CAR PRICE PREDICTION

```
# OBJECTIVE
```

One of the main areas of research in machine learning is the prediction of the price of cars. It is based on finance and the marketing domain. It is a major research topic in machine learning because the price of a car depends on many factors. Some of the factors that contribute a lot to the price of a car are: -Brand -Model -Horsepower -Mileage -Safety Features -GPS and many more. If one ignores the brand of the car, a car manufacturer primarily fixes the price of a car based on the features it can offer a customer. Later, the brand may raise the price depending on its goodwill, but the most important factors are what features a car gives you to add valueto life. So, in the section below, I will walk you through the task of training a car price prediction model with machine learning using the Python programming language.

```
# IMPORTING NECESSARY LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import pickle

df=pd.read_csv(r"C:\Users\HP\OneDrive\Desktop\quikr_car.csv")
df
```

```
                                          name    company  year
Price  \
0      Hyundai Santro Xing XO eRLX Euro III    Hyundai  2007
80,000
1                   Mahindra Jeep CL550 MDI   Mahindra  2006
4,25,000
2                 Maruti Suzuki Alto 800 Vxi     Maruti  2018  Ask For
Price
3    Hyundai Grand i10 Magna 1.2 Kappa VTVT    Hyundai  2014
3,25,000
4          Ford EcoSport Titanium 1.5L TDCi       Ford  2014
5,75,000
..                                        ...        ...   ...
...
887                                        Ta       Tara  zest
3,10,000
888                     Tata Zest XM Diesel       Tata  2018
2,60,000
889                      Mahindra Quanto C8   Mahindra  2013
3,90,000
890                  Honda Amaze 1.2 E i VTEC      Honda  2014
1,80,000
891                Chevrolet Sail 1.2 LT ABS  Chevrolet  2014
```

```
1,60,000
```

```
     kms_driven fuel_type
0    45,000 kms    Petrol
1        40 kms    Diesel
2    22,000 kms    Petrol
3    28,000 kms    Petrol
4    36,000 kms    Diesel
..          ...       ...
887         NaN       NaN
888  27,000 kms    Diesel
889  40,000 kms    Diesel
890      Petrol       NaN
891      Petrol       NaN

[892 rows x 6 columns]
```

```
df.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
```

```
df1=df.copy()
```

```
df.year.unique() # here in the year column we can see that there are years but there is a lot of daat that does
# not have year and are filled with random values
```

```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '...', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
       'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
       '2 bs', 'arry', 'Eon', 'o...', 'ture', 'emi', 'car', 'able',
'no.',
       'd...', 'SALE', 'digo', 'sell', 'd Ex', 'n...', 'e...', 'D...',
       ', Ac', 'go .', 'k...', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
       't xe', 'EV2', 'r...', 'zest'], dtype=object)
```

```
df=df[df["year"].str.isnumeric()] # we will only keep the cars that
have numeric value and rest will filter out and then will convert them
to integere dtype

df.year=df.year.astype(int)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_22912\3595414959.py: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
  df.year=df.year.astype(int)
```

```
df.year.dtype
```

```
dtype('int32')
```

```
df.Price.unique() # heer we have one ask for price in price column
```

```
array(['80,000', '4,25,000', 'Ask For Price', '3,25,000', '5,75,000',
       '1,75,000', '1,90,000', '8,30,000', '2,50,000', '1,82,000',
       '3,15,000', '4,15,000', '3,20,000', '10,00,000', '5,00,000',
       '3,50,000', '1,60,000', '3,10,000', '75,000', '1,00,000',
       '2,90,000', '95,000', '1,80,000', '3,85,000', '1,05,000',
       '6,50,000', '6,89,999', '4,48,000', '5,49,000', '5,01,000',
       '4,89,999', '2,80,000', '3,49,999', '2,84,999', '3,45,000',
       '4,99,999', '2,35,000', '2,49,999', '14,75,000', '3,95,000',
       '2,20,000', '1,70,000', '85,000', '2,00,000', '5,70,000',
       '1,10,000', '4,48,999', '18,91,111', '1,59,500', '3,44,999',
       '4,49,999', '8,65,000', '6,99,000', '3,75,000', '2,24,999',
       '12,00,000', '1,95,000', '3,51,000', '2,40,000', '90,000',
       '1,55,000', '6,00,000', '1,89,500', '2,10,000', '3,90,000',
       '1,35,000', '16,00,000', '7,01,000', '2,65,000', '5,25,000',
       '3,72,000', '6,35,000', '5,50,000', '4,85,000', '3,29,500',
       '2,51,111', '5,69,999', '69,999', '2,99,999', '3,99,999',
       '4,50,000', '2,70,000', '1,58,400', '1,79,000', '1,25,000',
       '2,99,000', '1,50,000', '2,75,000', '2,85,000', '3,40,000',
       '70,000', '2,89,999', '8,49,999', '7,49,999', '2,74,999',
       '9,84,999', '5,99,999', '2,44,999', '4,74,999', '2,45,000',
       '1,69,500', '3,70,000', '1,68,000', '1,45,000', '98,500',
       '2,09,000', '1,85,000', '9,00,000', '6,99,999', '1,99,999',
       '5,44,999', '1,99,000', '5,40,000', '49,000', '7,00,000',
'55,000',
       '8,95,000', '3,55,000', '5,65,000', '3,65,000', '40,000',
       '4,00,000', '3,30,000', '5,80,000', '3,79,000', '2,19,000',
       '5,19,000', '7,30,000', '20,00,000', '21,00,000', '14,00,000',
       '3,11,000', '8,55,000', '5,35,000', '1,78,000', '3,00,000',
       '2,55,000', '5,49,999', '3,80,000', '57,000', '4,10,000',
```

```
       '2,25,000', '1,20,000', '59,000', '5,99,000', '6,75,000',
'72,500',
       '6,10,000', '2,30,000', '5,20,000', '5,24,999', '4,24,999',
       '6,44,999', '5,84,999', '7,99,999', '4,44,999', '6,49,999',
       '9,44,999', '5,74,999', '3,74,999', '1,30,000', '4,01,000',
       '13,50,000', '1,74,999', '2,39,999', '99,999', '3,24,999',
       '10,74,999', '11,30,000', '1,49,000', '7,70,000', '30,000',
       '3,35,000', '3,99,000', '65,000', '1,69,999', '1,65,000',
       '5,60,000', '9,50,000', '7,15,000', '45,000', '9,40,000',
       '1,55,555', '15,00,000', '4,95,000', '8,00,000', '12,99,000',
       '5,30,000', '14,99,000', '32,000', '4,05,000', '7,60,000',
       '7,50,000', '4,19,000', '1,40,000', '15,40,000', '1,23,000',
       '4,98,000', '4,80,000', '4,88,000', '15,25,000', '5,48,900',
       '7,25,000', '99,000', '52,000', '28,00,000', '4,99,000',
       '3,81,000', '2,78,000', '6,90,000', '2,60,000', '90,001',
       '1,15,000', '15,99,000', '1,59,000', '51,999', '2,15,000',
       '35,000', '11,50,000', '2,69,000', '60,000', '4,30,000',
       '85,00,003', '4,01,919', '4,90,000', '4,24,000', '2,05,000',
       '5,49,900', '4,35,000', '1,89,700', '3,89,700', '3,60,000',
       '2,95,000', '1,14,990', '10,65,000', '4,70,000', '48,000',
       '1,88,000', '4,65,000', '1,79,999', '21,90,000', '23,90,000',
       '10,75,000', '4,75,000', '10,25,000', '6,15,000', '19,00,000',
       '14,90,000', '15,10,000', '18,50,000', '7,90,000', '17,25,000',
       '12,25,000', '68,000', '9,70,000', '31,00,000', '8,99,000',
       '88,000', '53,000', '5,68,500', '71,000', '5,90,000',
'7,95,000',
       '42,000', '1,89,000', '1,62,000', '35,999', '29,00,000',
'39,999',
       '50,500', '5,10,000', '8,60,000', '5,00,001'], dtype=object)
```

```
df=df[df["Price"]!='Ask For Price']
df
# now we will remove the commas from this and convert into appropriate
datatype
```

```
                                          name    company  year     Price
\
0      Hyundai Santro Xing XO eRLX Euro III    Hyundai  2007    80,000

1                    Mahindra Jeep CL550 MDI   Mahindra  2006  4,25,000

3    Hyundai Grand i10 Magna 1.2 Kappa VTVT    Hyundai  2014  3,25,000

4          Ford EcoSport Titanium 1.5L TDCi       Ford  2014  5,75,000

6                                  Ford Figo       Ford  2012  1,75,000

..                                       ...        ...   ...       ...

886                    Toyota Corolla Altis     Toyota  2009  3,00,000
```

```
888                     Tata Zest XM Diesel       Tata  2018  2,60,000

889                      Mahindra Quanto C8   Mahindra  2013  3,90,000

890                  Honda Amaze 1.2 E i VTEC      Honda  2014  1,80,000

891                Chevrolet Sail 1.2 LT ABS  Chevrolet  2014  1,60,000
```

```
       kms_driven fuel_type
0      45,000 kms    Petrol
1          40 kms    Diesel
3      28,000 kms    Petrol
4      36,000 kms    Diesel
6      41,000 kms    Diesel
..             ...       ...
886  1,32,000 kms    Petrol
888    27,000 kms    Diesel
889    40,000 kms    Diesel
890        Petrol       NaN
891        Petrol       NaN

[819 rows x 6 columns]
```

```
df["Price"]=df["Price"].str.replace(",","")
```

```
df["Price"]=df["Price"].astype(int)
```

```
# now we converted the Price column to suitable integere type and
cleaned it of unwanted values
```

```
df["kms_driven"]=df["kms_driven"].str.strip('kms')
```

```
df["kms_driven"]=df["kms_driven"].str.replace(',','')
```

```
df["kms_driven"] # herew we can see that we have petrol in rows we
will delete that
```

```
0       45000
1          40
3       28000
4       36000
6       41000
        ...
886    132000
888     27000
889     40000
890     Petrol
```

```
891     Petrol
Name: kms_driven, Length: 819, dtype: object
```

```
for x in df.index:
    if df.loc[x,"kms_driven"]=="Petrol":
        df.drop(x,inplace=True)


df["kms_driven"]=df["kms_driven"].astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 817 entries, 0 to 889
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        817 non-null    object
 1   company     817 non-null    object
 2   year        817 non-null    int32 
 3   Price       817 non-null    int32 
 4   kms_driven  817 non-null    int32 
 5   fuel_type   816 non-null    object
dtypes: int32(3), object(3)
memory usage: 35.1+ KB
```

```
# n0w we will look at the fuel type column
df=df[~df["fuel_type"].isna()]
```

```
df
```

```
                                          name   company  year   Price  \
0      Hyundai Santro Xing XO eRLX Euro III   Hyundai  2007   80000   
1                   Mahindra Jeep CL550 MDI  Mahindra  2006  425000   
3    Hyundai Grand i10 Magna 1.2 Kappa VTVT   Hyundai  2014  325000   
4          Ford EcoSport Titanium 1.5L TDCi      Ford  2014  575000   
6                                 Ford Figo      Ford  2012  175000   
..                                      ...       ...   ...     ...   
883             Maruti Suzuki Ritz VXI ABS    Maruti  2011  270000   
885              Tata Indica V2 DLE BS III      Tata  2009  110000   
886                   Toyota Corolla Altis    Toyota  2009  300000   
888                    Tata Zest XM Diesel      Tata  2018  260000   
889                     Mahindra Quanto C8  Mahindra  2013  390000   

     kms_driven fuel_type  
0         45000    Petrol  
1            40    Diesel  
3         28000    Petrol  
4         36000    Diesel  
6         41000    Diesel  
..          ...       ...  
```

```
883       50000    Petrol
885       30000    Diesel
886      132000    Petrol
888       27000    Diesel
889       40000    Diesel

[816 rows x 6 columns]
```

```
# we only want to keep the first 3 rows of the name as it is very
messy and the first will be suitable for classification
df["name"]=df["name"].str.split(" ").str.slice(0,3).str.join(' ')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_22912\2938324665.py:2:
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
  df["name"]=df["name"].str.split(" ").str.slice(0,3).str.join(' ')
```

```
df
```

```
                         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
3         Hyundai Grand i10   Hyundai  2014  325000       28000
Petrol
4    Ford EcoSport Titanium      Ford  2014  575000       36000
Diesel
6                 Ford Figo      Ford  2012  175000       41000
Diesel
..                      ...       ...   ...     ...         ...
...
883      Maruti Suzuki Ritz    Maruti  2011  270000       50000
Petrol
885          Tata Indica V2      Tata  2009  110000       30000
Diesel
886    Toyota Corolla Altis    Toyota  2009  300000      132000
Petrol
888            Tata Zest XM      Tata  2018  260000       27000
Diesel
889      Mahindra Quanto C8  Mahindra  2013  390000       40000
Diesel

[816 rows x 6 columns]
```

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

```
                        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
813    Toyota Corolla Altis    Toyota  2009   300000      132000
Petrol
814            Tata Zest XM      Tata  2018   260000       27000
Diesel
815      Mahindra Quanto C8  Mahindra  2013   390000       40000
Diesel

[816 rows x 6 columns]
```

```
df.describe()
```

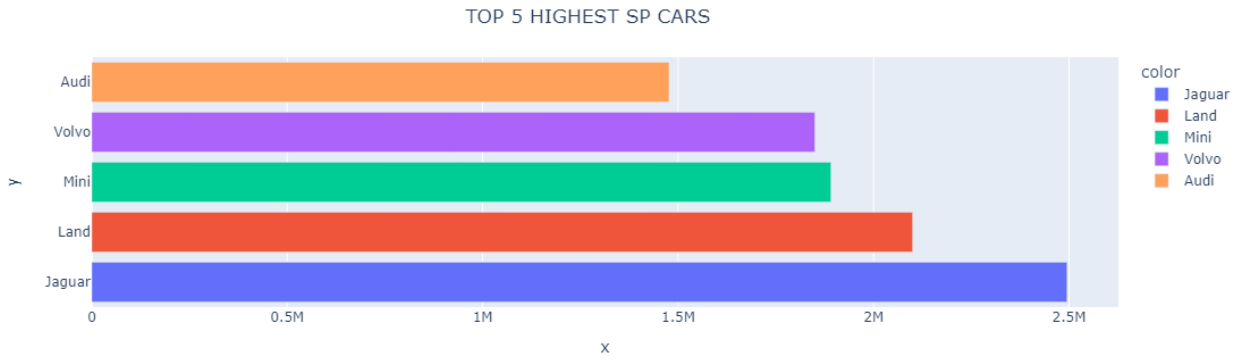
```
              year         Price     kms_driven
count   816.000000  8.160000e+02     816.000000
mean   2012.444853  4.117176e+05   46275.531863
std       4.002992  4.751844e+05   34297.428044
min    1995.000000  3.000000e+04       0.000000
25%    2010.000000  1.750000e+05   27000.000000
50%    2013.000000  2.999990e+05   41000.000000
75%    2015.000000  4.912500e+05   56818.500000
max    2019.000000  8.500003e+06  400000.000000
```

```
# now we wnat to see which company has cars in the market via selling
price top5
company=df.groupby("company").Price.mean().sort_values(ascending=False
)[:5]
company
```

```
company
Jaguar    2.495000e+06
Land      2.100000e+06
```

```
Mini      1.891111e+06
Volvo     1.850000e+06
Audi      1.476909e+06
Name: Price, dtype: float64

fig=px.bar(y=company.index,x=company.values,color=company.index)
fig.update_layout(title="TOP 5 HIGHEST SP CARS",title_x=0.47)
fig.show()
```



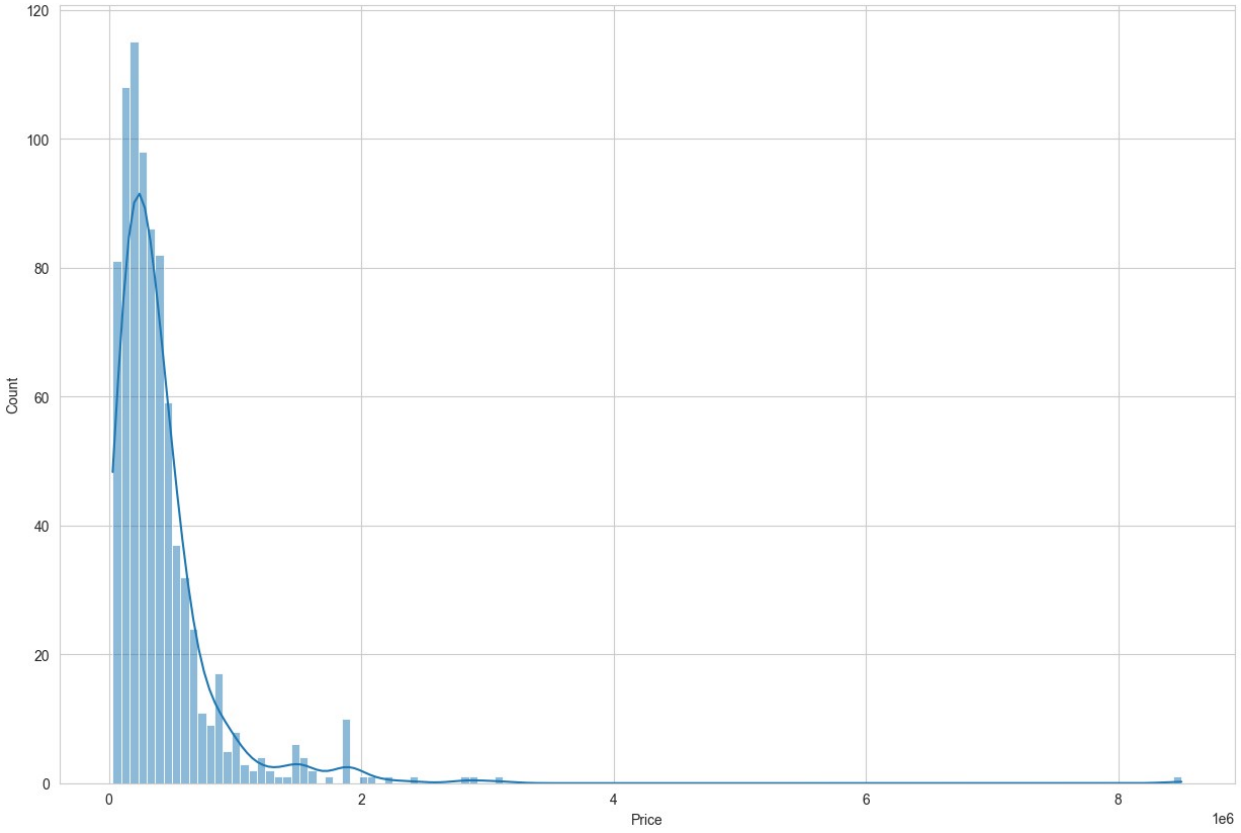

```
# i want to find out the fresh cars with year so lets segregate the
data
year=df[df["year"]==2000]
year
```

	name	company	year	Price	kms_driven	fuel_type
139	Hindustan Motors Ambassador	Hindustan	2000	70000	200000	Diesel
192	Maruti Suzuki Zen	Maruti	2000	55000	60000	Petrol
284	Hyundai Santro Xing	Hyundai	2000	59000	56450	Petrol
402	Honda City	Honda	2000	65000	80000	Petrol
547	Hyundai Santro	Hyundai	2000	51999	88000	Petrol
834	Maruti Suzuki Omni	Maruti	2000	35999	60000	Petrol
851	Maruti Suzuki 800	Maruti	2000	30000	33400	Petrol

```
# we will store the clean data in a csv ile
df.to_csv("cars.csv")
```

The price column in this dataset is supposed to be the column whose values we need to predict. So let's see the distribution of the values of the price column:

```
import seaborn as sns
sns.set_style("whitegrid")
plt.figure(figsize=(15, 10))
sns.histplot(df.Price,kde=True)
plt.show()
```



Model

```
# now we we will create a model thta will predict the car price on the
some features

# segregating the data required for prediction
X=df.drop(columns="Price")
y=df["Price"]

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

```
OneHotEncoder()
```

```
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline

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

lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
```

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',

OneHotEncoder(categories=[array(['Audi A3 Cabriolet', 'Audi A4 1.8',
'Audi A4 2.0', 'Audi A6 2.0',
       'Audi A8', 'Audi Q3 2.0', 'Audi Q5 2.0', 'Audi Q7', 'BMW 3
Series',
       'BMW 5 Series', 'BMW 7 Series', 'BMW X1', 'BMW X1 sDrive20d',
       'BMW X1 xDrive20d', 'Chevrolet Beat', 'Chevrolet Beat...

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)]),
                                                  ['name', 'company',
                                                   'fuel_type'])])),
                ('linearregression', LinearRegression())])
```

```
# training the model for prediction
scores=[]
for i in range(20000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
    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)
```

```
17205
```

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

```
0.8439597935929756
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,rando
m_state=np.argmax(scores))
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)

# creating a pickle file

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

# predicting a car price base don the desired parameters

pipe.predict(pd.DataFrame([['Maruti Suzuki
Swift','Maruti',2022,100000,'Petrol']],columns=['name','company','year
','kms_driven','fuel_type']))
```

```
array([478722.41878423])
```

------ PLEASE IGNORE THIS ------

```
df2=df[['year','kms_driven','fuel_type','Price']]
df2.reset_index(drop=True)
```

	year	kms_driven	fuel_type	Price
0	2007	45000	Petrol	80000
1	2006	40	Diesel	425000
2	2014	28000	Petrol	325000
3	2014	36000	Diesel	575000
4	2012	41000	Diesel	175000
..	...	...	...	...
811	2011	50000	Petrol	270000
812	2009	30000	Diesel	110000
813	2009	132000	Petrol	300000
814	2018	27000	Diesel	260000
815	2013	40000	Diesel	390000

```
[816 rows x 4 columns]
```

```
d={"Petrol":1,"Diesel":2}
```

```
df2["fuel_type"]=df2["fuel_type"].map(d)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_22912\1437950470.py: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
```

```
df2.dropna(inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_22912\1761232742.py:1:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

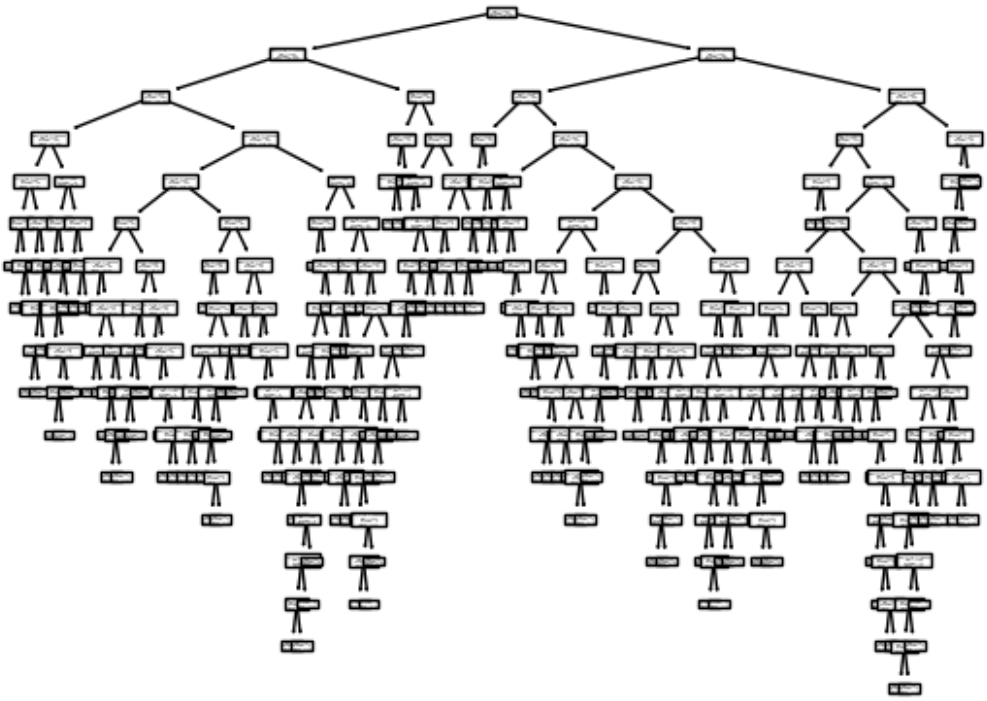
```
df2.isnull().sum()
```

```
year          0
kms_driven    0
fuel_type     0
Price         0
dtype: int64
```

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
features = ['year', 'kms_driven', 'Price']
X=df2[features]
y=df2["fuel_type"]

dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)

tree.plot_tree(dtree, feature_names=features)
plt.show()
```



```
print(dtree.predict([[2010,4500,4770]]))
```

```
[1.]

C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\
sklearn\base.py:464: UserWarning:

X does not have valid feature names, but DecisionTreeClassifier was
fitted with feature names
```

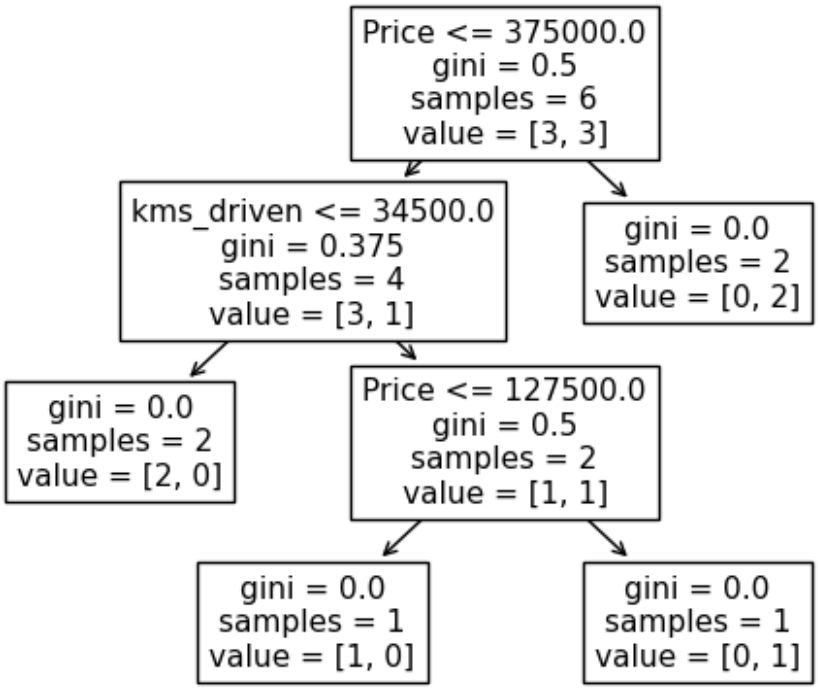
```
exp=df2.head(6)
exp
```

```
   year  kms_driven  fuel_type   Price
0  2007       45000        1.0   80000
1  2006          40        2.0  425000
3  2014       28000        1.0  325000
4  2014       36000        2.0  575000
6  2012       41000        2.0  175000
7  2013       25000        1.0  190000
```

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
features = ['year', 'kms_driven', 'Price']
```

```
X=exp[features]
y=exp["fuel_type"]
dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)

tree.plot_tree(dtree, feature_names=features)
plt.show()
```




```
dtree.predict([[2010,4500,4770]])
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\
sklearn\base.py:464: UserWarning:

X does not have valid feature names, but DecisionTreeClassifier was
fitted with feature names


array([1.])
```