

# Ozner Axel Leyva Mariscal

A01742377

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.diagnostic import het_breuschpagan

# Cargar los datos
df = pd.read_csv("precios_autos.csv")

# Seleccionar las variables del primer grupo y el precio
variables = ['wheelbase', 'fueltype', 'horsepower', 'price']
df_subset = df[variables]

df_subset.head()
```

```
Out[ ]:   wheelbase fueltype horsepower price
0      88.6     gas        111  13495.0
1      88.6     gas        111  16500.0
2      94.5     gas        154  16500.0
3      99.8     gas        102  13950.0
4      99.4     gas        115  17450.0
```

## Exploración de la base de datos

```
In [ ]: df_subset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 4 columns):
 #   Column      Non-Null Count Dtype  
--- 
 0   wheelbase    205 non-null   float64 
 1   fueltype     205 non-null   object  
 2   horsepower   205 non-null   int64   
 3   price        205 non-null   float64 
dtypes: float64(2), int64(1), object(1)
memory usage: 6.5+ KB
```

## Calcula medidas estadísticas apropiadas para las variables:

Cuantitativas:

```
In [ ]: stats_quantitative = df_subset[['wheelbase', 'horsepower', 'price']].describe()
print("\nEstadísticas descriptivas para variables cuantitativas:")
stats_quantitative
```

Estadísticas descriptivas para variables cuantitativas:

```
Out[ ]:   wheelbase  horsepower      price
count    205.000000  205.000000  205.000000
mean     98.756585  104.117073 13276.710571
std      6.021776  39.544167  7988.852332
min     86.600000  48.000000  5118.000000
25%    94.500000  70.000000  7788.000000
50%    97.000000  95.000000 10295.000000
75%   102.400000 116.000000 16503.000000
max   120.900000 288.000000 45400.000000
```

Cualitativas:

```
In [ ]: fuel_type_freq = df_subset['fueltype'].value_counts(normalize=True)
print("\nFrecuencias relativas para tipo de combustible:")
print(fuel_type_freq)
```

Frecuencias relativas para tipo de combustible:  
gas 0.902439  
diesel 0.097561  
Name: fueltype, dtype: float64

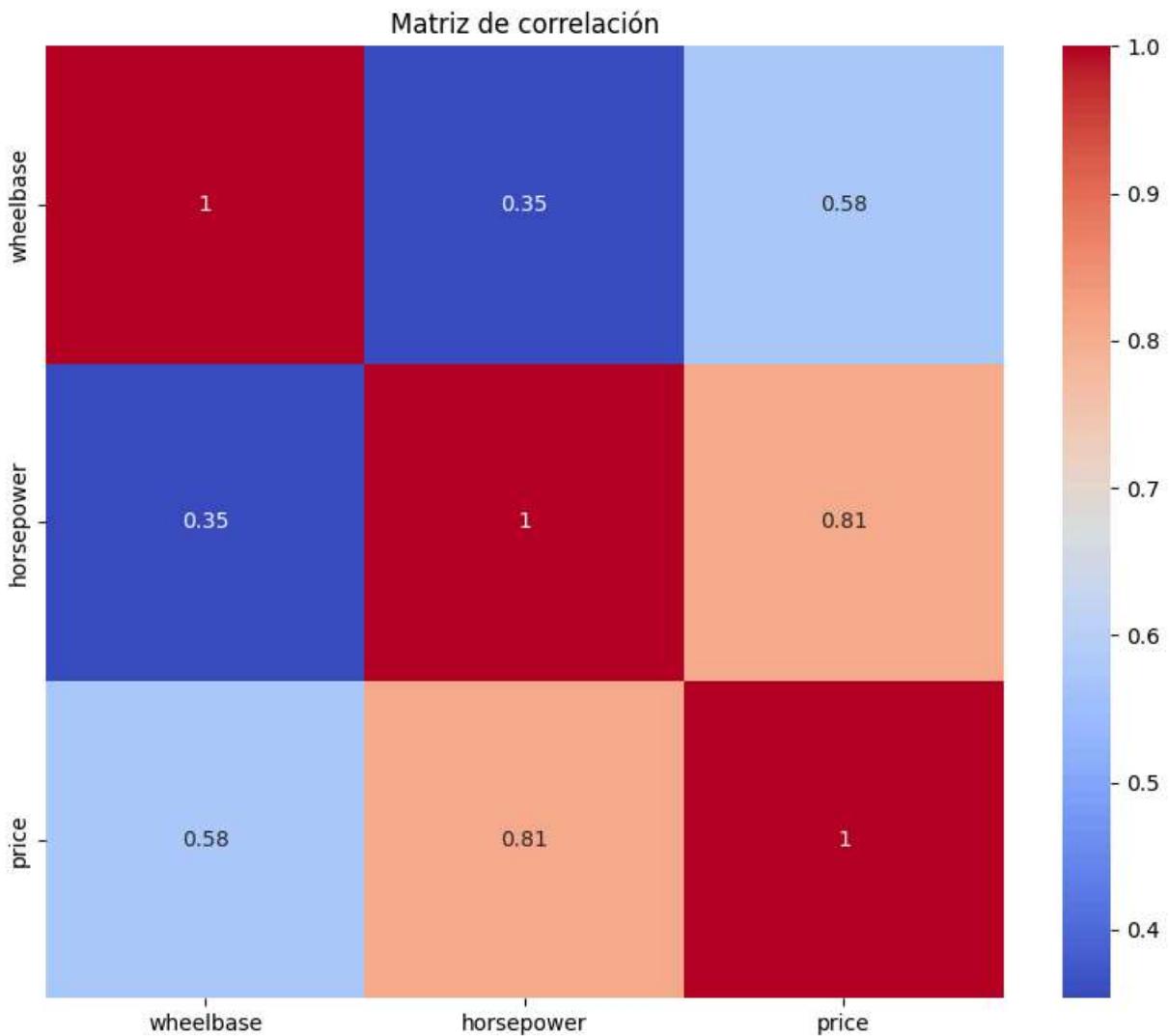
## Analiza la correlación entre las variables (analiza posible colinealidad entre las variables)

```
In [ ]: # Matriz de correlación
correlation_matrix = df_subset[['wheelbase', 'horsepower', 'price']].corr()
print("\nMatriz de correlación:")
print(correlation_matrix)

# Visualización de la matriz de correlación
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Matriz de correlación')
plt.show()
```

Matriz de correlación:

|            | wheelbase | horsepower | price    |
|------------|-----------|------------|----------|
| wheelbase  | 1.000000  | 0.353294   | 0.577816 |
| horsepower | 0.353294  | 1.000000   | 0.808139 |
| price      | 0.577816  | 0.808139   | 1.000000 |



Explora los datos usando herramientas de visualización (si lo consideras necesario):

```
In [ ]: # Boxplots para variables cuantitativas
plt.figure(figsize=(15, 5))
for i, var in enumerate(['wheelbase', 'horsepower', 'price']):
    plt.subplot(1, 3, i+1)
    sns.boxplot(x=df_subset[var])
    plt.title(f'Boxplot de {var}')
plt.tight_layout()
plt.show()

# Histogramas
plt.figure(figsize=(15, 5))
for i, var in enumerate(['wheelbase', 'horsepower', 'price']):
    plt.subplot(1, 3, i+1)
    sns.histplot(df_subset[var], kde=True)
```

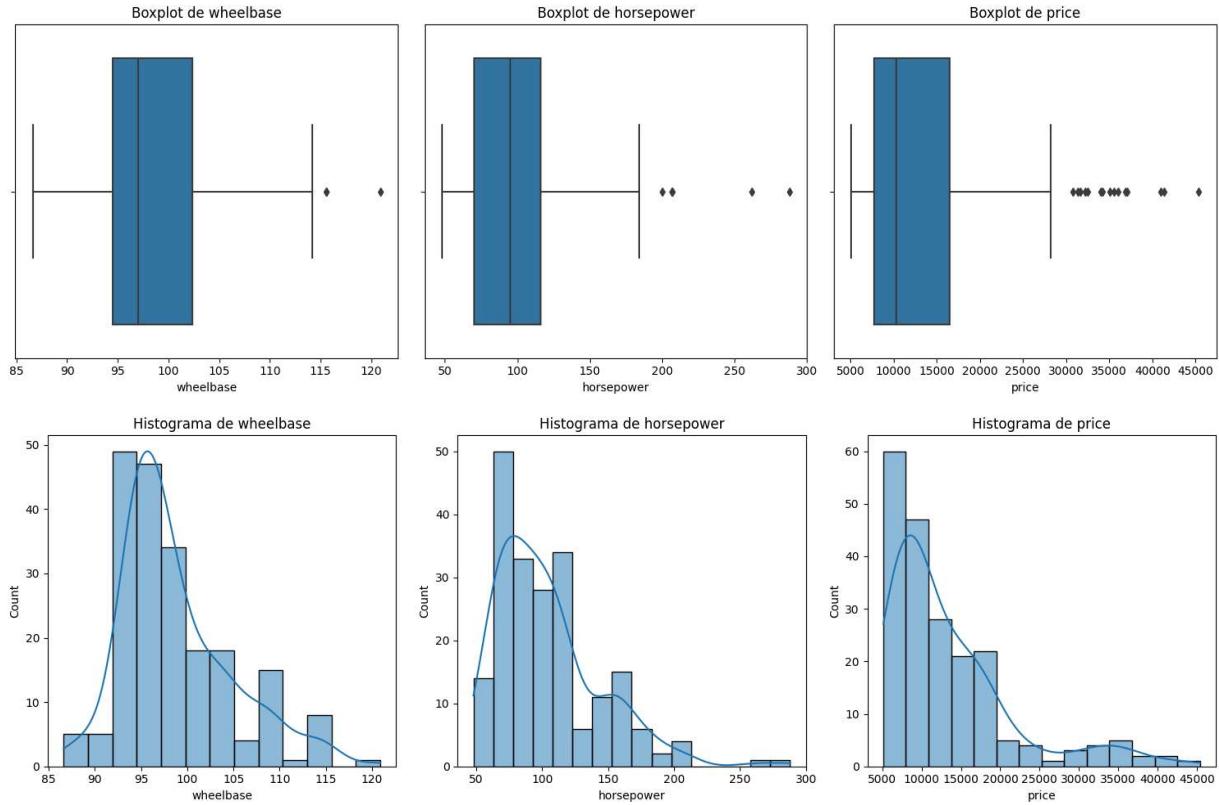
```

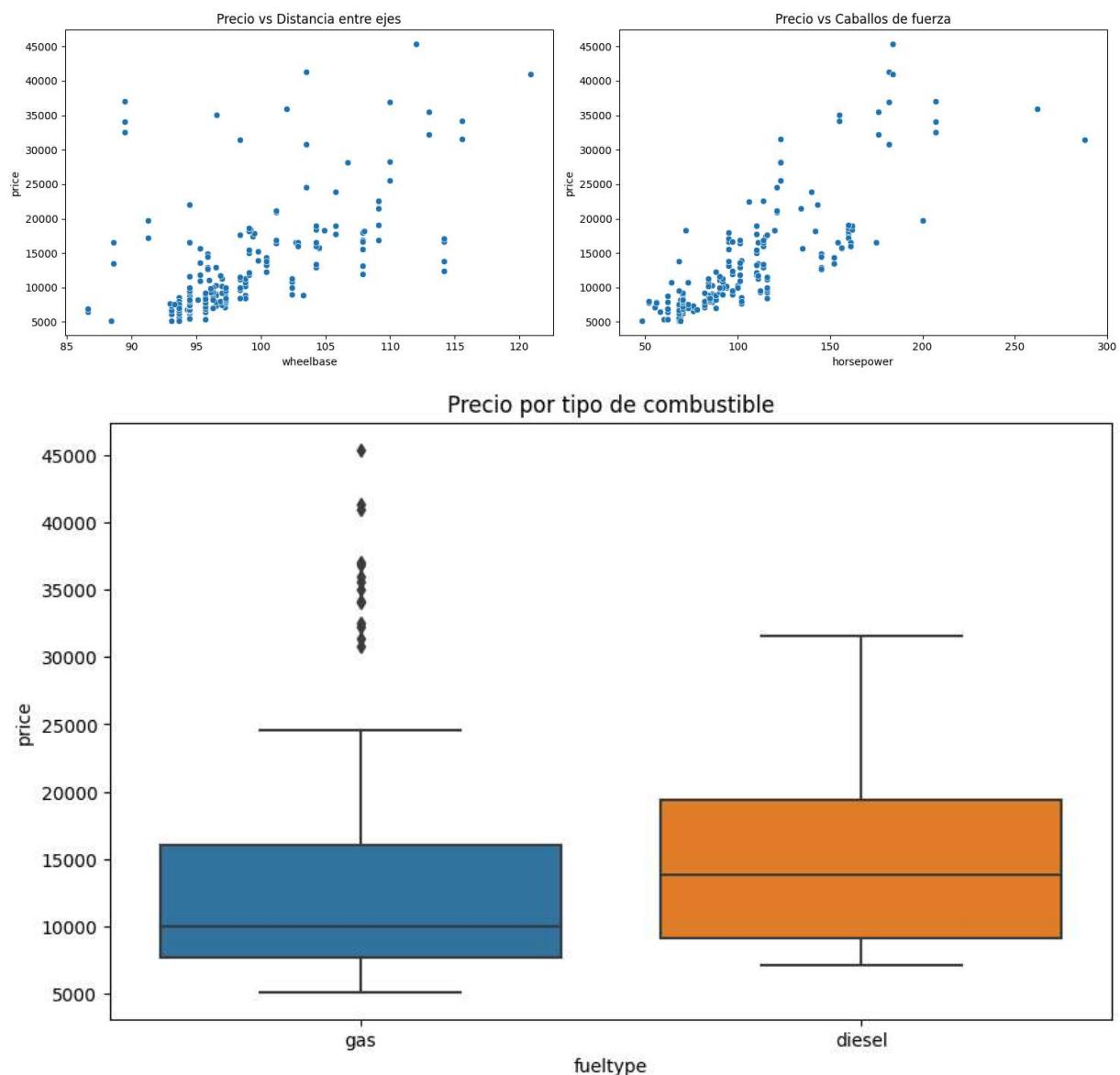
plt.title(f'Histograma de {var}')
plt.tight_layout()
plt.show()

# Diagramas de dispersión
plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.scatterplot(x='wheelbase', y='price', data=df_subset)
plt.title('Precio vs Distancia entre ejes')
plt.subplot(1, 2, 2)
sns.scatterplot(x='horsepower', y='price', data=df_subset)
plt.title('Precio vs Caballos de fuerza')
plt.tight_layout()
plt.show()

# Boxplot de precio por tipo de combustible
plt.figure(figsize=(10, 6))
sns.boxplot(x='fueltype', y='price', data=df_subset)
plt.title('Precio por tipo de combustible')
plt.show()

```





## Modelación y verificación del modelo

Encuentra la ecuación de regresión de mejor ajuste. Propón al menos 2 modelos de ajuste para encontrar la mejor forma de ajustar la variable precio.

```
In [ ]: # Codificar La variable categórica 'fueltype'
df_subset = pd.get_dummies(df_subset, columns=['fueltype'], drop_first=True)

# Modelo 1: Regresión Lineal múltiple
model1 = ols('price ~ wheelbase + horsepower + fueltype_gas', data=df_subset).fit()

# Modelo 2: Regresión con interacción entre wheelbase y horsepower
model2 = ols('price ~ wheelbase + horsepower + fueltype_gas + wheelbase:horsepower'
```

Valida la significancia del modelo con un alfa de 0.04 (incluye las hipótesis que pruebas y el valor frontera)

```
In [ ]: def analizar_significancia_modelo(model, alpha=0.04):
    print("\n--- Significancia del modelo ---")
    f_statistic = model.fvalue
    f_pvalue = model.f_pvalue
    print(f"Hipótesis nula: Todos los coeficientes son cero")
    print(f"Hipótesis alternativa: Al menos un coeficiente es diferente de cero")
    print(f"Valor F: {f_statistic}")
    print(f"Valor p: {f_pvalue}")
    print(f"Conclusión: {'Rechazamos' if f_pvalue < alpha else 'No rechazamos'} la hipótesis nula con alfa = {alpha}")

    print("Modelo 1:")
    analizar_significancia_modelo(model1)
    print("\nModelo 2:")
    analizar_significancia_modelo(model2)
```

Modelo 1:

```
--- Significancia del modelo ---
Hipótesis nula: Todos los coeficientes son cero
Hipótesis alternativa: Al menos un coeficiente es diferente de cero
Valor F: 220.6519241663889
Valor p: 2.526837482754609e-63
Conclusión: Rechazamos la hipótesis nula con alfa = 0.04
```

Modelo 2:

```
--- Significancia del modelo ---
Hipótesis nula: Todos los coeficientes son cero
Hipótesis alternativa: Al menos un coeficiente es diferente de cero
Valor F: 175.90074139556373
Valor p: 2.524217326089732e-64
Conclusión: Rechazamos la hipótesis nula con alfa = 0.04
```

De acuerdo a los tests con un alfa de 0.04, los dos modelos son significativos.

Valida la significancia de  $\beta_i$  con un alfa de 0.04 (incluye las hipótesis que pruebas y el valor frontera de cada una de ellas)

```
In [ ]: def analizar_significancia_coeficientes(model, alpha=0.04):
    print("\n--- Significancia de los coeficientes ---")
    for var in model.params.index:
        coef = model.params[var]
        p_value = model.pvalues[var]
        print(f"{var}:")
        print(f"  Coeficiente: {coef:.4f}")
        print(f"  Valor p: {p_value:.4f}")
        print(f"  Conclusión: {'Significativo' if p_value < alpha else 'No significativo' } con alfa = {alpha}")

    print("Modelo 1:")
    analizar_significancia_coeficientes(model1)
```

Modelo 1:

```
--- Significancia de los coeficientes ---  
Intercept:  
    Coeficiente: -34754.3253  
    Valor p: 0.0000  
    Conclusión: Significativo con alfa = 0.04  
wheelbase:  
    Coeficiente: 364.6572  
    Valor p: 0.0000  
    Conclusión: Significativo con alfa = 0.04  
horsepower:  
    Coeficiente: 148.3233  
    Valor p: 0.0000  
    Conclusión: Significativo con alfa = 0.04  
fueltype_gas:  
    Coeficiente: -3794.4500  
    Valor p: 0.0002  
    Conclusión: Significativo con alfa = 0.04
```

Todas las variables son significativas en el modelo 1.

```
In [ ]: print("\nModelo 2:")  
analizar_significancia_coeficientes(model2)
```

Modelo 2:

```
--- Significancia de los coeficientes ---  
Intercept:  
    Coeficiente: 10923.8671  
    Valor p: 0.4686  
    Conclusión: No significativo con alfa = 0.04  
wheelbase:  
    Coeficiente: -98.9541  
    Valor p: 0.5164  
    Conclusión: No significativo con alfa = 0.04  
horsepower:  
    Coeficiente: -205.7870  
    Valor p: 0.0621  
    Conclusión: No significativo con alfa = 0.04  
fueltype_gas:  
    Coeficiente: -4632.7984  
    Valor p: 0.0000  
    Conclusión: Significativo con alfa = 0.04  
wheelbase:horsepower:  
    Coeficiente: 3.6400  
    Valor p: 0.0014  
    Conclusión: Significativo con alfa = 0.04
```

Intercept, wheelbase y horsepower no son significativas en el modelo 2. Aunque horsepower falla la prueba por poco.

Indica cuál es el porcentaje de variación explicada por el modelo.

```
In [ ]: def analizar_variacion_explicada(model):
    r_squared = model.rsquared
    print(f"\n--- Porcentaje de variación explicada ---")
    print(f"R^2: {r_squared:.4f}")
    print(f"Porcentaje de variación explicada: {r_squared * 100:.2f}%")

    # Uso
    print("Modelo 1:")
    analizar_variacion_explicada(model1)
    print("\nModelo 2:")
    analizar_variacion_explicada(model2)
```

Modelo 1:

```
--- Porcentaje de variación explicada ---
R^2: 0.7671
Porcentaje de variación explicada: 76.71%
```

Modelo 2:

```
--- Porcentaje de variación explicada ---
R^2: 0.7787
Porcentaje de variación explicada: 77.87%
```

El modelo 2 explica más variación que el modelo 1 solo por +1%.

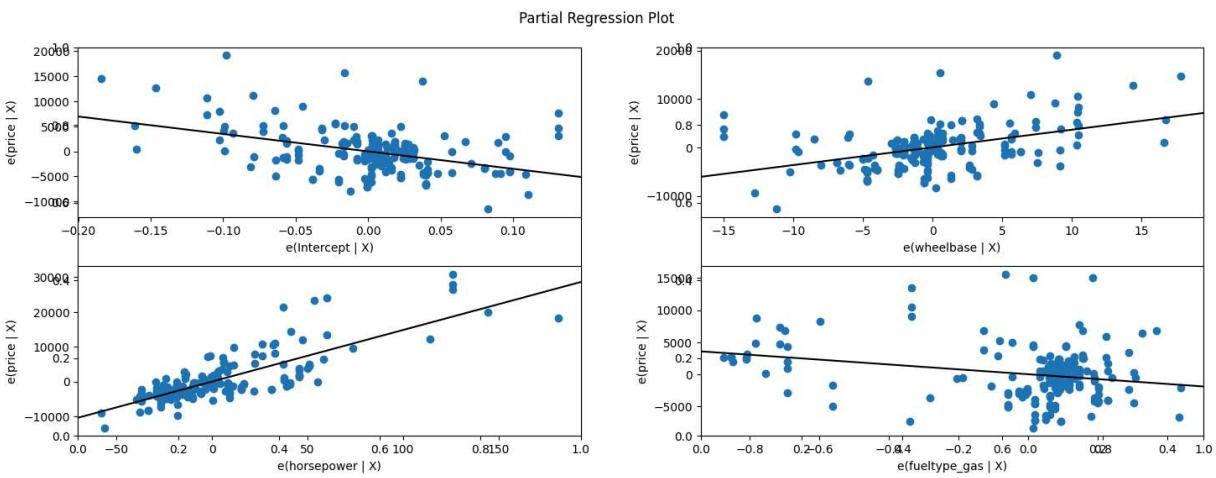
Dibuja el diagrama de dispersión de los datos por pares y la recta de mejor ajuste.

```
In [ ]: def graficar dispersion_ajuste(model):
    fig, ax = plt.subplots(1, 2, figsize=(15, 6))
    sm.graphics.plot_partregress_grid(model, fig=fig)
    plt.tight_layout()
    plt.show()

    # Uso
    print("Modelo 1:")
    graficar dispersion_ajuste(model1)
```

Modelo 1:

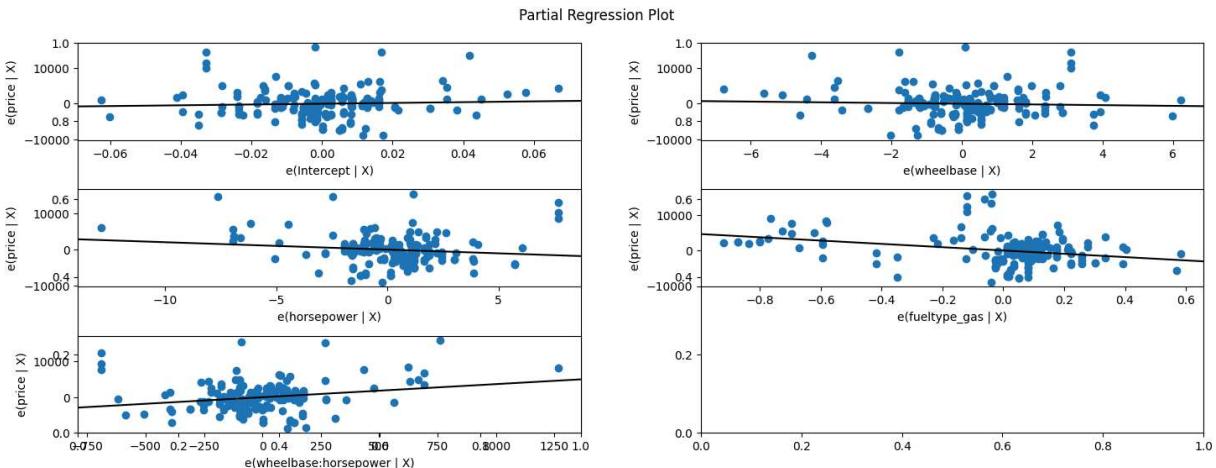
```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```



```
In [ ]: print("\nModelo 2:")
graficar dispersion_ajuste(model2)
```

Modelo 2:

```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```



### Analiza la validez de los modelos propuestos:

```
In [ ]: def analizar_residuos(model, alpha=0.04):
    residuals = model.resid
    fitted_values = model.fittedvalues

    print("\n--- Análisis de residuos ---")

    # Normalidad de Los residuos
    _, p_value = stats.normaltest(residuals)
    print(f"Test de normalidad de los residuos:")
    print(f"Valor p: {p_value:.4f}")
    print(f"Conclusión: Los residuos {'son' if p_value > alpha else 'no son'} norma

    # QQ-plot
    fig, ax = plt.subplots(figsize=(8, 6))
```

```

sm.qqplot(residuals, line='45', fit=True, ax=ax)
plt.title("Q-Q Plot de los residuos")
plt.show()

# Verificación de media cero
mean_residuals = np.mean(residuals)
print(f"\nMedia de los residuos: {mean_residuals:.4f}")

# Homocedasticidad
_, p_value, _, _ = het_breuschpagan(residuals, model.model.exog)
print(f"\nTest de Breusch-Pagan para homocedasticidad:")
print(f"Valor p: {p_value:.4f}")
print(f"Conclusión: Los residuos {'son' if p_value > alpha else 'no son'} homocedasticos")

# Gráfico de residuos vs valores ajustados
plt.figure(figsize=(10, 6))
plt.scatter(fitted_values, residuals)
plt.xlabel("Valores ajustados")
plt.ylabel("Residuos")
plt.title("Residuos vs Valores ajustados")
plt.axhline(y=0, color='r', linestyle='--')
plt.show()

# Uso
print("Modelo 1:")
analizar_residuos(model1)

```

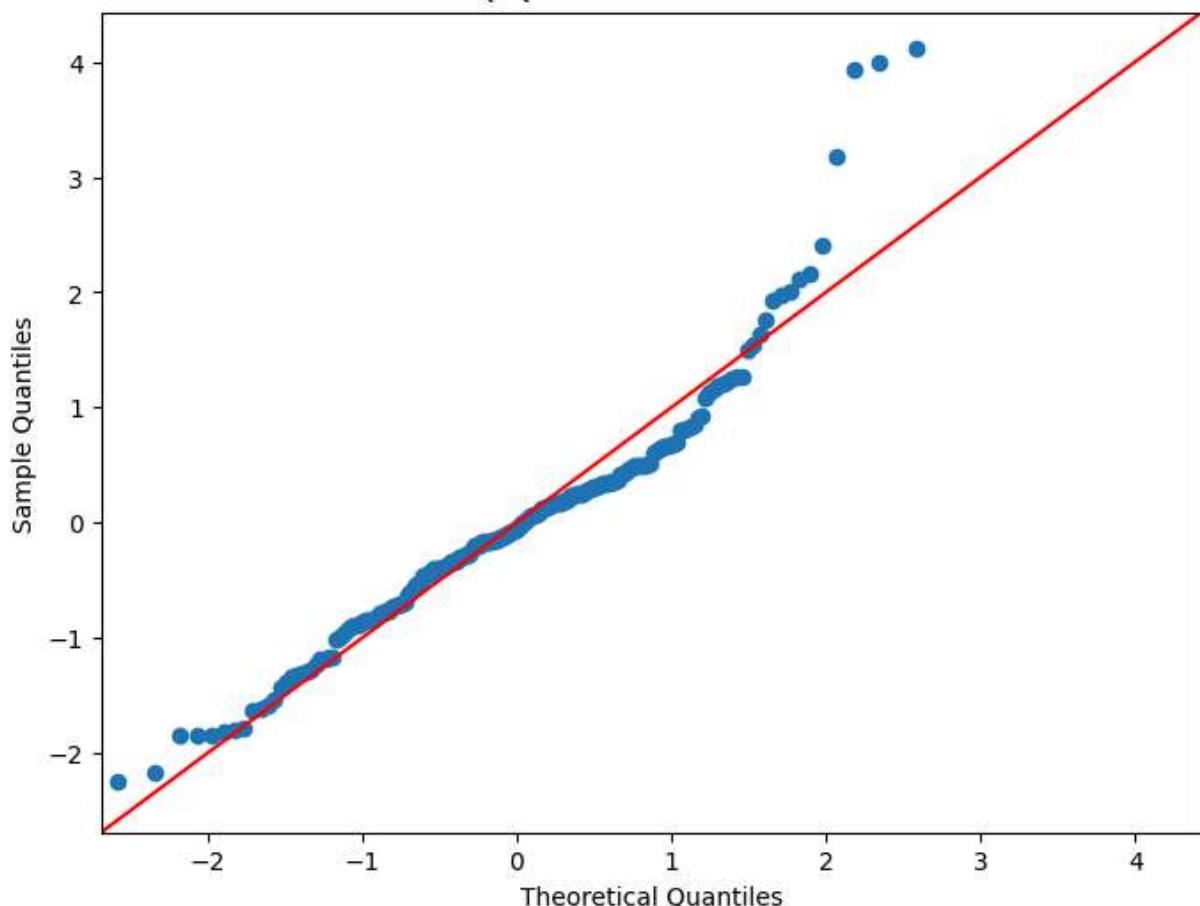
Modelo 1:

```

--- Análisis de residuos ---
Test de normalidad de los residuos:
Valor p: 0.0000
Conclusión: Los residuos no son normales con alfa = 0.04

```

Q-Q Plot de los residuos

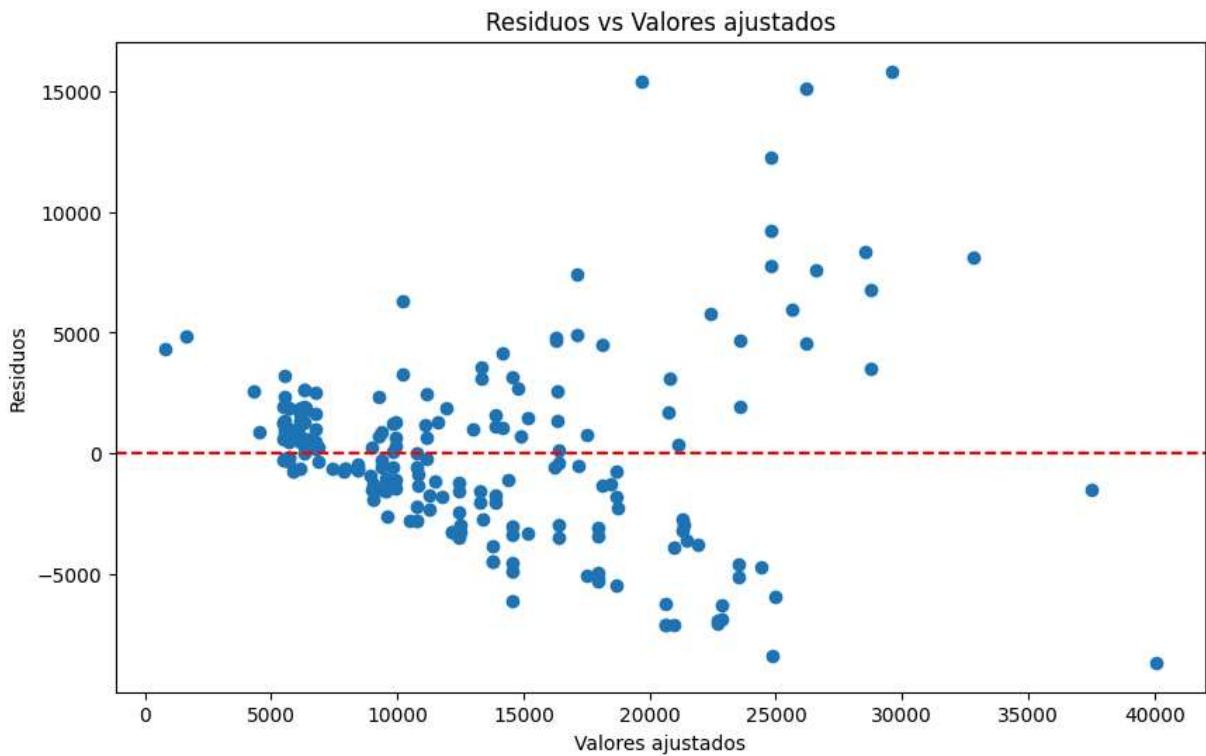


Media de los residuos: -0.0000

Test de Breusch-Pagan para homocedasticidad:

Valor p: 0.0000

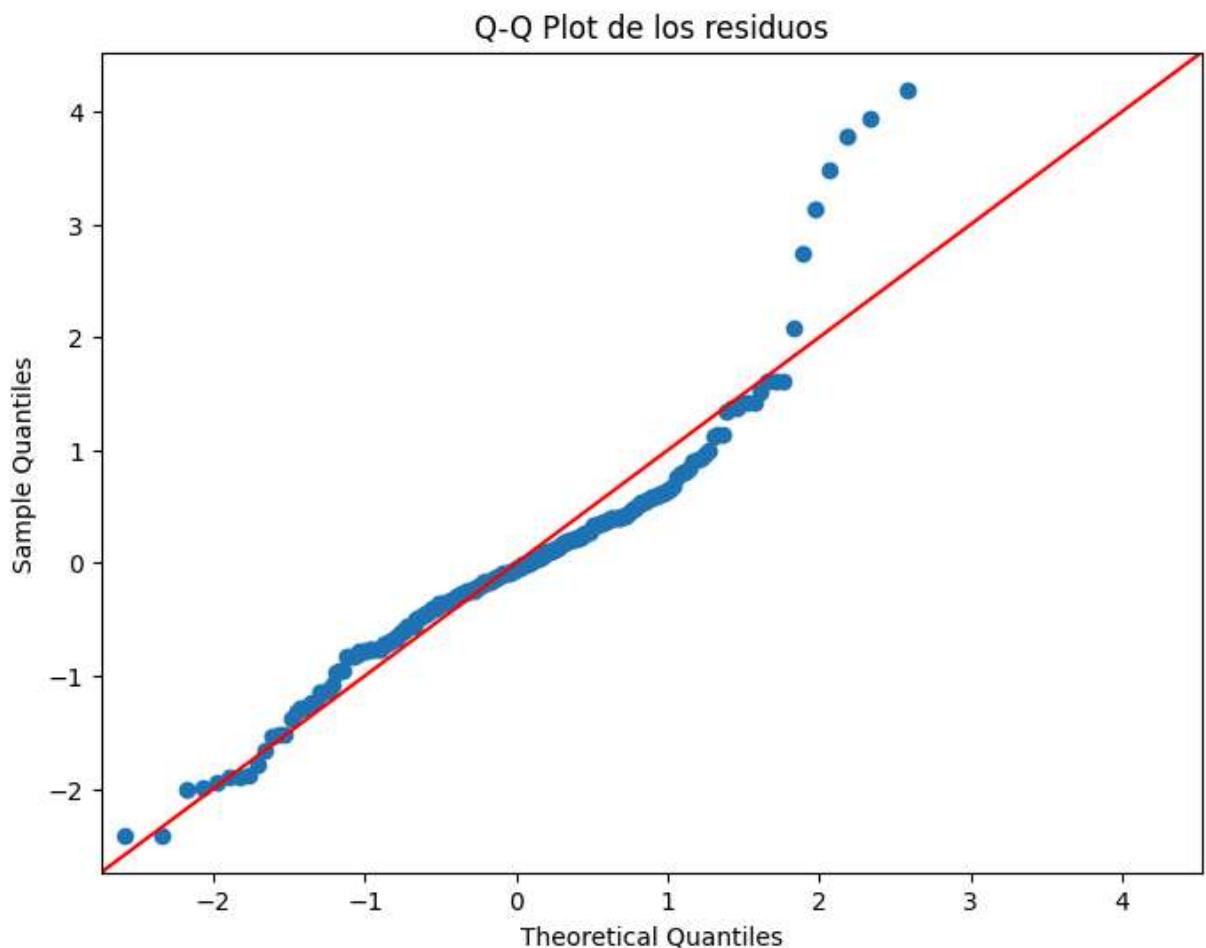
Conclusión: Los residuos no son homocedásticos con alfa = 0.04



```
In [ ]: print("\nModelo 2:")
    analizar_residuos(model2)
```

Modelo 2:

```
--- Análisis de residuos ---
Test de normalidad de los residuos:
Valor p: 0.0000
Conclusión: Los residuos no son normales con alfa = 0.04
```

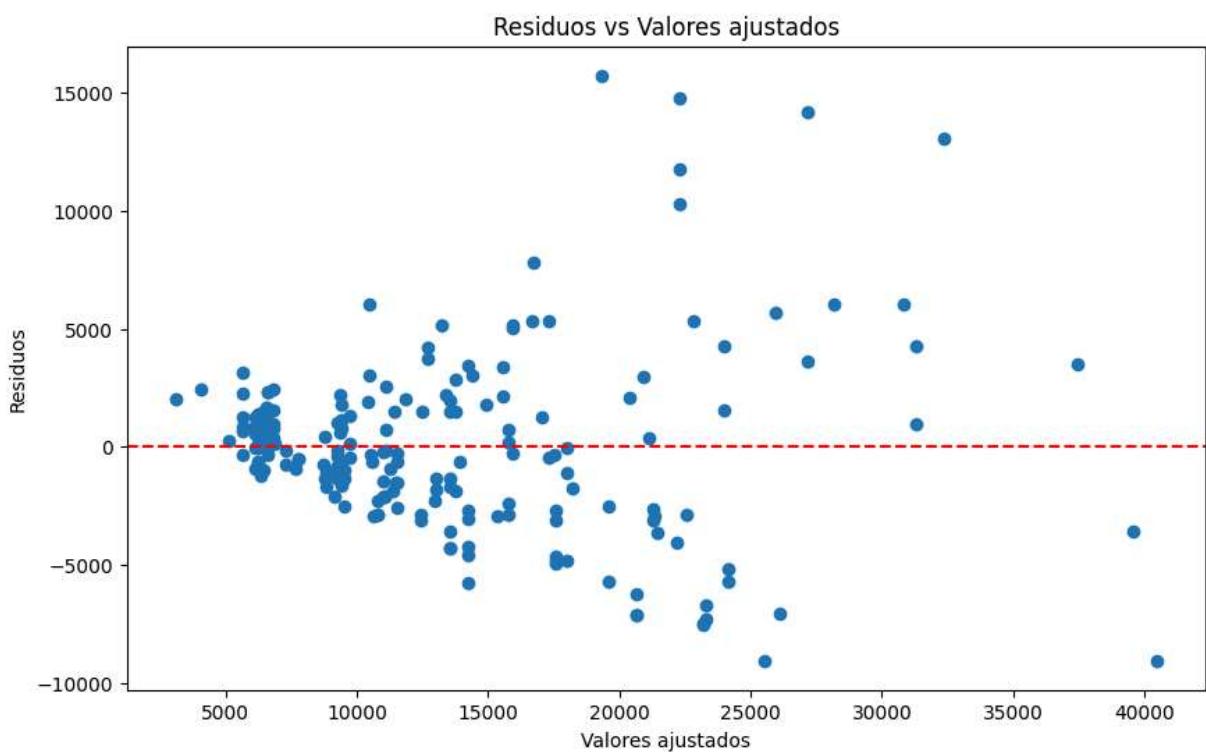


Media de los residuos: -0.0000

Test de Breusch-Pagan para homocedasticidad:

Valor p: 0.0000

Conclusión: Los residuos no son homocedásticos con alfa = 0.04



Emite una conclusión final sobre el mejor modelo de regresión lineal y contesta la pregunta central:

Concluye sobre el mejor modelo que encontraste y argumenta por qué es el mejor

Para resumir:

- Ninguno de los dos modelos cumple con homocedasticidad ni residuos normales ( $\alpha = 0.04$ ).
- Todas las variables en el modelo 1 son significativas.
- En el modelo 2, intercept, wheelbase y horsepower no son significativas.
- El modelo 2 explica más variación que el modelo 1 solo por +1%.

Por estas razones, el mejor modelo es el modelo 1. Debido a que todas las variables son significativas y explica solo 1% menos de variación que el modelo 2.

¿Cuáles de las variables asignadas influyen en el precio del auto? ¿de qué manera lo hacen?

Todas las variables asignadas influyen en el precio del auto en el modelo 1.

Debido a sus coeficientes, la interpretación de la influencia de wheelbase y horsepower en el modelo 1 es que entre más grande sea el valor de estas variables, mayor será el precio del auto.

## Intervalos de predicción y confianza

Con los datos de las variables asignadas construye la gráfica de los intervalos de confianza y predicción para la estimación y predicción del precio para el mejor modelo seleccionado:

```
In [ ]: def intervalos_prediccion_confianza(modelo, df, var_x, var_y, categoria):
    # Crear un DataFrame con valores para predicción
    x_range = np.linspace(df[var_x].min(), df[var_x].max(), 100)
    pred_data = pd.DataFrame({var_x: x_range, 'fueltype_gas': categoria})

    # Copiar los valores medios de las otras variables del DataFrame original
    for col in df.columns:
        if col not in [var_x, 'fueltype_gas', var_y]:
            pred_data[col] = df[col].mean()

    # Calcular predicciones e intervalos
    predictions = modelo.get_prediction(pred_data)
    summary_frame = predictions.summary_frame(alpha=0.05)

    # Graficar datos originales, predicciones e intervalos
    plt.figure(figsize=(12, 8))
    plt.scatter(df[df['fueltype_gas'] == categoria][var_x],
                df[df['fueltype_gas'] == categoria][var_y],
                alpha=0.5, label='Datos observados')
```

```

plt.plot(x_range, summary_frame['mean'], color='red', label='Predicción')
plt.fill_between(x_range, summary_frame['mean_ci_lower'], summary_frame['mean_ci_upper'],
                 color='red', alpha=0.1, label='Intervalo de confianza')
plt.fill_between(x_range, summary_frame['obs_ci_lower'], summary_frame['obs_ci_upper'],
                 color='green', alpha=0.1, label='Intervalo de predicción')

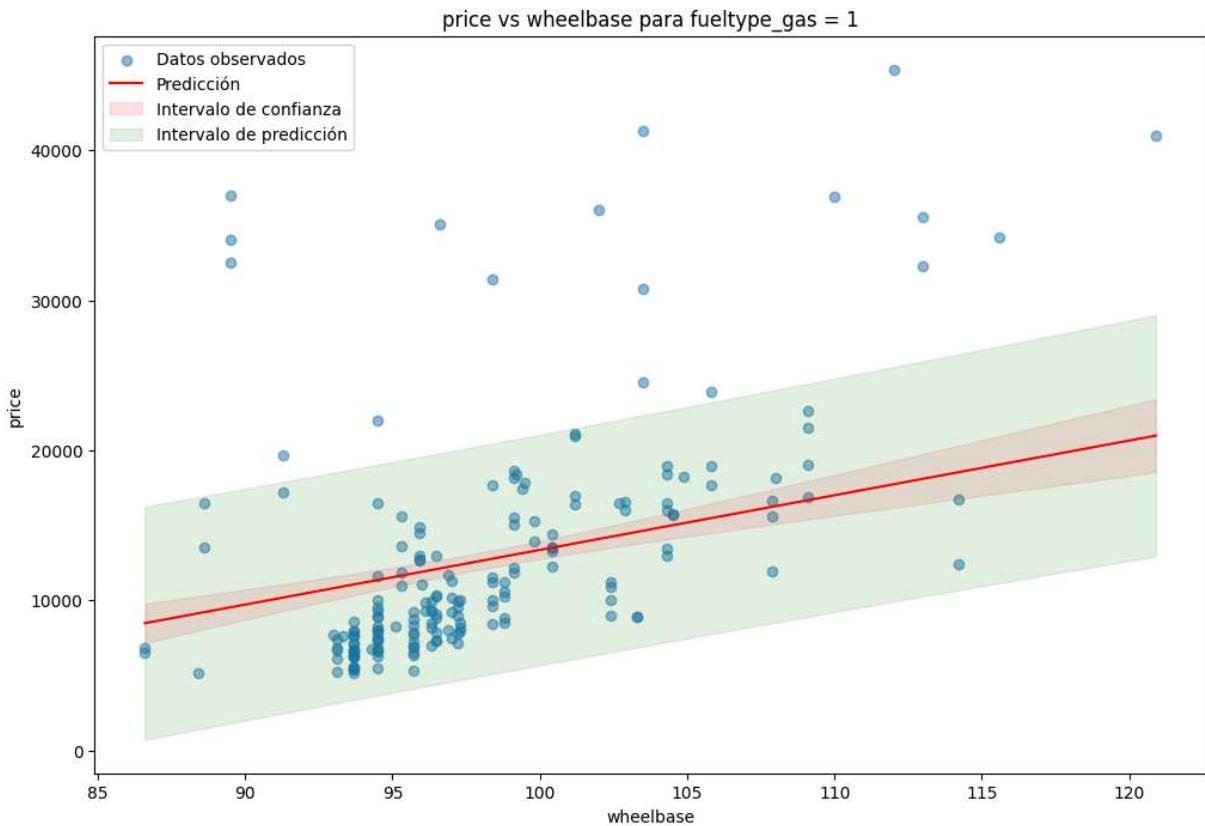
plt.xlabel(var_x)
plt.ylabel(var_y)
plt.title(f'{var_y} vs {var_x} para fueltype_gas = {categoria}')
plt.legend()
plt.show()

```

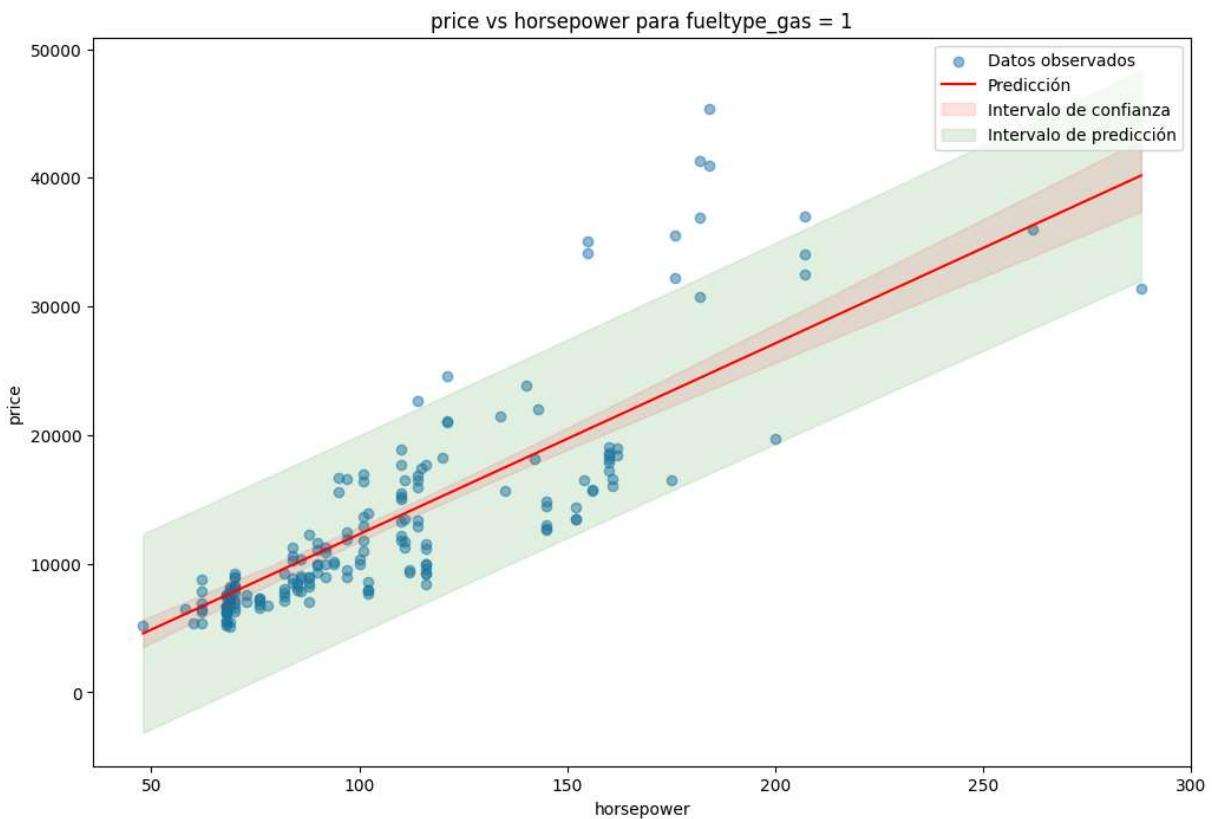
El último parámetro de la función es 'fueltype\_gas' al cual le vamos a poner 1. Lo que significa que el auto es de gasolina. Lo voy a hacer así ya que es más común.

In [ ]: # Aplicar la función para wheelbase y horsepower

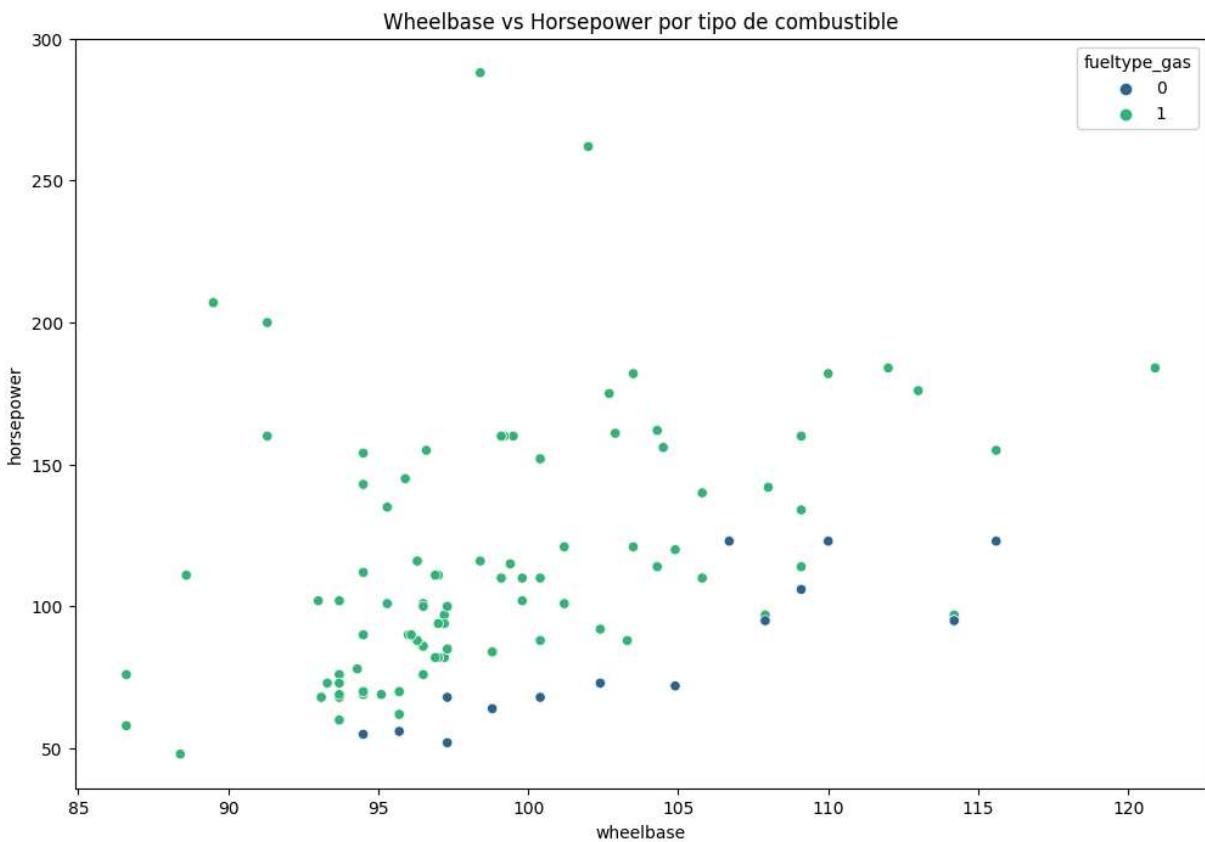
```
intervalos_prediccion_confianza(model1, df_subset, 'wheelbase', 'price', 1)
```



In [ ]: intervalos\_prediccion\_confianza(model1, df\_subset, 'horsepower', 'price', 1)



```
In [ ]: # Graficar por pares de variables numéricas
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df_subset, x='wheelbase', y='horsepower', hue='fueltype_gas',
plt.title('Wheelbase vs Horsepower por tipo de combustible')
plt.show()
```



Basándome en las gráficas de mi modelo, horsepower suele representar con más precisión que wheelbase el precio ya que hay muchos valores que estan dentro del intervalo de confianza.

Wheelbase también es bueno pero tiene más valores en su intervalo de predicción que en el de confianza.

### Más allá:

Contesta la pregunta referida a la agrupación de variables que propuso la empresa para el análisis: ¿propondrías una nueva agrupación de las variables a la empresa automovilística?

A pesar de que tiene sentido las agrupaciones propuestas por la empresa, propondría una nueva agrupación basado en lo más importante para el cliente, no solo para la empresa, ya que el cliente es el que va a comprar el auto, el que nos da la demanda.

Un grupo de ejemplo sería: citympg, peakrpm, horsepower, enginetype, carheight, carwidth, carlength, carbbody, fueltype, CarName, Symboling .

Retoma todas las variables y haz un análisis estadístico muy leve (medias y correlación) de cómo crees que se deberían agrupar para analizarlas.

```
In [ ]: df = pd.read_csv("precios_autos.csv")
df.head()
```

Out[ ]:

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

5 rows × 21 columns



In [ ]:

```
def basic_stat_analysis(df):
    # Identificar las variables numéricas y categóricas
    num_vars = df.select_dtypes(include='number').columns
    cat_vars = df.select_dtypes(include='object').columns

    # Análisis para variables numéricas
    if len(num_vars) > 0:
        print("Análisis de variables numéricas:")
        print(df[num_vars].describe().T[['mean', 'std', 'min', 'max']])

        # Calcular y mostrar la correlación entre las variables numéricas
        corr_matrix = df[num_vars].corr()

        # Graficar la matriz de correlación
        plt.figure(figsize=(10, 6))
        sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
        plt.title("Heatmap de Correlación entre Variables Numéricas")
        plt.show()
    else:
        print("No se encontraron variables numéricas.")

    # Análisis para variables categóricas
    if len(cat_vars) > 0:
        print("\nAnálisis de variables categóricas:")
        for var in cat_vars:
            print(f"\nFrecuencia de la variable {var}:")
            print(df[var].value_counts())
    else:
        print("No se encontraron variables categóricas.)
```

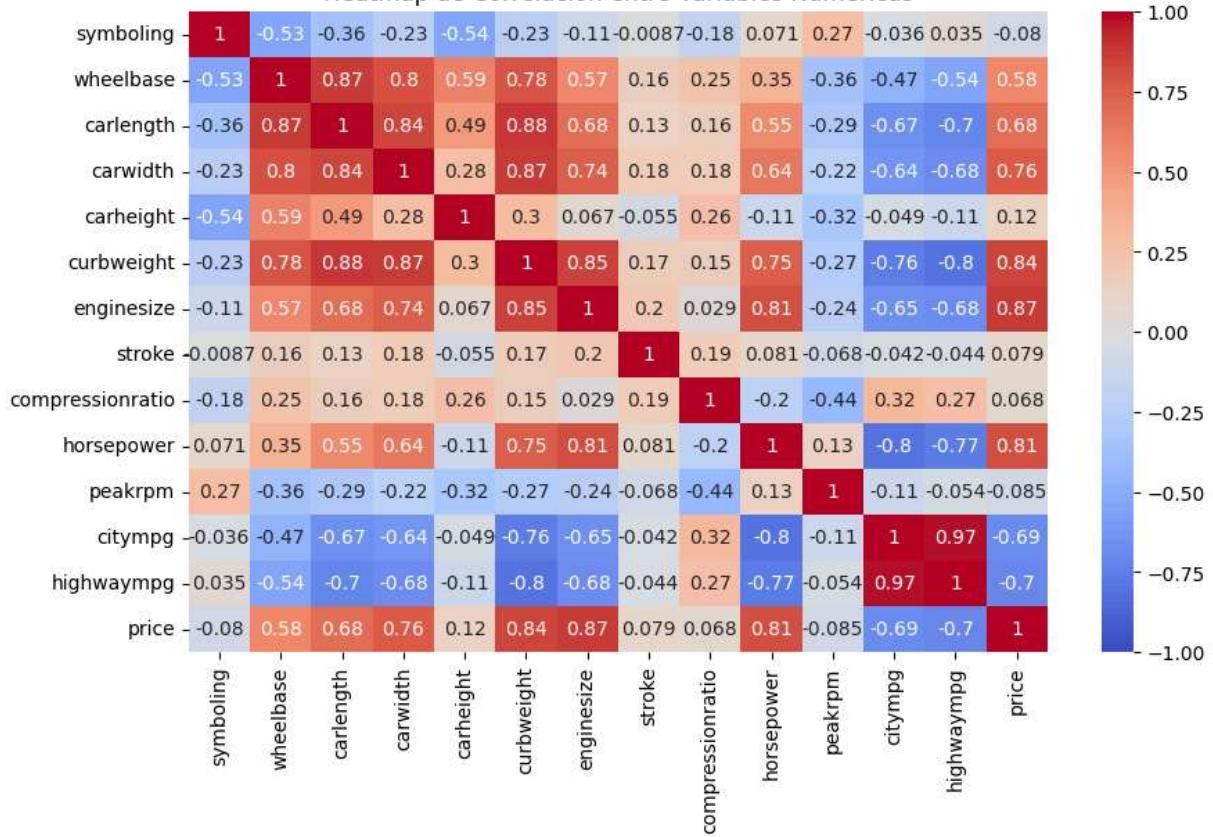
In [ ]:

```
basic_stat_analysis(df)
```

### Análisis de variables numéricas:

|                  | mean         | std         | min     | max      |
|------------------|--------------|-------------|---------|----------|
| symboling        | 0.834146     | 1.245307    | -2.00   | 3.00     |
| wheelbase        | 98.756585    | 6.021776    | 86.60   | 120.90   |
| carlength        | 174.049268   | 12.337289   | 141.10  | 208.10   |
| carwidth         | 65.907805    | 2.145204    | 60.30   | 72.30    |
| carheight        | 53.724878    | 2.443522    | 47.80   | 59.80    |
| curbweight       | 2555.565854  | 520.680204  | 1488.00 | 4066.00  |
| enginesize       | 126.907317   | 41.642693   | 61.00   | 326.00   |
| stroke           | 3.255415     | 0.313597    | 2.07    | 4.17     |
| compressionratio | 10.142537    | 3.972040    | 7.00    | 23.00    |
| horsepower       | 104.117073   | 39.544167   | 48.00   | 288.00   |
| peakrpm          | 5125.121951  | 476.985643  | 4150.00 | 6600.00  |
| citympg          | 25.219512    | 6.542142    | 13.00   | 49.00    |
| highwaympg       | 30.751220    | 6.886443    | 16.00   | 54.00    |
| price            | 13276.710571 | 7988.852332 | 5118.00 | 45400.00 |

Heatmap de Correlación entre Variables Numéricas



Análisis de variables categóricas:

Frecuencia de la variable CarName:

|                      |    |
|----------------------|----|
| toyota corona        | 6  |
| toyota corolla       | 6  |
| peugeot 504          | 6  |
| subaru dl            | 4  |
| mitsubishi mirage g4 | 3  |
|                      | .. |
| mazda glc 4          | 1  |
| mazda rx2 coupe      | 1  |
| maxda glc deluxe     | 1  |
| maxda rx3            | 1  |
| volvo 246            | 1  |

Name: CarName, Length: 147, dtype: int64

Frecuencia de la variable fueltype:

|        |     |
|--------|-----|
| gas    | 185 |
| diesel | 20  |

Name: fueltype, dtype: int64

Frecuencia de la variable carbody:

|             |    |
|-------------|----|
| sedan       | 96 |
| hatchback   | 70 |
| wagon       | 25 |
| hardtop     | 8  |
| convertible | 6  |

Name: carbody, dtype: int64

Frecuencia de la variable drivewheel:

|     |     |
|-----|-----|
| fwd | 120 |
| rwd | 76  |
| 4wd | 9   |

Name: drivewheel, dtype: int64

Frecuencia de la variable enginelocation:

|       |     |
|-------|-----|
| front | 202 |
| rear  | 3   |

Name: enginelocation, dtype: int64

Frecuencia de la variable enginetype:

|       |     |
|-------|-----|
| ohc   | 148 |
| ohcf  | 15  |
| ohcv  | 13  |
| dohc  | 12  |
| l     | 12  |
| rotor | 4   |
| dohcv | 1   |

Name: enginetype, dtype: int64

Frecuencia de la variable cylindernumber:

|       |     |
|-------|-----|
| four  | 159 |
| six   | 24  |
| five  | 11  |
| eight | 5   |
| two   | 4   |

```
three      1  
twelve     1  
Name: cylindernumber, dtype: int64
```

Vamos a hacer una agrupación de las variables que tengan una correlación mayor a 0.5 o menor a -0.5 con el precio. Para agrupar las más relevantes (según la correlación) con el precio.

```
In [ ]: correlation_matrix = df.corr()  
correlation_price = correlation_matrix['price']  
high_correlation_vars = correlation_price[(correlation_price > 0.5) | (correlation_<br>high_correlation_vars
```

```
C:\Users\ozner\AppData\Local\Temp\ipykernel_4492\2060387787.py:1: FutureWarning: The  
default value of numeric_only in DataFrame.corr is deprecated. In a future version,  
it will default to False. Select only valid columns or specify the value of numeric_<br>only to silence this warning.
```

```
correlation_matrix = df.corr()
```

```
Out[ ]: Index(['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize',  
               'horsepower', 'citympg', 'highwaympg', 'price'],  
              dtype='object')
```