CAR PRICE PREDICTION

In [ ]:
```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows',None)
```

Loading data

In [ ]:
```
data = pd.read_csv('/kaggle/input/car-data/CarPrice_Assignment.csv')
```

In [ ]:
```
data.head()
```

Out[ ]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginel
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

Analyzing

In [ ]:
```
data.shape
```

Out[ ]:
```
(205, 26)
```

In [ ]:
```
data.drop(columns=['car_ID', 'CarName'], inplace=True)
```

In [ ]:
```
def summary(df):
    sum = pd.DataFrame(df.dtypes, columns=['dtypes'])
    sum['missing#'] = df.isna().sum().values*100
    sum['missing%'] = (df.isna().sum().values*100)/len(df)
    sum['uniques'] = df.nunique().values
    sum['count'] = df.count().values
    #sum['skew'] = df.skew().values
    desc = pd.DataFrame(df.describe().T)
    sum['min'] = desc['min']
    sum['max'] = desc['max']
    sum['mean'] = desc['mean']
    return sum
```

```
summary(data).style.background_gradient(cmap='twilight_shifted_r')
```

Out[ ]:

	dtypes	missing#	missing%	uniques	count	min	max	me
symboling	int64	0	0.000000	6	205	-2.000000	3.000000	0.8341
fueltype	object	0	0.000000	2	205	nan	nan	n
aspiration	object	0	0.000000	2	205	nan	nan	n
doornumber	object	0	0.000000	2	205	nan	nan	n
carbody	object	0	0.000000	5	205	nan	nan	n
drivewheel	object	0	0.000000	3	205	nan	nan	n
enginelocation	object	0	0.000000	2	205	nan	nan	n
wheelbase	float64	0	0.000000	53	205	86.600000	120.900000	98.7565
carlength	float64	0	0.000000	75	205	141.100000	208.100000	174.0492
carwidth	float64	0	0.000000	44	205	60.300000	72.300000	65.9078
carheight	float64	0	0.000000	49	205	47.800000	59.800000	53.7248
curbweight	int64	0	0.000000	171	205	1488.000000	4066.000000	2555.5658
enginetype	object	0	0.000000	7	205	nan	nan	n
cylindernumber	object	0	0.000000	7	205	nan	nan	n
enginesize	int64	0	0.000000	44	205	61.000000	326.000000	126.9073
fuelsystem	object	0	0.000000	8	205	nan	nan	n
boreratio	float64	0	0.000000	38	205	2.540000	3.940000	3.3297
stroke	float64	0	0.000000	37	205	2.070000	4.170000	3.2554
compressionratio	float64	0	0.000000	32	205	7.000000	23.000000	10.1425
horsepower	int64	0	0.000000	59	205	48.000000	288.000000	104.1170
peakrpm	int64	0	0.000000	23	205	4150.000000	6600.000000	5125.1219
citympg	int64	0	0.000000	29	205	13.000000	49.000000	25.2195
highwaympg	int64	0	0.000000	30	205	16.000000	54.000000	30.7512
price	float64	0	0.000000	189	205	5118.000000	45400.000000	13276.7105

In [ ]:

```
data[['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight', 'e
```

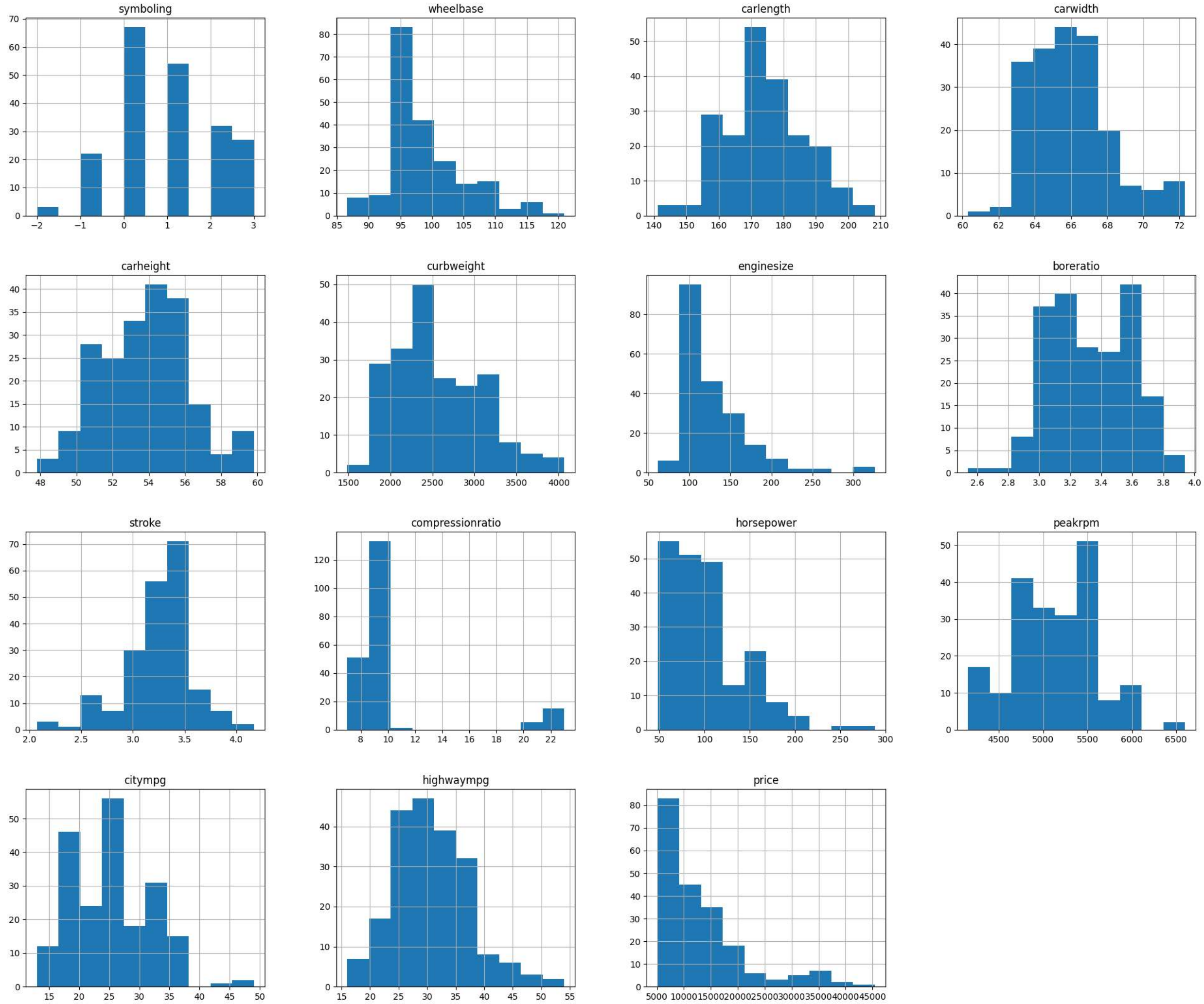
Out[ ]:
```
symboling           0.211072
wheelbase           1.050214
carlength           0.155954
carwidth            0.904003
carheight           0.063123
curbweight          0.681398
enginesize          1.947655
boreratio           0.020156
stroke             -0.689705
compressionratio    2.610862
horsepower          1.405310
peakrpm             0.075159
citympg             0.663704
highwaympg          0.539997
dtype: float64
```

> - We have 0 null values
> - We have 9 categorical columns and rest of them are integers
> - Compressionratio has outliers

Visualising data

In [ ]:
```
data.hist(figsize=(24, 20))
```

Out[ ]:
```
array([[<Axes: title={'center': 'symboling'}>,
        <Axes: title={'center': 'wheelbase'}>,
        <Axes: title={'center': 'carlength'}>,
        <Axes: title={'center': 'carwidth'}>],
       [<Axes: title={'center': 'carheight'}>,
        <Axes: title={'center': 'curbweight'}>,
        <Axes: title={'center': 'enginesize'}>,
        <Axes: title={'center': 'boreratio'}>],
       [<Axes: title={'center': 'stroke'}>,
        <Axes: title={'center': 'compressionratio'}>,
        <Axes: title={'center': 'horsepower'}>,
        <Axes: title={'center': 'peakrpm'}>],
       [<Axes: title={'center': 'citympg'}>,
        <Axes: title={'center': 'highwaympg'}>,
        <Axes: title={'center': 'price'}>, <Axes: >]], dtype=object)
```



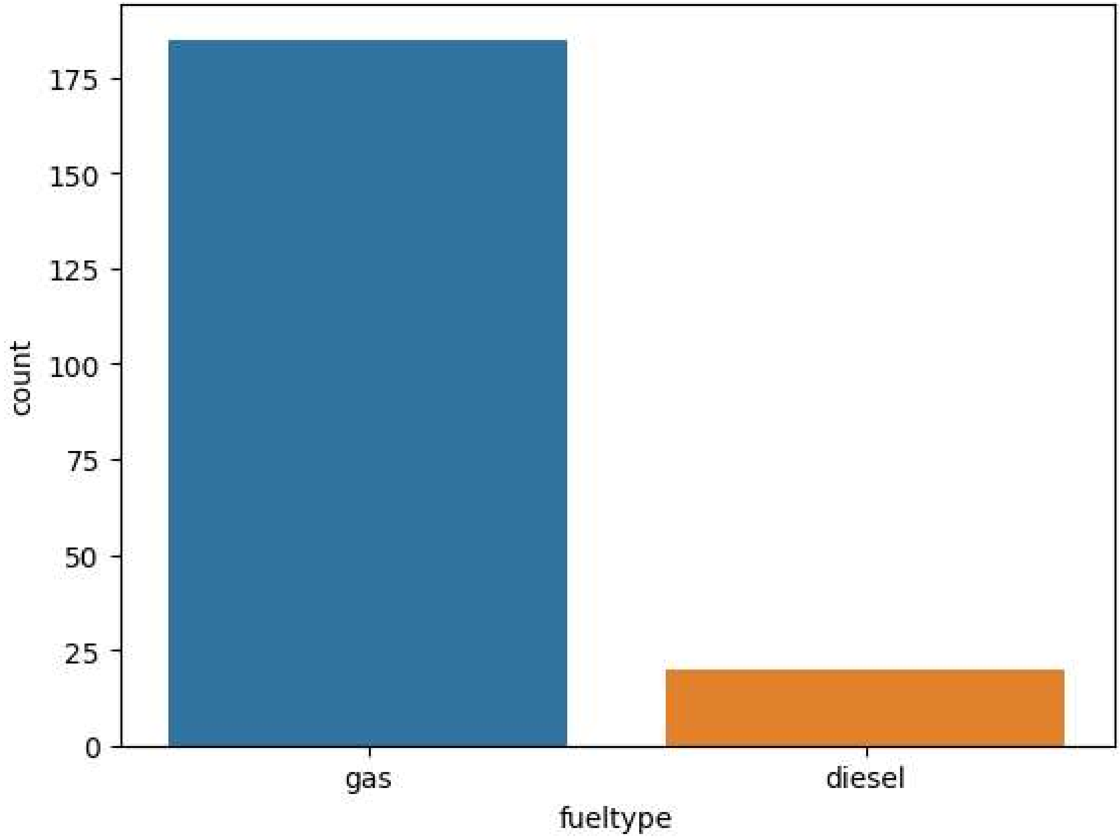
Visualizing categorical columns

In [ ]:

```python
import seaborn as sns
sns.countplot(x=data['fueltype'])
```

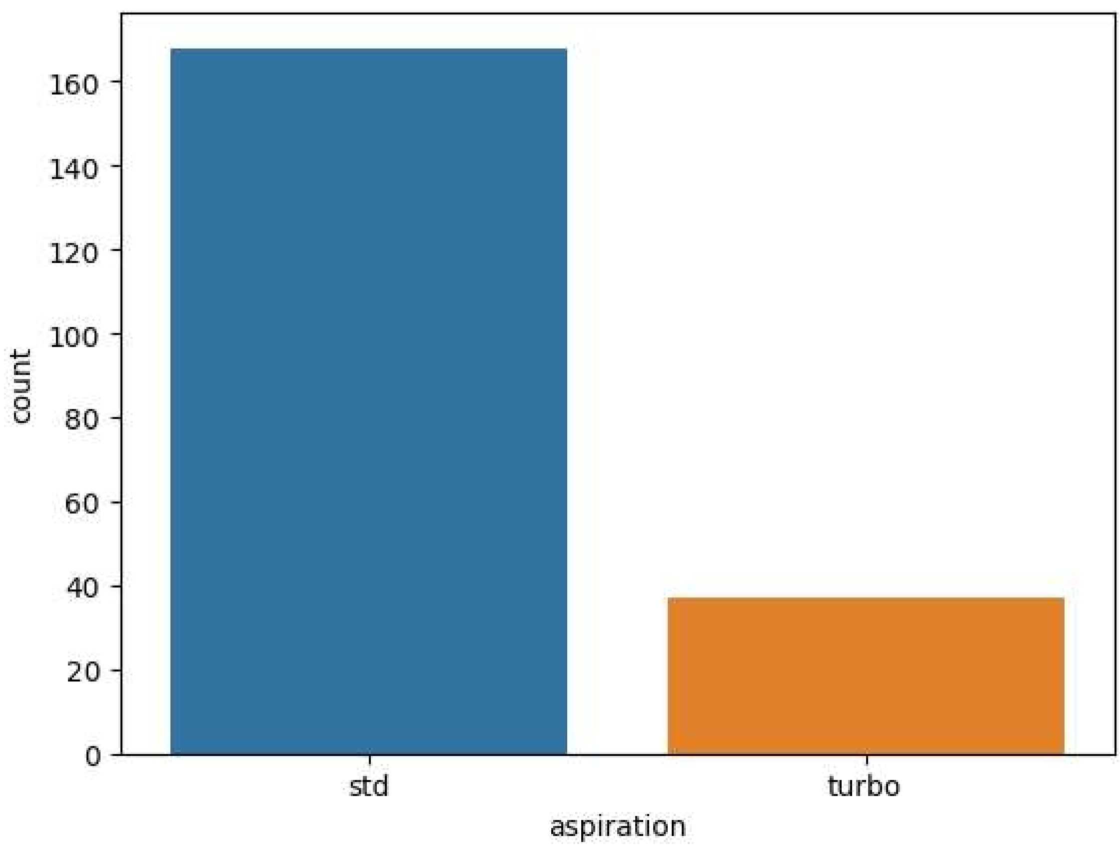
Out[ ]:

```
<Axes: xlabel='fueltype', ylabel='count'>
```



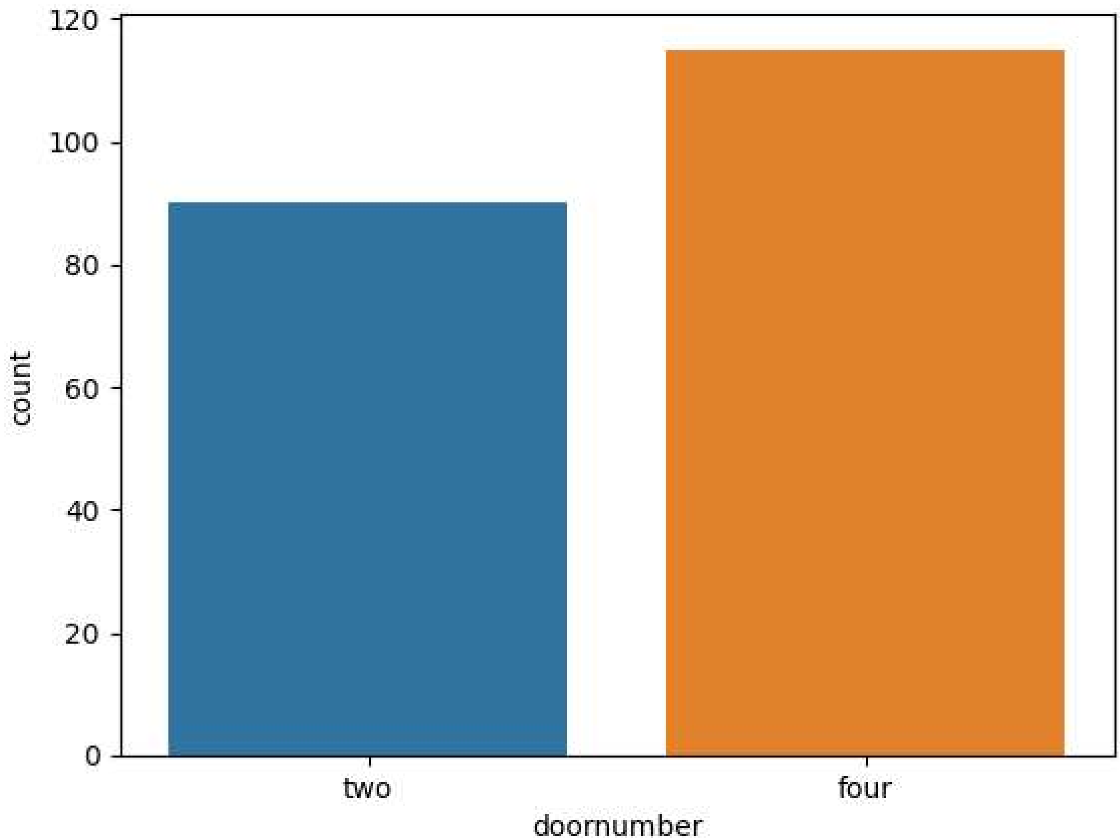
In [ ]:
```
sns.countplot(x=data['aspiration'])
```

Out[ ]:
```
<Axes: xlabel='aspiration', ylabel='count'>
```



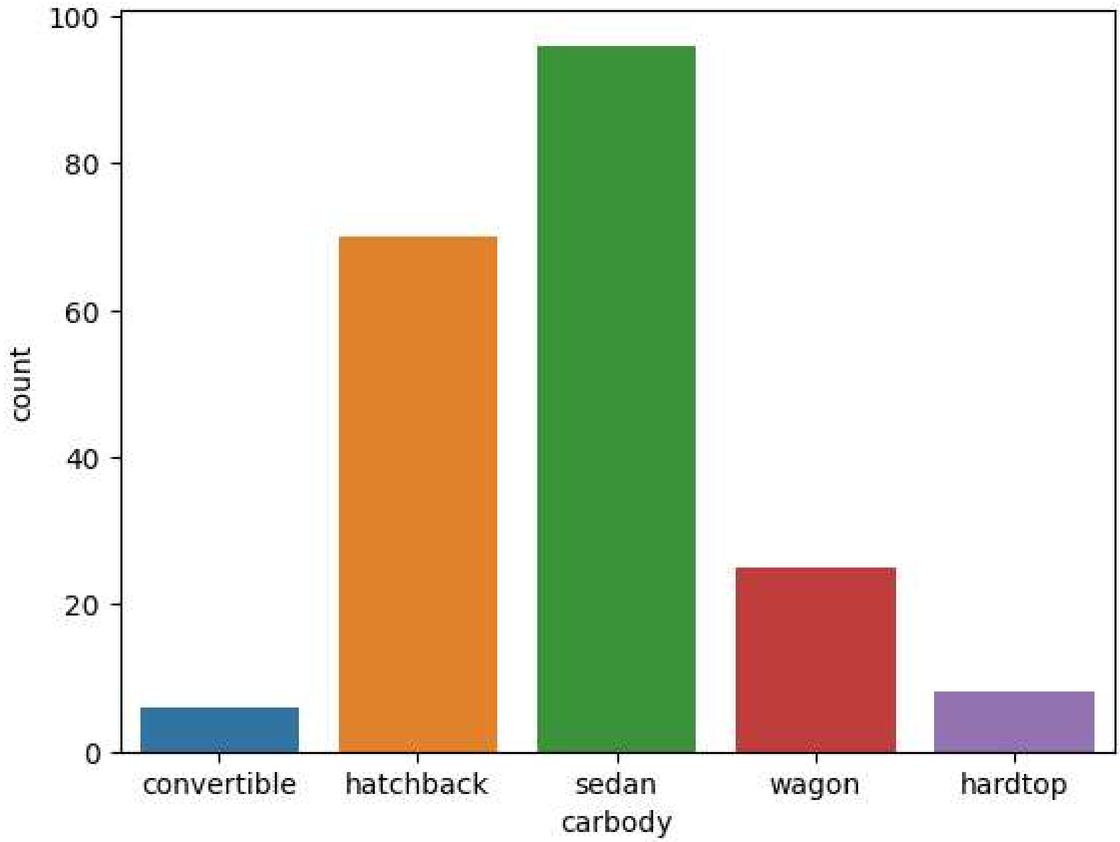
In [ ]:
```
sns.countplot(x=data['doornumber'])
```

Out[ ]: <Axes: xlabel='doornumber', ylabel='count'>



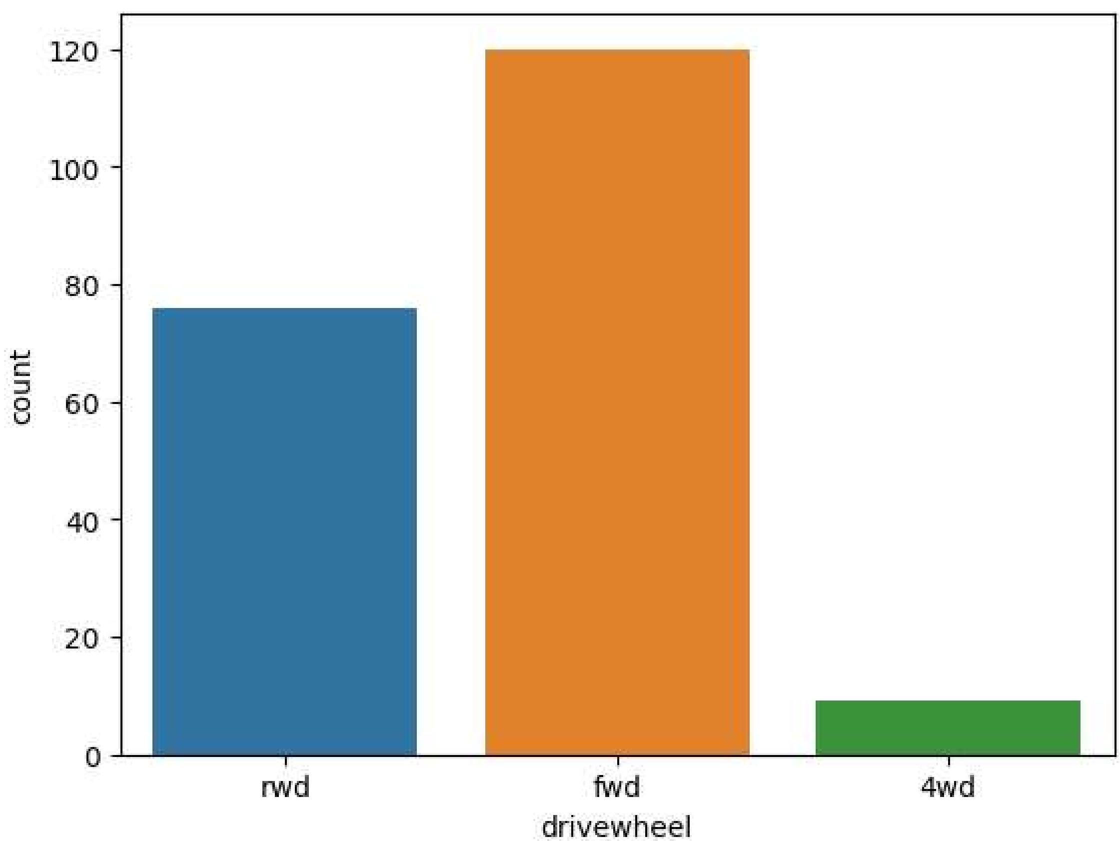
In [ ]:
```
sns.countplot(x=data['carbody'])
```

Out[ ]: <Axes: xlabel='carbody', ylabel='count'>



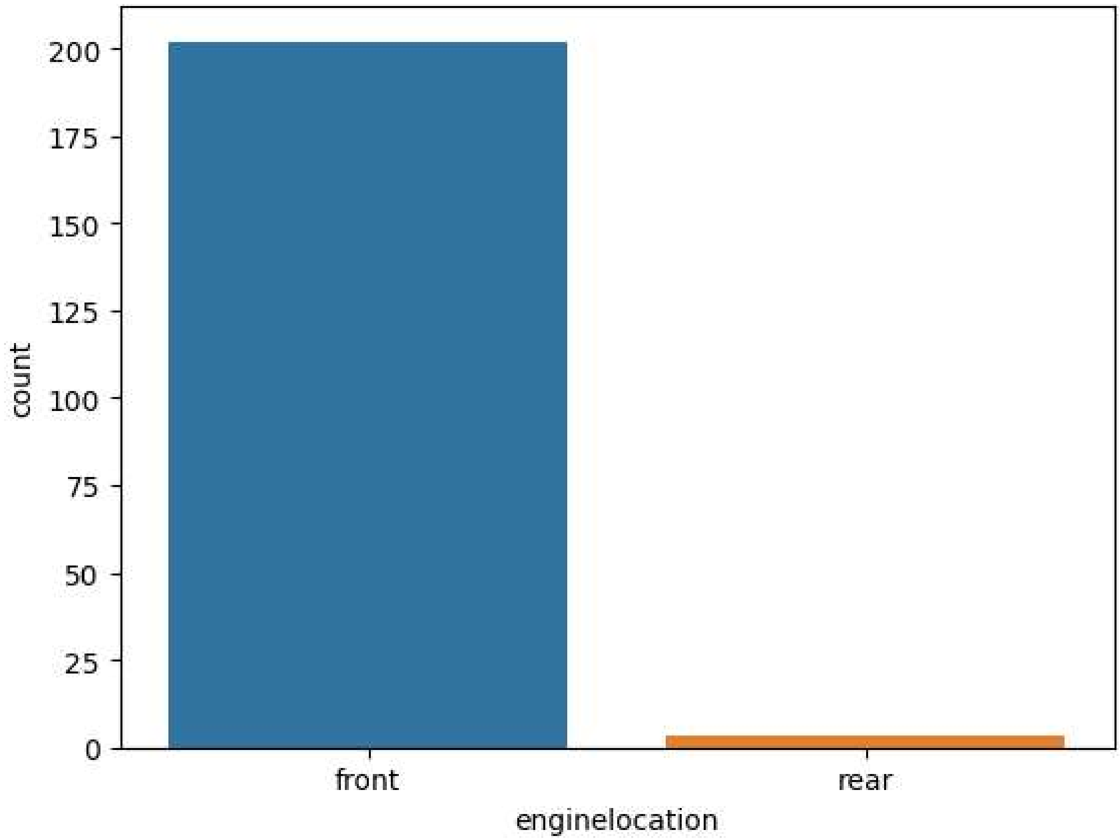
In [ ]:
```
sns.countplot(x=data['drivewheel'])
```

Out[ ]:
```
<Axes: xlabel='drivewheel', ylabel='count'>
```



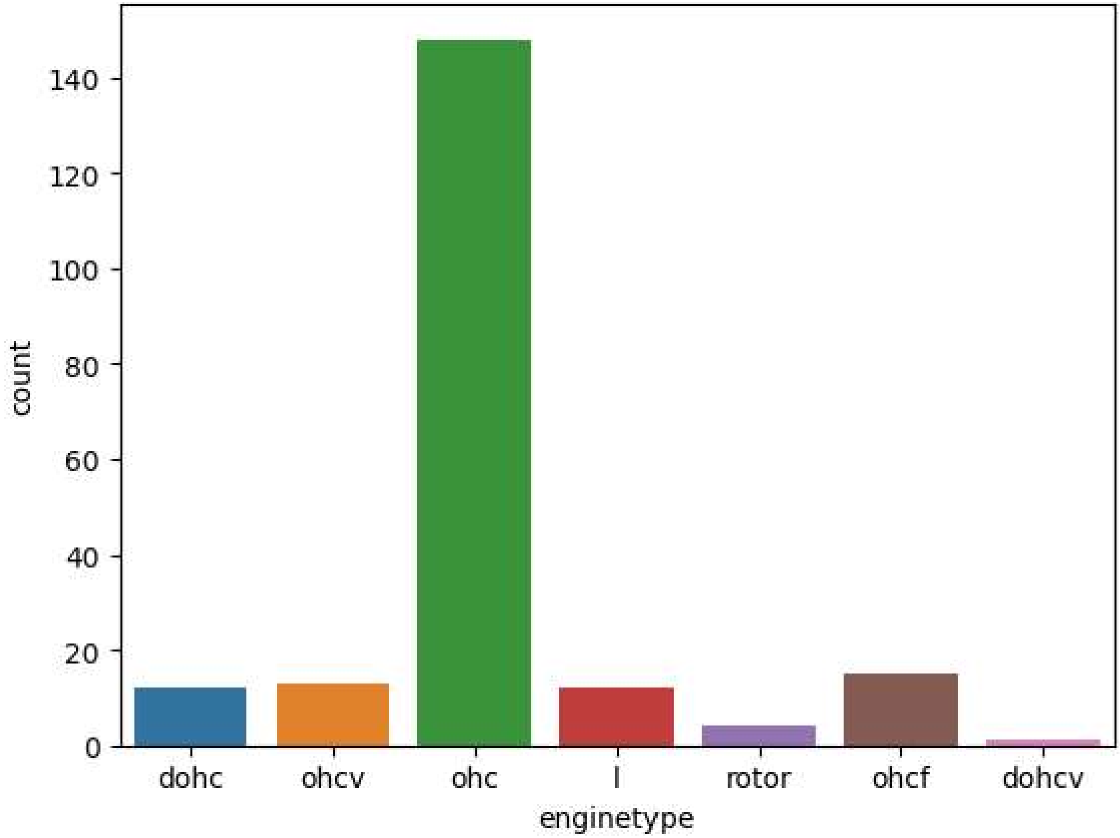
In [ ]:
```
sns.countplot(x=data['enginelocation'])
```

Out[ ]: <Axes: xlabel='enginelocation', ylabel='count'>



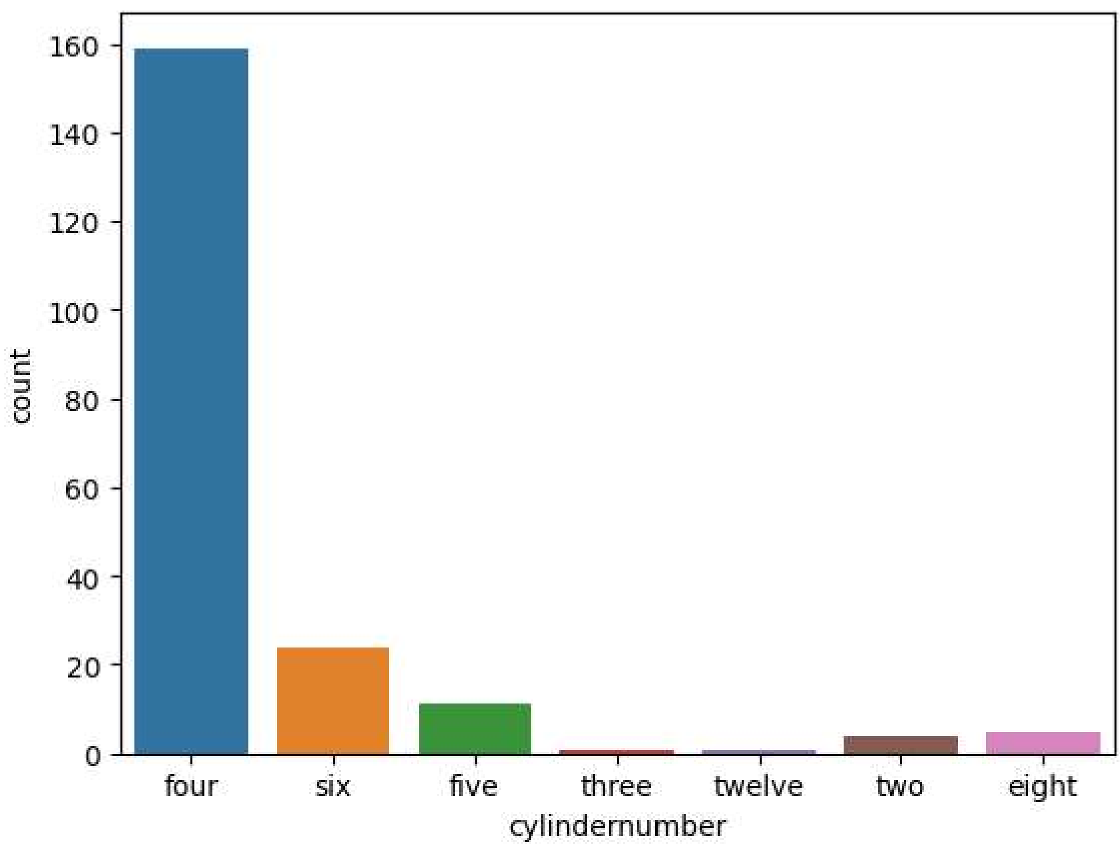
In [ ]:
```
sns.countplot(x=data['enginetype'])
```

Out[ ]: <Axes: xlabel='enginetype', ylabel='count'>



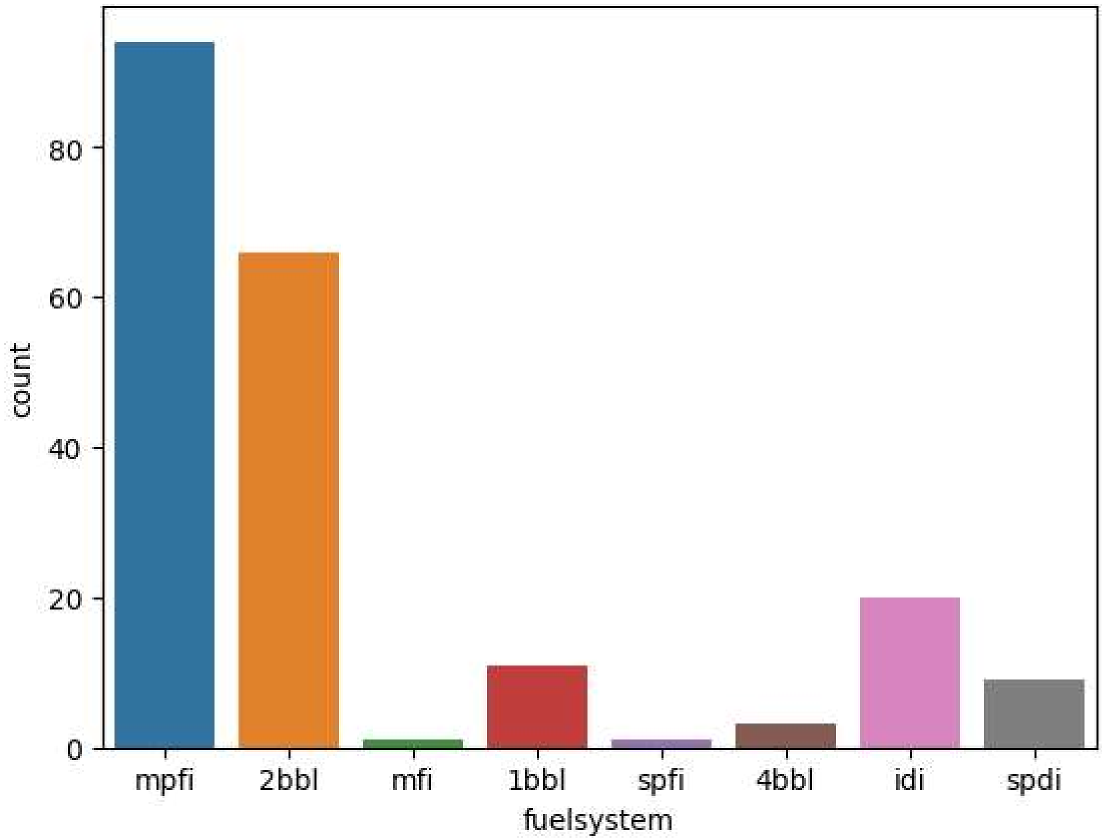
In [ ]:
```
sns.countplot(x=data['cylindernumber'])
```

Out[ ]:
```
<Axes: xlabel='cylindernumber', ylabel='count'>
```



In [ ]:
```
sns.countplot(x=data['fuelsystem'])
```

Out[ ]:
```
<Axes: xlabel='fuelsystem', ylabel='count'>
```



Removing outliers

In [ ]:
```
def remove_outlier(col):
    sorted(col)
    Q1,Q3=col.quantile([0.25,0.75])
    IQR=Q3-Q1
    lower_range=Q1-1.5*IQR
    upper_range=Q3+1.5*IQR
    return lower_range,upper_range
```

In [ ]:
```
low_leadtime,high_leadtime=remove_outlier(data['compressionratio'])
data['compressionratio']=np.where(data['compressionratio']>high_leadtime,high_leadtime
data['compressionratio']=np.where(data['compressionratio']<low_leadtime,low_leadtime,c
```

Encoding categorical columns

In [ ]:
```
lst=[]
for i in data.columns:
    if data[i].dtype=='object':
        lst.append(i)
lst
```

Out[ ]:
```
['fueltype',
 'aspiration',
 'doornumber',
 'carbody',
 'drivewheel',
 'enginelocation',
 'enginetype',
 'cylindernumber',
 'fuelsystem']
```

In [ ]:
```
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder()
data=pd.get_dummies(data,columns=['fueltype',
 'aspiration',
 'doornumber',
 'carbody',
 'drivewheel',
 'enginelocation',
 'enginetype',
 'cylindernumber',
 'fuelsystem'])
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(205, 53)
```

Applying models

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.linear_model import Lasso, Ridge
import xgboost as xgb
```

In [ ]:
```
y = data['price']
x = data.drop(columns=['price'])
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=
```

In [ ]:
```
lr = LinearRegression()
gb = GradientBoostingRegressor()
rf = RandomForestRegressor()
ls = Lasso()
rd = Ridge()
```

In [ ]:
```
lr.fit(x_train, y_train)
gb.fit(x_train, y_train)
rf.fit(x_train, y_train)
ls.fit(x_train, y_train)
rd.fit(x_train, y_train)
```

Out[ ]: ▾ Ridge

Ridge()

In [ ]:
```
plrtr = lr.predict(x_train)
plrte = lr.predict(x_test)

pgbtr = gb.predict(x_train)
pgbte = gb.predict(x_test)

prftr = rf.predict(x_train)
prfte = rf.predict(x_test)

plstr = ls.predict(x_train)
plste = ls.predict(x_test)

prdtr = rd.predict(x_train)
prdte = rd.predict(x_test)
```

In [ ]:
```
print(r2_score(y_train, plrtr))
print(r2_score(y_test, plrte))
```

```
0.9387450937876193
0.8839592012175543
```

In [ ]:
```
print(r2_score(y_train, pgbtr))
print(r2_score(y_test, pgbte))
```

```
0.9940039372167675
0.9511714314281142
```

In [ ]:
```
print(r2_score(y_train, prftr))
print(r2_score(y_test, prfte))
```

```
0.9875171514009213
0.9554492021406589
```

In [ ]:
```
print(r2_score(y_train, plstr))
print(r2_score(y_test, plste))
```

```
0.9386844106484762
0.8893497025186052
```

In [ ]:
```
print(r2_score(y_train, prdtr))
print(r2_score(y_test, prdte))
```

```
0.9326868128089326
0.9180301614576438
```

In [ ]:
```
models = pd.DataFrame(
    {
        'Model' : ['LR', 'LR', 'GB', 'GB', 'RF', 'RF', 'LS', 'LS', 'RD', 'RD'],
        'Group' : [
            'train',
            'test',
            'train',
            'test',
            'train',
            'test',
            'train',
```

```
            'test',
            'train',
            'test',],
        'Accuracy2' : [
            r2_score(y_test, plrte)*100,
            r2_score(y_test, plrte)*100,
            r2_score(y_test, pgbte)*100,
            r2_score(y_test, pgbte)*100,
            r2_score(y_test, prfte)*100,
            r2_score(y_test, prfte)*100,
            r2_score(y_test, plste)*100,
            r2_score(y_test, plste)*100,
            r2_score(y_test, prdte)*100,
            r2_score(y_test, prdte)*100,
        ]

    }
)
```

In [ ]:

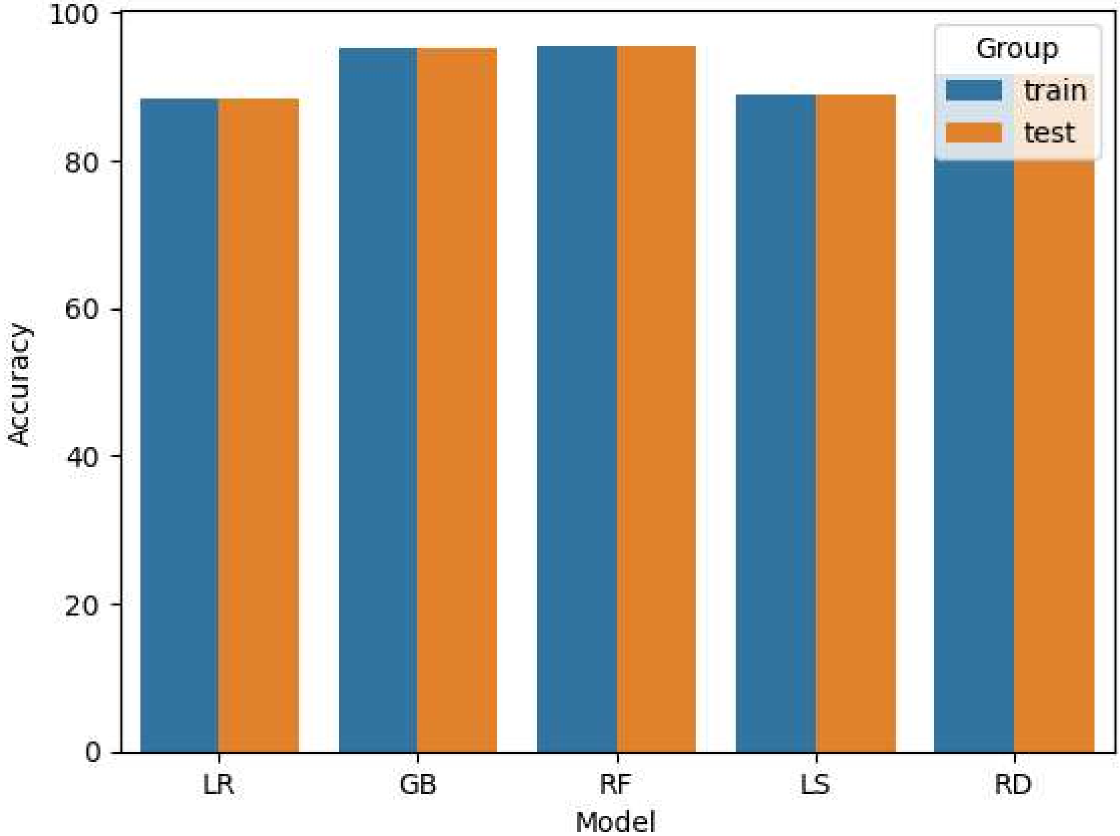
```
models
```

Out[ ]:

	Model	Group	Accuracy2
0	LR	train	88.395920
1	LR	test	88.395920
2	GB	train	95.117143
3	GB	test	95.117143
4	RF	train	95.544920
5	RF	test	95.544920
6	LS	train	88.934970
7	LS	test	88.934970
8	RD	train	91.803016
9	RD	test	91.803016

In [ ]:

```
import matplotlib.pyplot as plt
sns.barplot(
    x='Model',
    y='Accuracy2',
    hue='Group',
    data= models
)
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```


100
80
60
40
20
0
Accuracy
LR
GB
RF
LS
RD
Model
Group
train
test