Carol Arrieta Moreno

A01275465

Primera entrega (Portafolio Análisis)

```
import numpy as np
from pandas import DataFrame
import pandas as pd
import seaborn as sns
%matplotlib inline
```

```
#Conversion de CSV a DataFrame
train= pd.read_csv(r"precios_autos.csv")
train.head(10)
```

	symboling	CarName	fueltype	carbody	drivewheel	enginelocation	wheelbase	car
0	3	alfa-romero giulia	gas	convertible	rwd	front	88.6	
1	3	alfa-romero stelvio	gas	convertible	rwd	front	88.6	
2	1	alfa-romero Quadrifoglio	gas	hatchback	rwd	front	94.5	
3	2	audi 100 ls	gas	sedan	fwd	front	99.8	
4	2	audi 100ls	gas	sedan	4wd	front	99.4	
5	2	audi fox	gas	sedan	fwd	front	99.8	
6	1	audi 100ls	gas	sedan	fwd	front	105.8	
7	1	audi 5000	gas	wagon	fwd	front	105.8	
8	1	audi 4000	gas	sedan	fwd	front	105.8	
9	0	audi 5000s (diesel)	gas	hatchback	4wd	front	99.5	

10 rows × 21 columns

```
#Analisis de la Estadistica Descrpitiva de los datos
print(train.describe())
```

```
         symboling   wheelbase   carlength    carwidth   carheight  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
mean     0.834146   98.756585  174.049268   65.907805   53.724878
std      1.245307    6.021776   12.337289    2.145204    2.443522
min     -2.000000   86.600000  141.100000   60.300000   47.800000
25%      0.000000   94.500000  166.300000   64.100000   52.000000
50%      1.000000   97.000000  173.200000   65.500000   54.100000
75%      2.000000  102.400000  183.100000   66.900000   55.500000
max      3.000000  120.900000  208.100000   72.300000   59.800000

         curbweight  enginesize      stroke  compressionratio  horsepower  \
count   205.000000  205.000000  205.000000        205.000000  205.000000
mean   2555.565854  126.907317    3.255415         10.142537  104.117073
std     520.680204   41.642693    0.313597          3.972040   39.544167
min    1488.000000   61.000000    2.070000          7.000000   48.000000
25%    2145.000000   97.000000    3.110000          8.600000   70.000000
50%    2414.000000  120.000000    3.290000          9.000000   95.000000
75%    2935.000000  141.000000    3.410000          9.400000  116.000000
max    4066.000000  326.000000    4.170000         23.000000  288.000000

            peakrpm     citympg  highwaympg         price
count   205.000000  205.000000  205.000000    205.000000
mean   5125.121951   25.219512   30.751220  13276.710571
std     476.985643    6.542142    6.886443   7988.852332
min    4150.000000   13.000000   16.000000   5118.000000
25%    4800.000000   19.000000   25.000000   7788.000000
50%    5200.000000   24.000000   30.000000  10295.000000
75%    5500.000000   30.000000   34.000000  16503.000000
max    6600.000000   49.000000   54.000000  45400.000000
```

```
#Comando para ver el tipo de dato de cada columna en el dataframe y cantidad de valores no nu
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   symboling         205 non-null    int64  
 1   CarName           205 non-null    object 
 2   fueltype          205 non-null    object 
 3   carbody           205 non-null    object 
 4   drivewheel        205 non-null    object 
 5   enginelocation    205 non-null    object 
 6   wheelbase         205 non-null    float64
 7   carlength         205 non-null    float64
 8   carwidth          205 non-null    float64
 9   carheight         205 non-null    float64
 10  curbweight        205 non-null    int64  
 11  enginetype        205 non-null    object 
 12  cylindernumber    205 non-null    object 
 13  enginesize        205 non-null    int64  
 14  stroke            205 non-null    float64
 15  compressionratio  205 non-null    float64
```

```
 16  horsepower        205 non-null    int64
 17  peakrpm           205 non-null    int64
 18  citympg           205 non-null    int64
 19  highwaympg        205 non-null    int64
 20  price             205 non-null    float64
dtypes: float64(7), int64(7), object(7)
memory usage: 33.8+ KB
```

```
#Transformacion Binaria a la columna fueltype
from sklearn import preprocessing
label = preprocessing.LabelEncoder()

train['fueltype']= label.fit_transform(train['fueltype'])
print(train['fueltype'].unique())
```

```
[1 0]
```

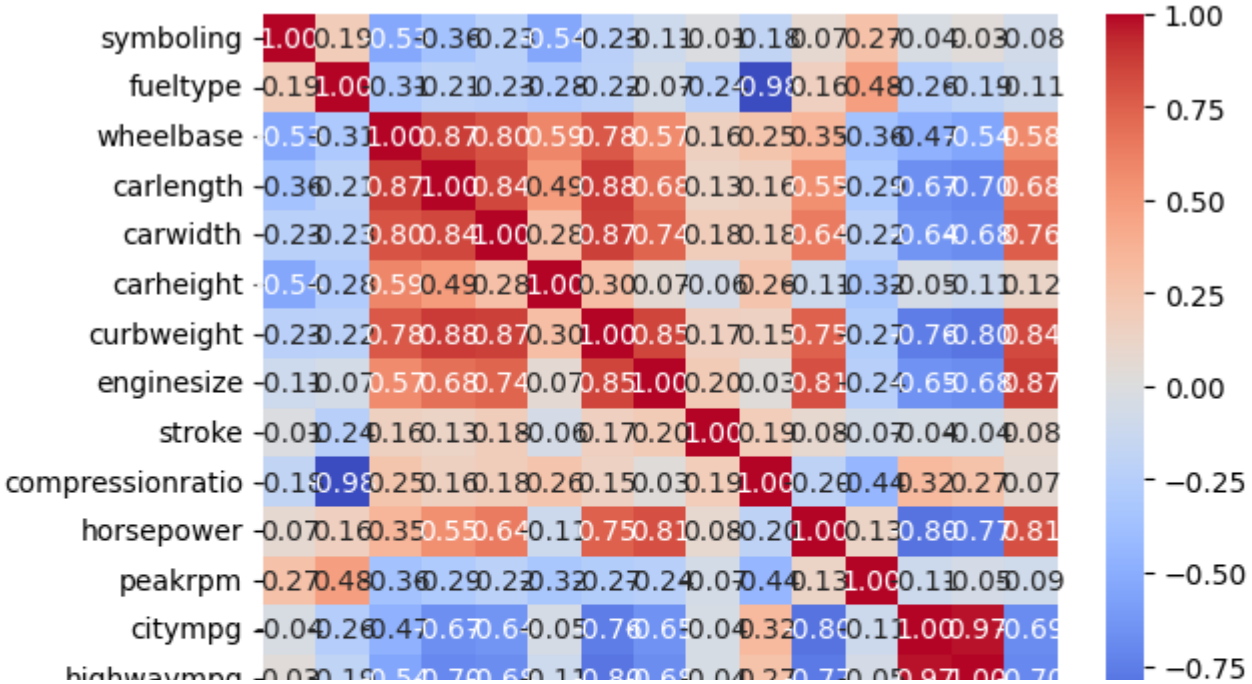
```
train.head()
```

	symboling	CarName	fueltype	carbody	drivewheel	enginelocation	wheelbase	ca
0	3	alfa-romero giulia	1	convertible	rwd	front	88.6	
1	3	alfa-romero stelvio	1	convertible	rwd	front	88.6	
2	1	alfa-romero Quadrifoglio	1	hatchback	rwd	front	94.5	
3	2	audi 100 ls	1	sedan	fwd	front	99.8	
4	2	audi 100ls	1	sedan	4wd	front	99.4	

5 rows × 21 columns

```
# Mapa de correlación entre cada variable
sns.heatmap(train.corr(), annot = True, fmt = '.2f', cmap = 'coolwarm')
```

```
<ipython-input-8-7bfdaa43e6ed>:2: FutureWarning: The default value of numeric_only in Da
  sns.heatmap(train.corr(), annot = True, fmt = '.2f', cmap = 'coolwarm')
<Axes: >
```



De acuerdo con el mapa de calor, las variables que mas impacto tienen en el precio son horsepower, enginesize, curbweight, carwidth, carlength y wheelbase. Por lo tanto son las variables que se utilizarán.

Limpieza de Datos

```
#Contamos los datos no nulos
train.count()
```

```
symboling           205
CarName             205
fueltype            205
carbody             205
drivewheel          205
enginelocation      205
wheelbase           205
carlength           205
carwidth            205
carheight           205
curbweight          205
enginetype          205
cylindernumber      205
enginesize          205
stroke              205
compressionratio    205
horsepower          205
peakrpm             205
```

```
citympg            205
highwaympg         205
price              205
dtype: int64
```

```
#Contamos los datos nulos de todas las columnas
train.isnull().sum()
```

```
symboling          0
CarName            0
fueltype           0
carbody            0
drivewheel         0
enginelocation     0
wheelbase          0
carlength          0
carwidth           0
carheight          0
curbweight         0
enginetype         0
cylindernumber     0
enginesize         0
stroke             0
compressionratio   0
horsepower         0
peakrpm            0
citympg            0
highwaympg         0
price              0
dtype: int64
```

No existen valores nulos por los que dejamos el dataset sin ningun cambio

Variables Cuantitativas

Variables con valores numéricos

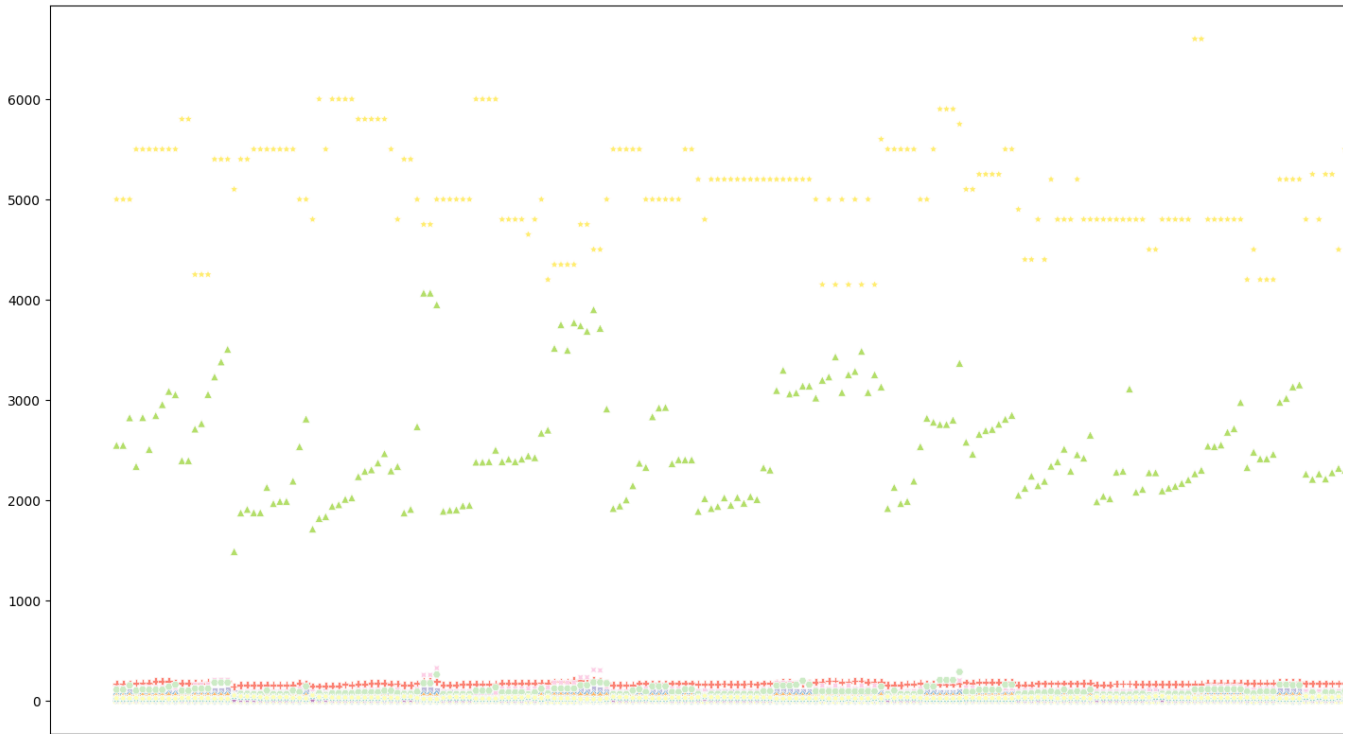
```
#coeficiente de correlacion
data=train.drop(['price'],1)
correlacion = data.corr()
print(correlacion)
```

```
                  symboling  fueltype  wheelbase  carlength  carwidth  \
symboling          1.000000  0.194311  -0.531954  -0.357612 -0.232919
fueltype           0.194311  1.000000  -0.308346  -0.212679 -0.233880
wheelbase         -0.531954 -0.308346   1.000000   0.874587  0.795144
carlength         -0.357612 -0.212679   0.874587   1.000000  0.841118
carwidth          -0.232919 -0.233880   0.795144   0.841118  1.000000
carheight         -0.541038 -0.284631   0.589435   0.491029  0.279210
curbweight        -0.227691 -0.217275   0.776386   0.877728  0.867032
```

```
enginesize        -0.105790 -0.069594   0.569329   0.683360  0.735433
stroke            -0.008735 -0.241829   0.160959   0.129533  0.182942
compressionratio  -0.178515 -0.984356   0.249786   0.158414  0.181129
horsepower         0.070873  0.163926   0.353294   0.552623  0.640732
peakrpm            0.273606  0.476883  -0.360469  -0.287242 -0.220012
citympg           -0.035823 -0.255963  -0.470414  -0.670909 -0.642704
highwaympg         0.034606 -0.191392  -0.544082  -0.704662 -0.677218

                  carheight  curbweight  enginesize    stroke  \
symboling         -0.541038   -0.227691   -0.105790 -0.008735
fueltype          -0.284631   -0.217275   -0.069594 -0.241829
wheelbase          0.589435    0.776386    0.569329  0.160959
carlength          0.491029    0.877728    0.683360  0.129533
carwidth           0.279210    0.867032    0.735433  0.182942
carheight          1.000000    0.295572    0.067149 -0.055307
curbweight         0.295572    1.000000    0.850594  0.168790
enginesize         0.067149    0.850594    1.000000  0.203129
stroke            -0.055307    0.168790    0.203129  1.000000
compressionratio   0.261214    0.151362    0.028971  0.186110
horsepower        -0.108802    0.750739    0.809769  0.080940
peakrpm           -0.320411   -0.266243   -0.244660 -0.067964
citympg           -0.048640   -0.757414   -0.653658 -0.042145
highwaympg        -0.107358   -0.797465   -0.677470 -0.043931

                  compressionratio  horsepower   peakrpm   citympg  highwaympg
symboling                -0.178515    0.070873  0.273606 -0.035823    0.034606
fueltype                 -0.984356    0.163926  0.476883 -0.255963   -0.191392
wheelbase                 0.249786    0.353294 -0.360469 -0.470414   -0.544082
carlength                 0.158414    0.552623 -0.287242 -0.670909   -0.704662
carwidth                  0.181129    0.640732 -0.220012 -0.642704   -0.677218
carheight                 0.261214   -0.108802 -0.320411 -0.048640   -0.107358
curbweight                0.151362    0.750739 -0.266243 -0.757414   -0.797465
enginesize                0.028971    0.809769 -0.244660 -0.653658   -0.677470
stroke                    0.186110    0.080940 -0.067964 -0.042145   -0.043931
compressionratio          1.000000   -0.204326 -0.435741  0.324701    0.265201
horsepower               -0.204326    1.000000  0.131073 -0.801456   -0.770544
peakrpm                  -0.435741    0.131073  1.000000 -0.113544   -0.054275
citympg                   0.324701   -0.801456 -0.113544  1.000000    0.971337
highwaympg                0.265201   -0.770544 -0.054275  0.971337    1.000000
<ipython-input-11-e34befd2d916>:2: FutureWarning: In a future version of pandas all argu
  data=train.drop(['price'],1)
<ipython-input-11-e34befd2d916>:3: FutureWarning: The default value of numeric_only in D
  correlacion = data.corr()
```

```
import matplotlib.pyplot as plt

# Diagrama de dispercion
plt.figure(figsize=(20, 10))
sns.scatterplot(data=train.drop(['price'],1) , palette="Set3")


# Show the plot
plt.show()
```

```
<ipython-input-12-6d5d94ff3a8e>:5: FutureWarning: In a future version of pandas all argu
  sns.scatterplot(data=train.drop(['price'],1) , palette="Set3")
```

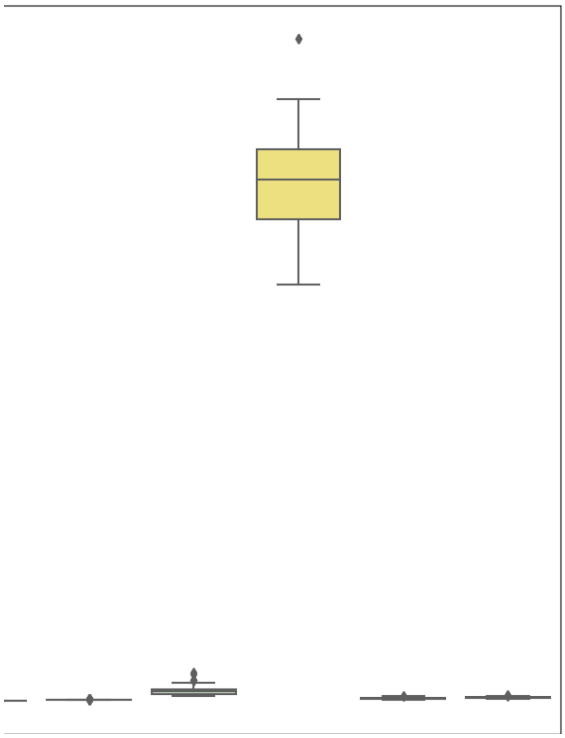



```
# Histograma
plt.figure(figsize=(20, 10))
sns.histplot(data=train.drop(['price'],1) , palette="Set3")

# Show the plot
plt.show()
```

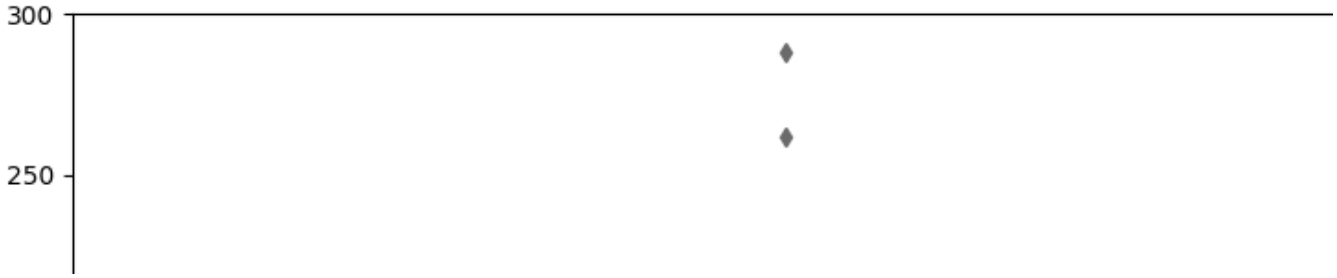
```
<ipython-input-13-78fe92ec995b>:3: FutureWarning: In a future version of pandas all argu
  sns.histplot(data=train.drop(['price'],1) , palette="Set3")
```

```
# Boxplot
plt.figure(figsize=(20, 10))
sns.boxplot(data=train.drop(['price'],1) , palette="Set3")

# Show the plot
plt.show()
```

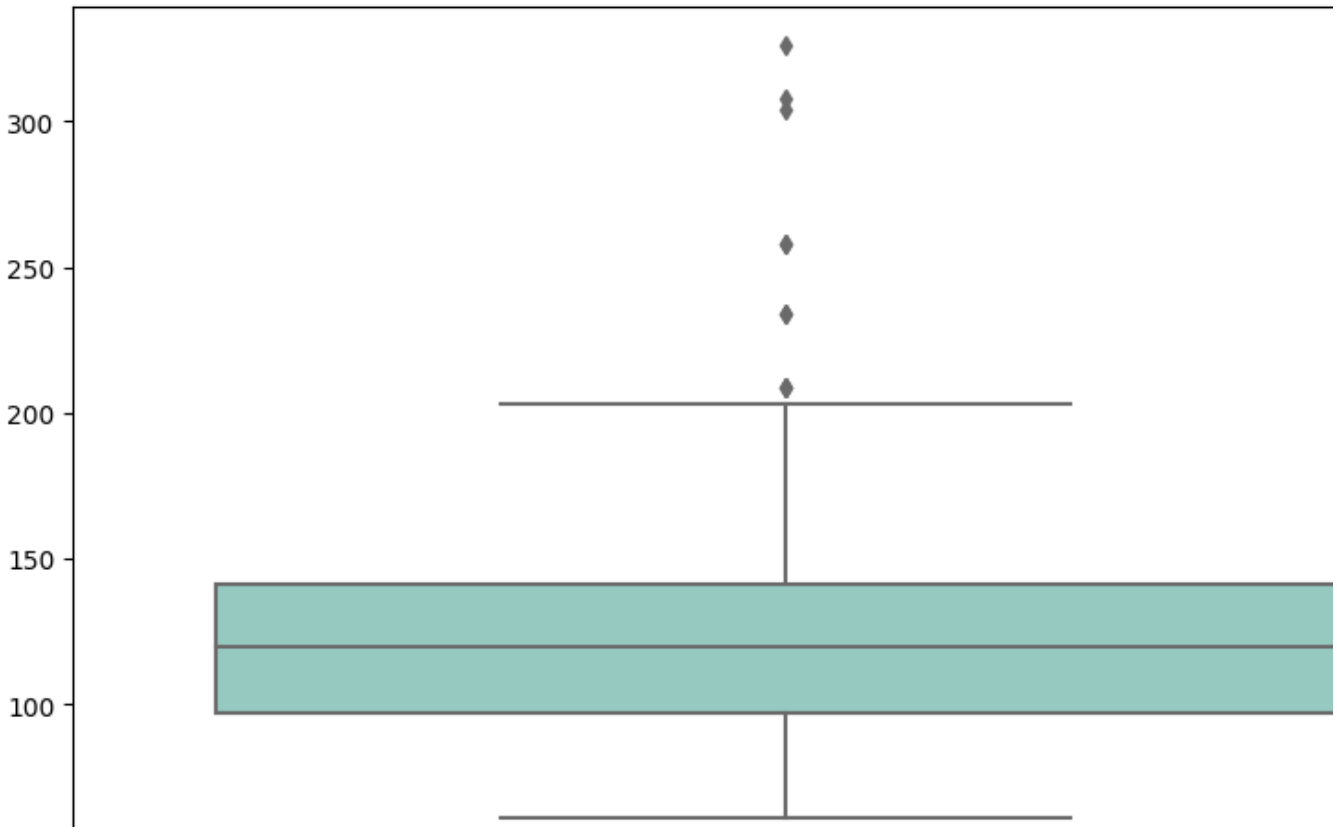
```
/ersion of pandas all arguments of DataFrame.drop except for the argument 'labels' will b
```



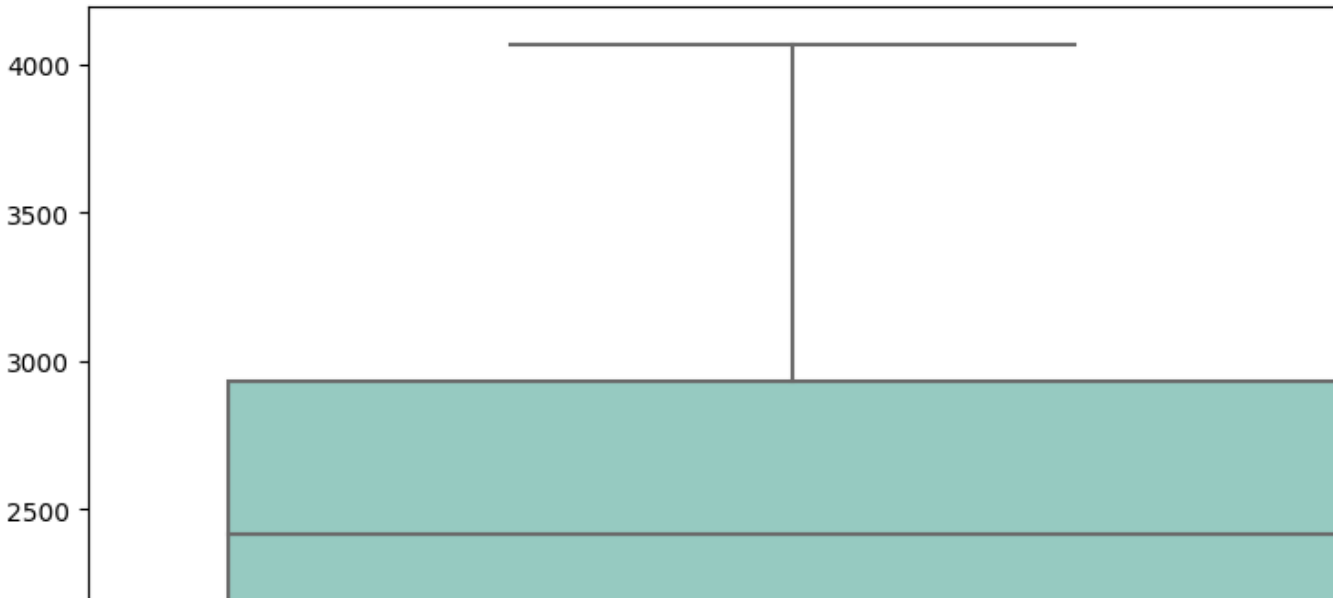
```
# Create a boxplot using Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(data=train.horsepower, palette="Set3")

# Show the plot
plt.show()
```



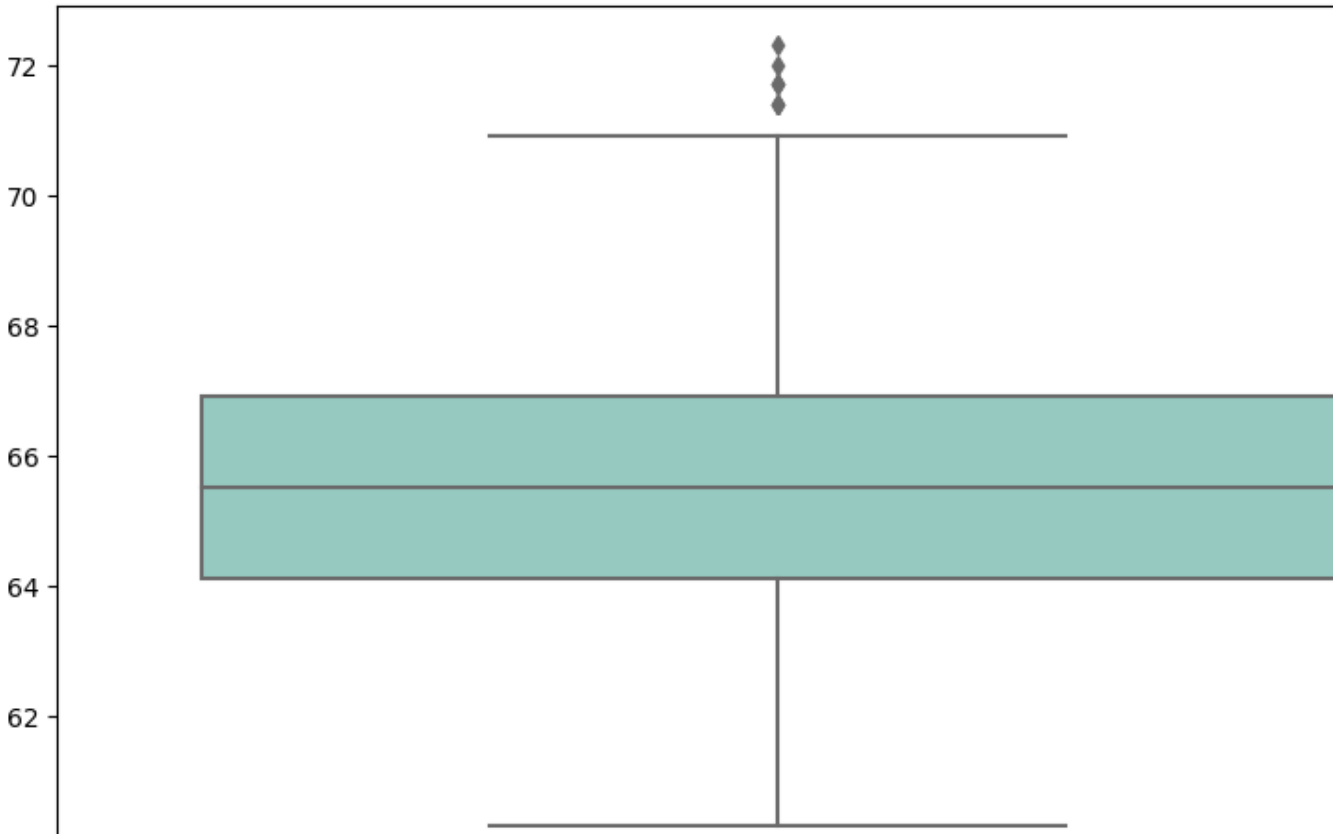

```
# Create a boxplot using Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(data=train.enginesize, palette="Set3")

# Show the plot
plt.show()
```




```
# Create a boxplot using Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(data=train.curbweight, palette="Set3")

# Show the plot
plt.show()
```

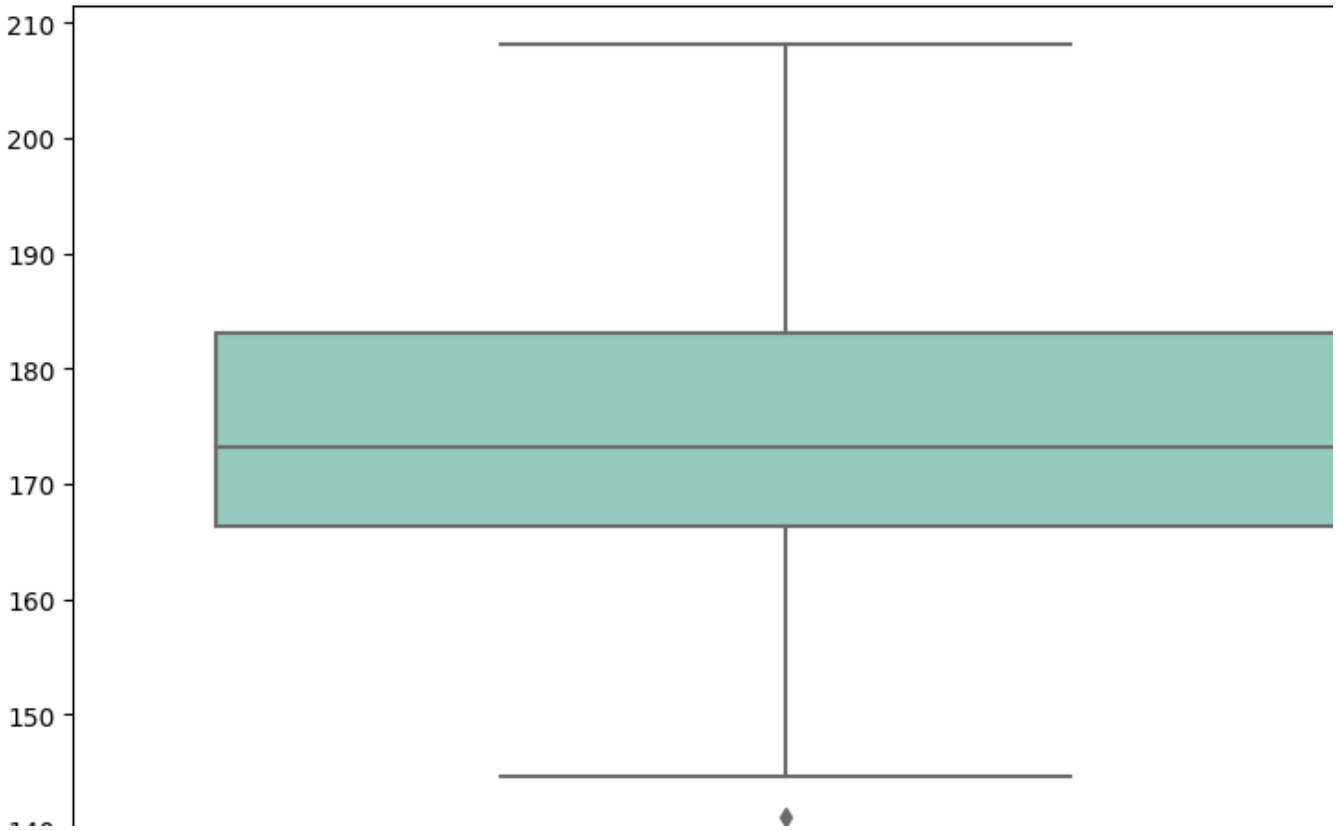



```
# Create a boxplot using Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(data=train.carwidth, palette="Set3")

# Show the plot
plt.show()
```

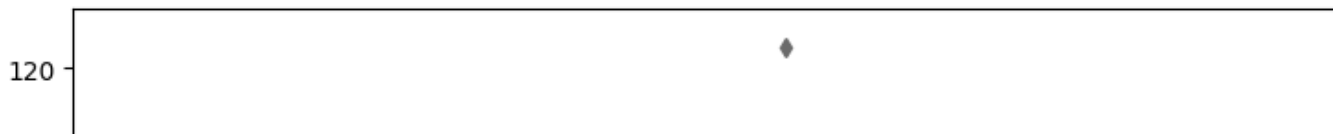



```
# Create a boxplot using Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(data=train.carlength, palette="Set3")
```

```
# Show the plot
plt.show()
```




```
# Create a boxplot using Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(data=train.wheelbase, palette="Set3")

# Show the plot
plt.show()
```



Observamos como existen datos atipicos en nuestro dataset, por lo cual es necesario ver como nos afectan para despues tomar una decisión sobre que hacer con estos, si escalarlos o normalizarlos

Variables categóricas

Variables sin orden lógico, no tienen una forma de medición

```
resumen_categorias = data.value_counts()
print(resumen_categorias)
```

```
symboling  CarName                 fueltype  carbody    drivewheel  enginelocation  whe
-2         volvo 145e (sw)         1         sedan      rwd         front           104
 1         toyota corolla liftback 1         sedan      rwd         front           94
           nissan rogue            1         sedan      fwd         front           94
           nissan titan            1         wagon      fwd         front           94
           plymouth cricket        1         hatchback  fwd         front           93

 0         subaru baja             1         sedan      fwd         front           97
           subaru brz              1         sedan      fwd         front           97
           subaru dl               1         wagon      4wd         front           96

 3         vw rabbit               1         hatchback  fwd         front           94
Length: 205, dtype: int64
```

```
plt.figure(figsize=(20, 6))
sns.countplot(data=train.drop(['price'],1))

plt.show()
```

```
<ipython-input-22-25fe2b914150>:2: FutureWarning: In a future version of pandas all argu
  sns.countplot(data=train.drop(['price'],1))
```

Conclusión primera parte

Muchas de las variables cuentan con datos atipicos, se tendra que ver si estos afectan nuestros siguientes pasos en el modelo, de igual manera la forma de medida de cada variable es en diferentes rangos, por lo que si realizaremos una regresión, necesitaremos normalizar los valores para tener los mejores resultados posibles.

Las 6 variables con las que estaré trabajando son horsepower, enginesize, curbweight, carwidth, carlength y wheelbase, esto debido a que en el mapa de correlación se observó que son las que mas impacto tienen en el precio de los autos.

Construcción de un modelo estadístico base

La pregunta base que vamos a buscar responder es si las diferentes variables que tenemos se diferencian mucho una de otras afectando así el precio del automovil.

Para esto se hará uso de dos herramientas estadisticas, Anova y pruebas de hipótesis.

ANOVA lo utilizaremos para comparar medias en múltiples grupos, mientras que las pruebas de hipótesis va a un enfoque más general para evaluar afirmaciones.

```
import scipy.stats as stats

# Realiza una prueba t de Student de dos muestras
t_statistic, p_value = stats.ttest_ind(train.horsepower, train.enginesize)

# Compara el valor p con el nivel de significancia
alpha = 0.05
p_value
```

```
2.5344402766634857e-08
```

```
if p_value < alpha:
    print("Rechazamos la hipótesis nula.")
else:
    print("No podemos rechazar la hipótesis nula.")
```

```
Rechazamos la hipótesis nula.
```

```
# Realiza una prueba t de Student de dos muestras
t_statistic, p_value = stats.ttest_ind(train.curbweight, train.carwidth)
```

```
# Compara el valor p con el nivel de significancia
alpha = 0.05
p_value
```

```
8.584918879099727e-226
```

```
if p_value < alpha:
    print("Rechazamos la hipótesis nula.")
else:
    print("No podemos rechazar la hipótesis nula.")
```

```
Rechazamos la hipótesis nula.
```

```
# Realiza una prueba t de Student de dos muestras
t_statistic, p_value = stats.ttest_ind(train.carlength, train.wheelbase)

# Compara el valor p con el nivel de significancia
alpha = 0.05
p_value
```

```
2.214091958718416e-248
```

```
if p_value < alpha:
    print("Rechazamos la hipótesis nula.")
else:
    print("No podemos rechazar la hipótesis nula.")
```

```
Rechazamos la hipótesis nula.
```

```
# Realizamos un ANOVA
f_statistic, p_value = stats.f_oneway(train.horsepower, train.enginesize, train.curbweight, t

if p_value < 0.05:  # 0.05 de umbral
    print("Existen diferencias significativas entre al menos dos grupos.")
else:
    print("No hay diferencias significativas entre los grupos.")
```

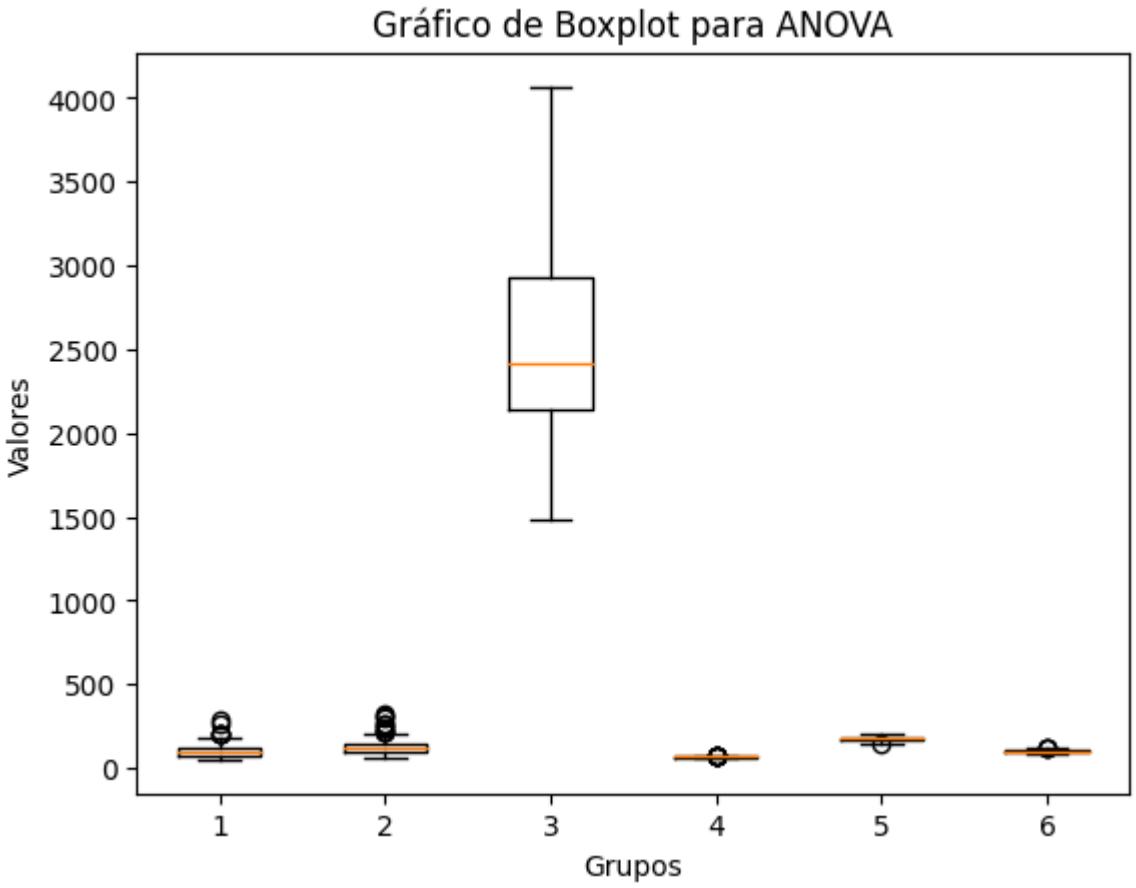
```
Existen diferencias significativas entre al menos dos grupos.
```

```
data = [train.horsepower, train.enginesize, train.curbweight, train.carwidth, train.carlength
```

```
import seaborn as sns
import matplotlib.pyplot as plt

print("Valor F:", f_statistic)
print("Valor p:", p_value)
```

```
# Crear un gráfico de barras para visualizar los datos

plt.boxplot(data)
plt.title('Gráfico de Boxplot para ANOVA')
plt.xlabel('Grupos')
plt.ylabel('Valores')

# Mostrar el gráfico
plt.show()
```

```
Valor F: 4456.267132126095
Valor p: 0.0
```

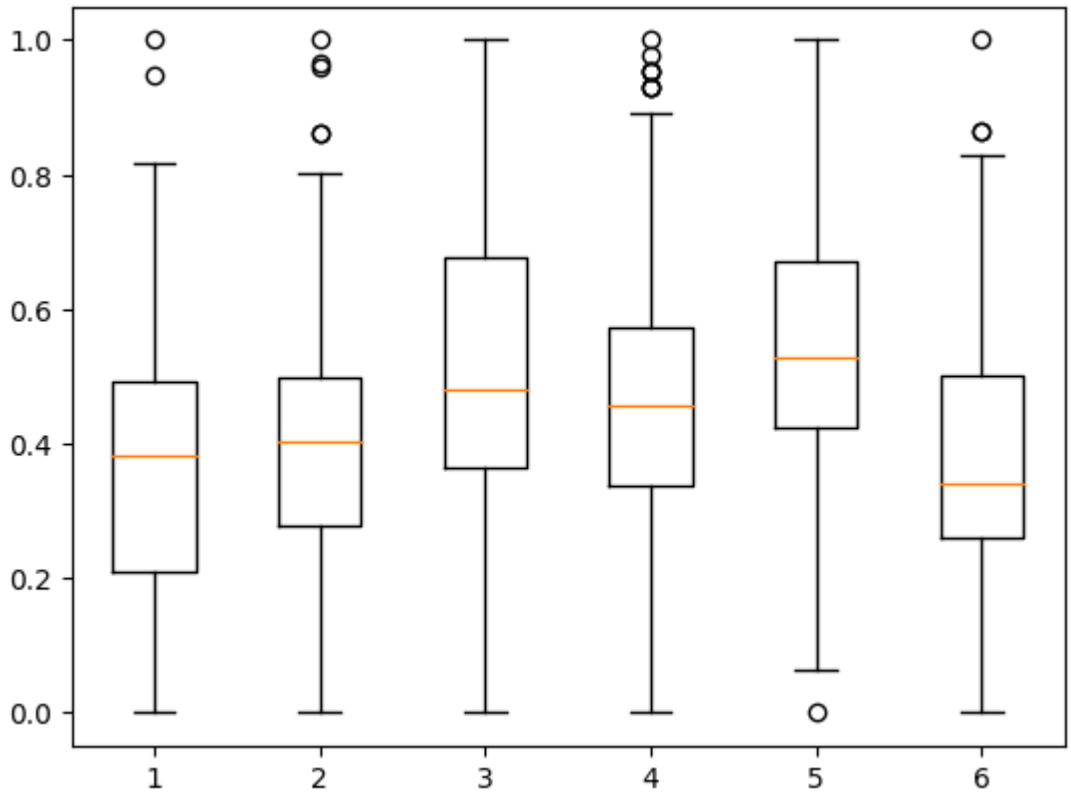


Vamos a eliminar datos atipicos y realizar el proceso de normalización para ver como cambia nuestro resultado

```
train.horsepower = np.log(train.horsepower)
train.enginesize = np.log(train.enginesize)
train.curbweight = np.log(train.curbweight)
train.carwidth = np.log(train.carwidth)
train.carlength = np.log(train.carlength)
train.wheelbase = np.log(train.wheelbase)
```

```
# Calcula los valores mínimos y máximos de cada característica
min_vals = np.min(train.horsepower, axis=0)
max_vals = np.max(train.horsepower, axis=0)

# Realiza la normalización Min-Max
horsepower = (train.horsepower - min_vals) / (max_vals - min_vals)



# Calcula los valores mínimos y máximos de cada característica
min_vals = np.min(train.enginesize, axis=0)
max_vals = np.max(train.enginesize, axis=0)

# Realiza la normalización Min-Max
enginesize = (train.enginesize - min_vals) / (max_vals - min_vals)



# Calcula los valores mínimos y máximos de cada característica
min_vals = np.min(train.curbweight, axis=0)
max_vals = np.max(train.curbweight, axis=0)

# Realiza la normalización Min-Max
curbweight = (train.curbweight - min_vals) / (max_vals - min_vals)


# Calcula los valores mínimos y máximos de cada característica
min_vals = np.min(train.carwidth, axis=0)
max_vals = np.max(train.carwidth, axis=0)

# Realiza la normalización Min-Max
carwidth = (train.carwidth - min_vals) / (max_vals - min_vals)


# Calcula los valores mínimos y máximos de cada característica
min_vals = np.min(train.carlength, axis=0)
max_vals = np.max(train.carlength, axis=0)

# Realiza la normalización Min-Max
carlength = (train.carlength - min_vals) / (max_vals - min_vals)


# Calcula los valores mínimos y máximos de cada característica
min_vals = np.min(train.wheelbase, axis=0)
max_vals = np.max(train.wheelbase, axis=0)

# Realiza la normalización Min-Max
wheelbase = (train.wheelbase - min_vals) / (max_vals - min_vals)

X_normalized = [horsepower, enginesize, curbweight, carwidth, carlength, wheelbase]
```

```
# Crear un gráfico de barras para visualizar los datos

plt.boxplot(X_normalized)
# Mostrar el gráfico
plt.show()
```



Ya que normalizamos los datasets y eliminamos loa valores atipicos volvemos a calcular el anova y realizamos la prueba de hipotesis para ver que tanto afectó

```
# Realizamos un ANOVA
f_statistic, p_value = stats.f_oneway(horsepower, enginesize, curbweight, carwidth, carlength

if p_value < 0.05:  # 0.05 de umbral
    print("Existen diferencias significativas entre al menos dos grupos.")
else:
    print("No hay diferencias significativas entre los grupos.")
```

```
Existen diferencias significativas entre al menos dos grupos.
```

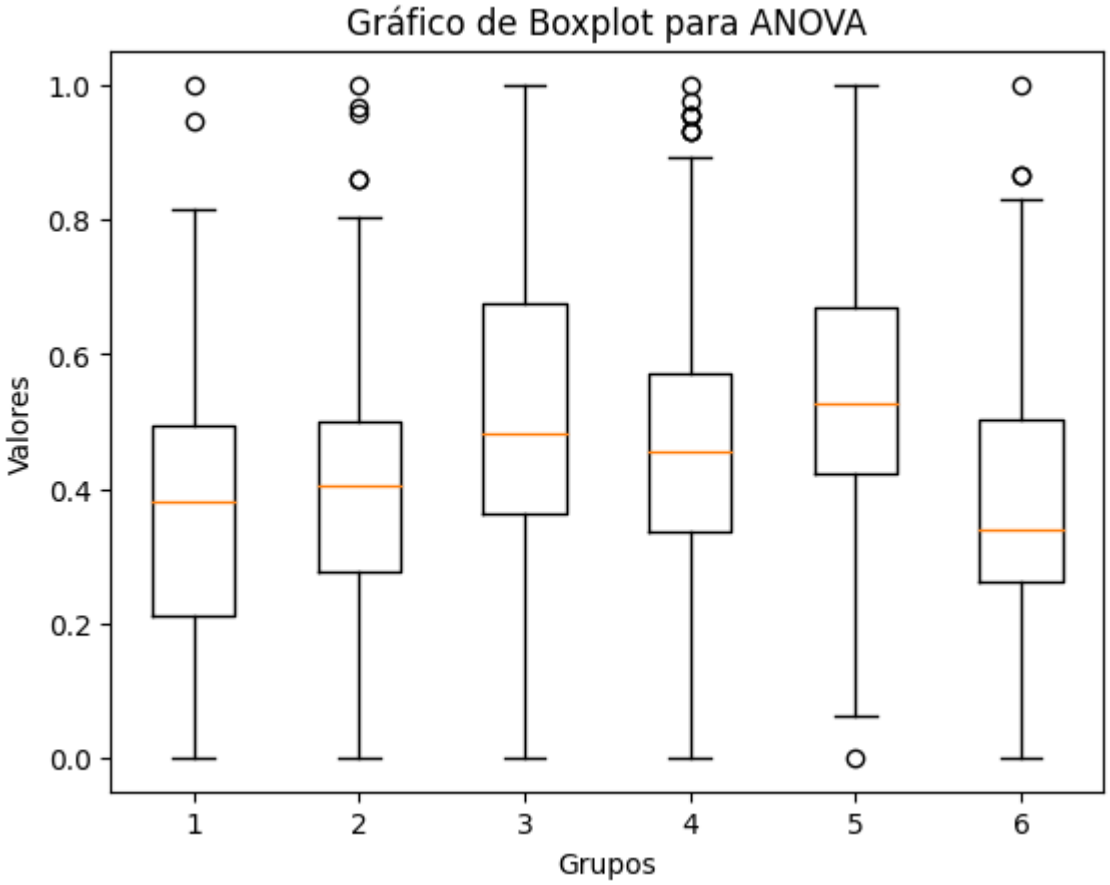
```
p_value
```

```
1.1817110158648524e-24
```

```
data = [horsepower, enginesize, curbweight, carwidth, carlength, wheelbase]
```

```
import seaborn as sns
import matplotlib.pyplot as plt

print("Valor F:", f_statistic)
print("Valor p:", p_value)

# Crear un gráfico de barras para visualizar los datos

plt.boxplot(data)
plt.title('Gráfico de Boxplot para ANOVA')
plt.xlabel('Grupos')
plt.ylabel('Valores')

# Mostrar el gráfico
plt.show()
```

```
Valor F: 25.625277295026756
Valor p: 1.1817110158648524e-24
```



Conclusión segunda parte

En este caso realicé una normalización a pesar de que no se fuera a realizar una regresión para ver el impacto que tenia en el ANOVA, fue mas una prueba que yo queria hacer para ver como afectaba,

sin embargo la normalización es más para cuando se realizará regresion y se necesita que los rangos sean los mismos.

Para los datos atipicos no los eliminé por completo porque al borrarlos completamente se perderá algo de información ya que son varios los datos atipicos, por los que se disminuyeron en cierta medida pero no los eliminé completamente.

Podemos ver que al inicio, antes de la normalización las variables se diferenciaban mucho de las otras en cuanto al rango de medida. Y como respuesta a la pregunta generada es que si, las cariables en conjunto se diferencian mucho de ellas mismas por lo que afectan al precio en si, y se podría intentar calcular el precio de un automovil con cada variable individualmente o encontrando el conjunto de variables que tengan la relación suficiente para obtener los mejores resultados.

✓ 0s completed at 9:55 PM