MLDL FINAL PROJECT CODE DOCUMENT

-Likitha Guthikonda

811150752

lguthiko@kent.edu

n [135…

```
import pandas                  as pd
import numpy                   as np
import matplotlib.pyplot       as plt
import seaborn                 as sns
import matplotlib.image        as mpimg

from IPython.core.display    import HTML
from IPython.display         import Image
from tabulate                import tabulate
#from mpl_toolkits.basemap    import Basemap
from scipy.stats             import chi2_contingency

from sklearn.preprocessing   import RobustScaler, MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from boruta                  import BorutaPy
from sklearn.ensemble        import RandomForestRegressor
from sklearn.linear_model    import LinearRegression, Lasso
from sklearn.ensemble        import RandomForestRegressor
import xgboost                 as xgb

from sklearn.metrics         import mean_absolute_error, mean_squared_error
```

```
def mean_percentage_error( y, yhat ):
    return np.mean( ( y - yhat ) / y )

def jupyter_settings():
    %matplotlib inline
    %pylab inline

    plt.style.use( 'bmh' )
    plt.rcParams['figure.figsize'] = [25, 12]
    plt.rcParams['font.size'] = 24

    display( HTML( '<style>.container { width:100% !important; }</style>') )
    pd.options.display.max_columns = None
    pd.options.display.max_rows = None
    pd.set_option( 'display.expand_frame_repr', False )

    sns.set()

def cramer_v( x, y ):
    cm = pd.crosstab( x, y ).values
    n = cm.sum()
    r, k = cm.shape

    chi2 = chi2_contingency( cm )[0]
    chi2corr = max( 0, chi2 - (k-1)*(r-1)/(n-1) )

    kcorr = k - (k-1)**2/(n-1)
    rcorr = r - (r-1)**2/(n-1)

    return np.sqrt( (chi2corr/n) / ( min( kcorr-1, rcorr-1 ) ) )

def ml_error( model_name, y, yhat ):
    mae = mean_absolute_error( y, yhat )
    mape = mean_absolute_percentage_error( y, yhat )
    rmse = np.sqrt( mean_squared_error( y, yhat ) )

    return pd.DataFrame( { 'Model Name': model_name,
                           'MAE': mae,
                           'MAPE': mape,
                           'RMSE': rmse }, index=[0] )

def mean_absolute_percentage_error( y, yhat ):
    return np.mean( np.abs( (y - yhat ) / y ) )
```

```python
def mean_absolute_percentage_error( y, yhat ):
    return np.mean( np.abs( (y - yhat ) / y ) )

def cross_validation( X_training, kfold, model_name, model, verbose=False ):

    mae_list = []
    mape_list = []
    rmse_list = []

    for k in reversed( range( 1, kfold+1 ) ):
        if verbose:
            print( '\nKFold Number: {}'.format( k ) )

        # filtering dataset
        training = X_training
        validation = X_training

        # training and validation dataset
        # training
        xtraining = training.drop( ['price' ], axis=1 )
        ytraining = training['price']

        #validation
        xvalidation = validation.drop( ['price'], axis=1 )
        yvalidation = validation['price']

        # model
        m = model.fit( xtraining, ytraining )

        # prediction
        yhat = m.predict( xvalidation )

        # performance
        m_result = ml_error( model_name, np.expm1( yvalidation ), np.expm1( yhat ) )

        # store performance of each kfold interation
        mae_list.append( m_result['MAE'] )
        mape_list.append( m_result['MAPE'] )
        rmse_list.append( m_result['RMSE'] )

    return pd.DataFrame( { 'Model Name': model_name,
                           'MAE CV': np.round( np.mean( mae_list ), 2 ).astype( str ) + ' +/- ' + np.round( np.std( mae_list ), 2 ).astype( str ),
                           'MAPE CV': np.round( np.mean( mape_list ), 2 ).astype( str ) + ' +/- ' + np.round( np.std( mape_list ), 2 ).astype( str ),
                           'RMSE CV': np.round( np.mean( rmse_list ), 2 ).astype( str ) + ' +/- ' + np.round( np.std( rmse_list ), 2 ).astype( str )
```

In [3]:

```python
jupyter_settings()
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [4]:

```python
df_raw = pd.read_csv('data/vehicles.csv', low_memory=False, error_bad_lines=False)
```

In [5]:

```python
df_raw.sample()
```

Out[5]:

	id	url	region	region_url	price	year	manufacturer	model	condition	cylinders	fuel	odometer
420470	7116954402	https://charleston.craigslist.org/ctd/d/columb...	charleston	https://charleston.craigslist.org	22995	2017.0	ford	f150 crew cab	NaN	NaN	gas	65650.0

```python
df1 = df_raw.copy()
```

```python
df1.columns

# The columns already have a label that I want and easy to understand.
```

```
Index(['id', 'url', 'region', 'region_url', 'price', 'year', 'manufacturer',
       'model', 'condition', 'cylinders', 'fuel', 'odometer', 'title_status',
       'transmission', 'vin', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'description', 'county', 'state', 'lat', 'long'],
      dtype='object')
```

```python
print( 'Number of Rows: {}'.format( df1.shape[0] ) )
print( 'Number of Cols: {}'.format( df1.shape[1] ) )
# Evaluate the possibilite do use this project in your computer
```

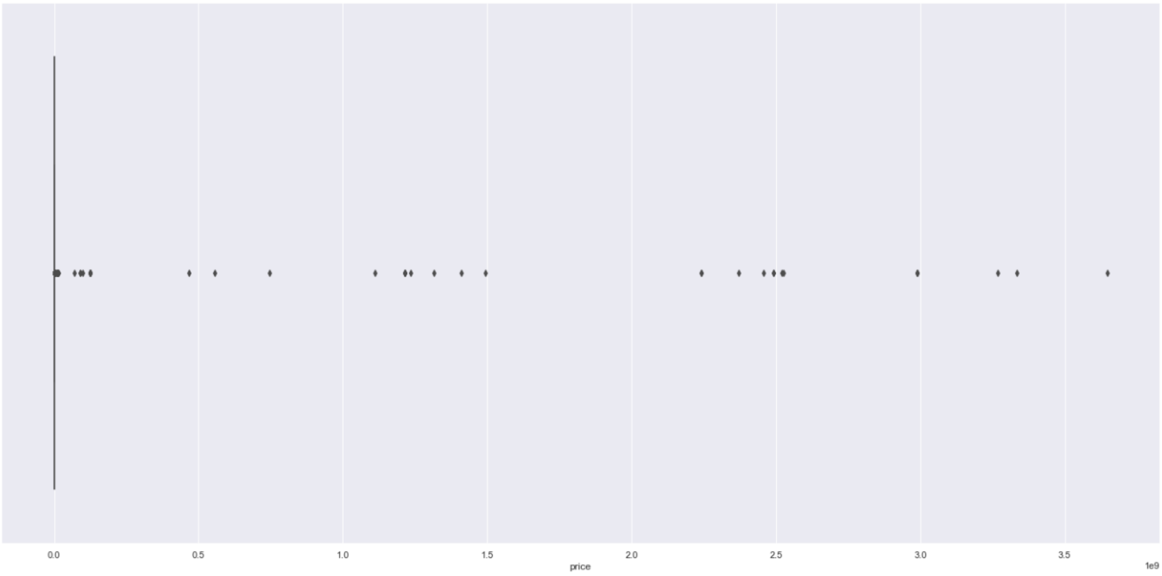
```
Number of Rows: 435849
Number of Cols: 25
```

```python
df1.dtypes
# At first, the types of the variables are corrected.
```

```
id                int64
url              object
region           object
region_url       object
price             int64
year            float64
manufacturer     object
model            object
condition        object
cylinders        object
fuel             object
odometer        float64
title_status     object
transmission     object
vin              object
drive            object
size             object
type             object
paint_color      object
image_url        object
description      object
county          float64
state            object
lat             float64
long            float64
dtype: object
```

```python
sns.boxplot( df1['price'] )
```

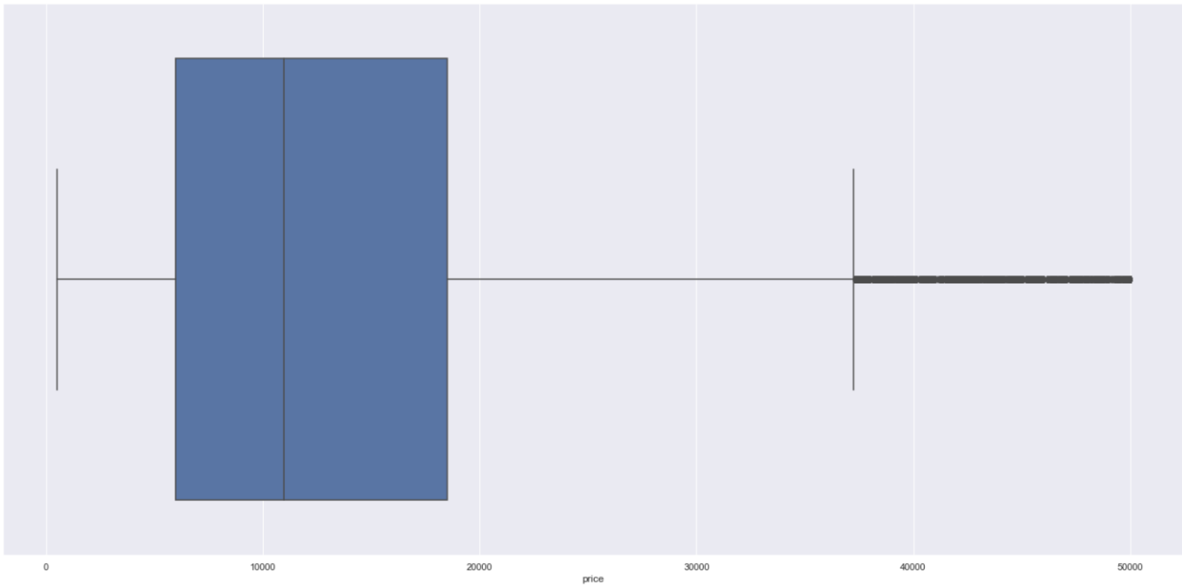
<AxesSubplot:xlabel='price'>




```
df1 = df1[(df1['price']>500) & (df1['price']<50000)]
```

```
sns.boxplot( df1['price'] )
```

<AxesSubplot:xlabel='price'>



In [13]:
```
df1.shape
```

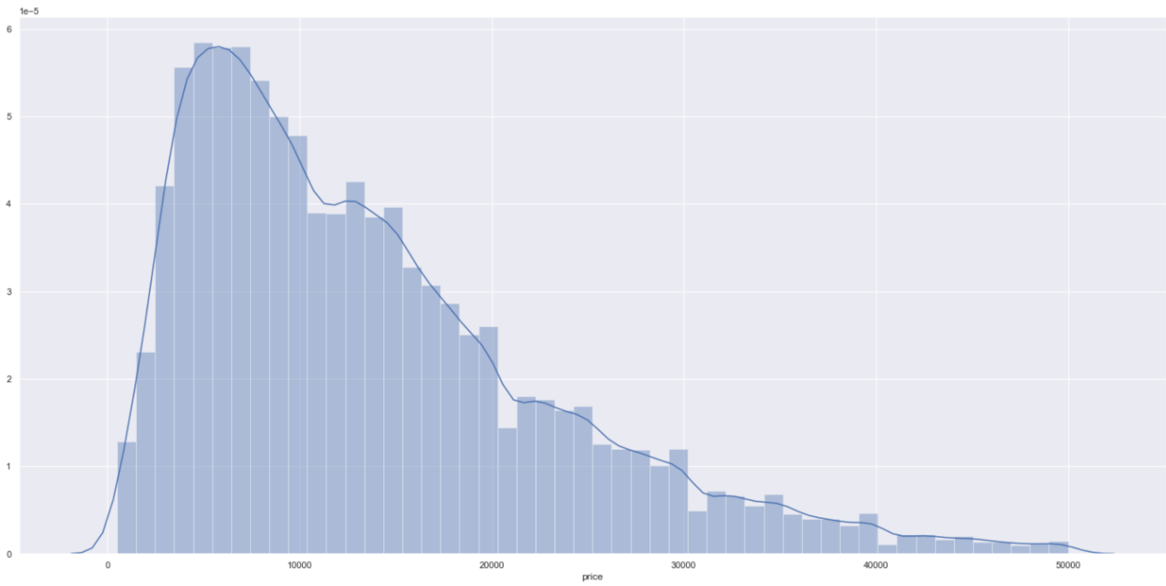
Out[13]: (390861, 25)

In [14]:
```python
df1.isna().sum()
```

Out[14]:
```
id                   0
url                  0
region               0
region_url           0
price                0
year               839
manufacturer     16650
model             5393
condition       157677
cylinders       143975
fuel              2703
odometer         63633
title_status      1584
transmission      1848
vin             178573
drive           106624
size            264047
type            103913
paint_color     117140
image_url           22
description         24
county          390861
state                0
lat               3333
long              3333
dtype: int64
```

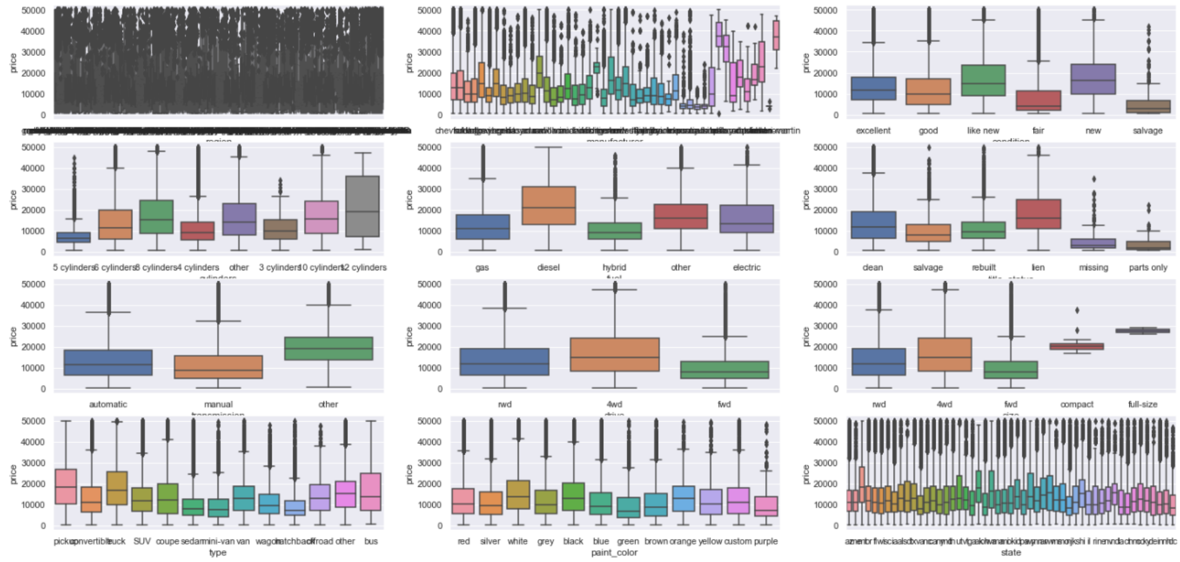
```python
sns.distplot( df1['price'] )
# Observing the distribution of the target variable, we can already see the presence of outliers. This problem will be addressed later.
```

```
<AxesSubplot:xlabel='price'>
```



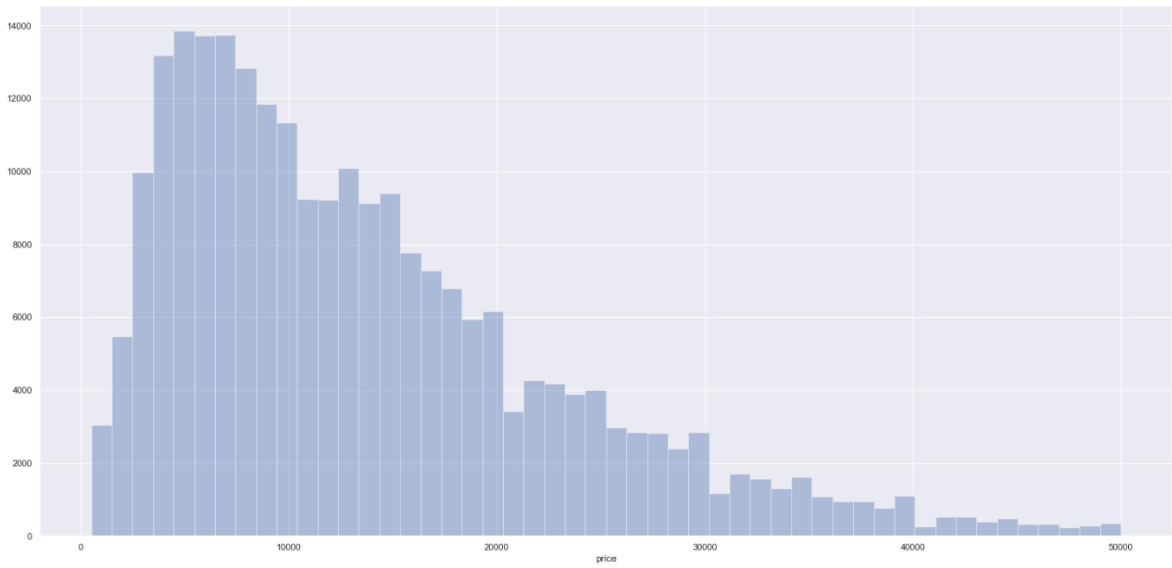
```
aux1 = df1[ df1['price'] > 0 ]


plt.subplot(4, 3, 1)
sns.boxplot( x= 'region', y='price' , data=aux1 )

plt.subplot(4, 3, 2)
sns.boxplot( x= 'manufacturer', y='price' , data=aux1 )

plt.subplot(4, 3, 3)
sns.boxplot( x= 'condition', y='price' , data=aux1 )

plt.subplot(4, 3, 4)
sns.boxplot( x= 'cylinders', y='price' , data=aux1 )

plt.subplot(4, 3, 5)
sns.boxplot( x= 'fuel', y='price' , data=aux1 )

plt.subplot(4, 3, 6)
sns.boxplot( x= 'title_status', y='price' , data=aux1 )

plt.subplot(4, 3, 7)
sns.boxplot( x= 'transmission', y='price' , data=aux1 )

plt.subplot(4, 3, 8)
sns.boxplot( x= 'drive', y='price' , data=aux1 )

plt.subplot(4, 3, 9)
sns.boxplot( x= 'size', y='price' , data=aux1 )

plt.subplot(4, 3, 10)
sns.boxplot( x= 'type', y='price' , data=aux1 )

plt.subplot(4, 3, 11)
sns.boxplot( x= 'paint_color', y='price' , data=aux1 )

plt.subplot(4, 3, 12)
sns.boxplot( x= 'state', y='price' , data=aux1 )
```

```
<AxesSubplot:xlabel='state', ylabel='price'>
```



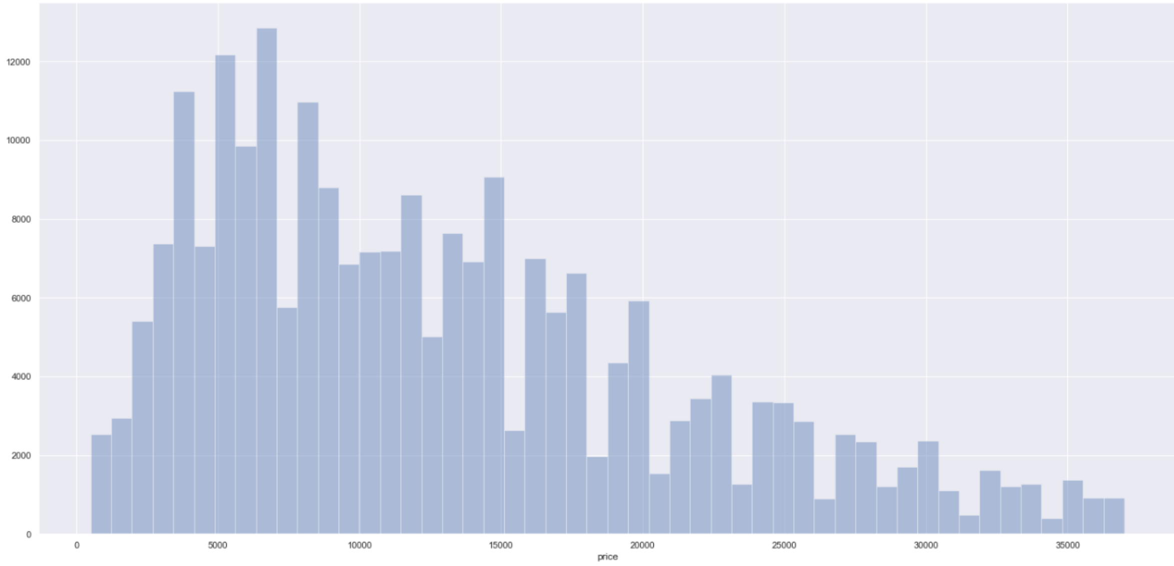

```python
sns.distplot( df4['price'], kde=False )
```

```
<AxesSubplot:xlabel='price'>
```

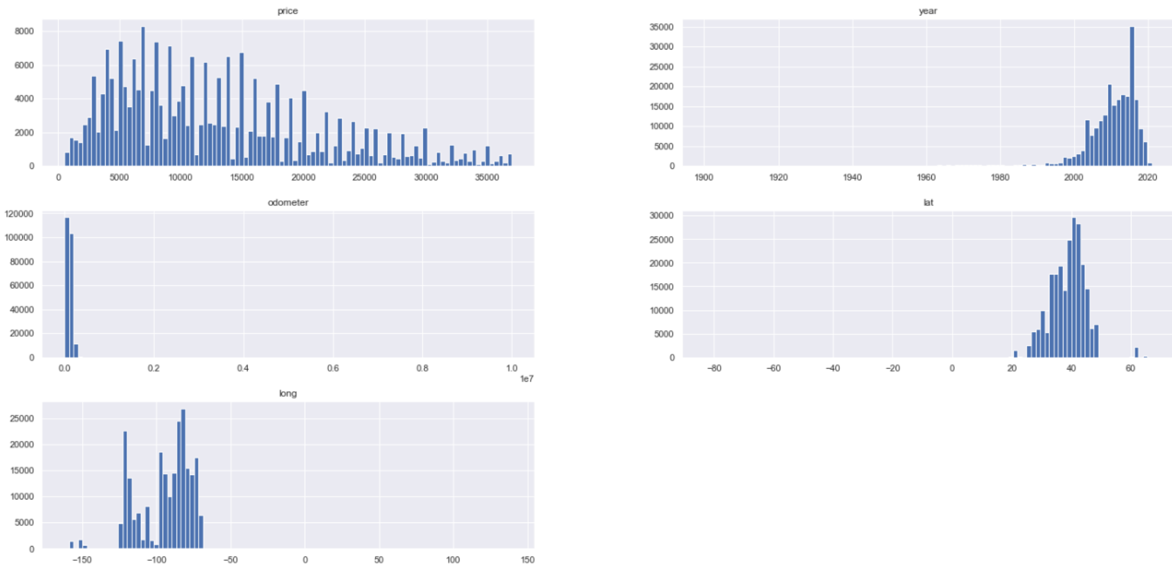


```python
sns.distplot( df4['price'], kde=False )
# To better understand the shape of the response variable, I took some possible outliers.
# I still haven't removed the outliers because I want to use them at EDA and see if some of them can define the value of cars that are worth much abov
```

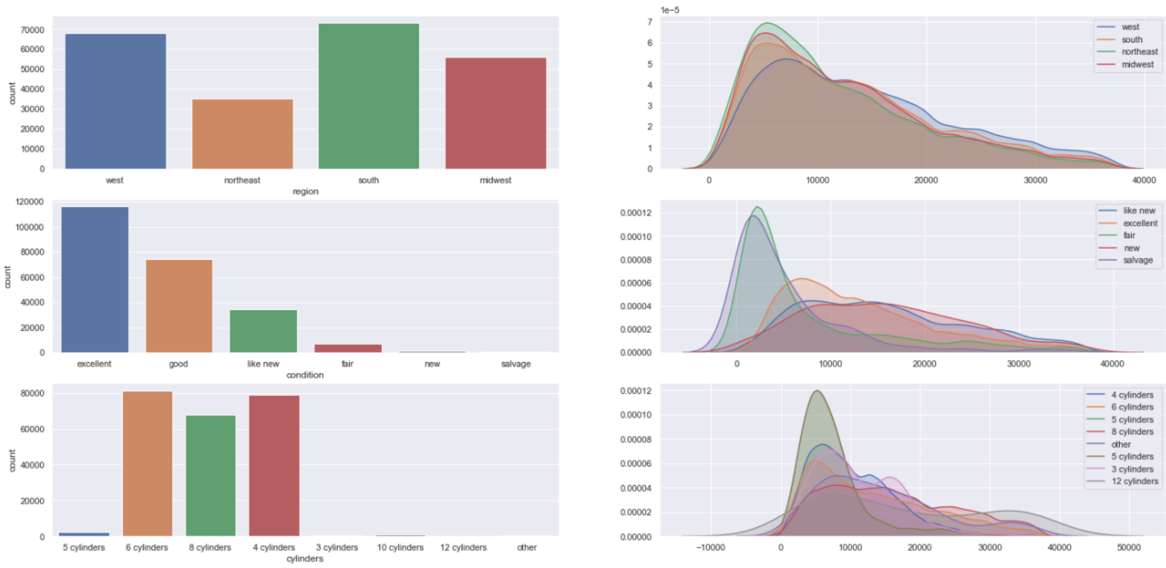
<AxesSubplot:xlabel='price'>



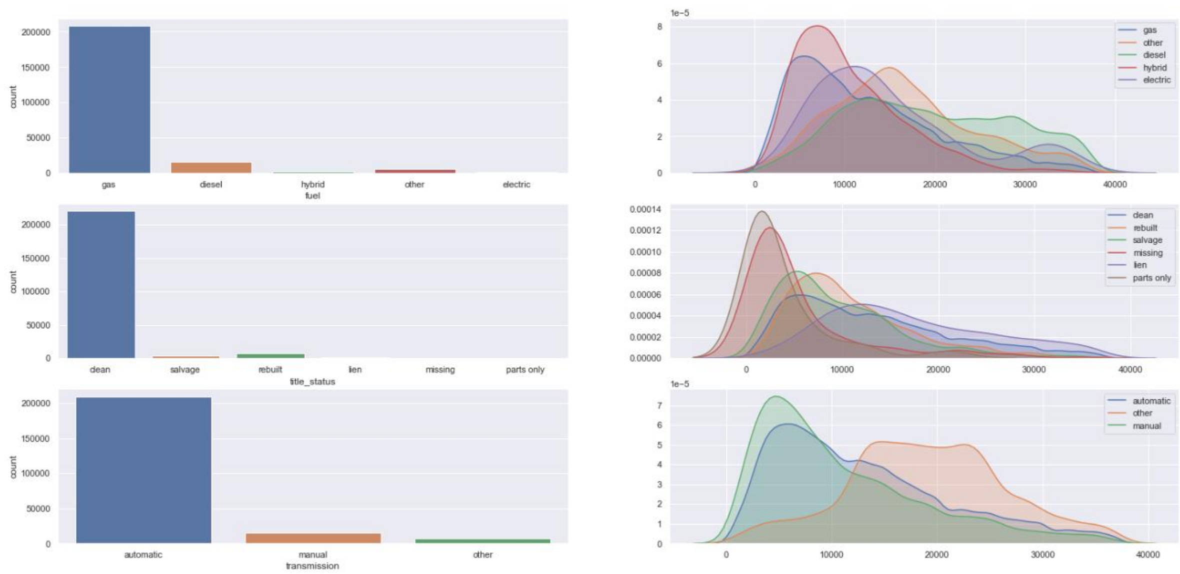

```python
# After the Feature Engineering let's reset the numerical attributes.

num_attributes = df4.select_dtypes( include=['int64', 'float64'] )
num_attributes.hist( bins=100 );
```



```
# region

plt.subplot( 3, 2, 1)
sns.countplot( df4['region'] )

plt.subplot( 3, 2, 2)
sns.kdeplot( df4[df4['region'] == 'west']['price'], label='west', shade=True )
sns.kdeplot( df4[df4['region'] == 'south']['price'], label='south', shade=True )
sns.kdeplot( df4[df4['region'] == 'northeast']['price'], label='northeast', shade=True )
sns.kdeplot( df4[df4['region'] == 'midwest']['price'], label='midwest', shade=True )

# condition

plt.subplot( 3, 2, 3)
sns.countplot( df4['condition'] )

plt.subplot( 3, 2, 4)
sns.kdeplot( df4[df4['condition'] == 'like new']['price'], label='like new', shade=True )
sns.kdeplot( df4[df4['condition'] == 'excellent']['price'], label='excellent', shade=True )
sns.kdeplot( df4[df4['condition'] == 'unknown']['price'], label='unknown', shade=True )
sns.kdeplot( df4[df4['condition'] == 'fair']['price'], label='fair', shade=True )
sns.kdeplot( df4[df4['condition'] == 'new']['price'], label='new', shade=True )
sns.kdeplot( df4[df4['condition'] == 'salvage']['price'], label='salvage', shade=True )

# cylinders

plt.subplot( 3, 2, 5)
sns.countplot( df4['cylinders'] )

plt.subplot( 3, 2, 6)
sns.kdeplot( df4[df4['cylinders'] == '4 cylinders']['price'], label='4 cylinders', shade=True )
sns.kdeplot( df4[df4['cylinders'] == 'unknown']['price'], label='unknown', shade=True )
sns.kdeplot( df4[df4['cylinders'] == '6 cylinders']['price'], label='6 cylinders', shade=True )
sns.kdeplot( df4[df4['cylinders'] == '5 cylinders']['price'], label='5 cylinders', shade=True )
sns.kdeplot( df4[df4['cylinders'] == '8 cylinders']['price'], label='8 cylinders', shade=True )
sns.kdeplot( df4[df4['cylinders'] == 'other']['price'], label='other', shade=True )
sns.kdeplot( df4[df4['cylinders'] == '5 cylinders']['price'], label='5 cylinders', shade=True )
sns.kdeplot( df4[df4['cylinders'] == '3 cylinders']['price'], label='3 cylinders', shade=True )
sns.kdeplot( df4[df4['cylinders'] == '12 cylinders']['price'], label='12 cylinders', shade=True )
```

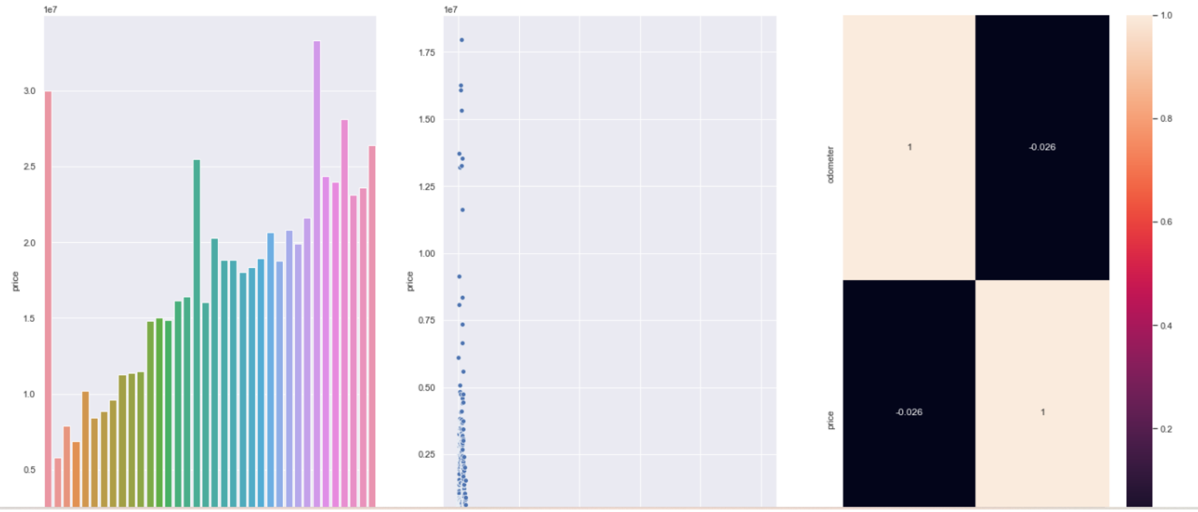


```
# fuel

plt.subplot( 3, 2, 1)
sns.countplot( df4['fuel'] )

plt.subplot( 3, 2, 2)
sns.kdeplot( df4[df4['fuel'] == 'gas']['price'], label='gas', shade=True )
sns.kdeplot( df4[df4['fuel'] == 'other']['price'], label='other', shade=True )
sns.kdeplot( df4[df4['fuel'] == 'diesel']['price'], label='diesel', shade=True )
sns.kdeplot( df4[df4['fuel'] == 'unknown']['price'], label='unknown', shade=True )
sns.kdeplot( df4[df4['fuel'] == 'hybrid']['price'], label='hybrid', shade=True )
sns.kdeplot( df4[df4['fuel'] == 'electric']['price'], label='electric', shade=True )

# title_status

plt.subplot( 3, 2, 3)
sns.countplot( df4['title_status'] )

plt.subplot( 3, 2, 4)
sns.kdeplot( df4[df4['title_status'] == 'clean']['price'], label='clean', shade=True )
sns.kdeplot( df4[df4['title_status'] == 'rebuilt']['price'], label='rebuilt', shade=True )
sns.kdeplot( df4[df4['title_status'] == 'salvage']['price'], label='salvage', shade=True )
sns.kdeplot( df4[df4['title_status'] == 'unknown']['price'], label='unknown', shade=True )
sns.kdeplot( df4[df4['title_status'] == 'missing']['price'], label='missing', shade=True )
sns.kdeplot( df4[df4['title_status'] == 'lien']['price'], label='lien', shade=True )
sns.kdeplot( df4[df4['title_status'] == 'parts only']['price'], label='parts only', shade=True )

# transmission

plt.subplot( 3, 2, 5)
sns.countplot( df4['transmission'] )

plt.subplot( 3, 2, 6)
sns.kdeplot( df4[df4['transmission'] == 'automatic']['price'], label='automatic', shade=True )
sns.kdeplot( df4[df4['transmission'] == 'other']['price'], label='other', shade=True )
sns.kdeplot( df4[df4['transmission'] == 'manual']['price'], label='manual', shade=True )
```

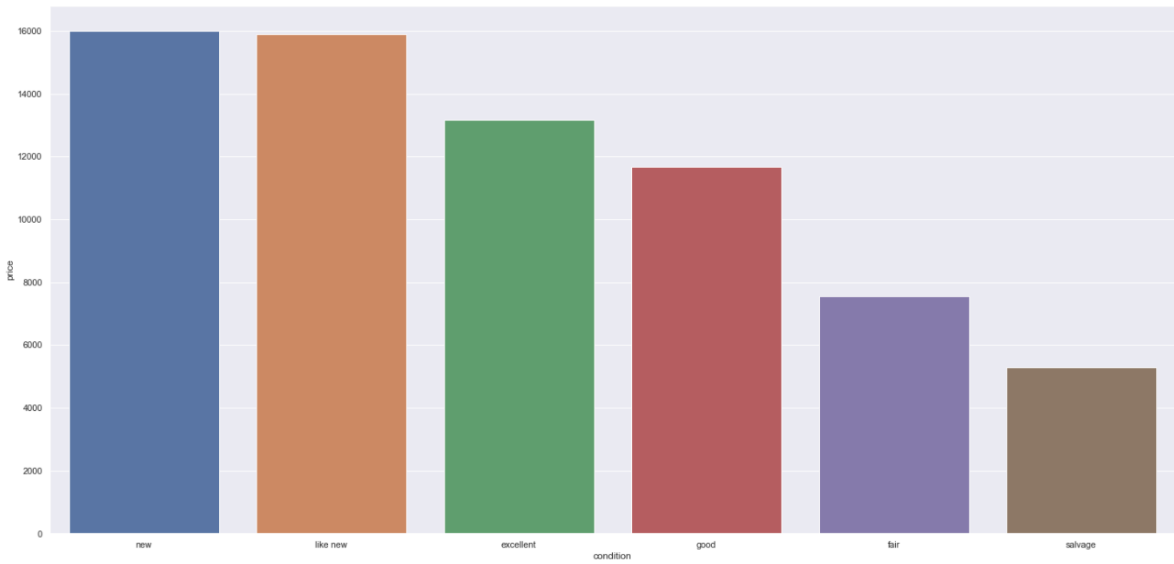



```
# drive
```

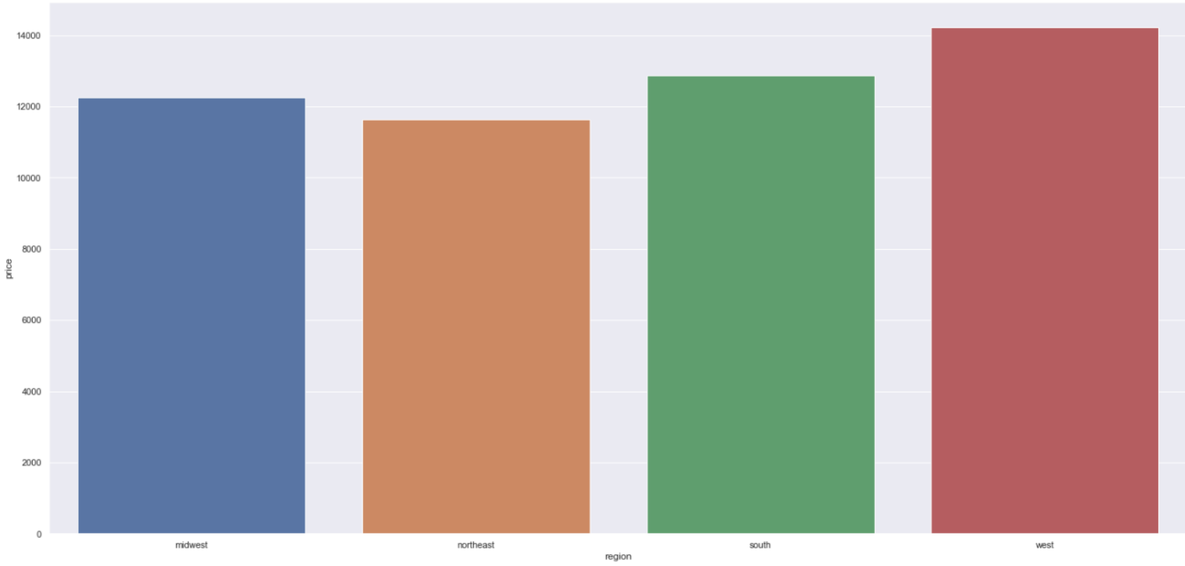
```
plt.subplot(1,3,1)
bins = list( np.arange(0, 37000, 1000) )
aux1['odometer_binned'] = pd.cut( aux1['odometer'], bins=bins )
aux2 = aux1[['odometer_binned', 'price']].groupby( 'odometer_binned' ).sum().reset_index()
sns.barplot( x='odometer_binned', y='price', data=aux2)
plt.xticks( rotation=90)

plt.subplot(1,3,2)
sns.scatterplot( x='odometer', y='price', data=aux1 )

plt.subplot(1,3,3)
sns.heatmap( aux1.corr( method='pearson' ), annot=True );

# As the odometer values are continuous numbers, to better visualize I arranged their values in bins.
# Looking at these three graphs, it became clear that the price of cars goes down while the odometer goes up.
# Looking at the heatmap we see that the influence on the price is very small anyway.
```



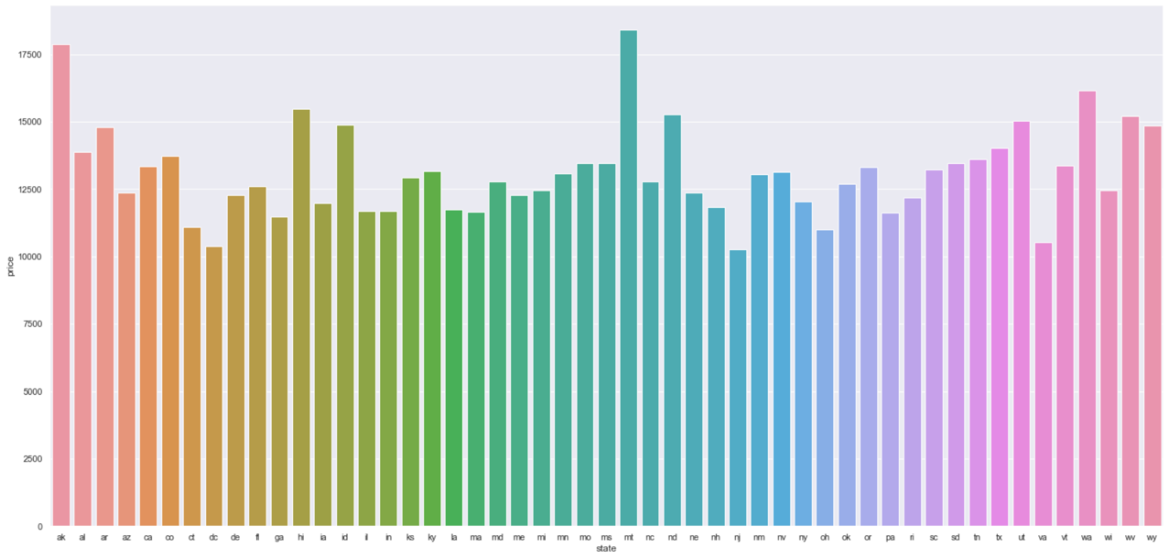

```
aux1 = df4[['condition', 'price']].groupby( 'condition' ).mean().reset_index()
sns.barplot( x='condition', y='price', data=aux1, order=['new', 'like new', 'excellent', 'good', 'fair', 'salvage']);

# Adjusting the order of the condition of the cars from the best condition to the worst condition, we see a drop in price as the condition of the car
```



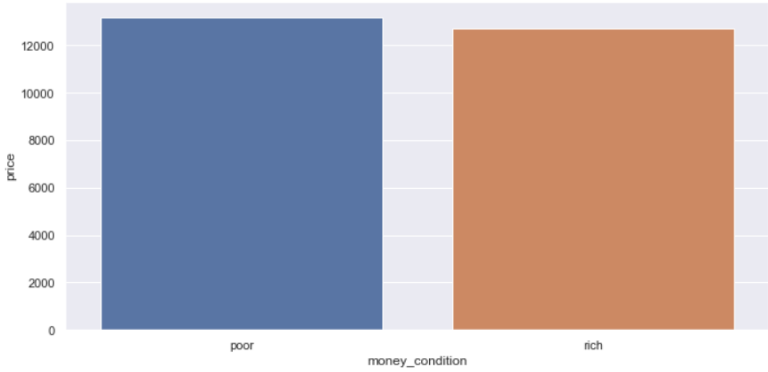
```
aux1 = df4[['region', 'price']].groupby( 'region' ).mean().reset_index()
sns.barplot( x='region', y='price', data=aux1);

# Just "west" region cost more. The "northeast" cost less.
```



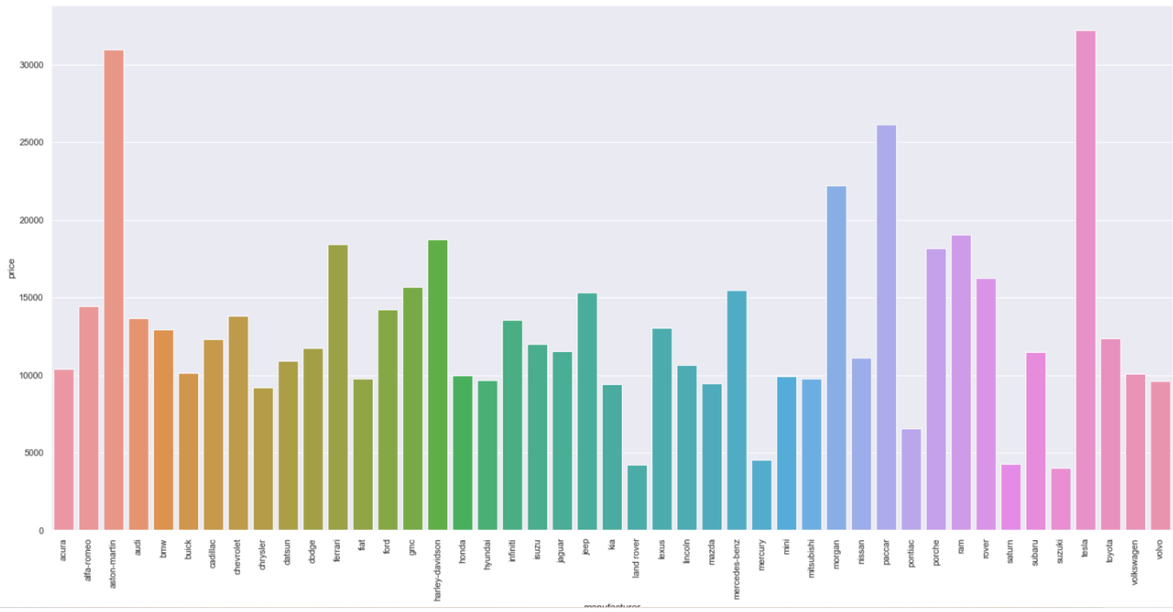

```
aux1 = df4[['state', 'price']].groupby( 'state' ).mean().reset_index()
sns.barplot( x='state', y='price', data=aux1);
```



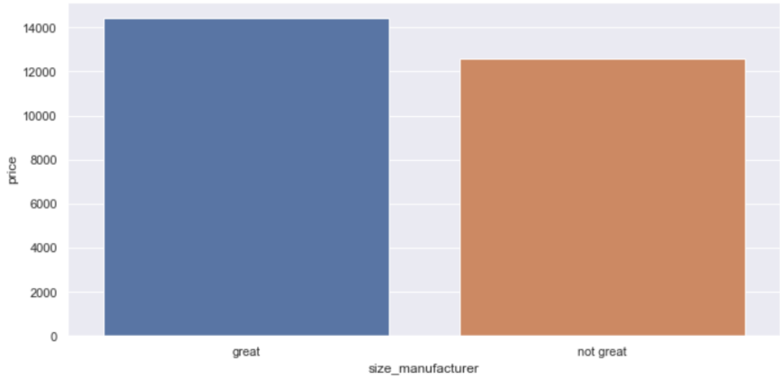
```python
# To visualize better the difference between rich states and poor states I will separate them.
# The top 10 states richest:
# nh, hi, mn, ut, nk, md, ma, va, co, nj.
aux1['money_condition'] = aux1['state'].apply( lambda x: 'rich' if x == 'nh' or x == 'hi' or x == 'mn' or x == 'ut' or x == 'nk' or x == 'md' or x ==
aux2 = aux1[['money_condition', 'price']].groupby( 'money_condition' ).mean().reset_index()

plt.subplot( 2, 2, 4 )
sns.barplot( x='money_condition', y='price', data=aux2 );
```



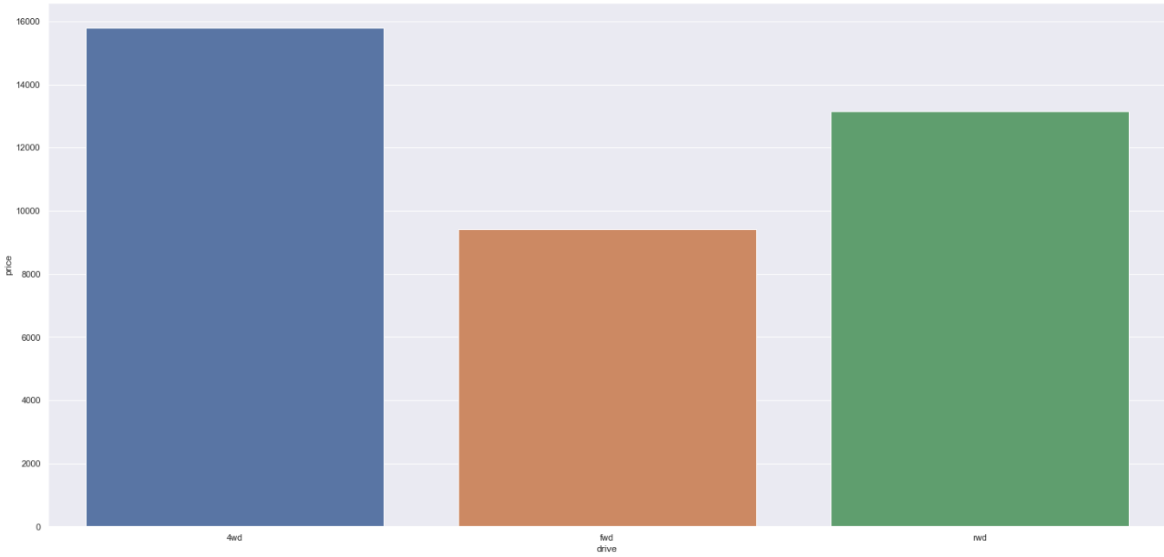

```python
aux1 = df4[['manufacturer', 'price']].groupby( 'manufacturer' ).mean().reset_index()
sns.barplot( x='manufacturer', y='price', data=aux1);
plt.xticks( rotation=90);
```



```python
# To visualize better the difference between great manufacturer and not so great manufacturer I will separate them.
# The top manufacturer are:
# toyota, gmc, chevrolet, volkswagen, ford, bmw, nissan, hyundai, honda, mercedes-benz, jaguar, ferrari and tesla
aux1['size_manufacturer'] = aux1['manufacturer'].apply( lambda x: 'great' if x == 'toyota' or x == 'gmc' or x == 'chevrolet' or x == 'volkswagen' o
aux2 = aux1[['size_manufacturer', 'price']].groupby( 'size_manufacturer' ).mean().reset_index()

plt.subplot( 2, 2, 4 )
sns.barplot( x='size_manufacturer', y='price', data=aux2 );
```

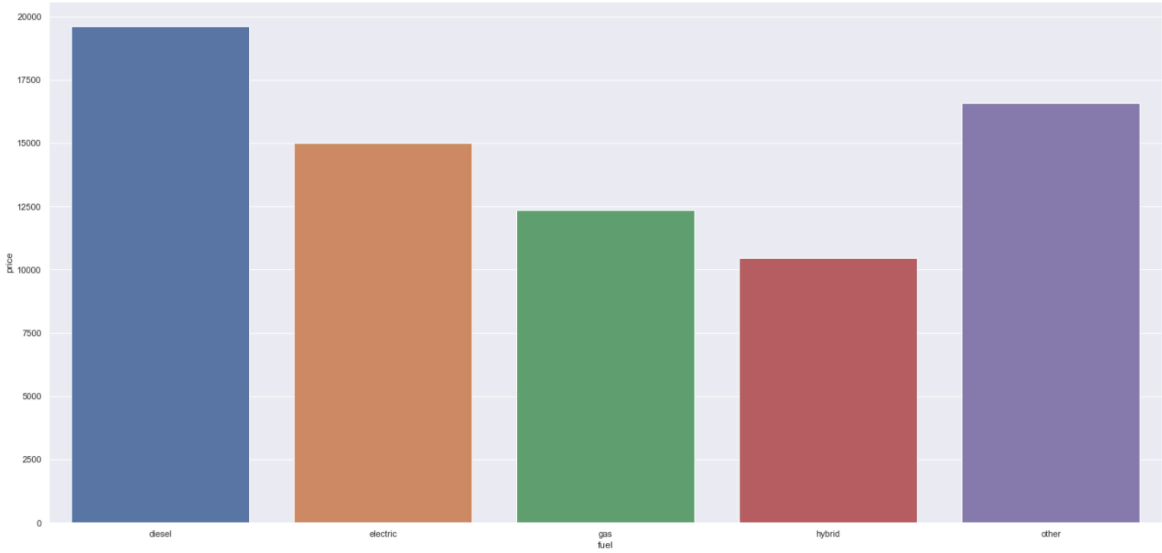



```python
aux1 = df4[['drive', 'price']].groupby( 'drive' ).mean().reset_index()
sns.barplot( x='drive', y='price', data=aux1);
```

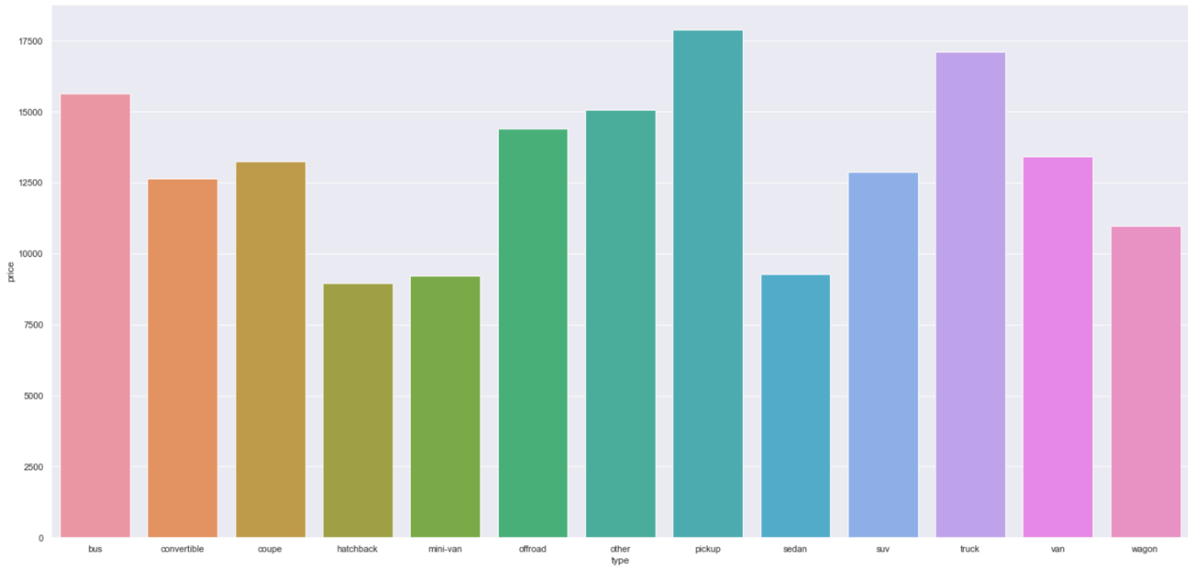


]:

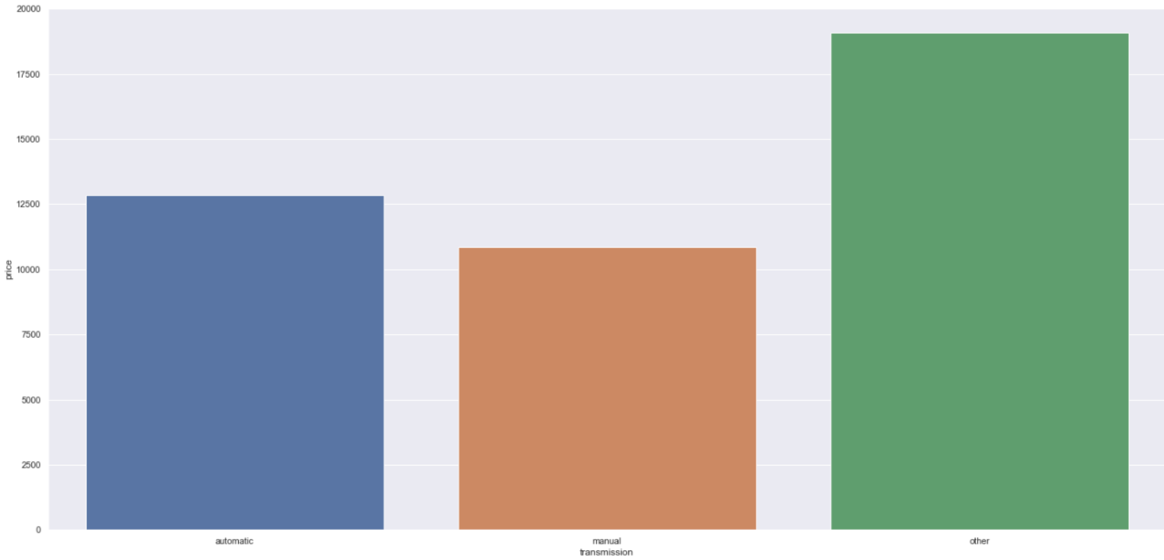
```
aux1 = df4[['fuel', 'price']].groupby( 'fuel' ).mean().reset_index()
sns.barplot( x='fuel', y='price', data=aux1);
```



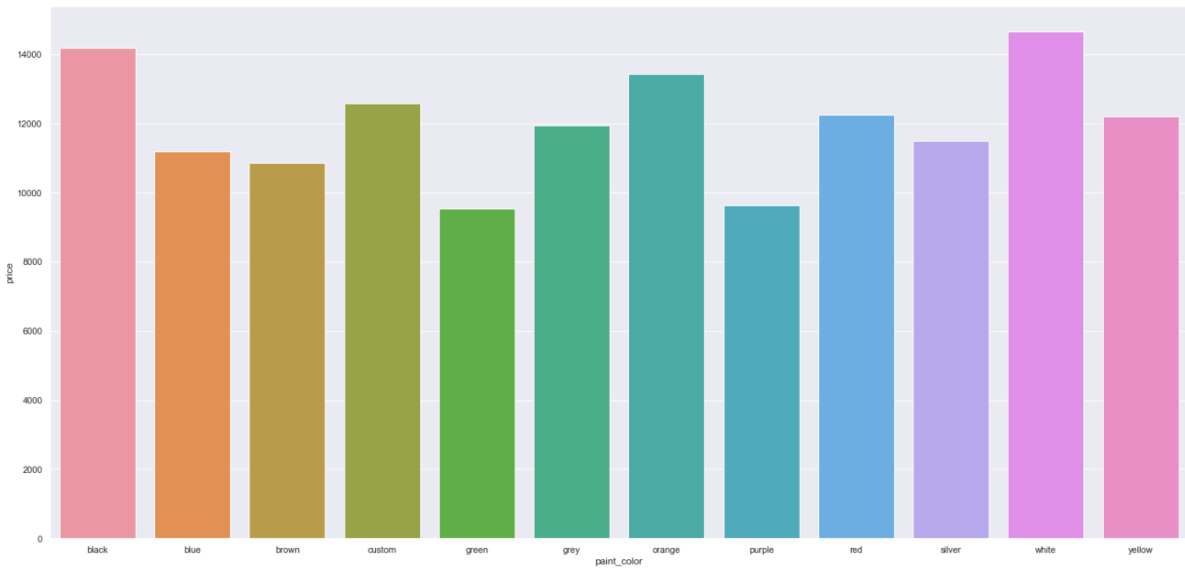

```
aux1 = df4[['type', 'price']].groupby( 'type' ).mean().reset_index()
sns.barplot( x='type', y='price', data=aux1);
```



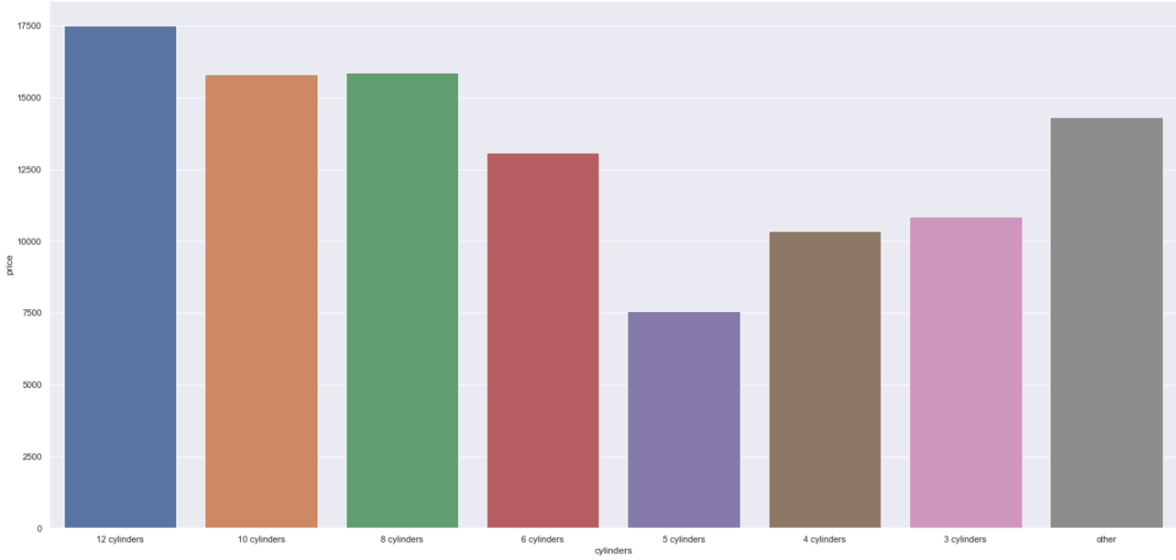
```python
aux1 = df4[['transmission', 'price']].groupby( 'transmission' ).mean().reset_index()
sns.barplot( x='transmission', y='price', data=aux1);
```



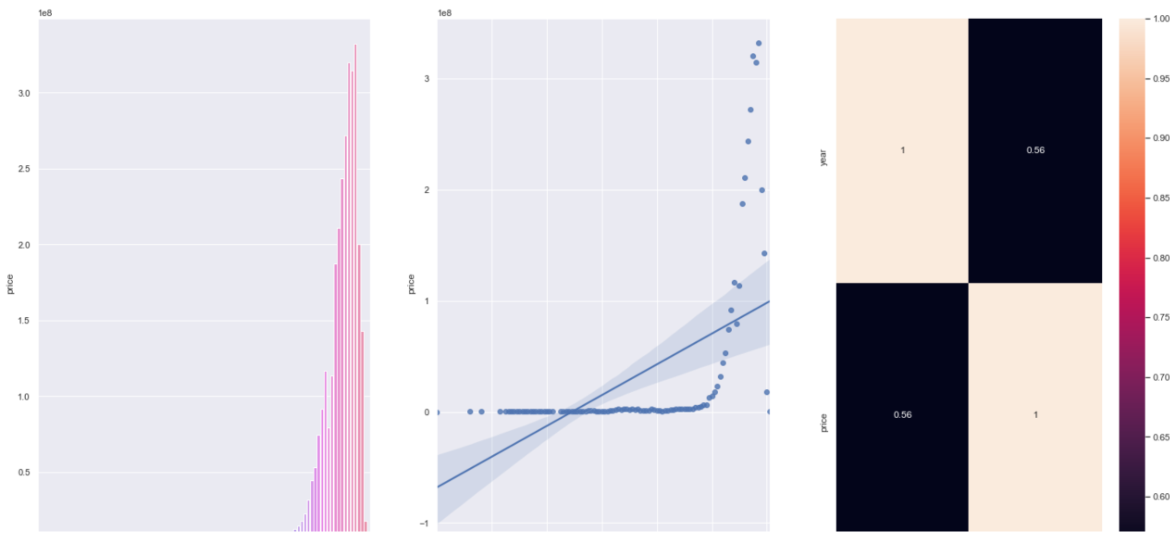

```python
aux1 = df4[['paint_color', 'price']].groupby( 'paint_color' ).mean().reset_index()
sns.barplot( x='paint_color', y='price', data=aux1);
```



```python
aux1 = df4[['cylinders', 'price']].groupby( 'cylinders' ).mean().reset_index()
sns.barplot( x='cylinders', y='price', data=aux1, order=['12 cylinders', '10 cylinders', '8 cylinders', '6 cylinders', '5 cylinders', '4 cylinders', '
```

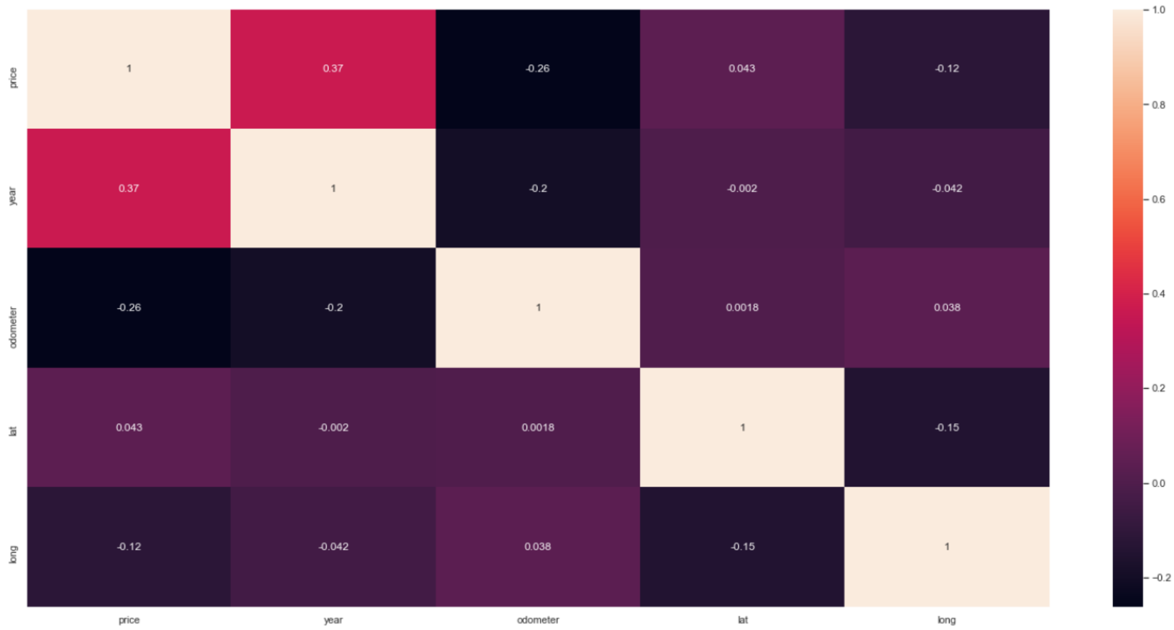



```python
aux1 = df4[['year', 'price']].groupby( 'year' ).sum().reset_index()

plt.subplot( 1, 3, 1 )
sns.barplot( x='year', y='price', data=aux1 );

plt.subplot( 1, 3, 2 )
sns.regplot( x='year', y='price', data=aux1 );

plt.subplot( 1, 3, 3 )
sns.heatmap( aux1.corr( method='pearson' ), annot=True );
```



```
correlation = num_attributes.corr( method='pearson' )
sns.heatmap( correlation, annot=True )
```

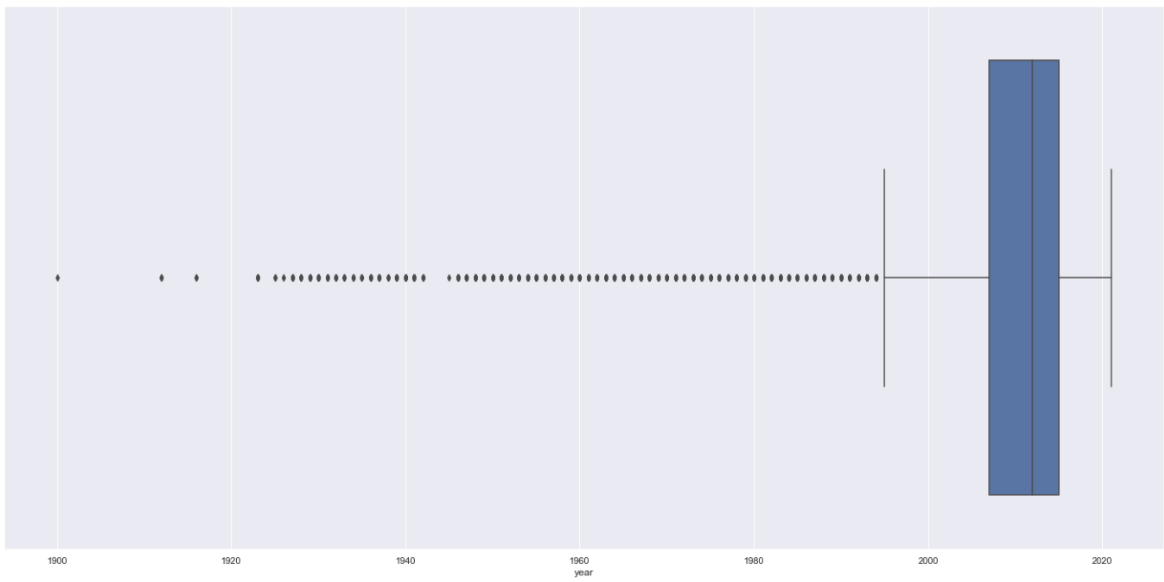
<AxesSubplot:>

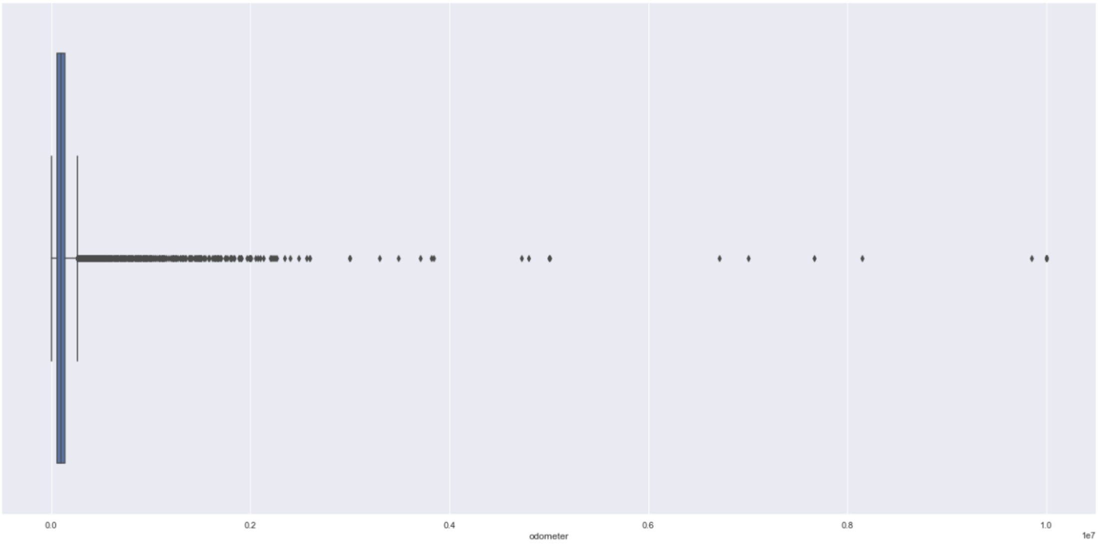



```
a = df5.select_dtypes( include=['int64', 'float64'] )
```

```
sns.boxplot( df5['year'] )
```

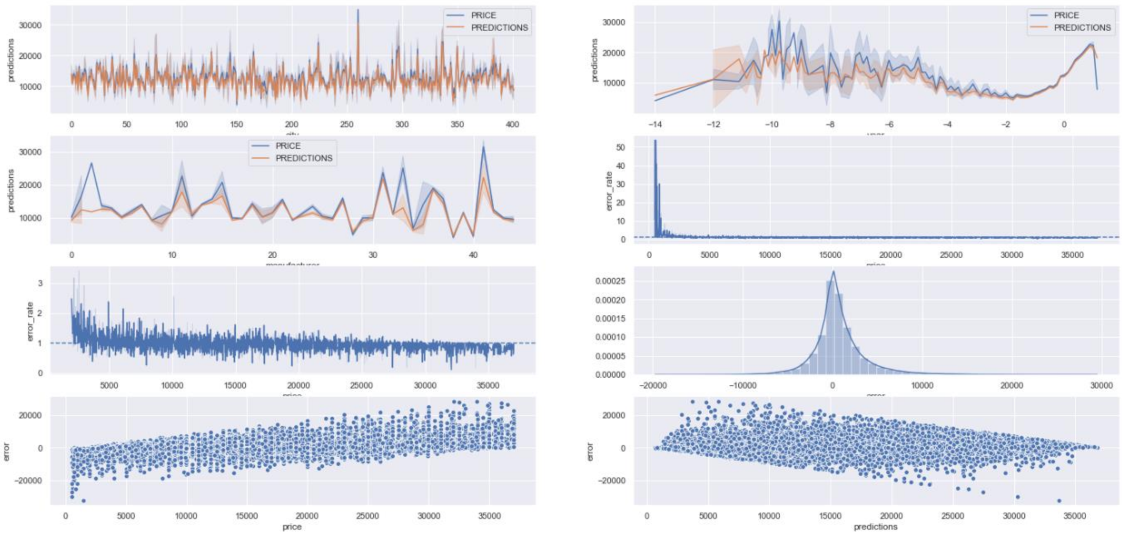
<AxesSubplot:xlabel='year'>



In [82]:
```
sns.boxplot( df5['odometer'] )
```
Out[82]:
```
<AxesSubplot:xlabel='odometer'>
```


In [83]:
```
sns.boxplot( df5['lat'] )
```

```
# Compare the real response with the one predicted and observe if the model is performing well.
plt.subplot( 4, 2, 1 )
sns.lineplot( x='city', y='price', data=df9, label='PRICE' )
sns.lineplot( x='city', y='predictions', data=df9, label='PREDICTIONS' )

plt.subplot( 4, 2, 2 )
sns.lineplot( x='year', y='price', data=df9, label='PRICE' )
sns.lineplot( x='year', y='predictions', data=df9, label='PREDICTIONS' )

plt.subplot( 4, 2, 3 )
sns.lineplot( x='manufacturer', y='price', data=df9, label='PRICE' )
sns.lineplot( x='manufacturer', y='predictions', data=df9, label='PREDICTIONS' )

# Insert a line passing the value 1 (one) to see the predictions in relation to a perfect prediction that would be the value 1 (one) itself.
plt.subplot( 4, 2, 4 )
sns.lineplot( x='price', y='error_rate', data=df9 )
plt.axhline( 1, linestyle='--' )

plt.subplot( 4, 2, 5 )
sns.lineplot( x=df9[df9['price']>2000]['price'], y='error_rate', data=df9 )
plt.axhline( 1, linestyle='--' )

# To see the distribution of errors.
plt.subplot( 4, 2, 6 )
sns.distplot( df9[df9['price']>4000]['error'] )

plt.subplot( 4, 2, 7 )
sns.scatterplot( df9['price'], df9['error'] )

plt.subplot( 4, 2, 8 )
sns.scatterplot( df9['predictions'], df9['error'] )
```



PRICE
PREDICTIONS
predictions
error_rate
error
price
predictions