In [73]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [74]:
```
df=pd.read_csv("Ford.csv")
```

In [75]:
```
df.head()
```

Out[75]:

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	Fiesta	2017	12000	Automatic	15944	Petrol	150	57.7	1.0
1	Focus	2018	14000	Manual	9083	Petrol	150	57.7	1.0
2	Focus	2017	13000	Manual	12456	Petrol	150	57.7	1.0
3	Fiesta	2019	17500	Manual	10460	Petrol	145	40.3	1.5
4	Fiesta	2019	16500	Automatic	1482	Petrol	145	48.7	1.0

In [76]:
```
df.shape
```

Out[76]:
```
(17966, 9)
```

In [77]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17966 entries, 0 to 17965
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         17966 non-null  object 
 1   year          17966 non-null  int64  
 2   price         17966 non-null  int64  
 3   transmission  17966 non-null  object 
 4   mileage       17966 non-null  int64  
 5   fuelType      17966 non-null  object 
 6   tax           17966 non-null  int64  
 7   mpg           17966 non-null  float64
 8   engineSize    17966 non-null  float64
dtypes: float64(2), int64(4), object(3)
memory usage: 1.2+ MB
```

In [78]:
```
df.describe()
```

Out[78]:

	year	price	mileage	tax	mpg	engineSize
count	17966.000000	17966.000000	17966.000000	17966.000000	17966.000000	17966.000000
mean	2016.866470	12279.534844	23362.608761	113.329456	57.906980	1.350807
std	2.050336	4741.343657	19472.054349	62.012456	10.125696	0.432367
min	1996.000000	495.000000	1.000000	0.000000	20.800000	0.000000
25%	2016.000000	8999.000000	9987.000000	30.000000	52.300000	1.000000
50%	2017.000000	11291.000000	18242.500000	145.000000	58.900000	1.200000
75%	2018.000000	15299.000000	31060.000000	145.000000	65.700000	1.500000
max	2060.000000	54995.000000	177644.000000	580.000000	201.800000	5.000000

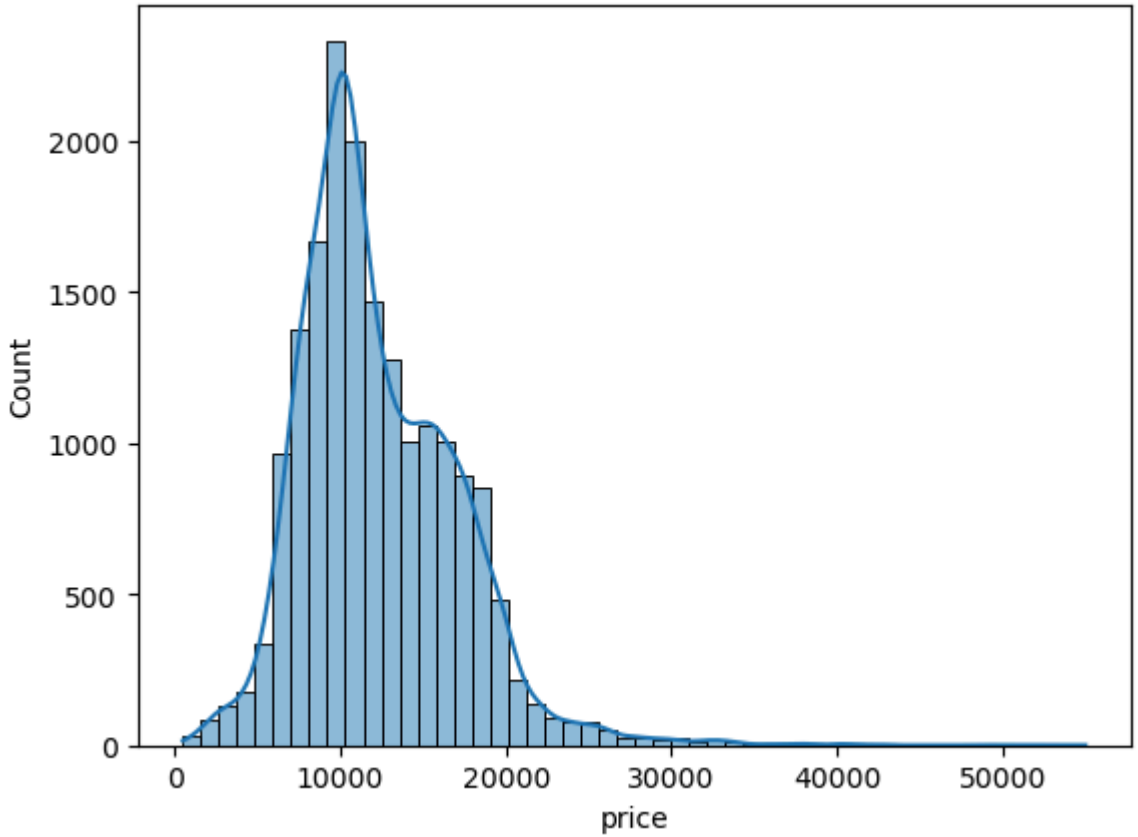
In [79]:
```
df.isnull().sum()
```

Out[79]:
```
model           0
year            0
price           0
transmission    0
mileage         0
fuelType        0
tax             0
mpg             0
engineSize      0
dtype: int64
```

EDA

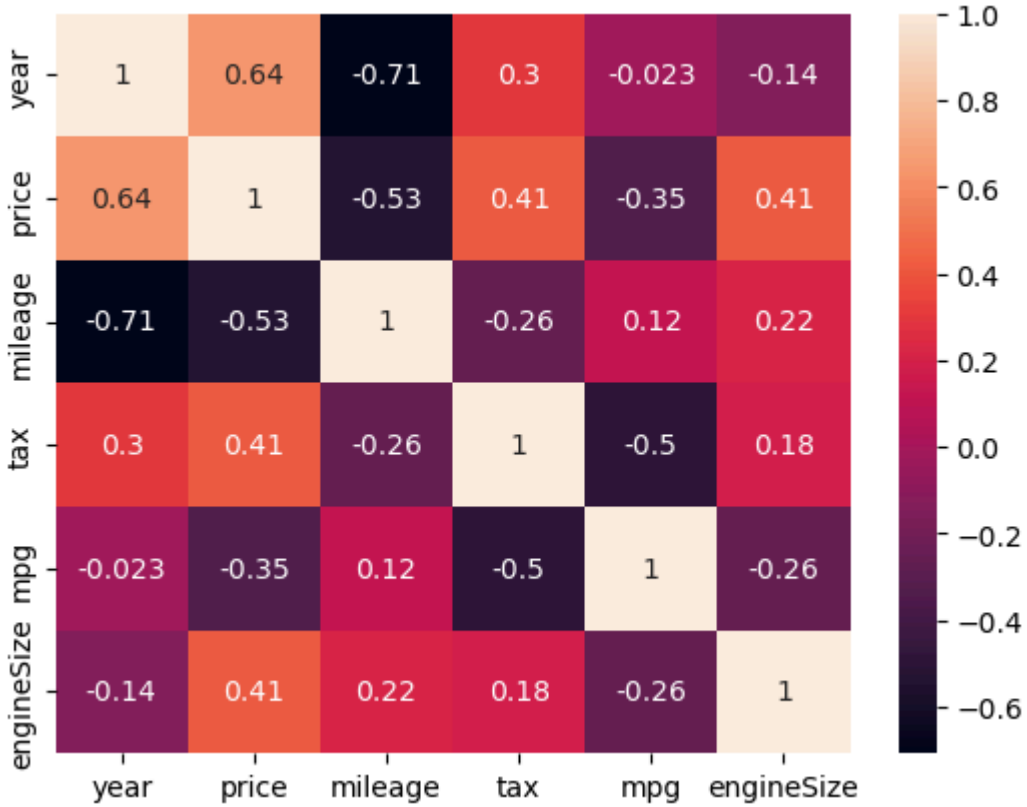
In [80]:
```
sns.histplot(df['price'],bins=50,kde=True)
```

Out[80]:
```
<Axes: xlabel='price', ylabel='Count'>
```



In [81]:
```
sns.heatmap(df.corr(numeric_only=True), annot=True)
```

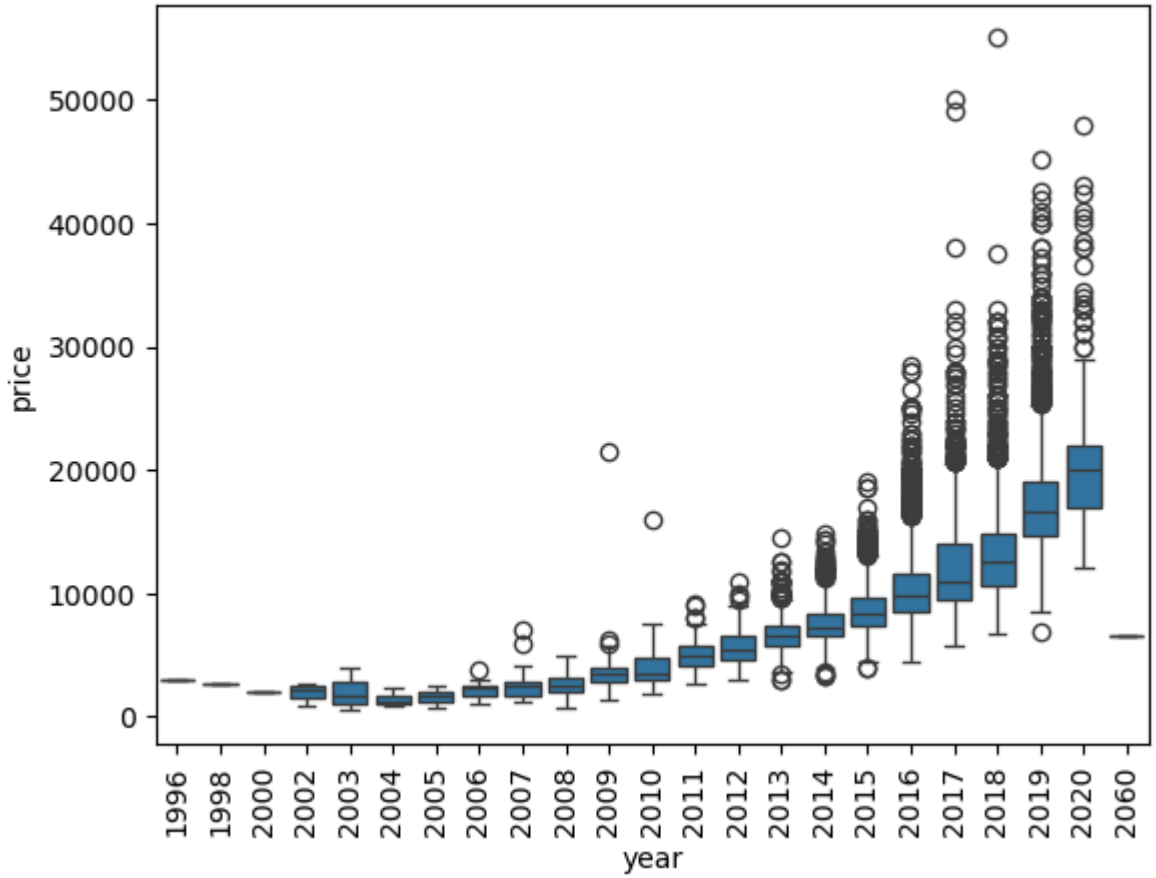
Out[81]: <Axes: >



In [82]:
```
sns.boxplot(data=df, x='year', y= 'price')
plt.xticks(rotation=90)
```

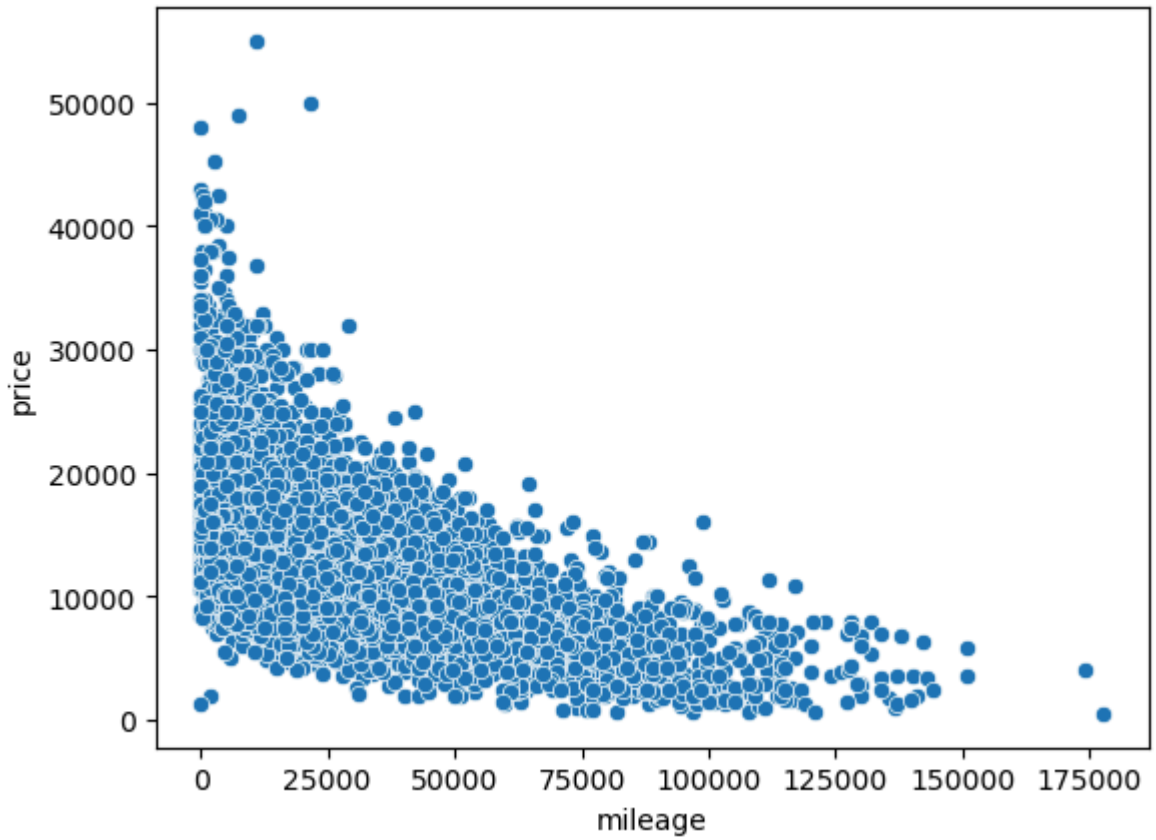
```
Out[82]: ([0,
           1,
           2,
           3,
           4,
           5,
           6,
           7,
           8,
           9,
           10,
           11,
           12,
           13,
           14,
           15,
           16,
           17,
           18,
           19,
           20,
           21,
           22],
          [Text(0, 0, '1996'),
           Text(1, 0, '1998'),
           Text(2, 0, '2000'),
           Text(3, 0, '2002'),
           Text(4, 0, '2003'),
           Text(5, 0, '2004'),
           Text(6, 0, '2005'),
           Text(7, 0, '2006'),
           Text(8, 0, '2007'),
           Text(9, 0, '2008'),
           Text(10, 0, '2009'),
           Text(11, 0, '2010'),
           Text(12, 0, '2011'),
           Text(13, 0, '2012'),
           Text(14, 0, '2013'),
           Text(15, 0, '2014'),
           Text(16, 0, '2015'),
```

```
Text(17, 0, '2016'),
Text(18, 0, '2017'),
Text(19, 0, '2018'),
Text(20, 0, '2019'),
Text(21, 0, '2020'),
Text(22, 0, '2060')])
```



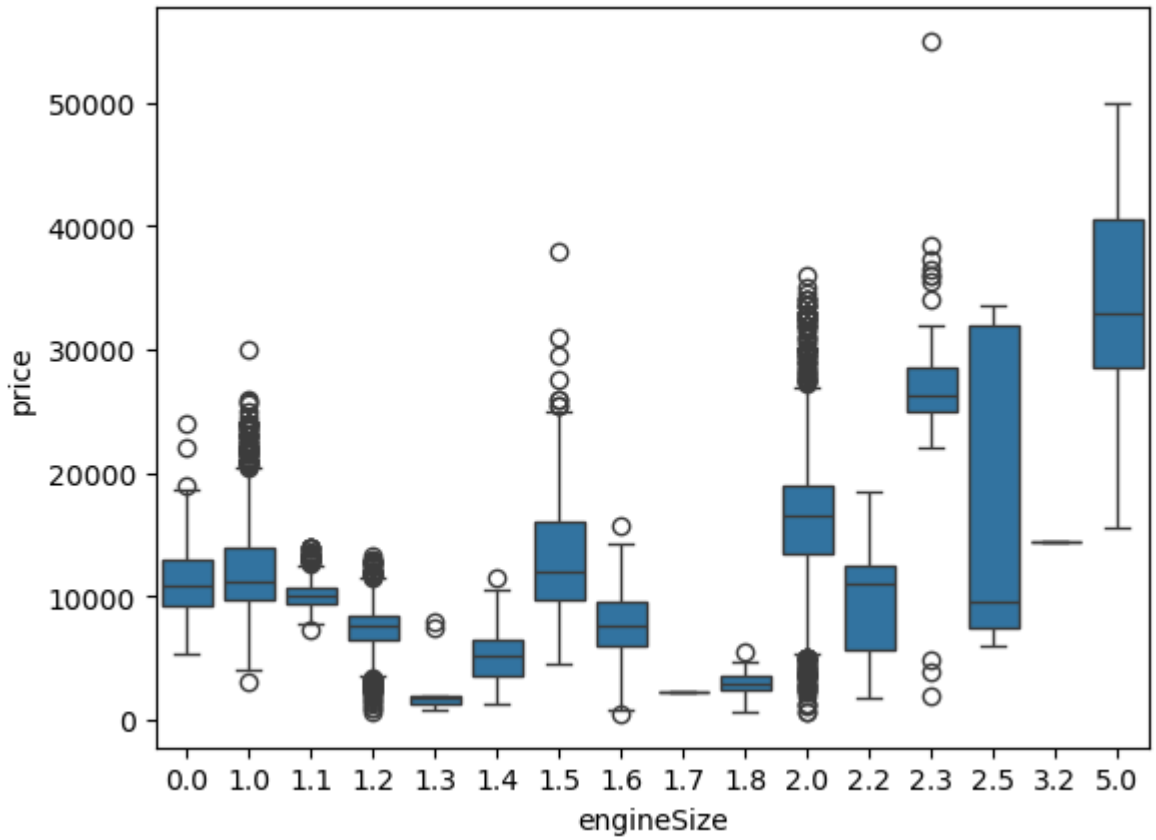
In [83]:
```
sns.scatterplot(data=df, x='mileage', y= 'price')
```

Out[83]:
```
<Axes: xlabel='mileage', ylabel='price'>
```



In [84]:
```
sns.boxplot(data=df, x='engineSize', y= 'price')
```

Out[84]:
```
<Axes: xlabel='engineSize', ylabel='price'>
```

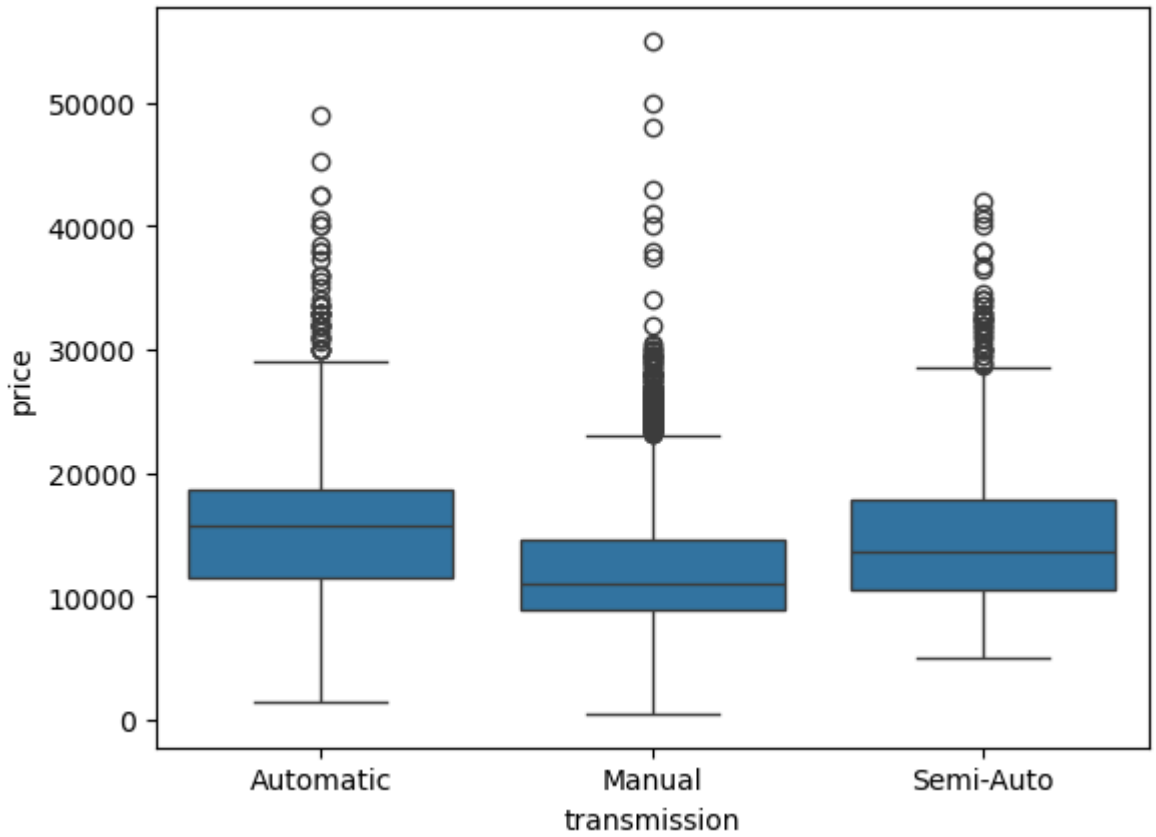


In [85]:
```
df.columns
```

Out[85]:
```
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType', 'tax',
       'mpg', 'engineSize'],
      dtype='object')
```

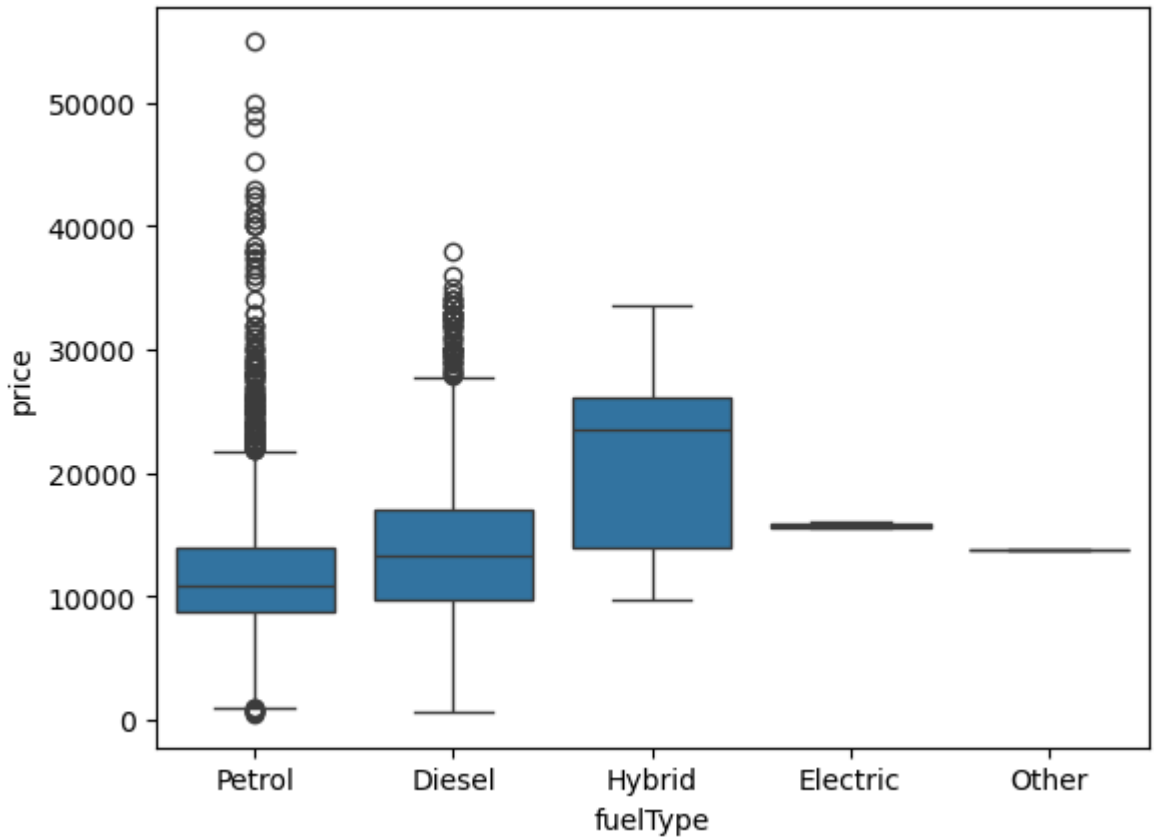
In [86]:
```
sns.boxplot(data=df, x='transmission', y='price')
```

Out[86]:
```
<Axes: xlabel='transmission', ylabel='price'>
```



In [87]:
```
sns.boxplot(data=df, x='fuelType', y='price')
```

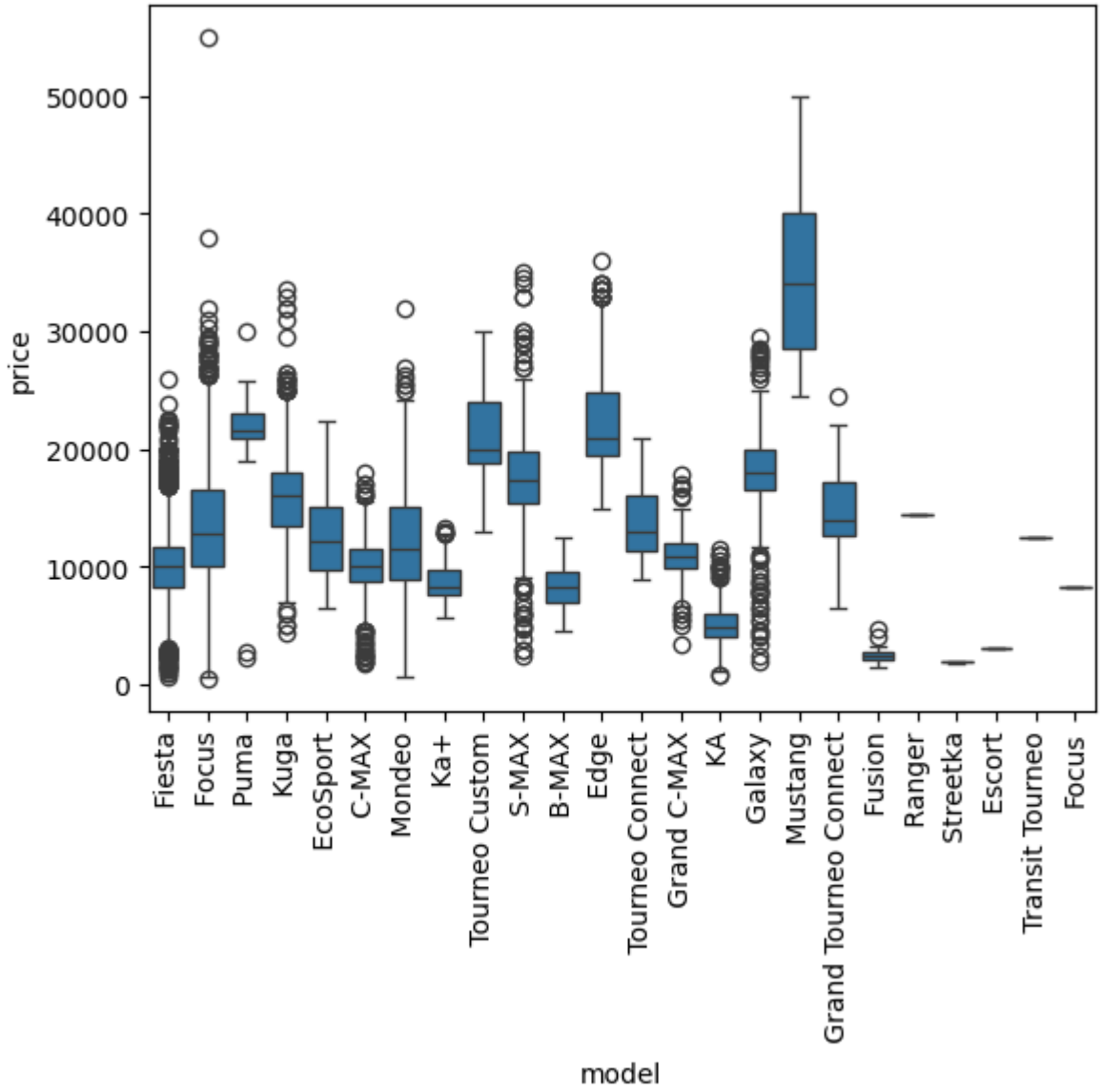
Out[87]:
```
<Axes: xlabel='fuelType', ylabel='price'>
```



In [88]:
```
sns.boxplot(data=df, x='model', y='price')
plt.xticks(rotation=90)
```

Out[88]:
```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23],
 [Text(0, 0, ' Fiesta'),
  Text(1, 0, ' Focus'),
  Text(2, 0, ' Puma'),
  Text(3, 0, ' Kuga'),
  Text(4, 0, ' EcoSport'),
  Text(5, 0, ' C-MAX'),
  Text(6, 0, ' Mondeo'),
  Text(7, 0, ' Ka+'),
  Text(8, 0, ' Tourneo Custom'),
  Text(9, 0, ' S-MAX'),
  Text(10, 0, ' B-MAX'),
  Text(11, 0, ' Edge'),
  Text(12, 0, ' Tourneo Connect'),
  Text(13, 0, ' Grand C-MAX'),
  Text(14, 0, ' KA'),
  Text(15, 0, ' Galaxy'),
```

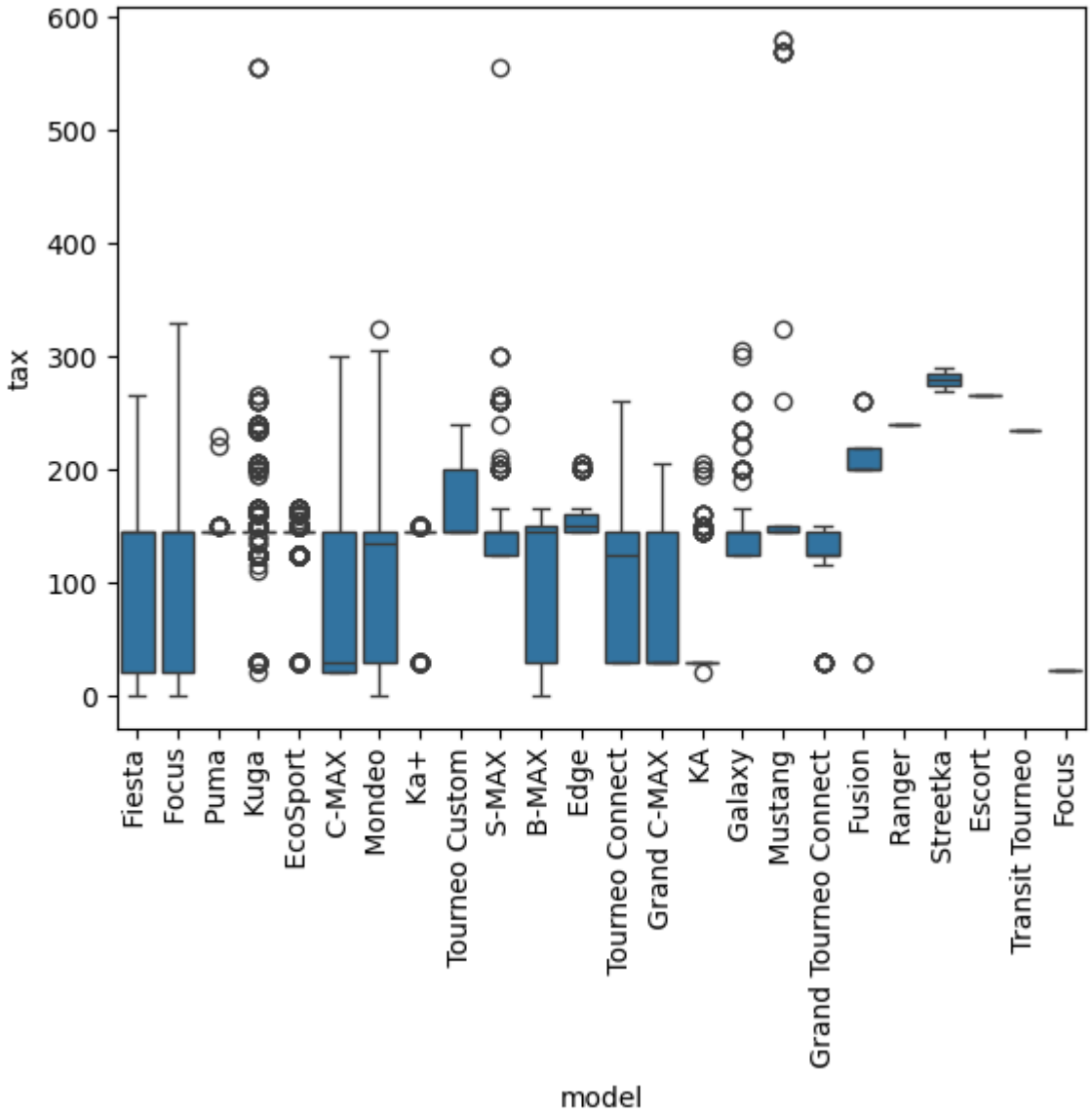
```
Text(16, 0, ' Mustang'),
Text(17, 0, ' Grand Tourneo Connect'),
Text(18, 0, ' Fusion'),
Text(19, 0, ' Ranger'),
Text(20, 0, ' Streetka'),
Text(21, 0, ' Escort'),
Text(22, 0, ' Transit Tourneo'),
Text(23, 0, 'Focus')])
```



In [89]:
```
sns.boxplot(data=df, x='model', y='tax')
plt.xticks(rotation=90)
```

Out[89]:
```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23],
 [Text(0, 0, ' Fiesta'),
  Text(1, 0, ' Focus'),
  Text(2, 0, ' Puma'),
  Text(3, 0, ' Kuga'),
  Text(4, 0, ' EcoSport'),
  Text(5, 0, ' C-MAX'),
  Text(6, 0, ' Mondeo'),
  Text(7, 0, ' Ka+'),
  Text(8, 0, ' Tourneo Custom'),
  Text(9, 0, ' S-MAX'),
  Text(10, 0, ' B-MAX'),
  Text(11, 0, ' Edge'),
  Text(12, 0, ' Tourneo Connect'),
  Text(13, 0, ' Grand C-MAX'),
  Text(14, 0, ' KA'),
  Text(15, 0, ' Galaxy'),
```

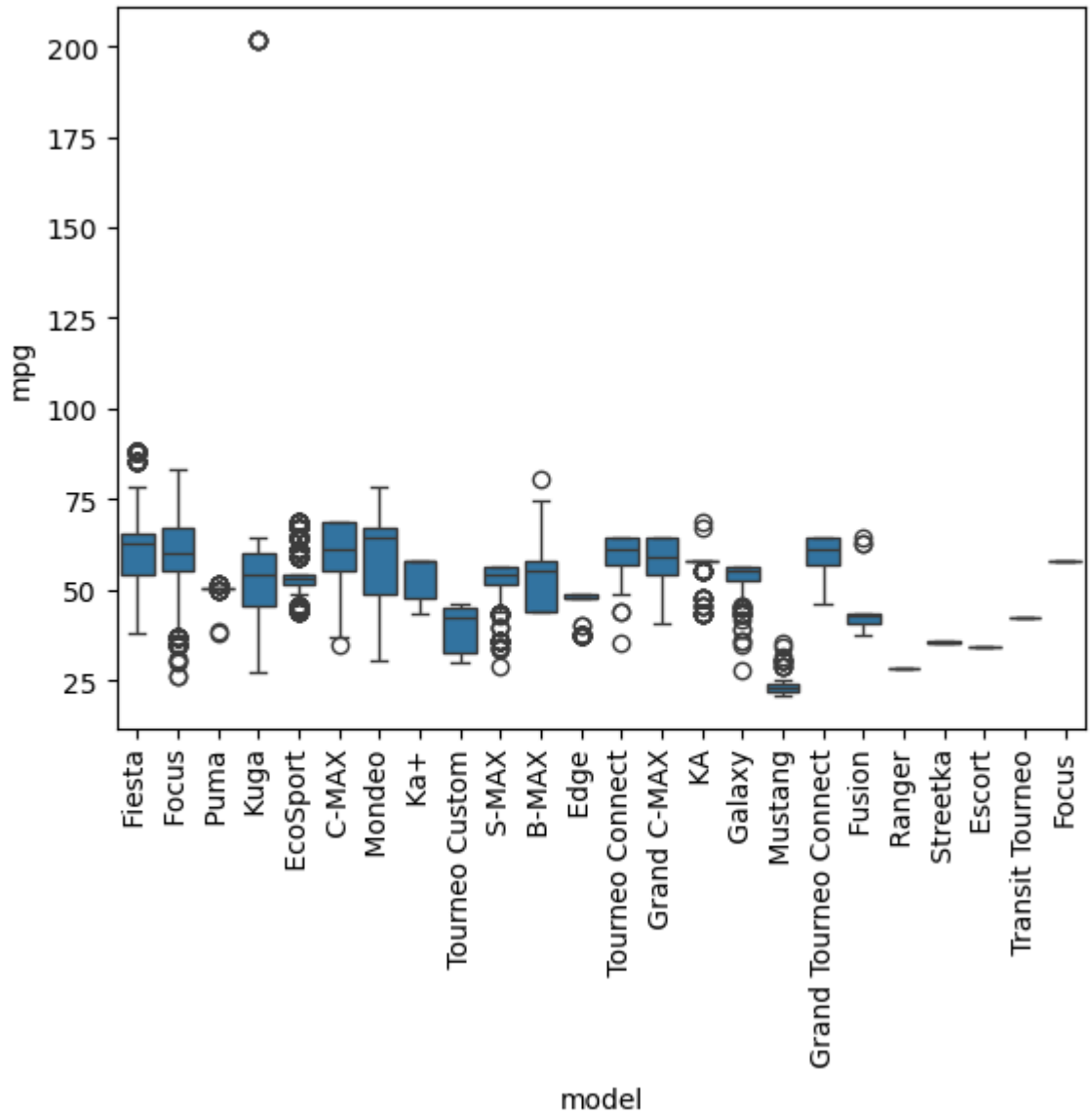
```
Text(16, 0, ' Mustang'),
Text(17, 0, ' Grand Tourneo Connect'),
Text(18, 0, ' Fusion'),
Text(19, 0, ' Ranger'),
Text(20, 0, ' Streetka'),
Text(21, 0, ' Escort'),
Text(22, 0, ' Transit Tourneo'),
Text(23, 0, 'Focus')])
```



In [90]:
```
sns.boxplot(data=df, x='model', y='mpg')
plt.xticks(rotation=90)
```

Out[90]:
```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23],
 [Text(0, 0, ' Fiesta'),
  Text(1, 0, ' Focus'),
  Text(2, 0, ' Puma'),
  Text(3, 0, ' Kuga'),
  Text(4, 0, ' EcoSport'),
  Text(5, 0, ' C-MAX'),
  Text(6, 0, ' Mondeo'),
  Text(7, 0, ' Ka+'),
  Text(8, 0, ' Tourneo Custom'),
  Text(9, 0, ' S-MAX'),
  Text(10, 0, ' B-MAX'),
  Text(11, 0, ' Edge'),
  Text(12, 0, ' Tourneo Connect'),
  Text(13, 0, ' Grand C-MAX'),
  Text(14, 0, ' KA'),
  Text(15, 0, ' Galaxy'),
```

```
Text(16, 0, ' Mustang'),
Text(17, 0, ' Grand Tourneo Connect'),
Text(18, 0, ' Fusion'),
Text(19, 0, ' Ranger'),
Text(20, 0, ' Streetka'),
Text(21, 0, ' Escort'),
Text(22, 0, ' Transit Tourneo'),
Text(23, 0, 'Focus')])
```



In [92]:
```
X=df.drop(columns=['price'], axis=1)
y=df['price']
```

In [93]:
```
X
```

Out[93]:

	model	year	transmission	mileage	fuelType	tax	mpg	engineSize
0	Fiesta	2017	Automatic	15944	Petrol	150	57.7	1.0
1	Focus	2018	Manual	9083	Petrol	150	57.7	1.0
2	Focus	2017	Manual	12456	Petrol	150	57.7	1.0
3	Fiesta	2019	Manual	10460	Petrol	145	40.3	1.5
4	Fiesta	2019	Automatic	1482	Petrol	145	48.7	1.0
...	...	...	...	...	...	...	...	...
17961	B-MAX	2017	Manual	16700	Petrol	150	47.1	1.4
17962	B-MAX	2014	Manual	40700	Petrol	30	57.7	1.0
17963	Focus	2015	Manual	7010	Diesel	20	67.3	1.6
17964	KA	2018	Manual	5007	Petrol	145	57.7	1.2
17965	Focus	2015	Manual	5007	Petrol	22	57.7	1.0

17966 rows × 8 columns

Converting string values into neumerical values

In [94]:
```
df.columns
```

Out[94]:
```
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType', 'tax',
       'mpg', 'engineSize'],
      dtype='object')
```

In [95]:
```
X_one_encode = pd.get_dummies(X, columns= ['model', 'transmission', 'fuelType'],drop_first=True)
```

In [96]:
```
X_one_encode
```

Out[96]:

	year	mileage	tax	mpg	engineSize	model_ C-MAX	model_ EcoSport	model_ Edge	model_ Escort	model_ Fiesta	...	model_ Tourneo Connect	model_ Tourneo Custom	model_ Transit Tourneo	model_Focus
0	2017	15944	150	57.7	1.0	False	False	False	False	True	...	False	False	False	False
1	2018	9083	150	57.7	1.0	False	False	False	False	False	...	False	False	False	False
2	2017	12456	150	57.7	1.0	False	False	False	False	False	...	False	False	False	False
3	2019	10460	145	40.3	1.5	False	False	False	False	True	...	False	False	False	False
4	2019	1482	145	48.7	1.0	False	False	False	False	True	...	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17961	2017	16700	150	47.1	1.4	False	False	False	False	False	...	False	False	False	False
17962	2014	40700	30	57.7	1.0	False	False	False	False	False	...	False	False	False	False
17963	2015	7010	20	67.3	1.6	False	False	False	False	False	...	False	False	False	False
17964	2018	5007	145	57.7	1.2	False	False	False	False	False	...	False	False	False	False
17965	2015	5007	22	57.7	1.0	False	False	False	False	False	...	False	False	False	True

17966 rows × 34 columns

In [97]:
```
X_one_encode= X_one_encode.astype(int)
```

In [98]:
```
from sklearn.preprocessing import LabelEncoder

columns = ['model', 'transmission', 'fuelType']

Xlabel = X.copy()    # make a safe copy
label_encoders = {}

for col in columns:   # use col instead of i for clarity
    le = LabelEncoder()
    # Convert to string in case of nulls
```

```
    Xlabel[col] = le.fit_transform(Xlabel[col].astype(str))
    label_encoders[col] = le
```

In [99]:
```
Xlabel
```

Out[99]:

	model	year	transmission	mileage	fuelType	tax	mpg	engineSize
0	5	2017	0	15944	4	150	57.7	1.0
1	6	2018	1	9083	4	150	57.7	1.0
2	6	2017	1	12456	4	150	57.7	1.0
3	5	2019	1	10460	4	145	40.3	1.5
4	5	2019	0	1482	4	145	48.7	1.0
...	...	...	...	...	...	...	...	...
17961	0	2017	1	16700	4	150	47.1	1.4
17962	0	2014	1	40700	4	30	57.7	1.0
17963	6	2015	1	7010	0	20	67.3	1.6
17964	11	2018	1	5007	4	145	57.7	1.2
17965	23	2015	1	5007	4	22	57.7	1.0

17966 rows × 8 columns

In [100…
```
from sklearn.preprocessing import StandardScaler
```

In [101…
```
from sklearn.preprocessing import StandardScaler

numerical_cols = ['year', 'mileage', 'tax', 'mpg', 'engineSize']

scaler = StandardScaler()  # fixed spelling + capitalization
X_one_encode[numerical_cols] = scaler.fit_transform(X_one_encode[numerical_cols])
```

In [102…
```
X_one_encode
```

Out[102...

	year	mileage	tax	mpg	engineSize	model_ C-MAX	model_ EcoSport	model_ Edge	model_ Escort	model_ Fiesta	...	model_ Tourneo Connect	model_ Tourneo Custom	model Transi Tourne
0	0.065128	-0.380998	0.591358	-0.042122	-0.447984	0	0	0	0	1	...	0	0	
1	0.552866	-0.733359	0.591358	-0.042122	-0.447984	0	0	0	0	0	...	0	0	
2	0.065128	-0.560132	0.591358	-0.042122	-0.447984	0	0	0	0	0	...	0	0	
3	1.040605	-0.662640	0.510727	-1.721198	-0.447984	0	0	0	0	1	...	0	0	
4	1.040605	-1.123724	0.510727	-0.931045	-0.447984	0	0	0	0	1	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17961	0.065128	-0.342172	0.591358	-1.029814	-0.447984	0	0	0	0	0	...	0	0	
17962	-1.398088	0.890398	-1.343791	-0.042122	-0.447984	0	0	0	0	0	...	0	0	
17963	-0.910349	-0.839822	-1.505053	0.945569	-0.447984	0	0	0	0	0	...	0	0	
17964	0.552866	-0.942690	0.510727	-0.042122	-0.447984	0	0	0	0	0	...	0	0	
17965	-0.910349	-0.942690	-1.472801	-0.042122	-0.447984	0	0	0	0	0	...	0	0	

17966 rows × 34 columns

In [104...
```
Xlabel.columns
```

Out[104...
```
Index(['model', 'year', 'transmission', 'mileage', 'fuelType', 'tax', 'mpg',
       'engineSize'],
      dtype='object')
```

In [106...
```
Xlabel[['model', 'year', 'transmission', 'mileage', 'fuelType', 'tax', 'mpg',
       'engineSize']] = scaler.fit_transform(Xlabel[['model', 'year', 'transmission', 'mileage', 'fuelType', 'tax', 'mpg',
       'engineSize']] )
```

In [107...
```
Xlabel
```

Out[107...

	model	year	transmission	mileage	fuelType	tax	mpg	engineSize
0	-0.460699	0.065128	-2.670032	-0.380998	0.688777	0.591358	-0.020442	-0.811386
1	-0.211477	0.552866	0.041351	-0.733359	0.688777	0.591358	-0.020442	-0.811386
2	-0.211477	0.065128	0.041351	-0.560132	0.688777	0.591358	-0.020442	-0.811386
3	-0.460699	1.040605	0.041351	-0.662640	0.688777	0.510727	-1.738890	0.345070
4	-0.460699	1.040605	-2.670032	-1.123724	0.688777	0.510727	-0.909294	-0.811386
...	...	...	...	...	...	...	...	...
17961	-1.706810	0.065128	0.041351	-0.342172	0.688777	0.591358	-1.067312	0.113779
17962	-1.706810	-1.398088	0.041351	0.890398	0.688777	-1.343791	-0.020442	-0.811386
17963	-0.211477	-0.910349	0.041351	-0.839822	-1.454098	-1.505053	0.927668	0.576362
17964	1.034634	0.552866	0.041351	-0.942690	0.688777	0.510727	-0.020442	-0.348804
17965	4.025302	-0.910349	0.041351	-0.942690	0.688777	-1.472801	-0.020442	-0.811386

17966 rows × 8 columns

Creating the model

In [111...

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Let's perform train_test_split and replacing the X with X_one_encoding

In [112...

```
X_train, X_test, y_train, y_test = train_test_split(X_one_encode, y, test_size=0.33, random_state=42)
```

In [113…
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[113…
```
▾ LinearRegression
LinearRegression()
```

Let's prform prediction

In [114…
```
y_pred= model.predict(X_test)
```

In [115…
```
y_pred          #Predicted Data
```

Out[115…
```
array([ 6888.75487917,  9328.62297666,  9420.53085767, ...,
       19099.96373064,  4948.44438264, 10424.59544226])
```

In [116…
```
y_test        #Real Data
```

Out[116…
```
17610     6995
7076      8999
1713      7998
1611      5491
16830     3790
         ...
6015     19000
10301    10940
15006    21999
5396      6995
6087     10299
Name: price, Length: 5929, dtype: int64
```

Comparing real and predicted data

In [118…
```
r2=r2_score(y_test,y_pred)
r2
```

Out[118…
```
0.8396626991294073
```

Model is 83% accurate

In [120…
```
#Let's see adjusted r2


n= X_test.shape[0]
p= X_test.shape[1]
adjusted_r2 = 1-((1-r2)*(n-1)/(n-p-1))
print("Adjusted R2 Score: ", adjusted_r2)          #it shoudl be similar to above one.
```

```
Adjusted R2 Score:  0.8387377808685318
```

let's do it for label encoded data

In [121…
```
X_train, X_test, y_train, y_test = train_test_split(Xlabel, y, test_size=0.33, random_state=42)
```

In [122…
```
model2 = LinearRegression()
model2.fit(X_train, y_train)
```

Out[122…
```
▾ LinearRegression
LinearRegression()
```

In [123…
```
y_pred= model2.predict(X_test)
```

In [124…
```
y_pred
```

Out[124…
```
array([ 6157.52473246,  9286.53653694,  9519.82817502, ...,
       19580.21385231,  7384.33202962,  9960.74790235])
```

In [125…
```
y_test
```

Out[125…
```
17610     6995
7076      8999
1713      7998
1611      5491
16830     3790
          ...
6015     19000
10301    10940
15006    21999
5396      6995
6087     10299
Name: price, Length: 5929, dtype: int64
```

Huge differance found after using the label encoding for prediction!

In [126…
```
r2=r2_score(y_test,y_pred)
r2
```

Out[126…
```
0.731021555739114
```

CONCLUSION

This project applied machine learning techniques to predict Ford car prices using features such as model, year, transmission, mileage, fuel type, tax, mpg, and engine size. By combining data preprocessing (label encoding, one-hot encoding, scaling) with exploratory data analysis and modeling, the project identified key price drivers and built a system capable of delivering accurate predictions.

From a business perspective, the model offers several high-value applications:

1. Dealerships can use it to set competitive and fair prices, improving sales turnover while maintaining profitability.

2. Online car marketplaces can integrate such a system to provide instant, data-driven price estimates, enhancing customer trust and engagement.

3. Buyers and sellers gain a reliable benchmark to negotiate fair deals, reducing pricing asymmetry in the used car market.

4. Finance and insurance companies can use predicted car values for better loan approvals and premium calculations.

5. Key insights show that year of manufacture and mileage are the strongest predictors of resale value, while engine size, transmission, fuel type, and model introduce additional differentiation across price ranges. The ability of the model to learn from historical data demonstrates how predictive analytics can be leveraged to address real-world pricing challenges in the automotive sector.

6. Overall, this project highlights my ability to transform raw data into actionable insights, apply end-to-end machine learning workflows, and deliver solutions with clear business impact. With further enhancements, the system can evolve into a robust, production-ready car price recommendation tool, showcasing how data science creates measurable value across industries