```
pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
import pandas as pd
import numpy as np
```

```
car=pd.read_csv('https://github.com/rajtilakls2510/car_price_predictor/raw/master/quikr_car.csv')
```

```
car.head()
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        892 non-null    object
 1   company     892 non-null    object
 2   year        892 non-null    object
 3   Price       892 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
dtypes: object(6)
memory usage: 41.9+ KB
```

```
car.tail()
```

	name	company	year	Price	kms_driven	fuel_type
887	Ta	Tara	zest	3,10,000	NaN	NaN
888	Tata Zest XM Diesel	Tata	2018	2,60,000	27,000 kms	Diesel
889	Mahindra Quanto C8	Mahindra	2013	3,90,000	40,000 kms	Diesel
890	Honda Amaze 1.2 E i VTEC	Honda	2014	1,80,000	Petrol	NaN
891	Chevrolet Sail 1.2 LT ABS	Chevrolet	2014	1,60,000	Petrol	NaN

```
car=car[car['year'].str.isnumeric()]
```

```
car['year']=car['year'].astype(int)
```

```
<ipython-input-7-c95edc1f455b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  car['year']=car['year'].astype(int)
```

```
car=car[car['Price']!="Ask For Price"]
```

```
car["Price"]=car['Price'].str.replace(',','').astype(int)
```

```
car["kms_driven"]= car["kms_driven"].str.split(' ').str.get(0).str.replace(",","")
```

```
car=car[car['kms_driven'].str.isnumeric()]
```

```
car=car[~car["fuel_type"].isna()]
```

```
car['name']=car['name'].str.split(" ").str.slice(0,3).str.join(" ")
```

```
car=car.reset_index(drop=True)
```

```
car
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
4	Ford Figo	Ford	2012	175000	41000	Diesel
...	...	...	...	...	...	...
811	Maruti Suzuki Ritz	Maruti	2011	270000	50000	Petrol
812	Tata Indica V2	Tata	2009	110000	30000	Diesel

```
car.describe()
```

	year	Price
count	816.000000	8.160000e+02
mean	2012.444853	4.117176e+05
std	4.002992	4.751844e+05
min	1995.000000	3.000000e+04
25%	2010.000000	1.750000e+05
50%	2013.000000	2.999990e+05
75%	2015.000000	4.912500e+05
max	2019.000000	8.500003e+06

```
car=car[car['Price']<6e6].reset_index(drop=True)
```

```
car.describe()
```

	year	Price
count	815.000000	8.150000e+02
mean	2012.442945	4.017933e+05
std	4.005079	3.815888e+05
min	1995.000000	3.000000e+04
25%	2010.000000	1.750000e+05
50%	2013.000000	2.999990e+05
75%	2015.000000	4.900000e+05
max	2019.000000	3.100000e+06

```
car.to_csv("cleaned car.csv")
```

```
x=car.drop(columns='Price')
y=car['Price']
```

Model

```
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)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

```
ohe=OneHotEncoder()
ohe.fit(x[['name','company','fuel_type']])
```

```
▾ OneHotEncoder
OneHotEncoder()
```

```
ohe.categories_
```

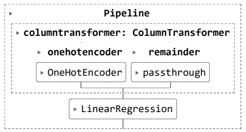
```
        'Maruti Suzuki Estilo', 'Maruti Suzuki Maruti',
        'Maruti Suzuki Omni', 'Maruti Suzuki Ritz', 'Maruti Suzuki S',
        'Maruti Suzuki SX4', 'Maruti Suzuki Stingray',
        'Maruti Suzuki Swift', 'Maruti Suzuki Versa',
        'Maruti Suzuki Vitara', 'Maruti Suzuki Wagon', 'Maruti Suzuki Zen',
        'Mercedes Benz A', 'Mercedes Benz B', 'Mercedes Benz C',
        'Mercedes Benz GLA', 'Mini Cooper S', 'Mitsubishi Lancer 1.8',
        'Mitsubishi Pajero Sport', 'Nissan Micra XL', 'Nissan Micra XV',
        'Nissan Sunny', 'Nissan Sunny XL', 'Nissan Terrano XL',
        'Nissan X Trail', 'Renault Duster', 'Renault Duster 110',
        'Renault Duster 110PS', 'Renault Duster 85', 'Renault Duster 85PS',
        'Renault Duster RxL', 'Renault Kwid', 'Renault Kwid 1.0',
        'Renault Kwid RXT', 'Renault Lodgy 85', 'Renault Scala RxL',
        'Skoda Fabia', 'Skoda Fabia 1.2L', 'Skoda Fabia Classic',
        'Skoda Laura', 'Skoda Octavia Classic', 'Skoda Rapid Elegance',
        'Skoda Superb 1.8', 'Skoda Yeti Ambition', 'Tata Aria Pleasure',
        'Tata Bolt XM', 'Tata Indica', 'Tata Indica V2', 'Tata Indica eV2',
        'Tata Indigo CS', 'Tata Indigo LS', 'Tata Indigo LX',
        'Tata Indigo Marina', 'Tata Indigo eCS', 'Tata Manza',
        'Tata Manza Aqua', 'Tata Manza Aura', 'Tata Manza ELAN',
        'Tata Nano', 'Tata Nano Cx', 'Tata Nano GenX', 'Tata Nano LX',
        'Tata Nano Lx', 'Tata Sumo Gold', 'Tata Sumo Grande',
        'Tata Sumo Victa', 'Tata Tiago Revotorq', 'Tata Tiago Revotron',
        'Tata Tigor Revotron', 'Tata Venture EX', 'Tata Vista Quadrajet',
        'Tata Zest Quadrajet', 'Tata Zest XE', 'Tata Zest XM',
        'Toyota Corolla', 'Toyota Corolla Altis', 'Toyota Corolla H2',
        'Toyota Etios', 'Toyota Etios G', 'Toyota Etios GD',
        'Toyota Etios Liva', 'Toyota Fortuner', 'Toyota Fortuner 3.0',
        'Toyota Innova 2.0', 'Toyota Innova 2.5', 'Toyota Qualis',
        'Volkswagen Jetta Comfortline', 'Volkswagen Jetta Highline',
        'Volkswagen Passat Diesel', 'Volkswagen Polo',
        'Volkswagen Polo Comfortline', 'Volkswagen Polo Highline',
        'Volkswagen Polo Highline1.2L', 'Volkswagen Polo Trendline',
        'Volkswagen Vento Comfortline', 'Volkswagen Vento Highline',
        'Volkswagen Vento Konekt', 'Volvo S80 Summum'], dtype=object),
 array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford',
        'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land',
        'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
        'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
       dtype=object),
 array(['Diesel', 'LPG', 'Petrol'], dtype=object)]
```

```
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                    remainder='passthrough')
```

```
lr=LinearRegression()
```

```
 pipe=make_pipeline(column_trans,lr)
```

```
pipe.fit(x_train,y_train)
```




```
y_pred=pipe.predict(x_test)
```

```
y_pred
```

```
array([ 476670.62399334, -104941.897735  ,  511726.14240716,
        320909.35976067, 1617911.23478079, 1157635.08334146,
        446012.75095767,  170511.30259977,  610162.80163135,
        568186.9082431 ,  305680.78333804,  234645.31215881,
        521786.13442309,  153080.3460394 ,  433713.2869643 ,
        380525.98821904,  362672.40810218,  514098.07002294,
        471237.91842301,  410278.51148297, 1209078.05355196,
        181971.65986201,   68483.73387857,  172743.72089379,
        609251.03764354,  442494.06617594,  316845.04090373,
        214617.00508091,  555061.82977204,  487633.07287505,
        394913.25218388,  329346.14180899,  363527.32345618,
        153189.53409889,  439098.10785604,  592928.59942895,
       1617911.23478079,  588558.7345705 ,  265107.97770253,
        676925.79362269,  258063.65100561,  195923.72749683,
        146809.88177666,  173341.45403829,   21388.53930601,
        -50534.14394173, -201001.41847686,  286706.30591857,
        208717.62398541,  392649.89732432,  289106.59637795,
       2117116.15949956,  693450.94973496,  484050.33163698,
        616450.68955216,  377597.72510912,  952732.3506612 ,
        653609.37176089,  259585.0511701 ,   39557.00231254,
        815694.40425037,  673515.48385921,  668016.61755353,
        196821.93137436,  171990.33642808,  106417.14700912,
        871017.87920818,  466896.96970379,  507400.4036385 ,
        475550.18073842,  214617.00508091,  460114.00050162,
        523843.00996248,  989547.25862695,  322347.02736774,
        331094.08775236,  506715.66335593,   71060.13483572,
       1104794.18990923,  227966.73521104,   34745.65002017,
        297611.91044702,  718684.66389856,  231512.82588215,
        -11231.8251798 ,  668630.12843335,  422092.83289357,
        345036.18867937,  167076.32832716,  595849.93100996,
        334777.83720094,  221514.81620365,  501765.9478021 ,
        182846.23818377,  332206.72242028,  546985.73851174,
       -300813.07175384,  251684.15341035,  398677.80143367,
         53176.0669436 ,  644418.58811878,  336210.0611577 ,
        126431.51528916,  399787.89535743,  385710.0204746 ,
        220102.45605031,  200499.69683249,  140109.42232703,
        291848.97306234,  335195.24927212,  -16465.70604396,
        359028.18344875, 1363644.90923327, 2117116.15949956,
        220391.69743967,  429367.41896215,  392449.48928171,
```

```
       423097.75331196,  464901.82552628,  187506.82201606,
       399242.52831818,  539200.0629462 ,  205017.80507192,
       590332.9244573 ,  375393.90808354,  272179.05362561,
       227784.62075458,  472093.15552774,  742519.16933358,
       517552.62888511,   65008.60652871,  148213.41681677,
       327306.87154171,  484523.81388462, -124469.66244447,
       814741.56443898,   26239.34514946,  399242.52831818,
       256796.67387599,  102218.34146774,  231059.9694847 ,
       278329.32842521,  297611.91044702,  462964.59284575,
       178685.14571695,  361941.26975483,  692102.6269937 ,
       361517.18562131,  281589.07263269,  465295.18743692,
       622422.83819205, 1040474.99101971,   78925.89158094,
       406768.86561207,  308014.73384885,  400242.14923292,
       193573.31501758, 2117261.8216615 ,  196681.0992495 ,
       216275.365889  ,  278891.83621439,  583687.06133346,
       375028.86310031])
```

```
r2_score(y_test,y_pred)
```

```
0.6405822393618714
```

```
scores=[]
for i in range(1000):
  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=i)
  lr=LinearRegression()
  pipe=make_pipeline(column_trans,lr)
  pipe.fit(x_train,y_train)
  y_pred=pipe.predict(x_test)
  scores.append(r2_score(y_test,y_pred))
```

```
np.argmax(scores)
```

```
433
```

```
scores[np.argmax(scores)]
```

```
0.8456515104452564
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=np.argmax(scores))
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(x_train,y_train)
y_pred=pipe.predict(x_test)
scores.append(r2_score(y_test,y_pred))
r2_score(y_test,y_pred)
```

```
0.8456515104452564
```

```
import pickle
```

```
pickle.dump(pipe,open('LinearRegressionModel.pkl','wb'))
```

```
pipe.predict(pd.DataFrame([['Maruti Suzuki Dzire','Maruti','2019','100','Petrol']],columns=['name','company','year','kms_driven','fuel_type']))
```

```
array([535674.25004546])
```