In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
%matplotlib inline
```

In [2]:
```python
df=pd.read_excel("Data_Train.xlsx")
df.head()
```

Out[2]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	1.75
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	12.50
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	4.50
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	6.00
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	17.74

In [3]:
```python
df.shape
```

Out[3]:
```
(6019, 12)
```

In [4]:
```python
df.dtypes
```

Out[4]:
```
Name                  object
Location              object
Year                   int64
Kilometers_Driven      int64
Fuel_Type             object
Transmission          object
Owner_Type            object
Mileage               object
Engine                object
Power                 object
Seats                float64
Price                float64
dtype: object
```

In [5]:
```python
df.isnull().sum()
```

Out[5]:
```
Name                  0
Location              0
Year                  0
Kilometers_Driven     0
Fuel_Type             0
Transmission          0
Owner_Type            0
Mileage               2
Engine               36
Power                36
Seats                42
Price                 0
dtype: int64
```

In [6]:
```python
df.nunique()
```

Out[6]:
```
Name                 1876
Location               11
Year                   22
Kilometers_Driven    3093
Fuel_Type               5
Transmission            2
Owner_Type              4
Mileage               442
Engine                146
Power                 372
Seats                   9
Price                1373
dtype: int64
```

In [7]:
```python
df['Engine'].value_counts()
```

Out[7]:
```
1197 CC    606
1248 CC    512
1498 CC    304
998 CC     259
2179 CC    240
          ...
2999 CC      1
2147 CC      1
2495 CC      1
3200 CC      1
1797 CC      1
Name: Engine, Length: 146, dtype: int64
```

In [8]:
```python
df['Engine'].fillna('1197 CC',inplace=True)
```

In [9]:
```python
df['Mileage'].value_counts()
```

Out[9]:
```
18.9 kmpl     172
17.0 kmpl     172
18.6 kmpl     119
20.36 kmpl     88
21.1 kmpl      86
             ...
27.28 kmpl      1
14.57 kmpl      1
22.8 km/kg      1
8.0 kmpl        1
17.24 kmpl      1
Name: Mileage, Length: 442, dtype: int64
```

In [10]:
```python
df['Mileage'].fillna('18.9 kmpl',inplace=True)
```

In [11]:
```python
df['Power'].value_counts()
```

Out[11]:
```
74 bhp        235
98.6 bhp      131
73.9 bhp      125
140 bhp       123
78.9 bhp      111
             ...
76.9 bhp        1
201 bhp         1
199.3 bhp       1
95 bhp          1
181.04 bhp      1
Name: Power, Length: 372, dtype: int64
```

In [12]:
```python
df['Power'].fillna('74 bhp',inplace=True)
```

In [13]:
```python
df['Mileage']=df['Mileage'].replace("0.0 kmpl", "17.0 kmpl")
```

In [14]:
```python
df['Power'] = df['Power'].replace("null bhp", "74 bhp")
```

In [15]:
```python
df.dropna(subset = ["Seats"], inplace=True)
```

In [16]:
```python
df.isnull().sum()
```

Out[16]:
```
Name                 0
Location             0
Year                 0
Kilometers_Driven    0
Fuel_Type            0
Transmission         0
Owner_Type           0
Mileage              0
Engine               0
Power                0
Seats                0
Price                0
dtype: int64
```

In [17]:
```python
df.dtypes
```

Out[17]:
```
Name                  object
Location              object
Year                   int64
Kilometers_Driven      int64
Fuel_Type             object
Transmission          object
Owner_Type            object
Mileage               object
Engine                object
Power                 object
Seats                float64
Price                float64
dtype: object
```

In [18]:
```python
df['Power'] = pd.to_numeric(df['Power'].str.lower().str.split().str.get(0), errors='coerce')
df['Engine'] = pd.to_numeric(df['Engine'].str.lower().str.split().str.get(0), errors='coerce')
```

In [19]:
```python
df.dtypes
```

Out[19]:
```
Name                  object
Location              object
Year                   int64
Kilometers_Driven      int64
Fuel_Type             object
Transmission          object
Owner_Type            object
Mileage               object
Engine                 int64
Power                float64
Seats                float64
Price                float64
dtype: object
```

In [20]:
```python
Common_Mileage_In_kmpl = []

for i in df.Mileage:
    if str(i).endswith('km/kg'):
        i = i[:-6]
        i = float(i)*1.40
        Common_Mileage_In_kmpl.append(float(i))

    elif str(i).endswith('kmpl'):
        i = i[:-6]
        Common_Mileage_In_kmpl.append(float(i))
```

In [21]:
```python
df['Mileage'] = Common_Mileage_In_kmpl
```

In [22]:
```python
Company = [i.split()[0] for i in df['Name']]
df.insert(0,'Company',Company)
```

In [23]:
```python
df.head()
```

Out[23]:

	Company	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price
0	Maruti	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	37.24	998	58.16	5.0	1.75
1	Hyundai	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.60	1582	126.20	5.0	12.50
2	Honda	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.00	1199	88.70	5.0	4.50
3	Maruti	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.70	1248	88.76	7.0	6.00
4	Audi	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.00	1968	140.80	5.0	17.74

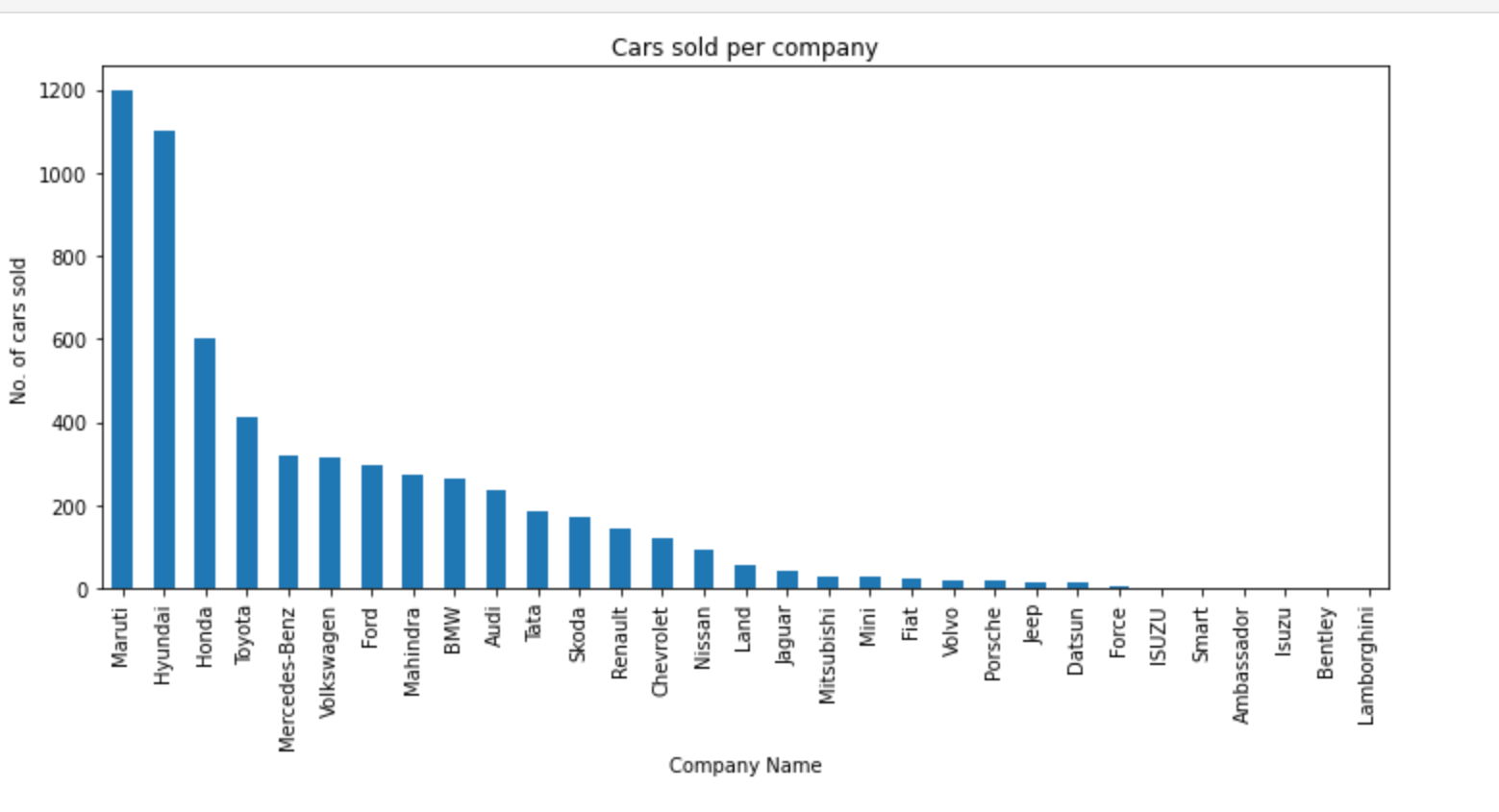
In [24]:
```python
df["Price"] = df["Price"]*100000
```

In [25]:
```python
df = df.astype({"Price":'int64'})
```

In [26]:
```python
df.dtypes
```

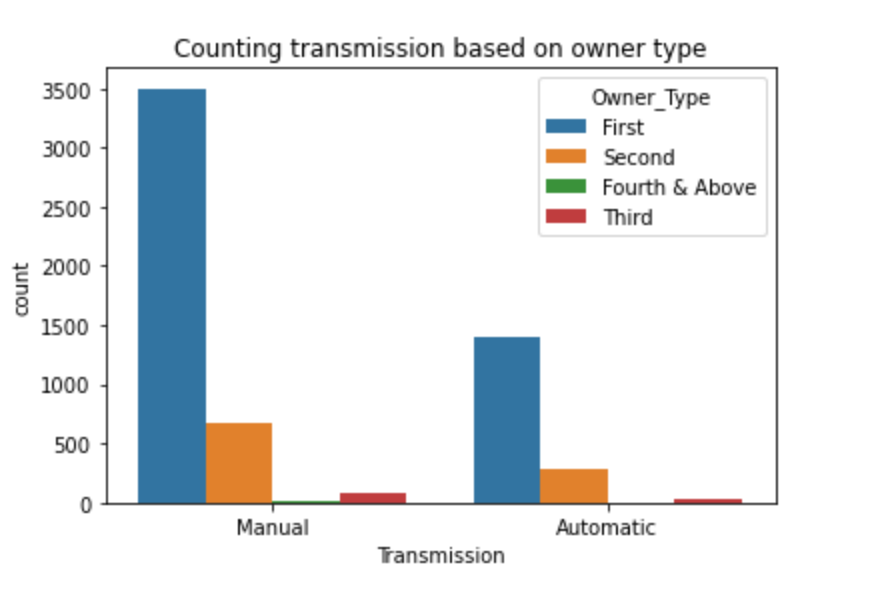
Out[26]:
```
Company               object
Name                  object
Location              object
Year                   int64
Kilometers_Driven      int64
Fuel_Type             object
Transmission          object
Owner_Type            object
Mileage              float64
Engine                 int64
Power                float64
Seats                float64
Price                  int64
dtype: object
```

In [27]:
```python
company_count=df['Company'].value_counts()
plt.figure(figsize=(12,5))
plt.xlabel('Company Name')
plt.ylabel('No. of cars sold')
plt.title('Cars sold per company')
company_count.plot(kind='bar')
plt.show()
```



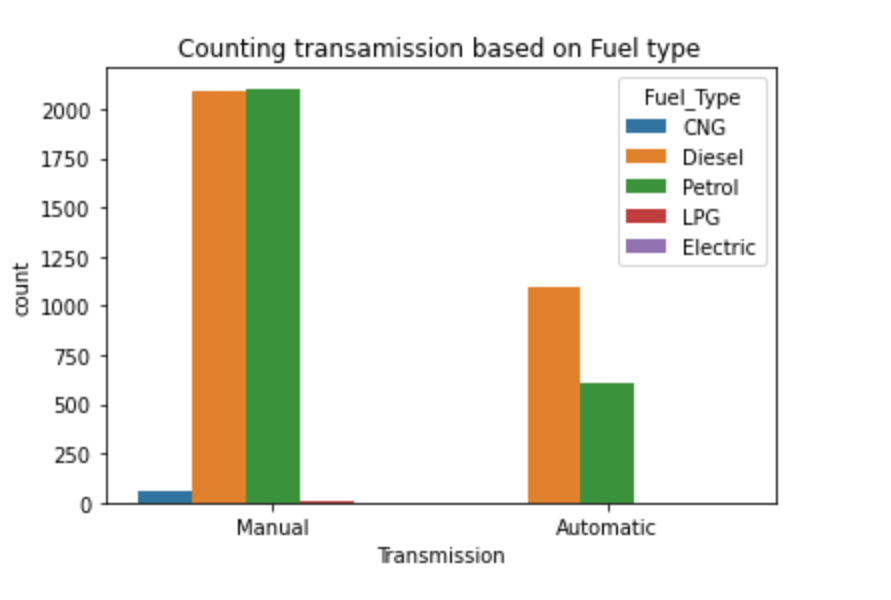
In [28]:
```python
sns.countplot(x = 'Transmission', hue ='Owner_Type', data = df )
plt.title('Counting transmission based on owner type')
```

Out[28]:
```
Text(0.5, 1.0, 'Counting transmission based on owner type')
```



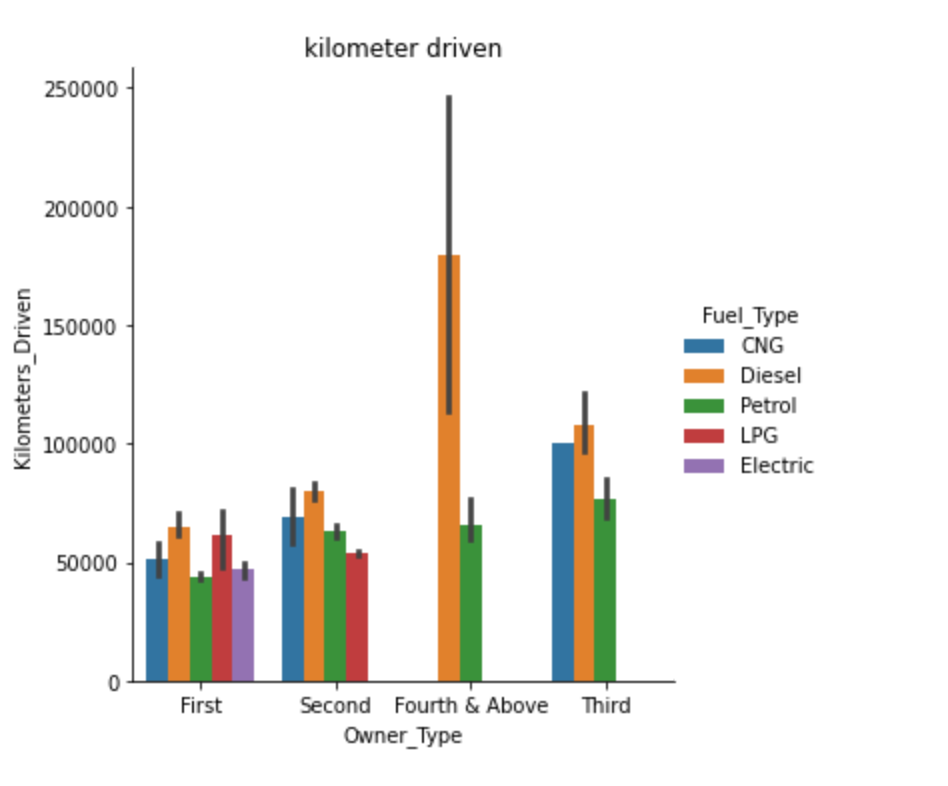
In [29]:
```python
sns.countplot(x = 'Transmission', hue ='Fuel_Type', data = df )
plt.title('Counting transamission based on Fuel type')
```

Out[29]:
```
Text(0.5, 1.0, 'Counting transamission based on Fuel type')
```



In [30]:
```python
sns.catplot(x = 'Owner_Type', y = 'Kilometers_Driven',hue = 'Fuel_Type',kind = 'bar', data = df)
plt.title('kilometer driven')
```

Out[30]:
```
Text(0.5, 1.0, 'kilometer driven')
```



In [31]:
```python
df['Number_Of_Year'] = 2022 - df['Year']
df.drop(['Year'],axis=1)
```

Out[31]:

	Company	Name	Location	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price	Number_Of_Year
0	Maruti	Maruti Wagon R LXI CNG	Mumbai	72000	CNG	Manual	First	37.24	998	58.16	5.0	175000	12
1	Hyundai	Hyundai Creta 1.6 CRDi SX Option	Pune	41000	Diesel	Manual	First	19.60	1582	126.20	5.0	1250000	7
2	Honda	Honda Jazz V	Chennai	46000	Petrol	Manual	First	18.00	1199	88.70	5.0	450000	11
3	Maruti	Maruti Ertiga VDI	Chennai	87000	Diesel	Manual	First	20.70	1248	88.76	7.0	600000	10
4	Audi	Audi A4 New 2.0 TDI Multitronic	Coimbatore	40670	Diesel	Automatic	Second	15.00	1968	140.80	5.0	1773999	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6014	Maruti	Maruti Swift VDI	Delhi	27365	Diesel	Manual	First	28.00	1248	74.00	5.0	475000	8
6015	Hyundai	Hyundai Xcent 1.1 CRDi S	Jaipur	100000	Diesel	Manual	First	24.00	1120	71.00	5.0	400000	7
6016	Mahindra	Mahindra Xylo D4 BSIV	Jaipur	55000	Diesel	Manual	Second	14.00	2498	112.00	8.0	290000	10
6017	Maruti	Maruti Wagon R VXI	Kolkata	46000	Petrol	Manual	First	18.00	998	67.10	5.0	265000	9
6018	Chevrolet	Chevrolet Beat Diesel	Hyderabad	47000	Diesel	Manual	First	25.40	936	57.60	5.0	250000	11

5977 rows × 13 columns

In [32]:
```python
df[df['Kilometers_Driven'] > 1000000]
```

Out[32]:

	Company	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price	Number_Of_Year
2328	BMW	BMW X5 xDrive 30d M Sport	Chennai	2017	6500000	Diesel	Automatic	First	15.9	2993	258.0	5.0	6500000	5

In [33]:
```python
kilometers_driven_mean = df['Kilometers_Driven'].mean()
df['Kilometers_Driven'] = [i if i < 1000000 else float(int(kilometers_driven_mean)) for i in df.Kilometers_Driven]
```

In [34]:
```python
df['Kilometers_Driven'].max()
```

Out[34]:
```
775000.0
```

In [35]:
```python
df = df.astype({"Kilometers_Driven":'int'})
```

In [36]:
```python
df['Kilometers_Driven'].max()
```

Out[36]:
```
775000
```

In [37]:
```python
df.shape
```

Out[37]:
```
(5977, 14)
```

In [38]:
```python
data = df.copy()
```

In [39]:
```python
#one hot Encoding
company_dummies = pd.get_dummies(df['Company'])
name_dummies = pd.get_dummies(df['Name'])
location_dummies = pd.get_dummies(df['Location'])
fuel_type_dummies = pd.get_dummies(df['Fuel_Type'])
transmission_dummies = pd.get_dummies(df['Transmission'])
owner_type_dummies = pd.get_dummies(df['Owner_Type'])
```

In [40]:
```python
features = df[['Number_Of_Year', 'Kilometers_Driven', 'Mileage', 'Engine', 'Power', 'Seats', 'Price']]
```

In [41]:
```python
df = pd.concat([features, company_dummies, name_dummies, location_dummies, fuel_type_dummies,
                transmission_dummies, owner_type_dummies], axis=1)
```

In [42]:
```python
df.rename(columns={'Fourth & Above': 'More'},inplace=True, errors='raise')
```

In [43]:
```python
df.dtypes
```

Out[43]:
```
Number_Of_Year         int64
Kilometers_Driven      int32
Mileage              float64
Engine                 int64
Power                float64
                      ...
Manual                 uint8
First                  uint8
More                   uint8
Second                 uint8
Third                  uint8
Length: 1917, dtype: object
```

In [44]:
```python
X = df.drop(['Price'], axis=1) #independent variables
Y = df['Price'] #target variable
```

In [45]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=0)
```

In [46]:
```python
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[46]:
```
((4183, 1916), (4183,), (1794, 1916), (1794,))
```

In [47]:
```python
dr=DecisionTreeRegressor()
dr.fit(X_train, y_train)
dr.score(X_test, y_test)
```

Out[47]:
```
0.7802114391377138
```

In [48]:
```python
rf=RandomForestRegressor()
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

Out[48]:
```
0.853471744969352
```

In [66]:
```python
gredR = GradientBoostingRegressor()
gredR.fit(X_train.values, y_train.values)
gredR.score(X_test.values,y_test.values)
```

Out[66]:
```
0.8616576053529588
```

In [50]:
```python
#hyper parameter tunning
from sklearn.model_selection import GridSearchCV
LR = {'learning_rate': [0.15, .1, .12, 1.0],'n_estimators':[50, 100, 150, 200, 250]}

tuning = GridSearchCV(estimator=GradientBoostingRegressor(),
                      param_grid=LR, scoring ='r2')
tuning.fit(X_train, y_train)
```

Out[50]:
```
GridSearchCV(estimator=GradientBoostingRegressor(),
             param_grid={'learning_rate': [0.15, 0.1, 0.12, 1.0],
                         'n_estimators': [50, 100, 150, 200, 250]},
             scoring='r2')
```

In [52]:
```python
tuning.best_params_, tuning.best_score_
```

Out[52]:
```
({'learning_rate': 0.12, 'n_estimators': 250}, 0.9065258858247107)
```

In [60]:
```python
gredR = GradientBoostingRegressor( n_estimators = 250, learning_rate=0.12)
gredR.fit(X_train.values, y_train.values)
gredR.score(X_test.values,y_test.values)
```

Out[60]:
```
0.8809790781465665
```

In [61]:
```python
dr.fit(X_train, y_train)
rmse = np.sqrt(mean_squared_error(y_test, predicted))
predicted = dr.predict(X_test)
print('RMSE decision tree:')
print(rmse)
```

```
RMSE decision tree:
443199.34004453884
```

In [62]:
```python
rf.fit(X_train, y_train)
rmse = np.sqrt(mean_squared_error(y_test, predicted))
predicted = rf.predict(X_test)
print('RMSE random forest:')
print(rmse)
```

```
RMSE random forest:
537886.7583541998
```

In [63]:
```python
gredR.fit(X_train, y_train)
predicted = gredR.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predicted))
print('RMSE Gradient Boosting:')
print(rmse)
```

```
RMSE Gradient Boosting:
398468.23572726286
```

In [73]:
```python
def predict_price(year, kilometers_driven, mileage, engine, power, seats, company, name, location, fuel_type, transmission, owner_type):
    company_index = np.where(X.columns==company)[0][0]
    name_index = np.where(X.columns==name)[0][0]
    location_index = np.where(X.columns==location)[0][0]
    fuel_type_index = np.where(X.columns==fuel_type)[0][0]
    transmission_index = np.where(X.columns==transmission)[0][0]
    owner_type_index = np.where(X.columns==owner_type)[0][0]
    data = np.zeros(len(X.columns))
    data[0] = year
    data[1] = kilometers_driven
    data[2] = mileage
    data[3] = engine
    data[4] = power
    data[5] = seats
    if company_index >= 0:
        data[company_index] = 1
    if name_index >= 0:
        data[name_index] = 1
    if location_index >= 0:
        data[location_index] = 1
    if fuel_type_index >= 0:
        data[fuel_type_index] = 1
    if transmission_index >= 0:
        data[transmission_index] = 1
    if owner_type_index >= 0:
        data[owner_type_index] = 1

    result = gredR.predict([data])
    return int(result[0])
```

In [76]:
```python
predict_price(3,50000, 19, 1582, 126.20, 5.0, 'Hyundai',
              'Hyundai Creta 1.6 CRDi SX Option', 'Hyderabad', 'Petrol', 'Manual', 'Second'
             )
```

Out[76]:
```
1002693
```

In [75]:
```python
predict_price(5,62221,17.4,1497,117.6, 5.0, 'Honda',
              'Honda City 1.5 E MT', 'Hyderabad', 'Petrol', 'Manual', 'Second'
             )
```

Out[75]:
```
892489
```

In [70]:
```python
predict_price(2,14000,21.01,1197,81.80, 5.0, 'Maruti',
              'Maruti Baleno Alpha', 'Hyderabad', 'Petrol', 'Manual', 'Second'
             )
```

Out[70]:
```
676086
```

In [ ]: