, solo 5
df_modelo.dropna(inplace=True)
```

```
df_modelo.isna().sum()
```

```
Fabricante     0
Modelo         0
Kilometraje    0
Año            0
Precio         0
dtype: int64
```

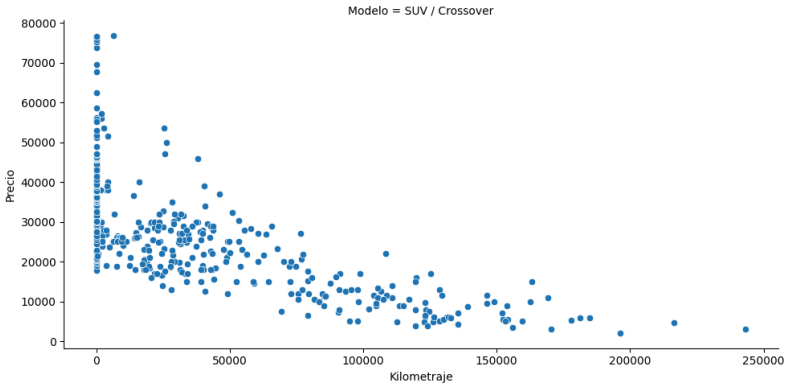
```
df_modelo
```

	Fabricante	Modelo	Kilometraje	Año	Precio
105	Ford	SUV / Crossover	53326.0	2016	25000
433	Ford	SUV / Crossover	3699.0	2020	37993
451	Ford	SUV / Crossover	79364.0	2014	6500
520	Ford	SUV / Crossover	123346.0	2010	7993
587	Ford	SUV / Crossover	177937.0	2010	5250
...	...	...	...	...	...
9978	Ford	SUV / Crossover	12.0	2020	30091
9980	Ford	SUV / Crossover	77222.0	2016	12995
9984	Ford	SUV / Crossover	12.0	2020	30091
9997	Ford	SUV / Crossover	20.0	2020	40596
9998	Ford	SUV / Crossover	10.0	2020	41380

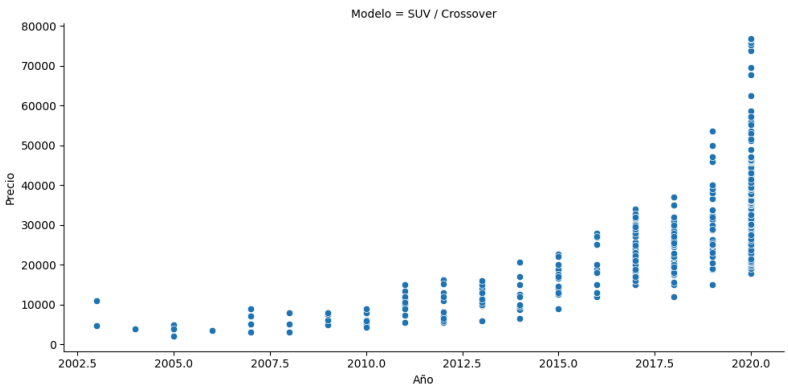
696 rows × 5 columns

Next steps: Generate code with `df_modelo` View recommended plots

```
g = sns.FacetGrid(df_modelo, col='Modelo', height=5, aspect=2)
g.map(sns.scatterplot, 'Kilometraje', 'Precio')
plt.show()
```




```
g = sns.FacetGrid(df_modelo, col='Modelo', height=5, aspect=2)
g.map(sns.scatterplot, 'Año', 'Precio')
plt.show()
```




```
df_modelo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 696 entries, 105 to 9998
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Fabricante   696 non-null    object
 1   Modelo       696 non-null    object
 2   Kilometraje  696 non-null    float64
 3   Año          696 non-null    int64
 4   Precio       696 non-null    int64
dtypes: float64(1), int64(2), object(2)
memory usage: 32.6+ KB
```

```
df_modelo['Modelo'] = df_modelo.Modelo.astype('category')
```

```
df_modelo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 696 entries, 105 to 9998
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Fabricante   696 non-null    object
 1   Modelo       696 non-null    category
 2   Kilometraje  696 non-null    float64
 3   Año          696 non-null    int64
 4   Precio       696 non-null    int64
dtypes: category(1), float64(1), int64(2), object(1)
memory usage: 28.0+ KB
```

```
df_modelo.describe(include = 'category')
```

	Modelo
count	696
unique	1
top	SUV / Crossover
freq	696

```
# Correlaciones
corr_km_precio = df_modelo['Kilometraje'].corr(df_modelo['Precio'], method='pearson')
corr_anio_precio = df_modelo['Año'].corr(df_modelo['Precio'], method='pearson')

print('La correlacion entre el kilometraje y el precio es:', corr_km_precio )
print('La correlacion entre el año de fabricacion y el precio es:', corr_anio_precio )
```

```
La correlacion entre el kilometraje y el precio es: -0.6277903181042908
La correlacion entre el año de fabricacion y el precio es: 0.6390774384739015
```

```
# variables independientes
x1 = df_modelo[['Año']]
x2 = df_modelo[['Kilometraje']]

# variable dependiente
y = df_modelo[['Precio']]

x1_train, x1_Test, y_train, y_Test, = train_test_split(x1, y, test_size = 0.2 , random_state = 0)

x2_train, x2_Test, y_train, y_Test, = train_test_split(x2, y, test_size = 0.2 , random_state = 0)

regressor1 = LinearRegression()
regressor1.fit(x1_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

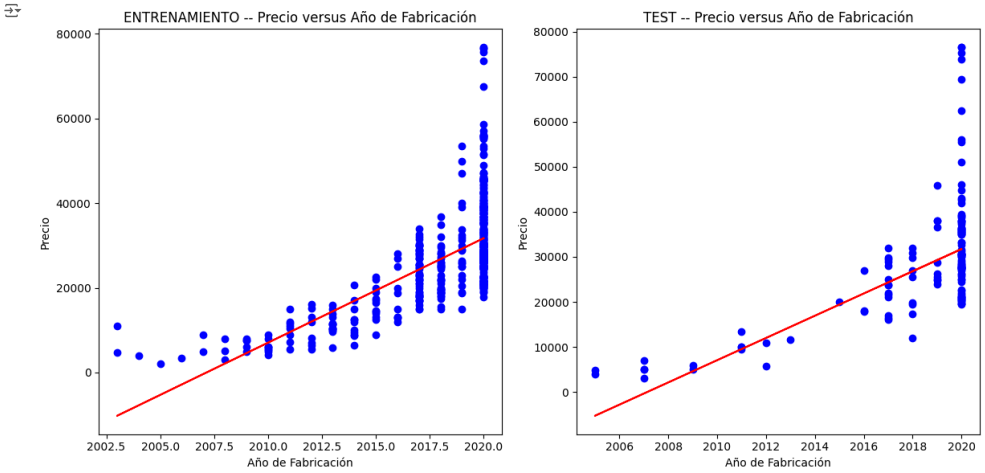
```
regressor2 = LinearRegression()
regressor2.fit(x2_train, y_train)
```

```
LinearRegression()
```

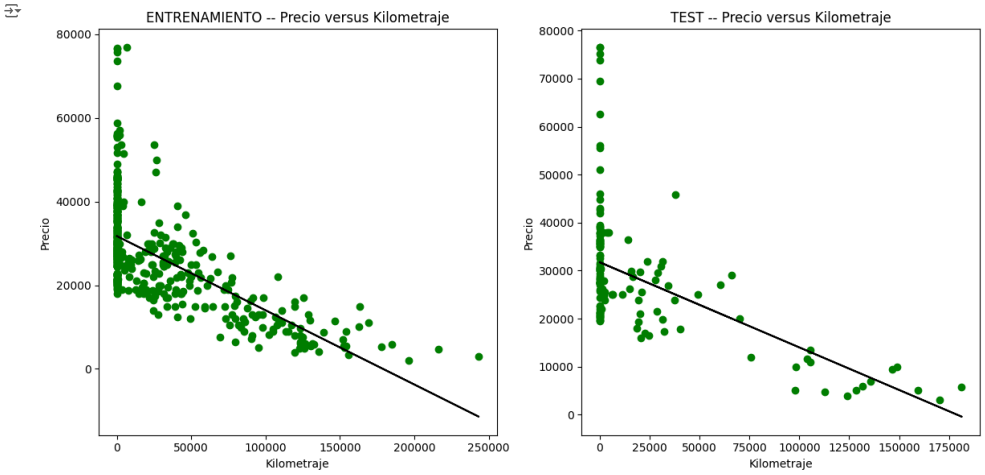
```
# pendiente
pend1 = regressor1.coef_
pend2 = regressor2.coef_

# interseccion
interc1 = regressor1.intercept_
interc2 = regressor2.intercept_

# relacion Año / Precio
print(f"El valor de la pendiente para la relacion Año-Precio es {pend1.round(13)} y de la interseccion es {interc1.round(3)}")

# relacion Kilometraje / Precio
print(f"El valor de la pendiente para la relacion Kilometraje-Precio es {pend2.round(13)} y de la interseccion es {interc2.round(3)}")
```

```
El valor de la pendiente para la relacion Año-Precio es [[2467.29987885]] y de la interseccion es [-4952221.444]
El valor de la pendiente para la relacion Kilometraje-Precio es [[-0.17732348]] y de la interseccion es [31729.925]
```

```
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# gráfico (entregamiento)
axs[0].scatter(x1_train, y_train, color='blue')
axs[0].plot(x1_train, regressor1.predict(x1_train), color='red')
axs[0].set_title('ENTRENAMIENTO -- Precio versus Año de Fabricación')
axs[0].set_ylabel('Precio')
axs[0].set_xlabel('Año de Fabricación')

# gráfico (test)
axs[1].scatter(x1_Test, y_Test, color='blue')
axs[1].plot(x1_Test, regressor1.predict(x1_Test), color='red')
axs[1].set_title('TEST -- Precio versus Año de Fabricación')
axs[1].set_ylabel('Precio')
axs[1].set_xlabel('Año de Fabricación')

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




```
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# gráfico (entregamiento)
axs[0].scatter(x2_train, y_train, color='green')
axs[0].plot(x2_train, regressor2.predict(x2_train), color='black')
axs[0].set_title('ENTRENAMIENTO -- Precio versus Kilometraje')
axs[0].set_ylabel('Precio')
axs[0].set_xlabel('Kilometraje')

# gráfico (test)
axs[1].scatter(x2_Test, y_Test, color='green')
axs[1].plot(x2_Test, regressor2.predict(x2_Test), color='black')
axs[1].set_title('TEST -- Precio versus Kilometraje')
axs[1].set_ylabel('Precio')
axs[1].set_xlabel('Kilometraje')

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




```
y_pred = regressor1.predict(x1_Test)
y_pred
```

```
       [31724.31112022],
       [31724.31112022],
       [31724.31112022],
       [26789.71136253],
       [26789.71136253],
       [ -350.58730478],
       [31724.31112022],
       [29257.01124138],
       [31724.31112022],
       [24322.41148368],
       [31724.31112022],
       [31724.31112022],
       [31724.31112022],
       [24322.41148368],
       [31724.31112022],
       [31724.31112022],
       [26789.71136253],
       [29257.01124138],
       [31724.31112022],
       [31724.31112022],
       [ -350.58730478],
       [31724.31112022],
       [21855.11160484],
       [14453.2119683 ],
       [31724.31112022],
       [31724.31112022],
       [11985.91208945],
       [26789.71136253],
       [31724.31112022],
       [29257.01124138],
       [31724.31112022],
       [29257.01124138],
       [31724.31112022],
       [31724.31112022],
       [ -350.58730478],
       [ 9518.61221061],
       [26789.71136253],
       [31724.31112022],
       [31724.31112022],
       [31724.31112022],
       [29257.01124138],
       [26789.71136253],
       [31724.31112022],
       [31724.31112022],
       [31724.31112022],
       [31724.31112022],
       [ 9518.61221061],
       [31724.31112022]])
```

```
y_Test
```

	Precio
8898	28990
6108	27928
8817	4790
9963	29495
6697	20698
...	...
8820	27440
3341	25750
5682	21540
7584	13400
8002	30724

140 rows × 1 columns

Next steps: Generate code with `y_Test` View recommended plots

```
# R2 o coef de determinacion
coef1 = regressor1.score(x1_train, y_train)
coef2 = regressor2.score(x2_train, y_train)

coef3 = regressor1.score(x1_Test, y_Test)
coef4 = regressor2.score(x2_Test, y_Test)


coef1, coef2, coef3, coef4
```

```
(0.4250762231873161,
 0.4109582689153001,
 0.3523793208377146,
 0.3386312815170367)
```

```
# Input para calcular precio con kilometraje y año de fabricacion
fab = int(input('Ingrese año de fabricacion de su vehiculo: '))
```