TEC DE JUAREZ
Forjando el futuro...

Proyecto Final

Irving Vaquero Flores (18111777)
Joaquin Toto Alvarado (18112358)

Maestro: Mario Macario.

Mater: Análisis inteligente de datos.

Fecha: 13/06/2022

Contenido

Detalles del Data Set

Para iniciar nuestra búsqueda de un tema de interés pudimos observar 3 distintos data set de los cuales nos decantamos por el que cumpliera más con nuestro tema de interés, el nombre de dicho data set es el siguiente "Car_Prices_Poland_Kaggle", dicho Data Set está localizado en el siguiente LINK

El conjunto de datos se reunió en enero de 2022. Datos de un conocido sitio de venta de automóviles en Polonia (que es público), el Data set nos muestra las siguientes columnas.

- mark = Marca del auto.
- Model = Modelo de la marca.
- generation_name = Nombre de la generación.
- year = Año del auto.
- mileage = Millas del auto.
- vol_engine = Volumen del motor.
- fuel = Tipo de combustible.
- city = Ciudad
- province = Provincia.
- Price = Precio.

Las hipótesis sobre lo que queremos lograr con estos datos son las siguiente:

1. El devaluó de los autos según los kilómetros recorridos y marca del vehículo: el porqué de esta idea es debido a que no todos los autos se devalúan de la misma manera aun que se corran de la misma manera durante el mismo tiempo debido a que la confiabilidad de algunas marcas suele ser mayor que otras ya que su trayectoria es mayor que el de algunas otras.
2. Como segunda idea podemos tener que tipo de combustible es utilizado para los autos de viaje ya que por lo regular cuando un auto tiene un alto millaje suelen ser para viajes.
3. En tercer lugar, tenemos que marcar de autos prefieren los ciudadanos de Polonia además de saber qué precios se manejan en cada una de

ellas por ende sabremos que auto son más baratos en cada una de estas ciudades.

Exploración del Data Set

Con la función info() sacamos la información del DataSet, como el tipo de variable de cada columna (Objeto, entero o flotante), la cantidad de registros que tiene y desde donde empieza su numeración, la cantidad de columnas, y por último, la cantidad de registros que no contienen registros nulos

Código

```
print(df.info())
```

Resultado

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Unnamed: 0       117927 non-null  int64
 1   mark             117927 non-null  object
 2   model            117927 non-null  object
 3   generation_name  87842 non-null   object
 4   year             117927 non-null  int64
 5   mileage          117927 non-null  int64
 6   vol_engine       117927 non-null  int64
 7   fuel             117927 non-null  object
 8   city             117927 non-null  object
 9   province         117927 non-null  object
 10  price            117927 non-null  int64
dtypes: int64(5), object(6)
memory usage: 9.9+ MB
None
```

Con la función describe() sacamos los valores básicos de la Media, Desviación estándar, mínimo, máximo, la cantidad de registros y los cuartiles de cada columna numérica.

Código

```
print(df.describe())
```

Resultado

```
None
        Unnamed: 0          year       mileage    vol_engine         price
count  117927.000000  117927.000000  1.179270e+05  117927.000000  1.179270e+05
mean    58963.000000    2012.925259  1.409768e+05    1812.057782  7.029988e+04
std     34042.736935       5.690135  9.236936e+04     643.613438  8.482458e+04
min         0.000000    1945.000000  0.000000e+00       0.000000  5.000000e+02
25%     29481.500000    2009.000000  6.700000e+04    1461.000000  2.100000e+04
50%     58963.000000    2013.000000  1.462690e+05    1796.000000  4.190000e+04
75%     88444.500000    2018.000000  2.030000e+05    1995.000000  8.360000e+04
max    117926.000000    2022.000000  2.800000e+06    7600.000000  2.399900e+06
```

Con esta función mostramos los primeros 5 registro del DataSet

Código

```
print(df.head())
```

Resultado

	Unnamed: 0	mark	model	generation_name	year	mileage	vol_engine	fuel	city	province	price
0	0	opel	combo	gen-d-2011	2015	139568	1248	Diesel	Janki	Mazowieckie	35900
1	1	opel	combo	gen-d-2011	2018	31991	1499	Diesel	Katowice	Śląskie	78501
2	2	opel	combo	gen-d-2011	2015	278437	1598	Diesel	Brzeg	Opolskie	27000
3	3	opel	combo	gen-d-2011	2016	47600	1248	Diesel	Korfantów	Opolskie	30800
4	4	opel	combo	gen-d-2011	2014	103000	1400	CNG	Tarnowskie Góry	Śląskie	35900

Con esta función mostramos los últimos 5 registro del Data Set

Código

```
print(df.tail())
```

Resultado

	Unnamed: 0	mark	model	generation_name	year	mileage	vol_engine	fuel	city	province	price
117922	117922	volvo	xc-90	gen-ii-2014-xc-90	2020	40000	1969	Hybrid	Katowice	Śląskie	222790
117923	117923	volvo	xc-90	gen-ii-2014-xc-90	2017	51000	1969	Diesel	Chechło Pierwsze	Łódzkie	229900
117924	117924	volvo	xc-90	gen-ii-2014-xc-90	2016	83500	1969	Gasoline	Pruszcz Gdański	Pomorskie	135000
117925	117925	volvo	xc-90	gen-ii-2014-xc-90	2017	174000	1969	Diesel	Kalisz	Wielkopolskie	154500
117926	117926	volvo	xc-90	gen-ii-2014-xc-90	2016	189020	1969	Gasoline	Sionna	Mazowieckie	130000

Además, con la función sample(INT) obtenemos la cantidad de datos de ejemplos que indiquemos en nuestro caso unos 10

Código

```
print(df.sample(10))
```

Resultado

	Unnamed: 0	mark	model	generation_name	year	mileage	vol_engine	fuel	city	province	pric e
37549	37549	volkswagen	golf	gen-vi-2008-2013	2009	175000	1968	Diesel	Ostrów Wielkopolski	Wielkopolskie	1990 0
56974	56974	mercedes-benz	glc-klasa	NaN	2021	10	1991	Gasoline	Łódź	Łódzkie	27777 7
27011	27011	bmw	seria-3	gen-e90-2005-2012	2010	219000	1995	Diesel	Bolesławiec	Dolnośląskie	2799 9
93507	93507	hyundai	tucson	gen-iii-2015-2020	2017	140000	1685	Diesel	Nowy Targ	Małopolskie	7290 0
113992	113992	volvo	s60	gen-ii-2010-s60	2012	165000	2521	Gasoline	Międzyrzec Podlaski	Lubelskie	4390 0
39814	39814	volkswagen	passat	gen-b5-1996-2000	2000	386120	2496	Diesel	Ożarów Mazowiecki	Mazowieckie	350 0
23099	23099	audi	q7	NaN	2012	146000	2967	Diesel	Rakoniewice	Wielkopolskie	9500 0
40399	40399	volkswagen	passat	gen-b6-2005-2010	2007	315000	1968	Diesel	Świecie	Kujawsko-pomorskie	1750 0
4932	4932	opel	corsa	gen-d-2006-2014	2010	121584	1598	Gasoline	Alwernia	Małopolskie	2980 0
71866	71866	toyota	c-hr	NaN	2020	20547	1798	Hybrid	Rumia	Pomorskie	10790 0

Pre procesamiento

Para un manejo más fácil de las columnas optamos por cambiar el nombre de la columna "Unnamed: 0" por "ID"

Código

```
df.rename(columns={"Unnamed: 0": "ID"},
                inplace=True)
```

Resultado

```
[25 rows x 10 columns]
                ID           year        mileage     vol_engine         price
count   117927.000000  117927.000000  1.179270e+05  117927.000000  1.179270e+05
mean     58963.000000    2012.925259  1.409768e+05    1812.057782  7.029988e+04
std      34042.736935       5.690135  9.236936e+04     643.613438  8.482458e+04
min          0.000000    1945.000000  0.000000e+00       0.000000  5.000000e+02
25%      29481.500000    2009.000000  6.700000e+04    1461.000000  2.100000e+04
50%      58963.000000    2013.000000  1.462690e+05    1796.000000  4.190000e+04
75%      88444.500000    2018.000000  2.030000e+05    1995.000000  8.360000e+04
max     117926.000000    2022.000000  2.800000e+06    7600.000000  2.399900e+06
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
```

Durante la inspección del Data Set notamos que en la columna "generation_name" se encuentran algunos datos faltantes

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Unnamed: 0       117927 non-null  int64
 1   mark             117927 non-null  object
 2   model            117927 non-null  object
 3   generation_name  87842 non-null   object
 4   year             117927 non-null  int64
 5   mileage          117927 non-null  int64
 6   vol_engine       117927 non-null  int64
 7   fuel             117927 non-null  object
 8   city             117927 non-null  object
 9   province         117927 non-null  object
 10  price            117927 non-null  int64
dtypes: int64(5), object(6)
memory usage: 9.9+ MB
None
```

Utilizamos el siguiente código para verificar y efectivamente en esta columna se encuentra una gran cantidad de datos faltantes

Código

```
print(df.isnull().sum())
```

Resultado

```
Unnamed: 0           0
mark                 0
model                0
generation_name    30085
year                 0
mileage              0
vol_engine           0
fuel                 0
city                 0
province             0
price                0
dtype: int64
```

Para determinar que haremos con dicha columna primero obtenemos 10 registros de ejemplo

Código

```
print(df['generation_name'].sample(10))
```

Resultado

```
114159      gen-ii-2010-s60
33421          gen-g01-2017
5916                    NaN
21719                   NaN
2657        gen-j-2009-2015
32179       gen-g11-12-2015
67420     gen-iii-2008-2016
77788          gen-iii-2013
72797                   NaN
33321          gen-g01-2017
```

Como observamos los datos de dicha columna muestra datos de generaciones de los autos que no son más que agrupación por año lo cual si es que utilizamos para algo este tipo de información lo podemos lograr mediante la columna de 'year', así que por el momento optaremos por eliminar dicha columna,

Código

```
del df["generation_name"]
print(df.isnull().sum())
```

Resultado

```
Name: generation_name, dtype: object
ID             0
mark           0
model          0
year           0
mileage        0
vol_engine     0
fuel           0
city           0
province       0
price          0
dtype: int64
```

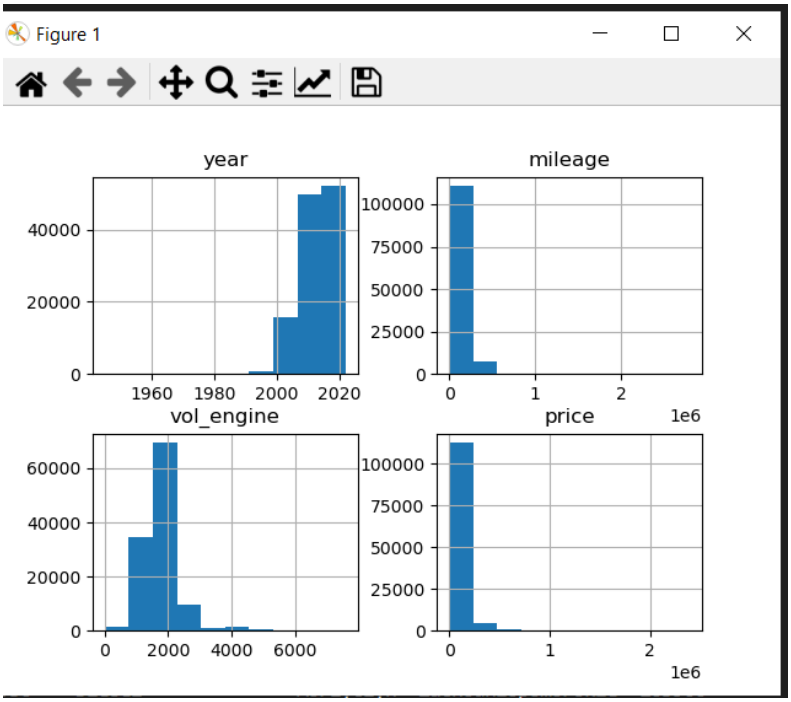
Ahora lo que haremos es visualizar los datos mediante graficas dado que al utilizar describe() notamos que hay datos muy dispersos

	ID	year	mileage	vol_engine	price
count	117927.000000	117927.000000	1.179270e+05	117927.000000	1.179270e+05
mean	58963.000000	2012.925259	1.409768e+05	1812.057782	7.029988e+04
std	34042.736935	5.690135	9.236936e+04	643.613438	8.482458e+04
min	0.000000	1945.000000	0.000000e+00	0.000000	5.000000e+02
25%	29481.500000	2009.000000	6.700000e+04	1461.000000	2.100000e+04
50%	58963.000000	2013.000000	1.462690e+05	1796.000000	4.190000e+04
75%	88444.500000	2018.000000	2.030000e+05	1995.000000	8.360000e+04
max	117926.000000	2022.000000	2.800000e+06	7600.000000	2.399900e+06

Código

```
df.drop(['ID'],1).hist()
plt.show()
```

Resultado

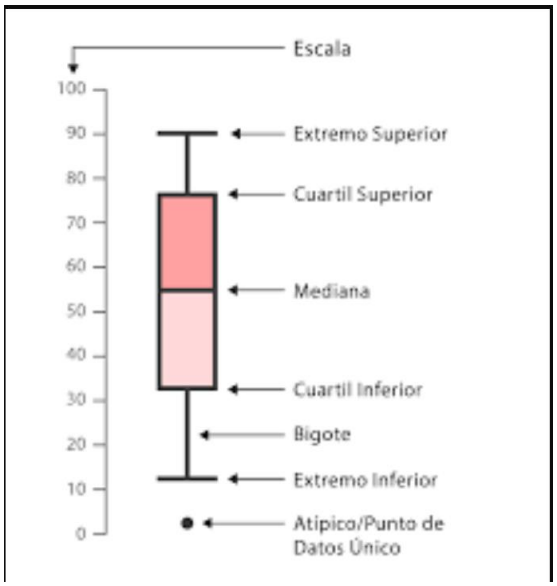


Localizadas las columnas con datos dispersos o bien ya siendo más técnicos datos Outliers – Inliers, asi que de manera manual haremos la eliminación de estos datos

Este proceso consiste en 6 pasos

1. Detectar las columnas donde queremos eliminar datos atípicos

2. Mediante una gráfica de caja y bigote detectamos que tanto de datos están fuera del rango habitual de los demás datos, que según la gráfica todo aquel dato que este fuera de los límites de los cuartiles será considerado un valor atípico



3. Siguiendo la gráfica tendremos que obtener los limites según el primer y tercer cuartil

4. Además, obtenemos el índice IQR (rango intercuartílico) el cual es una medida de variabilidad adecuada cuando la medida de posición central empleada ha sido la mediana. Se define como la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1)

5. Localizamos en el Data Set los datos atípicos

6. Eliminamos estos datos

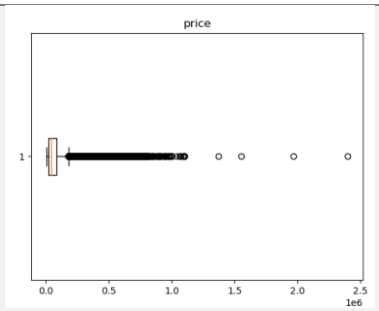
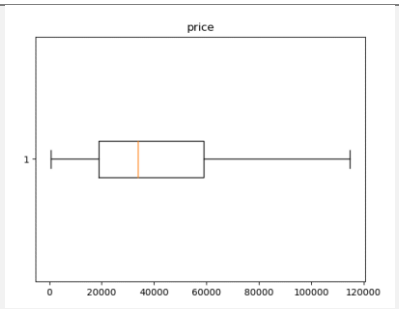
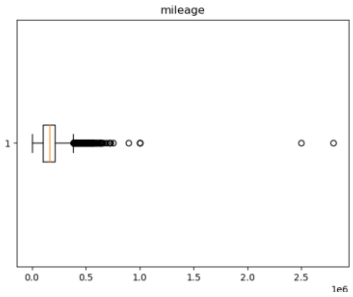
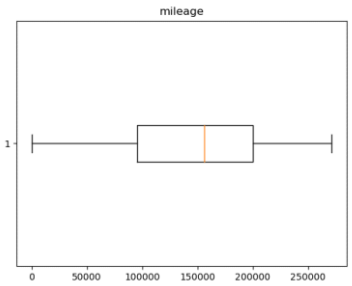
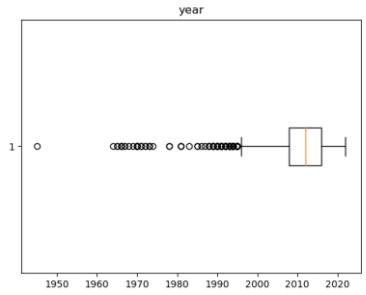
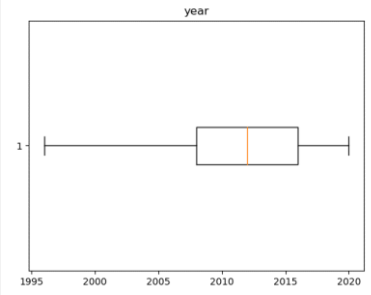
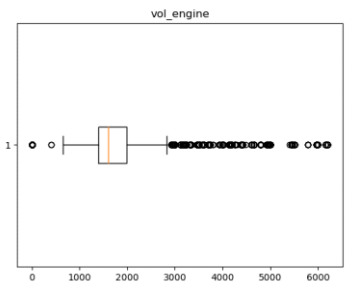
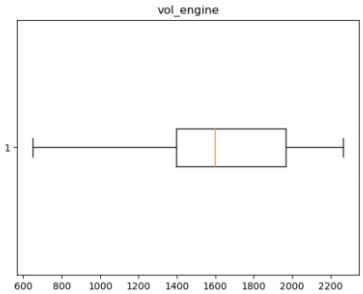
Para nuestro conjunto de datos lo aplicaremos en las columnas de Price, year, mileage y val_engine, lo cual sera reutilizar código.

Código

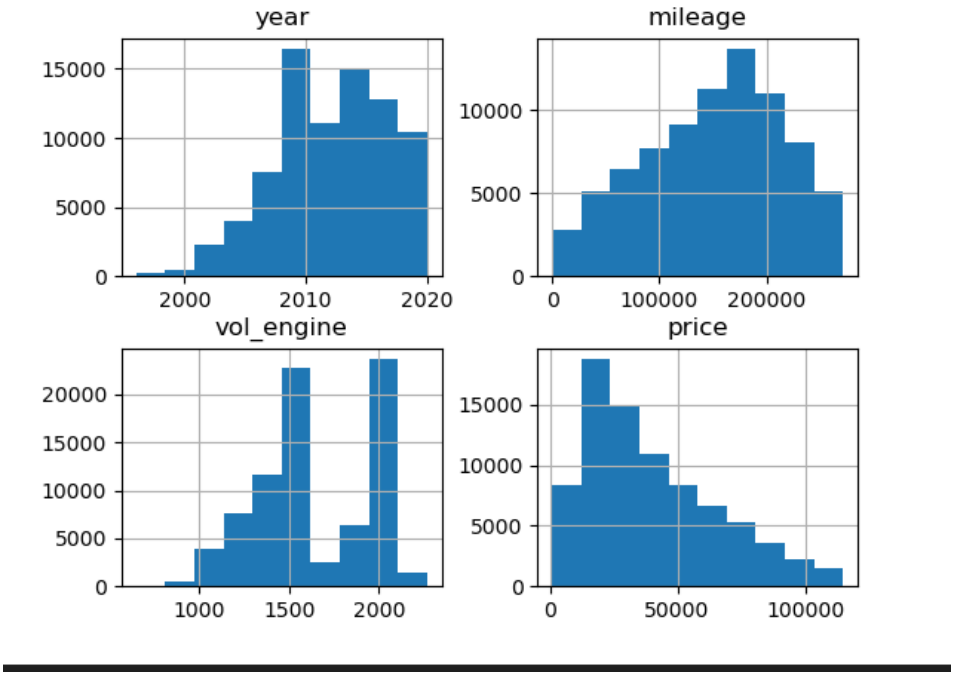
```
q1 = df['price'].quantile(0.25)
q3 = df['price'].quantile(0.75)
IQR = q3 - q1
bi_calculado = (q1 - 1.5 * IQR)
```

```
bs_calculado = (q1 + 1.5 * IQR)

ubicacion_outliers = (df['price'] >= bi_calculado) & (df['price'] <=
bs_calculado)

df = df[ubicacion_outliers]
```

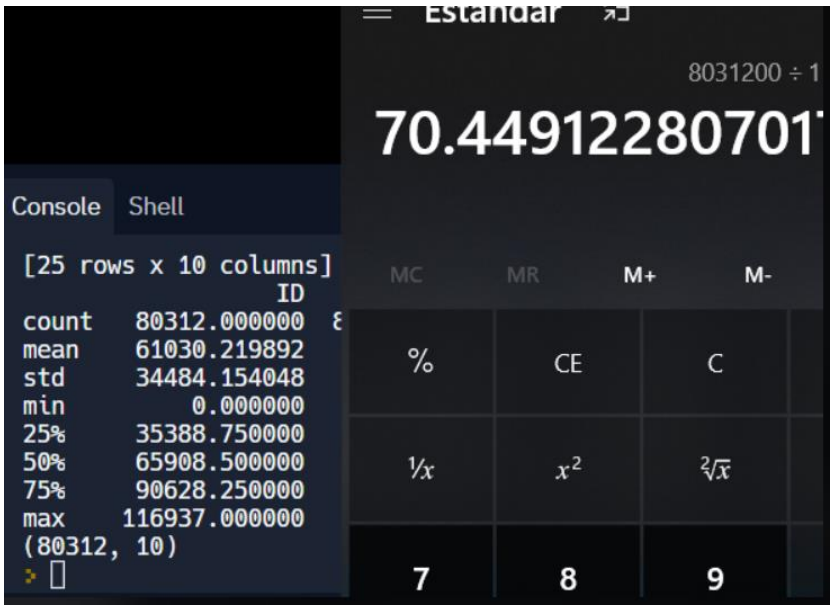
Resultado

Columna	*Con Outlier*	*Sin Outlier*
Price		
mileage		
year		
val_engine		

Una vez ejecutado el código para cada una de las columnas vemos gráficamente que están mucho mejor renqueados los datos



Para saber si continuamos con estos datos verificamos que cantidad de datos se eliminaros, mediante una regla de tres tenemos que después de la limpia de datos atípicos contamos con un aproximado del 70% de los datos así que continuaremos en esta ruta



Ya sabemos que trabajaremos con estos datos, analizamos un poco los datos para determinar cuáles columnas nos conviene etiquetar, entre las cuales localizamos dos en las cuales hay un cierto patrón de datos que se repiten las cuales son mark y fuel, así que haremos el tratamiento de dichas columnas con el siguiente código

Código

```
encoder = preprocessing.LabelEncoder()
encoder.fit(df["mark"])
aux_mark = encoder.transform(df["mark"])
dfMark_transform = pd.DataFrame(aux_mark)
df['mark_transform'] = dfMark_transform
```

donde el procedimiento es utilizar el método LabelEncoder el cual nos permite transformar string en datos numéricos, su funcionamiento consta de primero sacar las etiquetas o nombres de los patrones estos nombres los ordena de manera alfabética y les asigna un numero partiendo del 0, para verificar las equivalencias las guardamos en Data Set nuevos

Código

```
clasesmark = pd.DataFrame(list(encoder.classes_), columns=['Mark_Labels'])
aux_mark_etiquetado = encoder.transform(clasesmark["Mark_Labels"])
clasesmark['mark_transform'] = aux_mark_etiquetado
```

Resultado

	Mark_Labels	mark_transform
0	alfa-romeo	0
1	audi	1
2	bmw	2
3	chevrolet	3
4	citroen	4
5	fiat	5
6	ford	6
7	honda	7
8	hyundai	8
9	kia	9
10	mazda	10
11	mercedes-benz	11
12	mini	12
13	mitsubishi	13
14	nissan	14
15	opel	15
16	peugeot	16
17	renault	17
18	seat	18
19	skoda	19
20	toyota	20
21	volkswagen	21
22	volvo	22

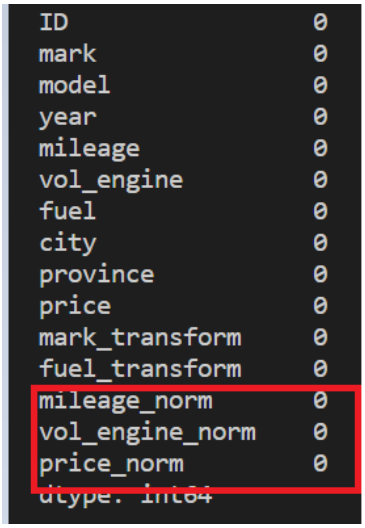
	Fuel_Labels	Fuel_transform
0	CNG	0
1	Diesel	1
2	Electric	2
3	Gasoline	3
4	Hybrid	4
5	LPG	5

Adema que notamos que los datos están muy dispersos así que optaremos por estandarizarlos con el siguiente código, Una vez terminado añadimos estos datos al Data Set principal

Código

```
scale= StandardScaler()
xnorm= df[['mileage','vol_engine','price']]
scaledx = scale.fit_transform(xnorm)
auxnorm = pd.DataFrame(scaledx,
columns=['mileage_norm','vol_engine_norm','price_norm'])
df[['mileage_norm','vol_engine_norm','price_norm']] =
auxnorm[['mileage_norm','vol_engine_norm','price_norm']]
```

Resultado



Teniendo esto ya tenemos nuestros datos listos para aplicarles la técnica que más nos convenga según los datos que requerimos, pero antes haremos una inspección rápida

También vamos a crear algunas cuantas categorías dependiendo de nuestras columnas

La primera categoría cuenta con 6 identificadores los cuales nos indican los precios relacionados a la categoría en sí.

Precio (price)

Categoría	MIN	MAX
Bajo	600	38300
Medio	38300	76000
Lujo	76000	114900+

Para la segunda categorización la haremos dependiendo del kilometraje

Kilometraje (mileage)

Categoría	MIN	MAX
Viaje	203625	271500+
Urbano	135750	203625
Rural	67875	135750
Agencia	1	67875

Para lograrlo ejecutamos el siguiente código definiendo los rangos y cada una de las categorías, mismo procedimiento para la segunda categorización.

Código

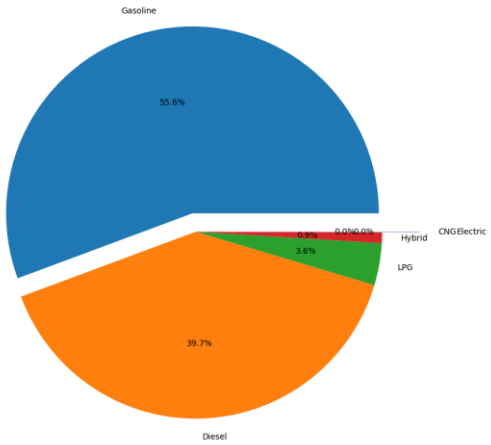
```
df['precios_cat'] = pd.cut(x=df['price'],
                     bins=[np.NINF, 19950, 38300,57450,76000,96750, np.inf],
                     labels=["bajo","bajo-m","medio","medio-a","alto","lujo"])
```

Resultado

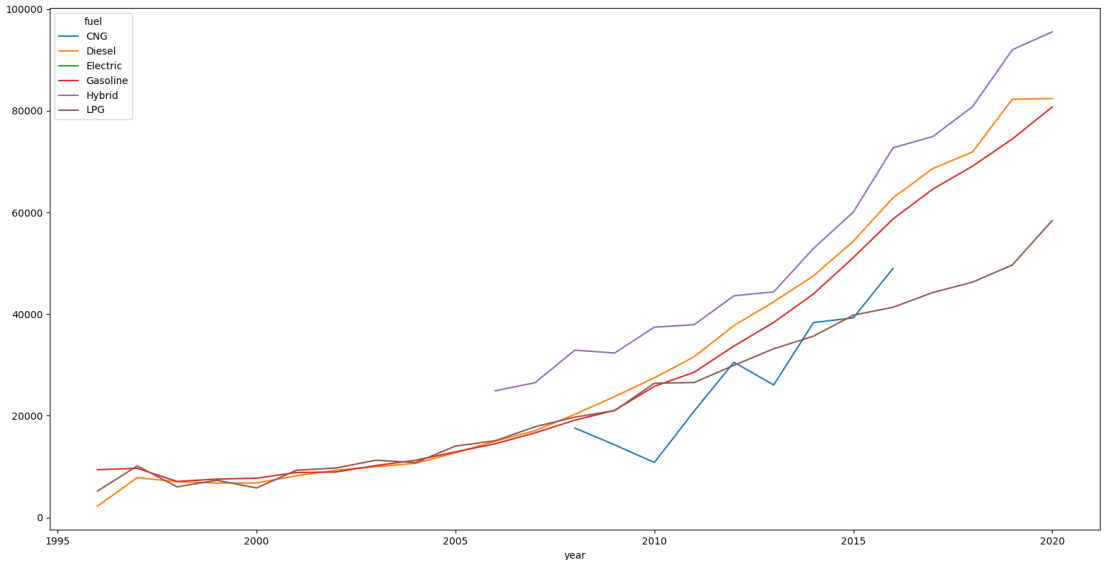
```
dtype: int64
       price precios_cat  mileage mileage_cat
23161  43500       medio    92317       Rural
17845  24900      bajo-m   249432       Viaje
76396  21900      bajo-m   122063       Rural
61430  50500       medio    30504     Agencia
39301  79900        alto    62000     Agencia
4110   19500        bajo   103580       Rural
61909  44900       medio   190000      Urbano
45659  76900        alto    39642     Agencia
31356   6500        bajo   168569      Urbano
27485  19990      bajo-m    55786     Agencia
53623  49900       medio    77500       Rural
5118   62500     medio-a    15000     Agencia
74273   9800        bajo   265083       Viaje
24936  44500       medio   182500      Urbano
52993  28900      bajo-m   133080       Rural
17195  17500        bajo   255380       Viaje
32962  36900      bajo-m   236000       Viaje
60081  37700      bajo-m   142000      Urbano
76328  46999       medio   260000       Viaje
22526  37500      bajo-m    93400       Rural
ID                    0
```

Interpretación de datos principales

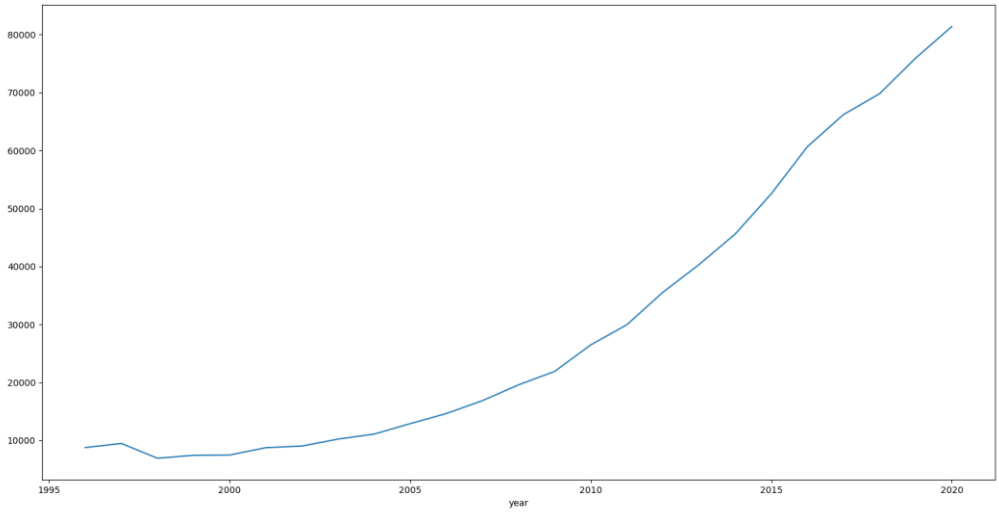
Ahora toca darles una interpretación a estos datos primeramente mediante una gráfica de pastel visualizamos que el principal tipo de combustible esta por Gasoline con un 56.6%, y que aparentemente un 0% usa un auto eléctrico esto podría no ser necesariamente que sean peores autos si no que como todos sabemos no es una tecnología accesible para todo el público especialmente para en el lado económico



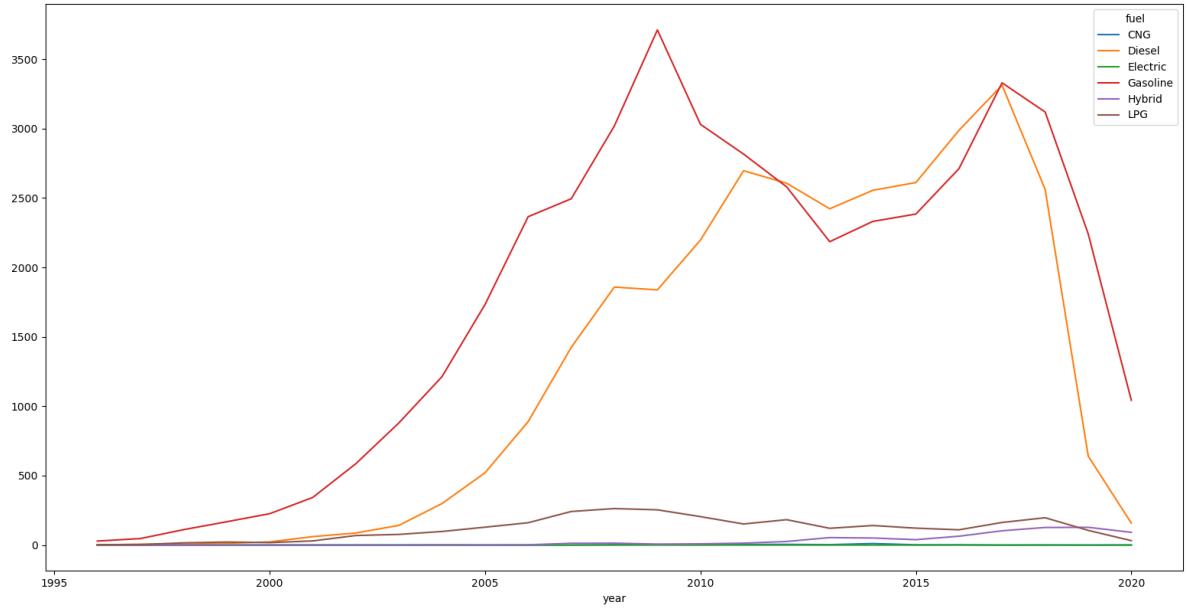
En relación de los precios con el tipo de gasolina notamos que sin importar el tipo de combustible el precio siempre ha estado en constante crecimiento



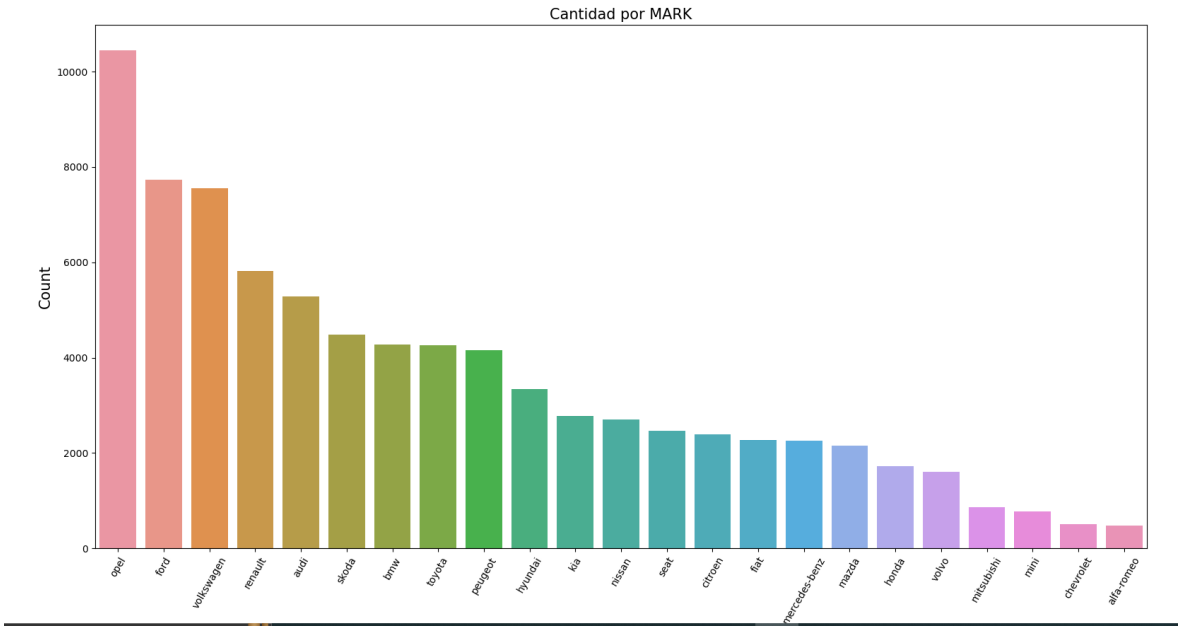
Generalizando siempre ha estado este aumento exponencial



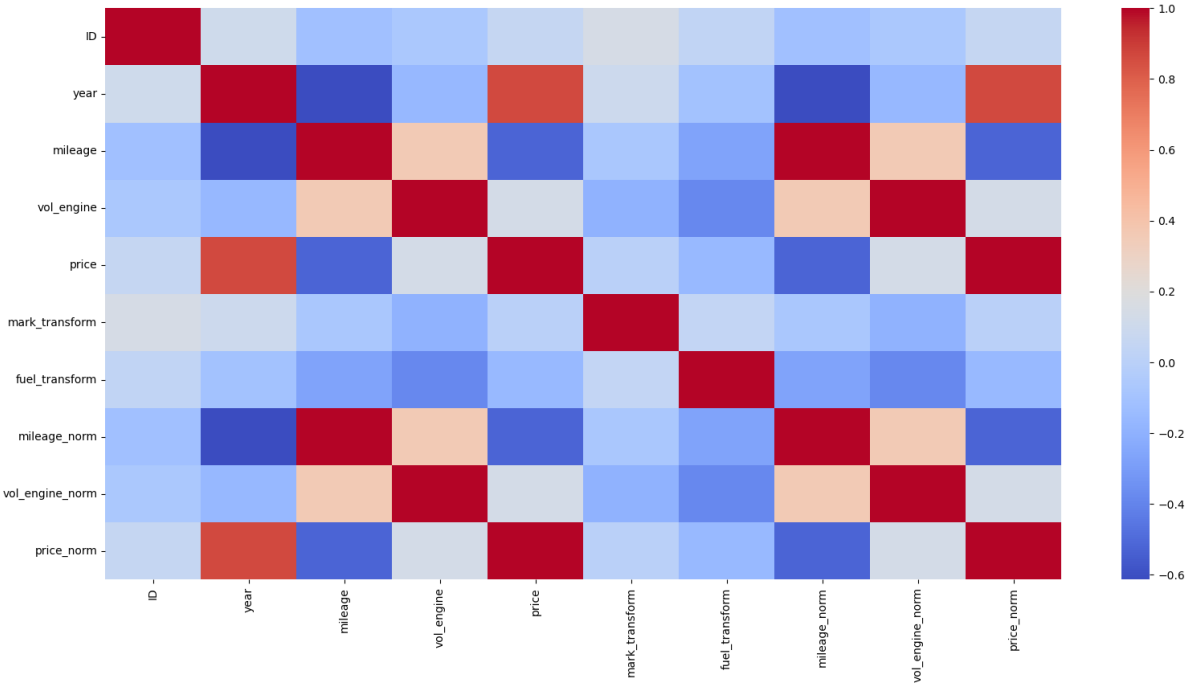
Además, haciendo una relación entre la cantidad de autos y el tipo de gasolina no influye en el precio porque como notamos los autos más vendidos son los que utilizan gasolina, pero curiosamente los que son más costosos son los que utilizan como combustible LPG



Mediante un histograma visualizamos en forma ordenada la popularidad en cuanto las marcas utilizadas

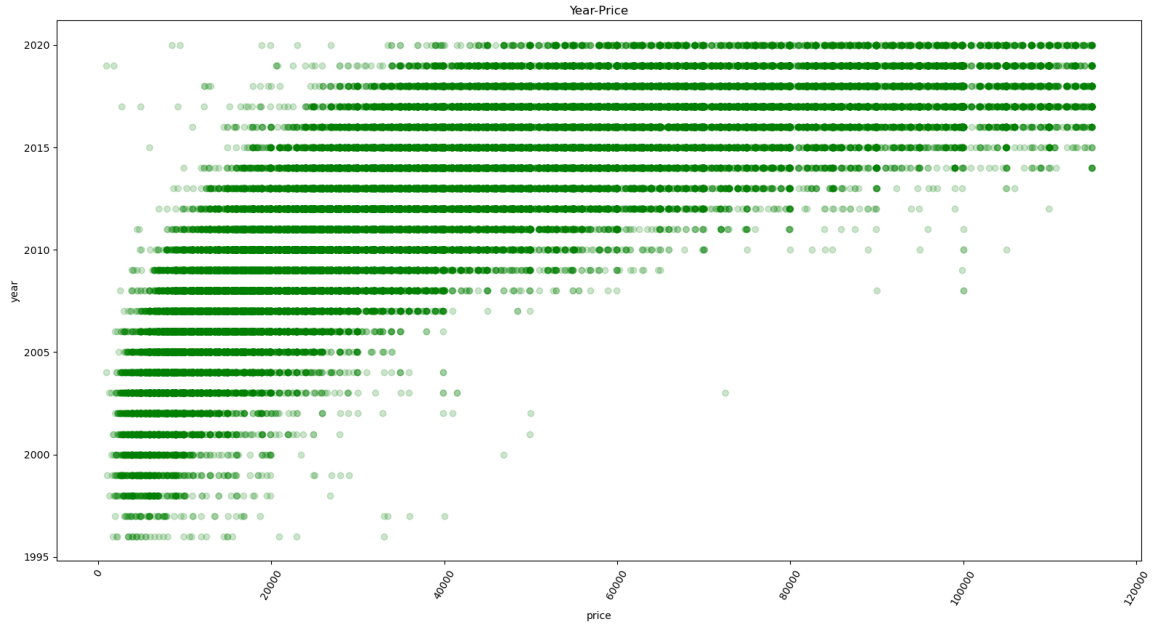


Ahora con una sencilla matriz de correlación intentamos darles una relación a las distintas columnas

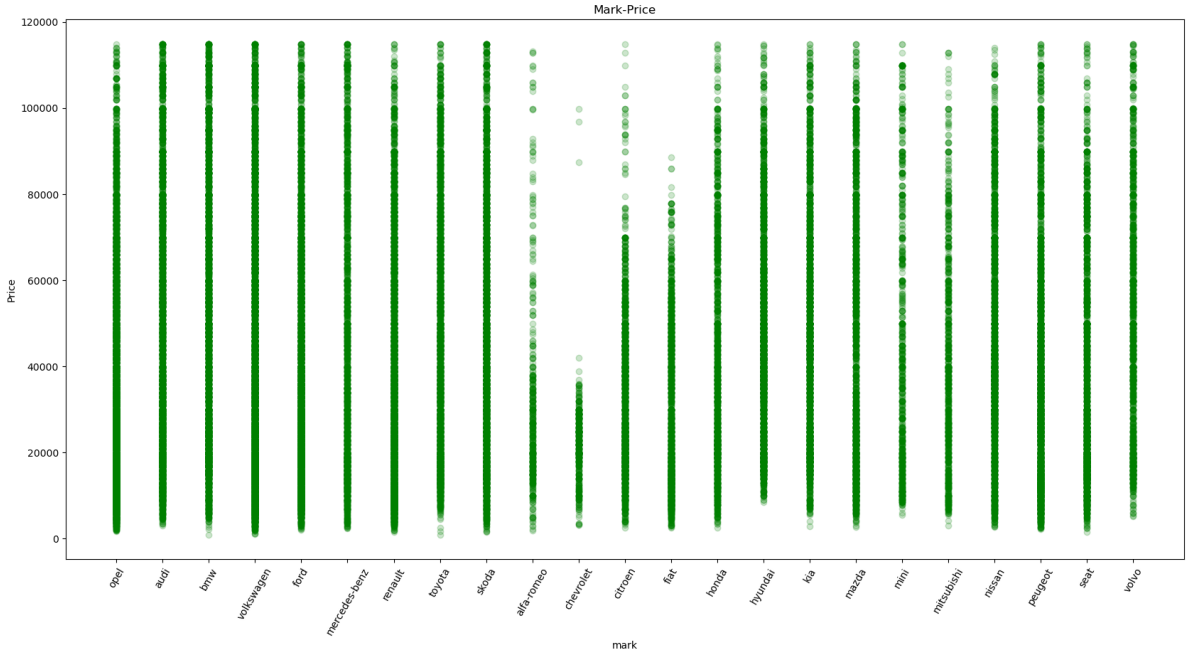


Vamos a desglosar mejor estos datos con las columnas que más se asemejan más bien lo realizaremos con nuestras columnas de interés

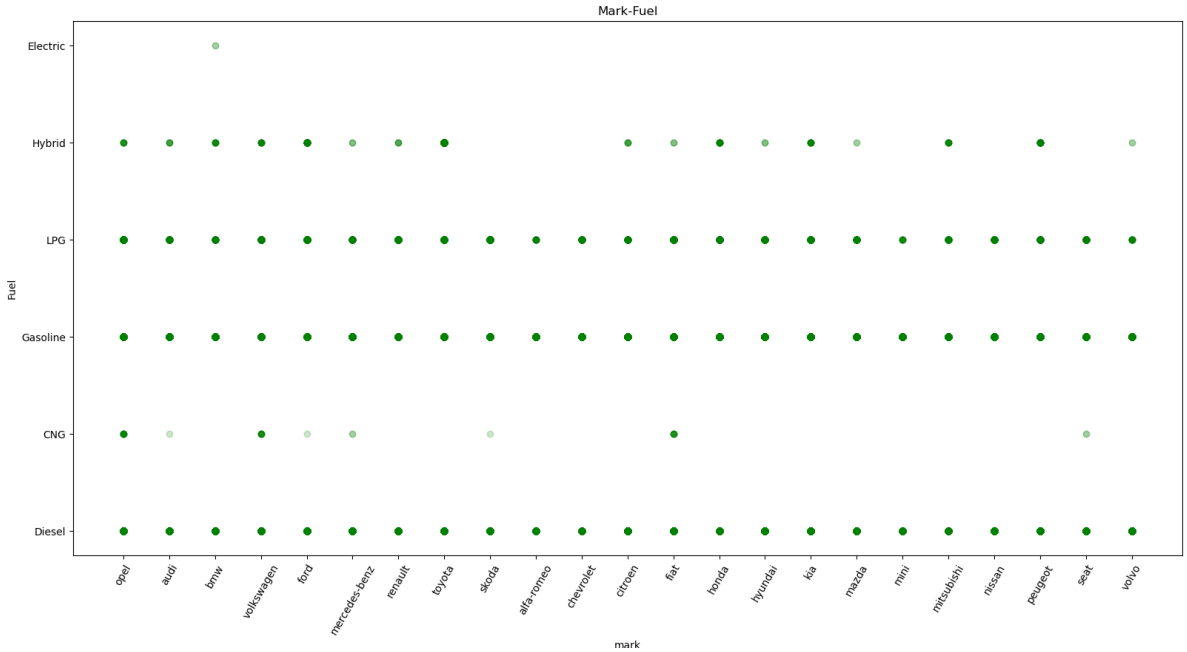
Notamos que en los años anteriores al 2010 el precio de los autos no eran tan elevados



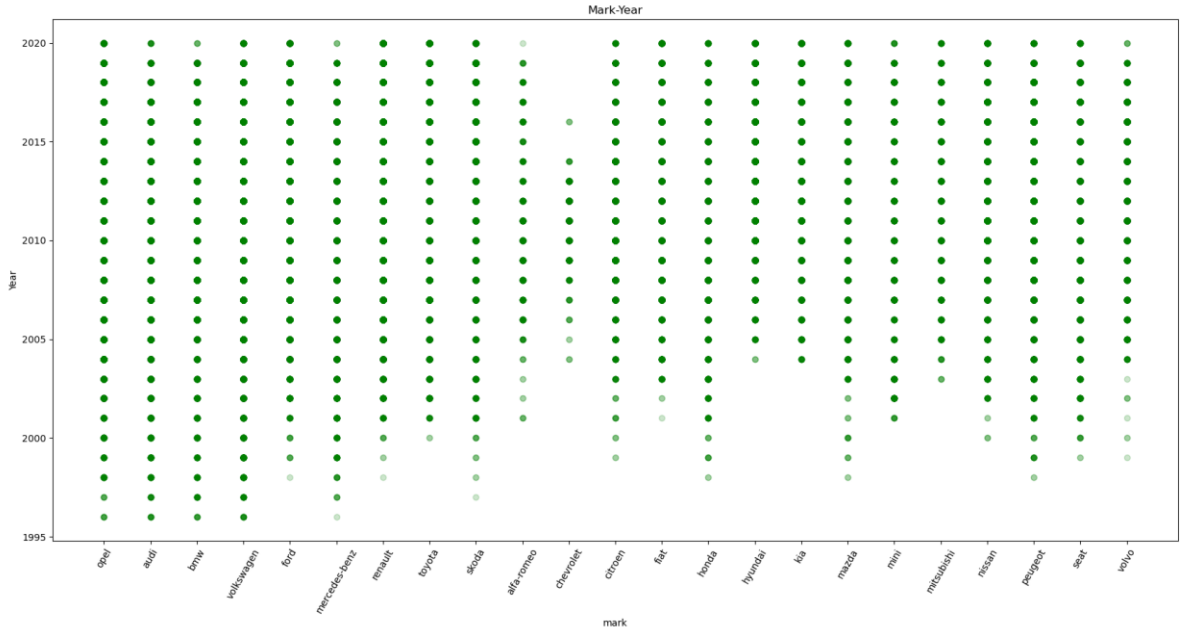
Que todas las marcas han estado como que estandarizada en los precios, menos la compañía de Chevrolet



En cuanto a los tipos de combustibles el tipo menos usado entre las marcas ha sido CNG



Por último, la relación años con marca nos deja ver que los menos utilizados son los Chevrolet



Pero hay algo más a fondo con la marca Chevrolet que no es tan gustado por el público a pesar de su bajo costo, analizando entre distintas paginas nos encontramos una que nos dice que tanto vehículo de una marca han sido arreglados, notamos que esta marca ronda entre los que más se han arreglado en un periodo de 1 año, para más información véase la página: https://www.caranddriver.com/es/coches/planeta-motor/a51339/las-10-marcas-de-coches-mas-fiables-y-las-menos-segun-la-ocu/

Fiabilidad por marcas	2018 Calificación (sobre 100)		
LEXUS	95	NISSAN	88
HONDA	93	SMART	88
PORSCHE	93	VOLVO	88
TOYOTA	93	FORD	88
MITSUBISHI	92	RENAULT	88
BMW	91	PEUGEOT	88
AUDI	91	SEAT	88
MAZDA	91	CITROËN	87
JAGUAR	91	LANCIA	86
KIA	90	CHEVROLET/DAEWOO	85
SUBARU	90	SSANGYONG	85
SUZUKI	89	OPEL	85
SKODA	89	LAND ROVER	85
DACIA	89	SAAB	85
MINI	89	FIAT	84
MERCEDES	89	ALFA ROMEO	83
HYUNDAI	89	CHRYSLER	82
VOLKSWAGEN	89	DODGE	80
JEEP	89		

Clasificador

Como clasificador utilizaremos lo que son arboles de decisión ya que nos permitirá evaluar de manera un poco más visual la toma de alguna decisión en este caso que tipo de auto comprar dependiendo de distintas cuestiones.

Comenzamos con elegir las columnas con las que trabajaremos, nos guiaremos de la siguiente lista.

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               80312 non-null  int32
 1   mark             80312 non-null  object
 2   model            80312 non-null  object
 3   year             80312 non-null  int64
 4   mileage          80312 non-null  int64
 5   vol_engine       80312 non-null  int64
 6   fuel             80312 non-null  object
 7   city             80312 non-null  object
 8   province         80312 non-null  object
 9   price            80312 non-null  int64
 10  mark_transform   80312 non-null  int32
 11  fuel_transform   80312 non-null  int32
 12  precios_cat      80312 non-null  category
 13  mileage_cat      80312 non-null  category
 14  mileage_norm     80312 non-null  float64
 15  vol_engine_norm  80312 non-null  float64
 16  price_norm       80312 non-null  float64
```

Para determinar que rango de precio le conviene comprar al usuario vamos a tomar como variables descriptivas las columnas de year, mark_transform, fuel_transform, mileage_norm y vol_engine_norm

```
X = df.iloc[:, [3,10,11,14,15]]
```

Y como variable objetivo precios_cat

```
Y = df.iloc[:,12]
```

A continuación, dividiremos los datos de entrenamiento y los datos de prueba, dando un 75% a los datos de entrenamiento

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.70,
random_state= 0)
```

Dentro de una variable cargamos el modelo donde la profundidad del árbol será de 5 ramificaciones además de que con la propiedad de class_weight balanceamos los datos que como vemos en la siguiente lista hay muchos más autos económicos que de lujo y medio

```
economico    45233
medio        25956
lujo          9123
```

```
arbol = DecisionTreeClassifier(max_depth=5,
                               class_weight="balanced",
                               )
```

Con la siguiente línea entrenamos el modelo

```
arbol_precios = arbol.fit(x_train,y_train)
```

Para mostrar de manera visual no utilizaremos el visualizador del programa si no que guardaremos la gráfica en un archivo para poder visualizarla de mejor manera

```
fig = plt.figure(figsize=(60,20))
tree.plot_tree(arbol_precios, feature_names=list(X.columns.values),
                            class_names=list(Y.values), filled=True,
fontsize=10)
fig.savefig("arbolddlo3.png")
```

Hacemos algunas predicciones y medimos su precisión además de mostrar su matriz de confusión

```
y_pred =arbol_precios.predict(x_test)
matriz = confusion_matrix(y_test, y_pred)
print(matriz)

precicion =  np.sum(matriz.diagonal())/np.sum(matriz)
print(precicion)

from sklearn.metrics import accuracy_score
exactitud = accuracy_score(y_test, y_pred)
print(exactitud)
```

```
[[27035    69  4532]
 [    3  4642  1709]
 [ 1488  3907 12834]]
0.7917430050338854
0.7917430050338854
```

Pruebas en el clasificador

Probaremos los resultados con la siguiente tabla de testeo

year	mark_transform	fuel_transform	mileage_norm	vol_engine_norm	Resultado
2010	1	0	1.965303 (256000)	1.634886 (1968)	**Economico**
2005	5	1	0.244688 (167933)	-1.668947 (998)	**Economico**
1990	6	3	-0.126525 (148933)	-0.408722 (1368)	**Economico**
2018	7	2	-0.066604 (152000)	0.374661 (1598)	**Lujo**
2000	8	4	-1.116591 (98258)	0.374661 (1598)	**Economico**
2005	9	5	-0.900272 (109330)	-0.306541 (1398)	**Economico**

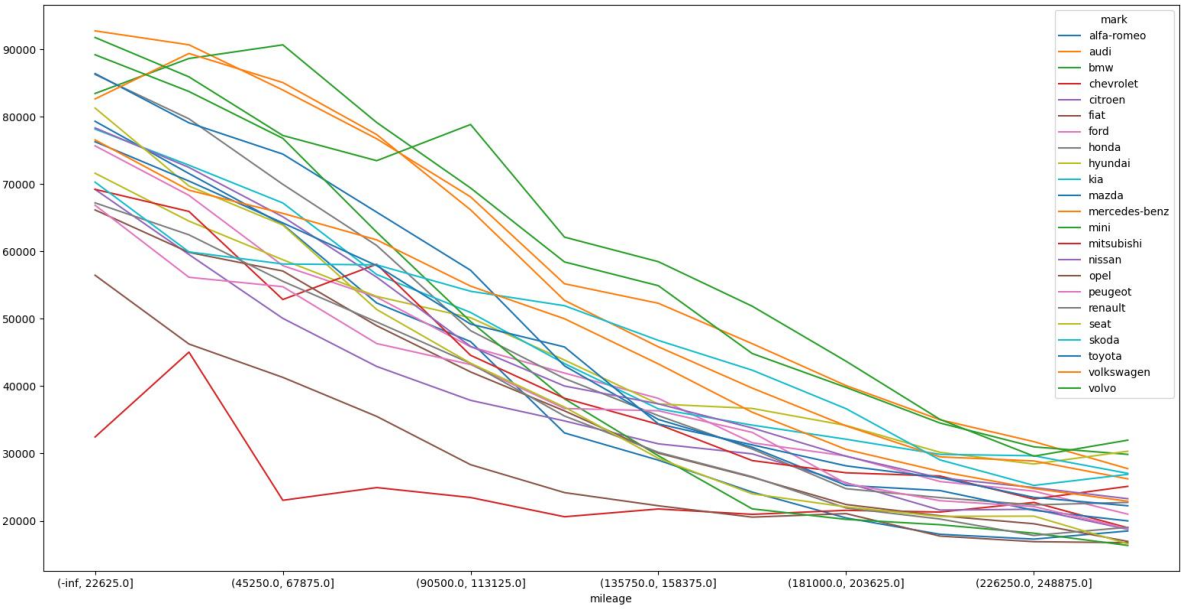
Una vez que ejecutamos el código para predecir obtenemos lo siguientes resultados, donde la penúltima columna muestra el auto que más le conviene al usuario según sus necesidades, además de un porcentaje de que el resultado sea correcto

```
 [9.99566557e-01 0.00000000e+00 4.33442867e-04]]
   year  mark_transform  fuel_transform  mileage_norm  vol_engine_norm resultadols  porcentaje
0  2010               1               0      1.965303         1.634886   economico    0.500714
1  2005               5               1      0.244688        -1.668947   economico    0.999567
2  1990               6               3     -0.126525        -0.408722   economico    0.999567
3  2018               7               2     -0.066604         0.374661        lujo    0.900498
4  2000               8               4     -1.116591         0.374661   economico    0.999567
5  2005               9               5     -0.900272        -0.306541   economico    0.999567
PS C:\Users\Admin\PycharmProjects\examenUnidadDos>
```

Tratamiento de las hipótesis

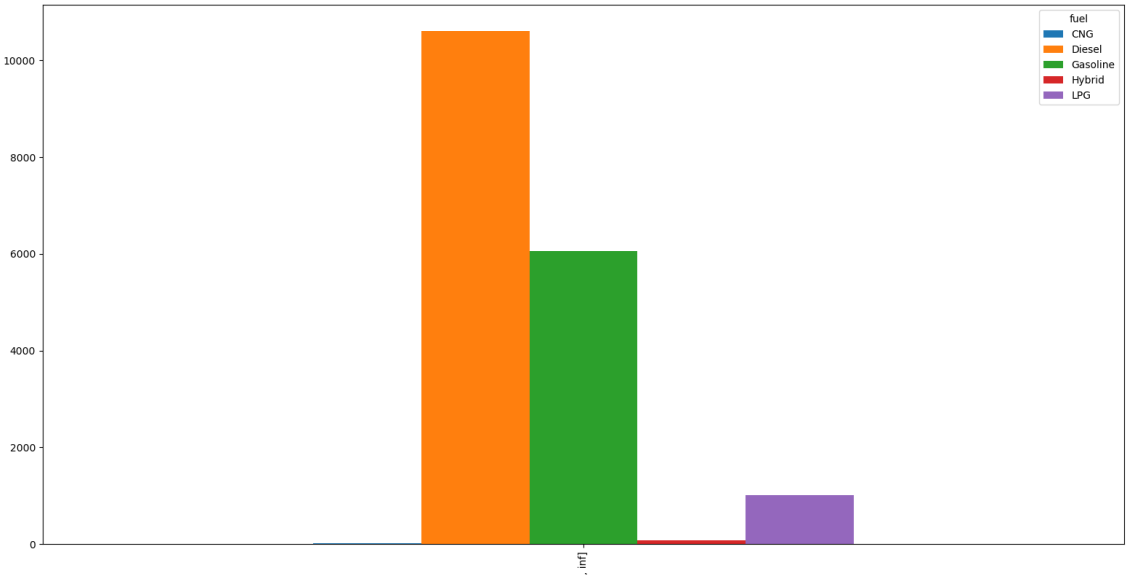
1. El devaluó de los autos según los kilómetros recorridos y marca del vehículo

   R= Como podemos observar independientemente de la marca del vehículo el precio de este siempre va decaer



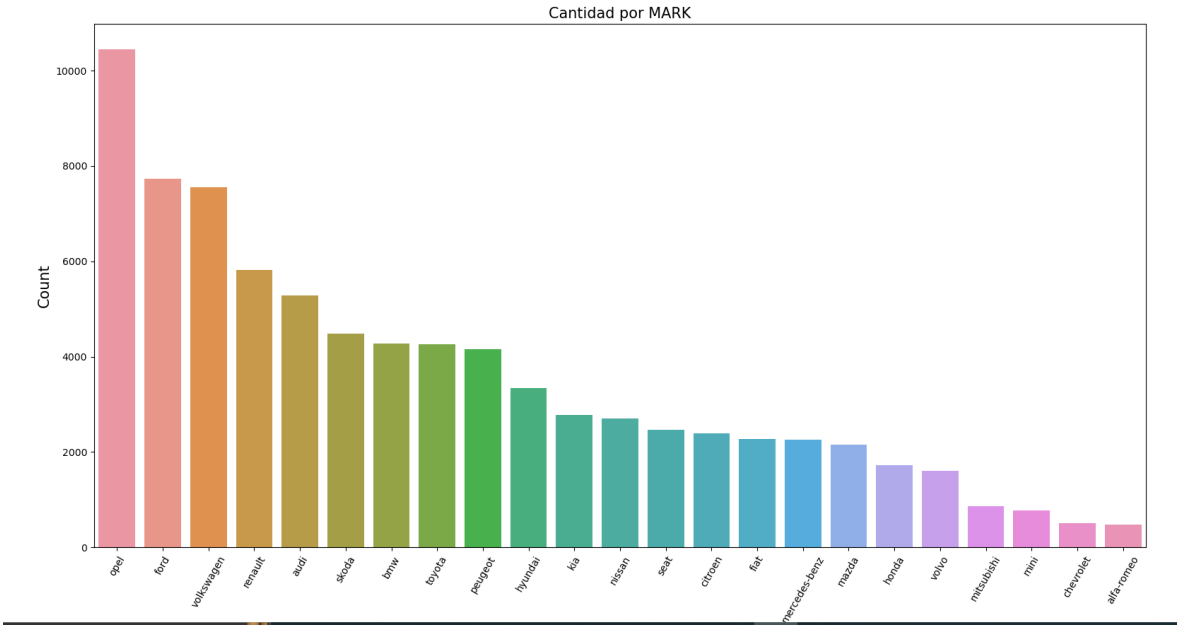
2. Como segunda idea podemos tener que tipo de combustible es utilizado para los autos de viaje

   R=Podemos observar que el diésel es el combustible preferido para este tipo de conductores



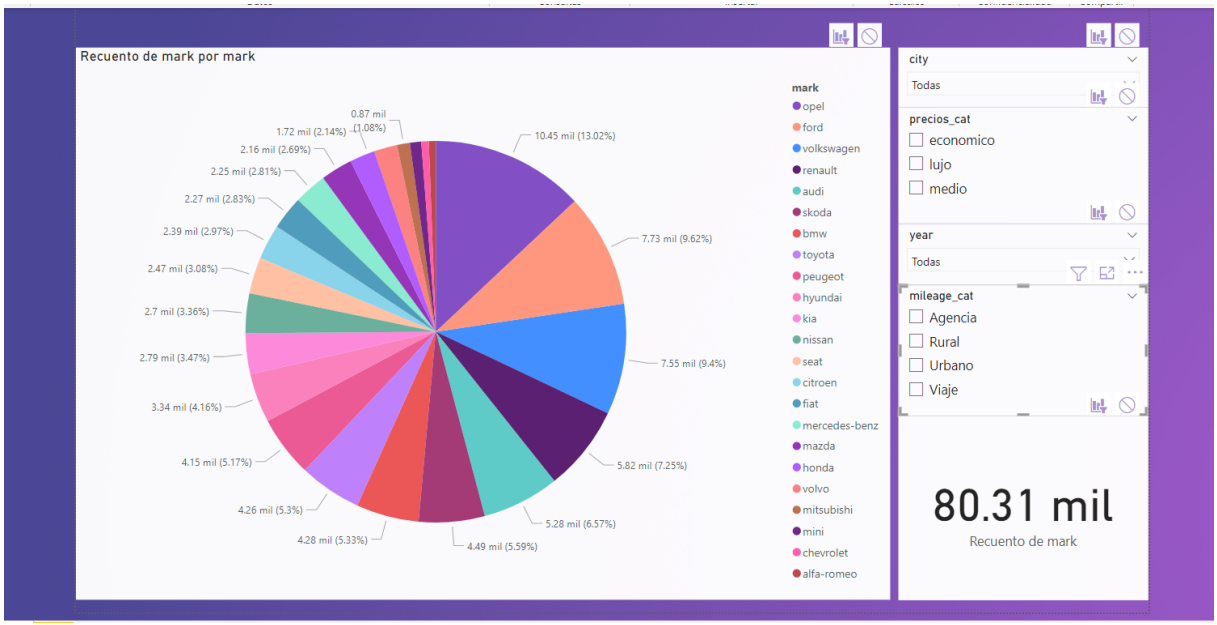
3. Que marca de autos prefieren los ciudadanos de Polonia.

R=Como vimos durante la realización del proyecto la marca preferida es opel



Extra

Para poder realizar un reporte más interactivo con el Data Set ya tratado haremos uso de una herramienta llamada Power Bi, en la cual aremos un recuento de las marcas dependiendo de filtros como lo podría ser la ciudad, las categorías de precios que realizamos, los años y las categorías del kilometraje



Conclusión

Para esta última parte del semestre hubo como primera complicación la selección del Data Set y con ello la creación de lo esperado o como nosotros lo llamamos las hipótesis del proyecto finalmente pensando en lo que queríamos mostrar como resultado y que cualquier persona que viera el proyecto pueda interactuar escogidos como clasificador el árbol de selección en el cual mediante selecciones de verdadero y falso se llega a un resultado, durante el desarrollo del proyecto hubo varias dificultades entre las cuales que al momento de eliminar datos atípicos y hacer una etiquetación se tienen que realizar un reseteo de índices, otra problemática fue escoger los rangos categóricos ya que si un rango estaba muy desbalanceado la precisión del modelo baja considerablemente, por utimo para poder darle un plus al proyecto realizamos lo que un reporte que contabiliza los autos por marca según los filtros que se seleccionen

Link del código en línea: https://replit.com/join/hyexvcimgo-irvingvaquero

Código

```
#-------Seccion de librerias BEGIN.-----#
import imp
import warnings
import pandas as pd
#import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
from pyparsing import col
import seaborn as sns
from sklearn import preprocessing
from sklearn import tree
from sklearn.preprocessing import StandardScaler
import numpy as np
import pingouin as pg
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
#-------Seccion de librerias END.-----#

#-------Configuracion de la consola(Salidas de codigo) BEGIN.-----#
warnings.filterwarnings("ignore") # ignorar alertas
pd.set_option('display.max_rows',30)
pd.set_option('display.max_columns',15)
pd.set_option('display.width',1000)
#-------Configuracion de la consola(Salidas de codigo) END-----#

#-------Lectura e inspeccion de datos BEGIN-----#
df = pd.read_csv("Car_Prices_Poland_Kaggle.csv") #Importacion de datos
print(df.info())                                 #Obtenemos los tipos de datos decada columna
```

```
print(df.describe())                              #Obtenemos valores estadisticos de las columna que tengan valores numericos
print(df.head())                                  #Obtenemos los primeros 5 registros
print(df.tail())                                  #Obtenemos los ultimos 10 rehistros
print(df.sample(10))                              #Obtenemos 10 registros de ejemplo
df.rename(columns={"Unnamed: 0": "ID"},           #Cambiamos el nombre de la columna "Unnamed: 0" por "ID"
              inplace=True)
#-------Configuracion de la consola(Salidas de codigo) END-----#

#-------Pre procesamiento BEGIN-----#
print(df.isnull().sum())                          #Deteccion de datos nulos
print(df['generation_name'].sample(10))           #Ejemplo de la columna con datos nulos
del df["generation_name"]                         #Eliminacion de la columna que contiene datos nulos
print(df.isnull().sum())                          #Verificamos que se elimino la columna
print(df.describe())                              #Localizacion de datos atipicos mediante maaximos y minimos en la columnas numericas
df.drop(['ID'],1).hist()                          #Histogramas sin la columna de ID
plt.show()                                        #Lanzamiento del histograma

#-------Eliminacion de Outliers - Inliers BEGIN-----#
columnas_dataSet = ['price', 'mileage', 'year', 'vol_engine']                #Defininimos un arreglo con el nombre de columnas con
outliers
for i in columnas_dataSet:                                                   #For para eliminacion de datos atipicos
    plt.title(i)                                                             #Titulo de la grafica
    plt.boxplot(df[i], vert=False)                                           #Carga de la columna en la grafica de caja y bigote
    plt.show()                                                               #Lanzamiento de la grafica
    q1 = df[i].quantile(0.25)                                                #Otencion del primer cuartil
    q3 = df[i].quantile(0.75)                                                #Obtencion del tercer cuartil
    IQR = q3 - q1                                                            #Obtencion del indice IQR
    bi_calculado = (q1 - 1.5 * IQR)                                          #Obtencion del limite inicial
    bs_calculado = (q1 + 1.5 * IQR)                                          #Obtencion del limite final
    ubicacion_outliers = (df[i] >= bi_calculado) & (df[i] <= bs_calculado)   #Obtencion de los datos NO atipicos
    df = df[ubicacion_outliers]                                              #Guardado del data set sin estos dato
```

```
    plt.title(i)                                                    #Titulo de la grafica
    plt.boxplot(df[i], vert=False)                                  #Carga de la columna en la grafica de caja y bigote
    plt.show()                                                      #Lanzamiento de la grafica


df.reset_index(inplace=True, drop=True)                             #Reseteo de indices predeterminado
df['ID'] = np.arange(0,len(df))                                     #Reseteo de nuestra columna de indices

df.drop(['ID'],1).hist()                                            #Histogramas sin la columna de ID
plt.show()                                                          #Lanzamiento del histograma
print(df.shape)                                                     #Obtenemos la cantidad de datos resultantes despues de la
limpia

#-------Eliminacion de Outliers - Inliers END-----#


#-------Creacion de etiquetas Begin----#
encoder = preprocessing.LabelEncoder()                                  #Cargamos el codificador
encoder.fit(df["mark"])                                                 #Pasamos la columna de donde tomara las etiquetas
aux_mark = encoder.transform(df["mark"])                                #Tranformamos la columna
dfMark_transform = pd.DataFrame(aux_mark)                               #Transformamos el resultado en un data set
df['mark_transform'] = dfMark_transform                                 #Añadiamos la columna a nuestro data set original

clasesmark = pd.DataFrame(list(encoder.classes_), columns=['Mark_Labels'])      #Creamos un data set con las equivalente
aux_mark_etiquetado = encoder.transform(clasesmark["Mark_Labels"])              #Guardamos los equivalentes
clasesmark['mark_transform'] = aux_mark_etiquetado                              #Lo añadimos al dataset de equivalencias

encoder = preprocessing.LabelEncoder()                                  #Cargamos el codificador
encoder.fit(df["fuel"])                                                 #Pasamos la columna de donde tomara las etiquetas
aux_mark = encoder.transform(df["fuel"])                                #Tranformamos la columna
dfMark_transform = pd.DataFrame(aux_mark)                               #Transformamos el resultado en un data set
df['fuel_transform'] = dfMark_transform                                 #Añadiamos la columna a nuestro data set original
```

```
clasesfuel = pd.DataFrame(list(encoder.classes_), columns=['Fuel_Labels'])      #Creamos un data set con las equivalente
aux_fuel_etiquetado = encoder.transform(clasesfuel["Fuel_Labels"])              #Guardamos los equivalentes
clasesfuel['Fuel_transform'] = aux_fuel_etiquetado                              #Lo añadimos al dataset de equivalencias

print(clasesmark)                                                               #Columna mark transformada
print(clasesfuel)                                                               #Columna fiel transformada
print(df.head(10))                                                              #Verificamos datos iniciales
print(df.tail(10))                                                              #Verificamos datos finales
print(df.isnull().sum())                                                        #Verificamos nulos
#-------Creacion de etiquetas END-----#

#-------Categorizacion de datos BEGIN-----#


df['precios_cat'] = pd.cut(x=df['price'],
                    bins=[np.NINF, 38300,76000, np.inf],
                    labels=["economico","medio","lujo"])
df['mileage_cat'] = pd.cut(x=df['mileage'],
                    bins=[np.NINF, 67845, 135750,203625, np.inf],
                    labels=["Agencia","Rural","Urbano","Viaje"])
print(df[['price','precios_cat','mileage','mileage_cat']].sample(20))
#-------Categorizacion de datos BEGIN-----#

#-------Normalizacion Begin -----#

scale= StandardScaler()                                                         #cargamos el metodo para estandarizar
xnorm= df[['mileage','vol_engine','price']]                                     #aasignamos las columnas a normalizar
scaledx = scale.fit_transform(xnorm)                                            #Aplicamos la normalizacion
auxnorm = pd.DataFrame(scaledx, columns=['mileage_norm','vol_engine_norm','price_norm'])                    #Guardamos en data set
auxiliar
```

```
df[['mileage_norm','vol_engine_norm','price_norm']] = auxnorm[['mileage_norm','vol_engine_norm','price_norm']]  #Guardamos en data set
original
print(df.isnull().sum())                                                                                          #Verificamos que se haya
insertado bien
#-------Normalizacion END-----#

#-------Pre procesamiento END-----#

#---------Interprestacion de datos BEGIN------#

pastel = df.fuel.value_counts()                              #Contabilizamos los datos de la columna fuel
explode = (0.1, 0, 0, 0, 0.2, 0.3)                           #Defininimos cual porcentaje estara separado
pastel.plot.pie(autopct = '%1.1f%%', explode = explode,)     #cargamos los dato
plt.axis("equal")
plt.show()

print(df.groupby(by='year')['price'].mean())                                    #En consola imprimimos la media de los precios por año
print(pd.crosstab(df['year'], df['fuel'], values=df['price'], aggfunc='mean'))  #En consola imprimimos la media del precio segun el año
y el tipo de combustible

pd.crosstab(df['year'], df['fuel']).plot()      #Grafica de cantidas de autos por tipo de combustible por añ
plt.show()                                      #Lanzamos grafica

pd.crosstab(df['year'], df['fuel'], values=df['price'], aggfunc='mean').plot()  #Grafica de la media de precios segun el combustible a
lo largo de los años
plt.show()                                                                      #Lanzamos la grafica

df.groupby(by='year')['price'].mean().plot()    #Grafica de la media de los precios a lo largo de los años
plt.show()                                      #Lanzamos la grafica

sns.countplot(x="mark", data=df, order=df["mark"].value_counts().index)         #Cargamos los datos
```

```
plt.xticks(rotation=60)                                                    #La etiqueta de la columnas la rotamos
plt.xlabel("Mark", size=15)                                                #Lo que mostrara el eje x
plt.ylabel("Count", size=15)                                               #Lo que mostrara el eje y
plt.title("Cantidad por MARK", size=15)                                    #Lo que mostrara el titulo
plt.show()                                                                 #Lazamos la grafica


car_corr = df.corr(method='spearman')           #defiimos el metodo por el cual se hara la matriz de correlacion
sns.heatmap(car_corr,                           #Cargamos los datos
            xticklabels=car_corr.columns,       #Definimimos el nombre para el eje x
            yticklabels=car_corr.columns,       #Definimimos el nombre para el eje y
            cmap='coolwarm'                     #Definimos los colores
            )
plt.show()                                      #Lanzamos la grafica
print('\n',format(' Matriz de correlacion datos numericos ','*^82'),'\n')
correr = pg.pairwise_corr(df, method='spearman')                    #defiimos el metodo por el cual se hara la matriz de correlacion
print(correr.sort_values(by=['p-unc'])[['X','Y','n','r','p-unc']])  #En consola mostramos la matriz


plt.scatter(df['price'],df['year'],c="green", alpha=0.2)   #Cargamos las columnas correlacionadas
plt.xticks(rotation=60)                                    #Rotamos las etiquetas
plt.xlabel("price")                                        #Etiqueta del eje x
plt.ylabel("year")                                         #Etiqueta del eje x
plt.title("Year-Price")                                    #Etiqueta del titulo
plt.show()                                                 #Lanzamos la grafica

plt.scatter(df['mileage'],df['vol_engine'],c="green", alpha=0.2)   #Cargamos las columnas correlacionadas
plt.xticks(rotation=60)                                            #Rotamos las etiquetas
plt.xlabel("mileage")                                              #Etiqueta del eje x
plt.ylabel("vol_engine")                                           #Etiqueta del eje x
plt.title("mileage-vol_engine")                                    #Etiqueta del titulo
plt.show()                                                         #Lanzamos la grafica
```

```
columnas_scatter = ['price', 'fuel', 'year']              #Defininimos un arreglo con el nombre de columnas que estarn correlacionadas
for i in columnas_scatter:                                #Inicamos for con el nombre de estas columnas
    plt.scatter(df['mark'],df[i],c="green", alpha=0.2)    #Cargamos las etiquetas
    plt.xticks(rotation=60)                               #Rotamos las etiquegtas
    plt.xlabel("mark")                                    #Etiqueta del eje x
    plt.ylabel(i)                                         #Etiqueta del eje y
    plt.title("Mark-"+i)                                  #Etiqueta del titulo
    plt.show()                                            #Lanzamos la grafica
print(df.info())
#---------Interprestacion de datos END------#

#---------Creacion del clasificador Begin------#
X = df.iloc[:, [3,10,11,14,15]]                                                  #Seleccion de variables descriptivas
Y = df.iloc[:,12]                                                                #Seleccion de variable objetivo
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.70, random_state= 0)  #Separacion de datos entrenamiento
arbol = DecisionTreeClassifier(max_depth=5,                                      #Maximo de niveles
                               class_weight="balanced",                          #Balanceamos datos
                               )
arbol_precios = arbol.fit(x_train,y_train)                                       #Entrenamos el modelo
fig = plt.figure(figsize=(140,20))                                               #Definimos tamaño de la figura
tree.plot_tree(arbol_precios, feature_names=list(X.columns.values),              #Cargamos la figura
                           class_names=list(Y.values), filled=True, fontsize=10)  #Defininimos letra
#plt.show()

fig.savefig("figuraExamen.png")                                                  #Guardamos en la computadora la img
y_pred =arbol_precios.predict(x_test)                                            #Hacemos la prediccion con los datos de
prueba
matriz = confusion_matrix(y_test, y_pred)                                        #Creamos la matriz de confucion
print(matriz)                                                                    #Lanzamos la matriz
```

```
precicion =  np.sum(matriz.diagonal())/np.sum(matriz)                                   #Calculamos precision/exactitud
print("Precision: ", precicion)                                                         #Lanzamos resultadi


exactitud = accuracy_score(y_test, y_pred)                                              #Metodo sklearn para precision
print("Precision: ", exactitud)                                                         #Lanzamos resultado

#---------Creacion del clasificador END------#


#---------Hipotesis Begin------#
age_groups = pd.cut(df['mileage'],                                                      #Seccionamos por
mileage
                    bins=[np.NINF, 22625, 45250,67875,90500,113125,135750,158375,181000,203625,226250,248875,271500, np.inf])
pd.crosstab(age_groups, df['mark'], values=df['price'], aggfunc='mean').plot()  #Grafica de la media de precios segun el combustible a
lo largo de los años
plt.show()                                                                              #lanzamos la grafica

viaje = pd.cut(df['mileage'], bins=[203625, np.inf])                                    #Seleccionamos por viaje
pd.crosstab(viaje, df['fuel']).plot.bar()                                               #Contamos segun el combustible
plt.show()                                                                              #Lanzamos la grafica
#---------Hipotesis End------#

#---------Testeo Begin------#
datos = [
    {'year': 2010, 'mark_transform': 1, 'fuel_transform': 0, 'mileage_norm': 256000, 'vol_engine_norm': 1968},
    {'year': 2005, 'mark_transform': 5, 'fuel_transform': 1, 'mileage_norm': 167933, 'vol_engine_norm': 998},
    {'year': 1990, 'mark_transform': 6, 'fuel_transform': 3, 'mileage_norm': 148933, 'vol_engine_norm': 1368},
    {'year': 2018, 'mark_transform': 7, 'fuel_transform': 2, 'mileage_norm': 152000, 'vol_engine_norm': 1598},
    {'year': 2000, 'mark_transform': 8, 'fuel_transform': 4, 'mileage_norm': 98258,  'vol_engine_norm': 1598},
```

```
    {'year': 2005, 'mark_transform': 9, 'fuel_transform': 5, 'mileage_norm': 109330, 'vol_engine_norm': 1398}
]
df_testeo = pd.DataFrame(datos)                                                    #Creamos un data set con los datos de prueba
scale= StandardScaler()                                                            #cargamos el metodo para estandarizar
xnorm2= df_testeo[['mileage_norm','vol_engine_norm']]                              #aasignamos las columnas a normalizar
scaledx2 = scale.fit_transform(xnorm2)                                             #Aplicamos la normalizacion
auxnorm2 = pd.DataFrame(scaledx2, columns=['mileage_norm','vol_engine_norm'])      #Guardamos en data set auxiliar
df_testeo[['mileage_norm','vol_engine_norm']] = auxnorm2[['mileage_norm','vol_engine_norm']]#Guardamos en data set original
y_predtesteo = arbol_precios.predict(df_testeo)                                    #Ejecutamos la prediccion
df_testeo['resultadols'] = pd.DataFrame(y_predtesteo)                              #Almacenamos el resulatdo
y_proba = arbol_precios.predict_proba(df_testeo.drop(['resultadols'], axis=1))     #Cargamos datos para la probabilidad
porcentaje = np.max(arbol_precios.predict_proba(df_testeo.drop(['resultadols'], axis=1)), axis=1)   #Ejecutamos la probabilidad
df_testeo['porcentaje'] = pd.DataFrame(porcentaje)                                 #Guardamos el resultado
print(df_testeo)                                                                   #Mostramos datos finales
df.to_csv('car_power_bi.csv')
#---------Testeo End------#
```