In [1]:

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

In [3]:

```
car = pd.read_csv("car data.csv")
car
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	M
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	M
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	M
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	M
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	Diesel	Dealer	M
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	M
298	city	2009	3.35	11.00	87934	Petrol	Dealer	M
299	city	2017	11.50	12.50	9000	Diesel	Dealer	M
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 9 columns

In [4]:

```
print(car['Selling_type'].unique())
print(car['Fuel_Type'].unique())
print(car['Transmission'].unique())
print(car['Owner'].unique())
```

```
['Dealer' 'Individual']
['Petrol' 'Diesel' 'CNG']
['Manual' 'Automatic']
[0 1 3]
```

In [5]:

```
car.describe()
```

Out[5]:

	Year	Selling_Price	Present_Price	Driven_kms	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.642584	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [6]:

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Selling_type   301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

In [7]:

```
car.duplicated().sum()
```

Out[7]:

```
2
```

In [8]:

```
car.drop_duplicates(inplace= True)
```

In [9]:

```
car.isnull().sum()
```

Out[9]:

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Driven_kms       0
Fuel_Type        0
Selling_type     0
Transmission     0
Owner            0
dtype: int64
```

In [10]:

```
car["Year"]= pd.to_datetime(car["Year"], format = '%Y').dt.year
```

In [11]:

```
car["Owner"] = car["Owner"].astype("int32")
car["Driven_kms"] = car["Driven_kms"].astype("int32")
```

In [12]:

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 299 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       299 non-null    object 
 1   Year           299 non-null    int32  
 2   Selling_Price  299 non-null    float64
 3   Present_Price  299 non-null    float64
 4   Driven_kms     299 non-null    int32  
 5   Fuel_Type      299 non-null    object 
 6   Selling_type   299 non-null    object 
 7   Transmission   299 non-null    object 
 8   Owner          299 non-null    int32  
dtypes: float64(2), int32(3), object(4)
memory usage: 19.9+ KB
```

In [13]:

```
car["Year"].unique()
```

Out[13]:

```
array([2014, 2013, 2017, 2011, 2018, 2015, 2016, 2009, 2010, 2012, 2003,
       2008, 2006, 2005, 2004, 2007])
```

In [14]:

```
car["Year"].nunique()
```

Out[14]:

```
16
```

In [15]:

```
car = car.drop(columns= "Car_Name")
```

In [16]:

```
car["current year"]= 2023
```

In [17]:

```
car['Age of car']= car["current year"]-car["Year"]
```

In [18]:
```
car
```

Out[18]:

	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Own
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
...	...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

299 rows × 10 columns

In [19]:
```
car = car.drop(columns= ["current year", "Year"])
car
```

Out[19]:

	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner	Ag (ca
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	1
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	1
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	
...	...	...	...	...	...	...	...	
296	9.50	11.60	33988	Diesel	Dealer	Manual	0	
297	4.00	5.90	60000	Petrol	Dealer	Manual	0	
298	3.35	11.00	87934	Petrol	Dealer	Manual	0	1
299	11.50	12.50	9000	Diesel	Dealer	Manual	0	
300	5.30	5.90	5464	Petrol	Dealer	Manual	0	

299 rows × 8 columns

In [20]:
```
car = pd.get_dummies(data=car,  drop_first= True)
```

In [21]:
```
car
```

Out[21]:

	Selling_Price	Present_Price	Driven_kms	Owner	Age of car	Fuel_Type_Diesel	Fuel_Type_Petrol
0	3.35	5.59	27000	0	9	False	True
1	4.75	9.54	43000	0	10	True	False
2	7.25	9.85	6900	0	6	False	True
3	2.85	4.15	5200	0	12	False	True
4	4.60	6.87	42450	0	9	True	False
...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	7	True	False
297	4.00	5.90	60000	0	8	False	True
298	3.35	11.00	87934	0	14	False	True
299	11.50	12.50	9000	0	6	True	False
300	5.30	5.90	5464	0	7	False	True

299 rows × 9 columns

In [22]:
```
g= ['Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Selling_type_Individual', 'Transm
car[g]= car[g].astype('int')
```

In [ ]:

In [23]:
```
car.head(3)
```

Out[23]:

	Selling_Price	Present_Price	Driven_kms	Owner	Age of car	Fuel_Type_Diesel	Fuel_Type_Petrol	Se
0	3.35	5.59	27000	0	9	0	1	
1	4.75	9.54	43000	0	10	1	0	
2	7.25	9.85	6900	0	6	0	1	

In [24]:
```
car.corr()
```

Out[24]:

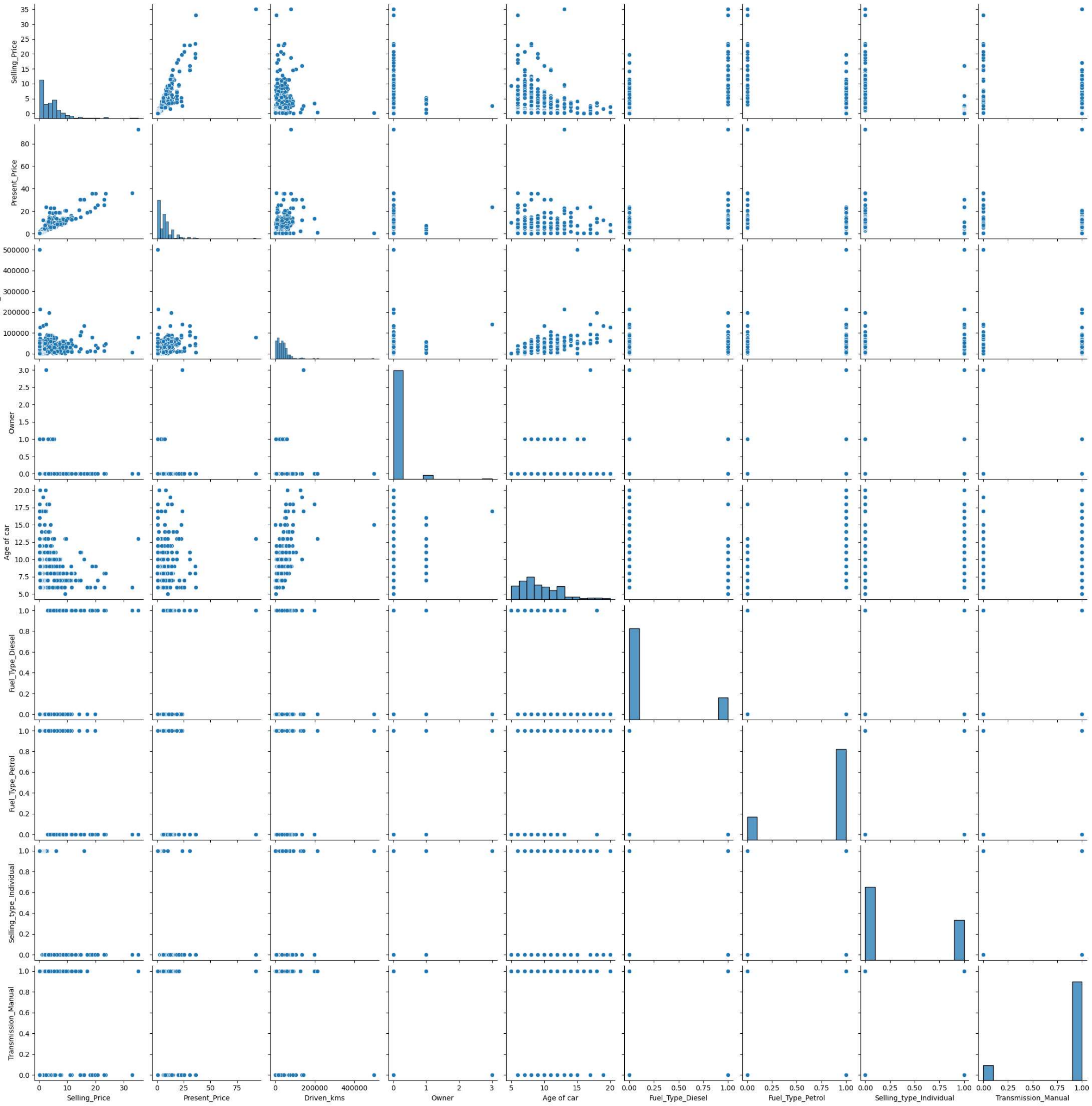
	Selling_Price	Present_Price	Driven_kms	Owner	Age of car	Fuel_Type_
Selling_Price	1.000000	0.876305	0.028566	-0.087880	-0.234369	0.
Present_Price	0.876305	1.000000	0.205224	0.009948	0.053167	0.
Driven_kms	0.028566	0.205224	1.000000	0.089367	0.525714	0.
Owner	-0.087880	0.009948	0.089367	1.000000	0.181639	-0.
Age of car	-0.234369	0.053167	0.525714	0.181639	1.000000	-0.
Fuel_Type_Diesel	0.543541	0.464934	0.173295	-0.051836	-0.056469	1.
Fuel_Type_Petrol	-0.531636	-0.456829	-0.173595	0.054102	0.052197	-0.
Selling_type_Individual	-0.553851	-0.511779	-0.101030	0.123646	0.036820	-0.
Transmission_Manual	-0.348869	-0.334326	-0.163881	-0.052166	-0.003434	-0.

In [25]:

```
import warnings
warnings.filterwarnings("ignore")
sns.pairplot(car)
```

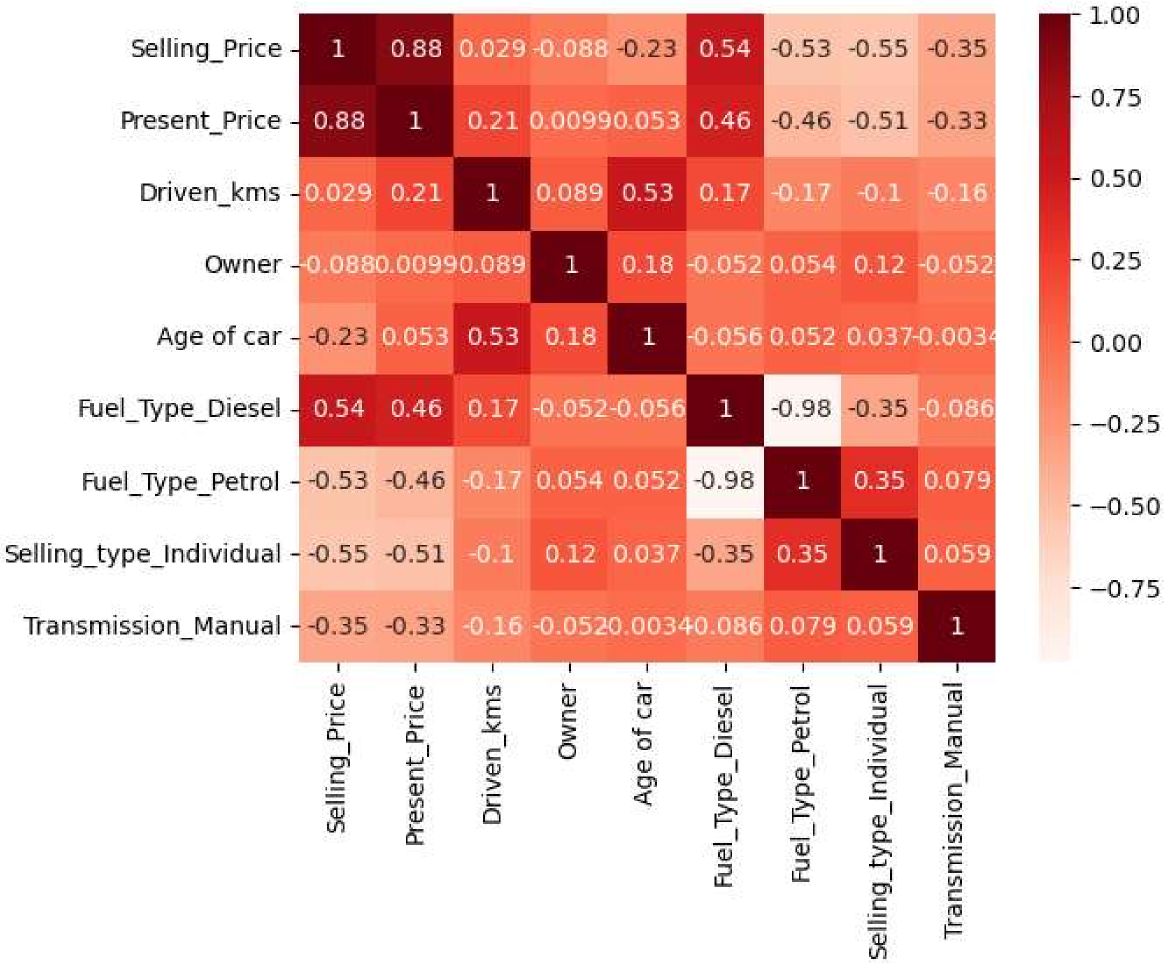
Out[25]:

```
<seaborn.axisgrid.PairGrid at 0x1db569738d0>
```



In [26]:
```
sns.heatmap(car.corr(), annot= True, cmap= 'Reds')
```

Out[26]:
```
<Axes: >
```



In [27]:
```
y = car['Selling_Price'] #DEPENDENT VARIABLE AND TARGET
x = car.drop(columns= ['Selling_Price']) # INPUT AND INDEPENDENT DATA
```

In [28]:
```
y
```

Out[28]:
```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
        ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 299, dtype: float64
```

In [29]:
```
x
```

Out[29]:

	Present_Price	Driven_kms	Owner	Age of car	Fuel_Type_Diesel	Fuel_Type_Petrol	Selling_type_Ir
0	5.59	27000	0	9	0	1	
1	9.54	43000	0	10	1	0	
2	9.85	6900	0	6	0	1	
3	4.15	5200	0	12	0	1	
4	6.87	42450	0	9	1	0	
...	...	...	...	...	...	...	
296	11.60	33988	0	7	1	0	
297	5.90	60000	0	8	0	1	
298	11.00	87934	0	14	0	1	
299	12.50	9000	0	6	1	0	
300	5.90	5464	0	7	0	1	

299 rows × 8 columns

In [30]:
```
x['Owner'].unique()
```

Out[30]:
```
array([0, 1, 3])
```

In [31]:
```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(x,y)
```

Out[31]:
```
ExtraTreesRegressor()
```

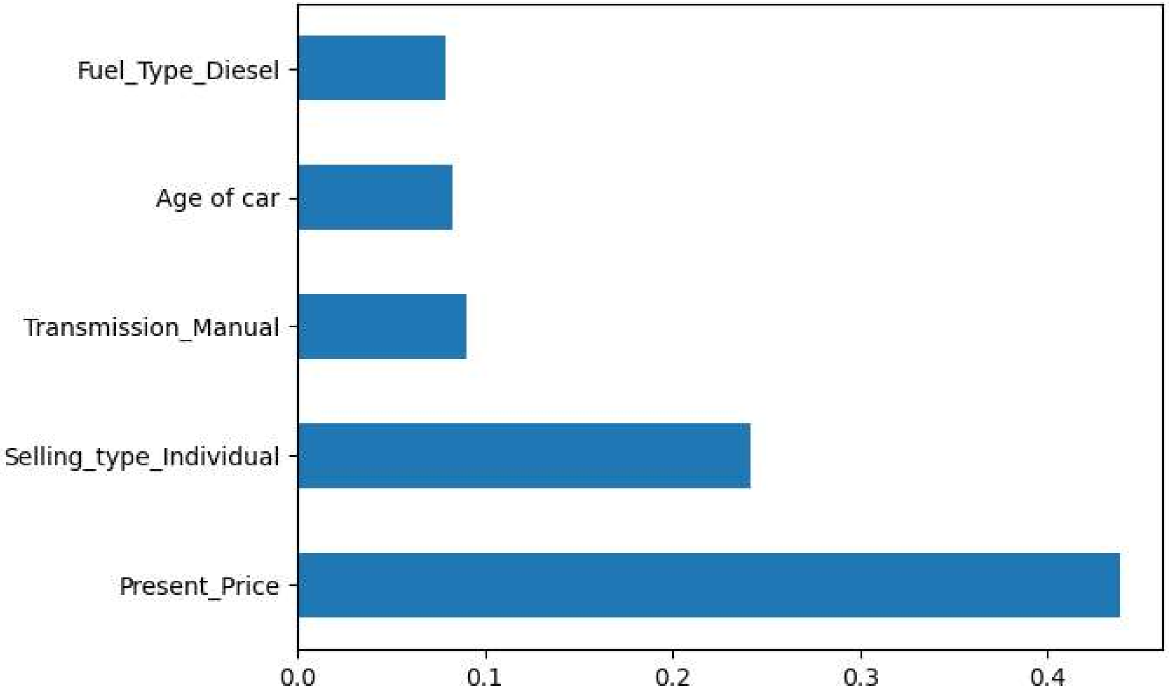
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [32]:
```
print(model.feature_importances_)
```

```
[0.4392558  0.03995501 0.00145351 0.08275982 0.07910522 0.02513653
 0.24188902 0.09044509]
```

In [33]:

```
feat_importances = pd.Series(model.feature_importances_, index=x.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



In [34]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_
```

In [35]:

```
x_train.shape
```

Out[35]:

```
(239, 8)
```

In [36]:

```
x_test.shape
```

Out[36]:

```
(60, 8)
```

In [38]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [39]:

```
parameters = {
    'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000],
    'criterion': ['squared_error', 'absolute_error', 'poisson', 'friedman_mse'
    'max_depth': [10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10, 20, 50],
    'min_samples_leaf': [1, 2, 5, 10],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

In [40]:

```
parameters
```

Out[40]:

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000],
 'criterion': ['squared_error', 'absolute_error', 'poisson', 'friedman_mse'],
 'max_depth': [10, 20, 30, 40, 50],
 'min_samples_split': [2, 5, 10, 20, 50],
 'min_samples_leaf': [1, 2, 5, 10],
 'max_features': ['auto', 'sqrt', 'log2']}
```

In [ ]:

```

```