Aim of the project is to create a model that will help to predict the price of a car whenever a new data is inputed... I used a previous data in this project

Importing the dependencies (libraries needed)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style("whitegrid")
import warnings
warnings.filterwarnings("ignore")
```

Loading the dataset

```
df = pd.read_csv('car_price.csv')
df
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	d
0	1	3	alfa-romero giulia	gas	std	two	convertible	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	
3	4	2	audi 100 ls	gas	std	four	sedan	
4	5	2	audi 100ls	gas	std	four	sedan	
...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	
202	203	-1	volvo 244dl	gas	std	four	sedan	
203	204	-1	volvo 246	diesel	turbo	four	sedan	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	

205 rows × 26 columns

```
df.shape
```

```
(205, 26)
```

```
df.columns
```

```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

```
df.dtypes
```

```
car_ID                int64
symboling             int64
CarName              object
fueltype             object
aspiration           object
doornumber           object
carbody              object
drivewheel           object
enginelocation       object
wheelbase           float64
carlength           float64
carwidth            float64
carheight           float64
curbweight            int64
enginetype           object
cylindernumber       object
enginesize            int64
fuelsystem           object
boreratio           float64
stroke              float64
compressionratio    float64
horsepower            int64
peakrpm               int64
citympg               int64
highwaympg            int64
price               float64
dtype: object
```

```
df.nunique()
```

```
car_ID              205
symboling             6
CarName             147
fueltype              2
aspiration            2
doornumber            2
```

```
carbody               5
drivewheel            3
enginelocation        2
wheelbase            53
carlength            75
carwidth             44
carheight            49
curbweight          171
enginetype            7
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

```
Is_there_missing_value = df.isnull().sum()
Is_there_missing_value
```

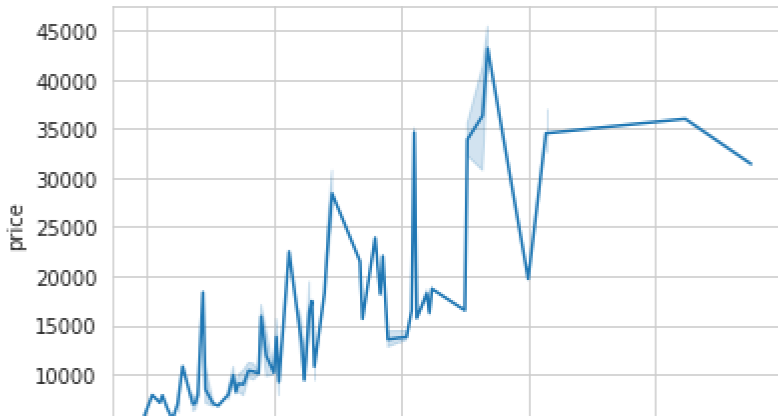
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

EDA

DATA VISUALIZATION

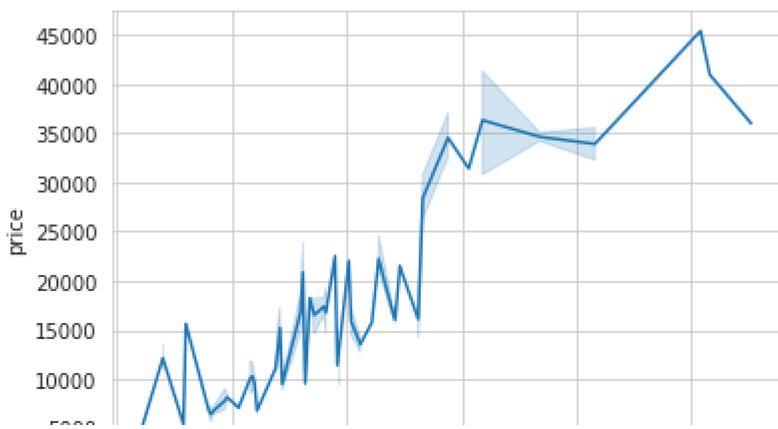
```
sns.lineplot(data=df, x='horsepower',y='price')
plt.xticks(rotation=90)
```

```
(array([  0.,  50., 100., 150., 200., 250., 300.]),
 <a list of 7 Text major ticklabel objects>)
```




```
sns.lineplot(data=df,x='enginesize',y='price')
plt.xticks(rotation=90)
```

```
(array([  0.,  50., 100., 150., 200., 250., 300., 350.]),
 <a list of 8 Text major ticklabel objects>)
```

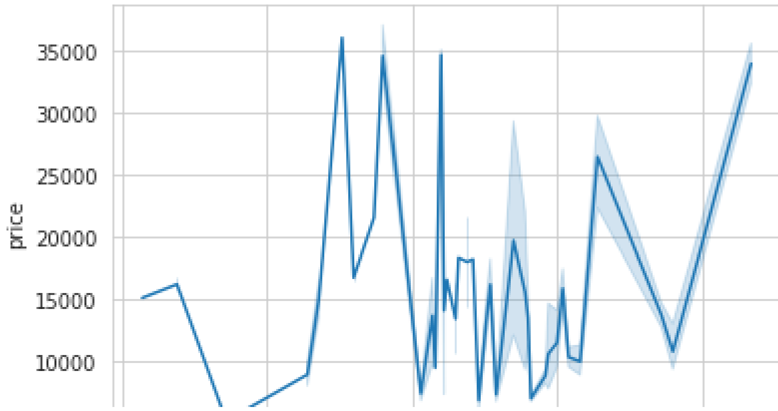



```
sns.lineplot(data=df,x='enginetype',y='price')
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6], <a list of 7 Text major ticklabel objects>)
```

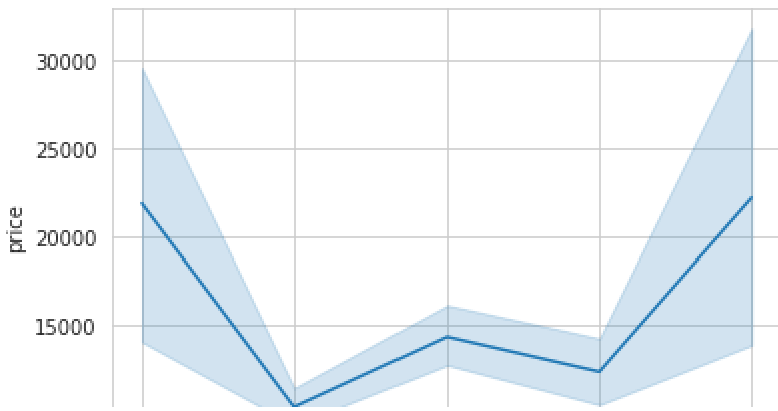
```
sns.lineplot(data=df,x='stroke',y='price')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c11a29b90>
```



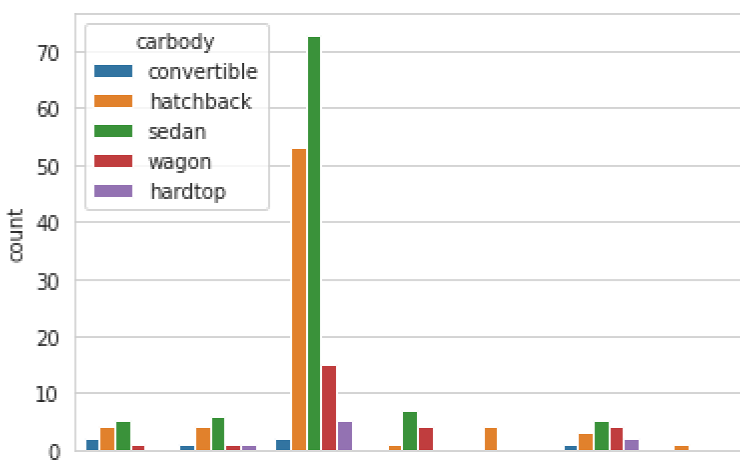

```
sns.lineplot(data=df,x='carbody',y='price')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c11a77450>
```



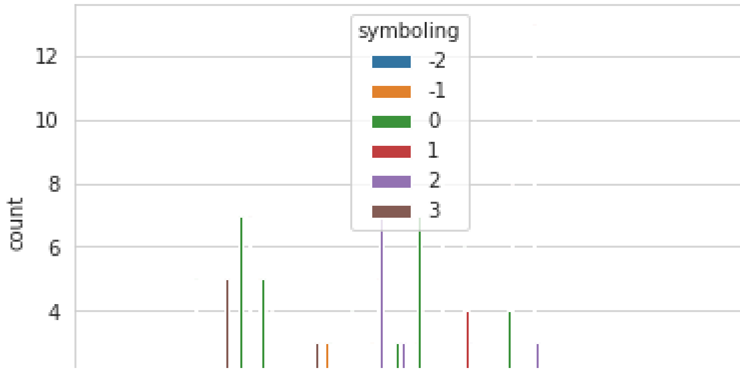

```
sns.countplot('enginetype',hue='carbody',data=df)
plt.xticks(rotation=90)
```

```
(array([0, 1, 2, 3, 4, 5, 6]), <a list of 7 Text major ticklabel objects>)
```




```
sns.countplot('highwaympg',hue='symboling',data=df)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 <a list of 30 Text major ticklabel objects>)
```




```
df = df.apply(pd.to_numeric, errors='coerce')
```

```
df
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	driv
0	1	3	NaN	NaN	NaN	NaN	NaN	
1	2	3	NaN	NaN	NaN	NaN	NaN	
2	3	1	NaN	NaN	NaN	NaN	NaN	
3	4	2	NaN	NaN	NaN	NaN	NaN	
4	5	2	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
200	201	-1	NaN	NaN	NaN	NaN	NaN	
201	202	-1	NaN	NaN	NaN	NaN	NaN	
202	203	-1	NaN	NaN	NaN	NaN	NaN	
203	204	-1	NaN	NaN	NaN	NaN	NaN	
204	205	-1	NaN	NaN	NaN	NaN	NaN	

205 rows × 26 columns

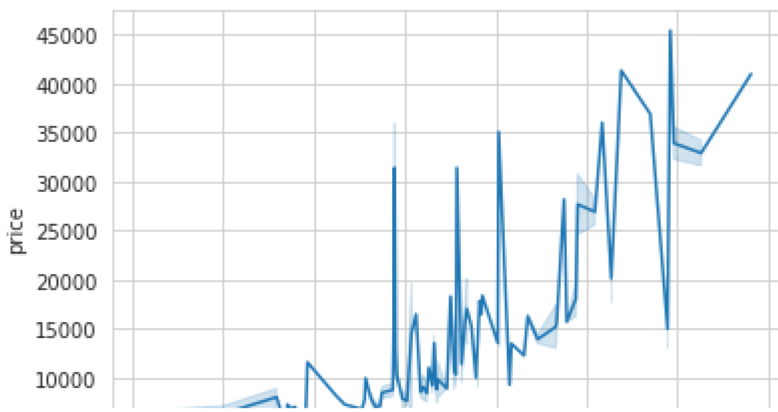
```
df.dropna(axis = 1,inplace=True)
```

```
df
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	e
0	1	3	88.6	168.8	64.1	48.8	2548	
1	2	3	88.6	168.8	64.1	48.8	2548	
2	3	1	94.5	171.2	65.5	52.4	2823	
3	4	2	99.8	176.6	66.2	54.3	2337	
4	5	2	99.4	176.6	66.4	54.3	2824	
...	...	...	...	...	...	...	...	
200	201	-1	109.1	188.8	68.9	55.5	2952	
201	202	-1	109.1	188.8	68.8	55.5	3049	
202	203	-1	109.1	188.8	68.9	55.5	3012	
203	204	-1	109.1	188.8	68.9	55.5	3217	

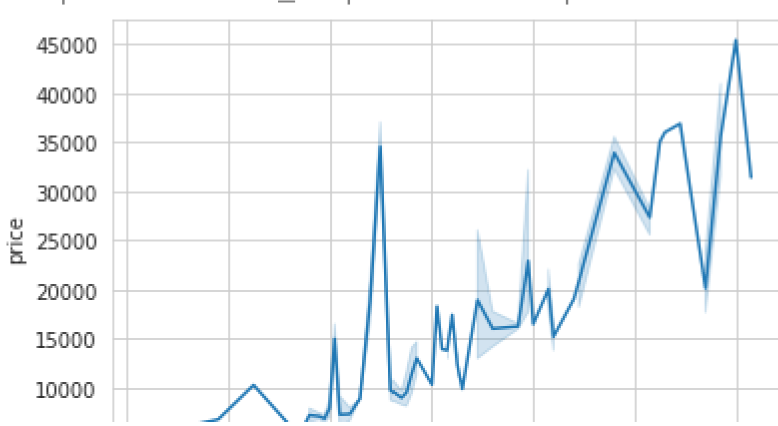
```
sns.lineplot(data=df,x='carlength',y='price')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0c11596450>



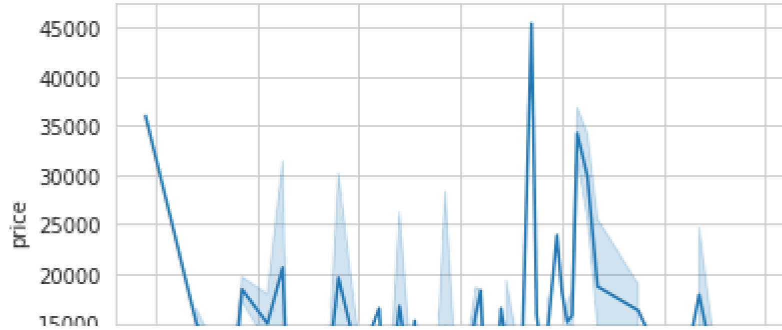

```
sns.lineplot(data=df,x='carwidth',y='price')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0c114fdb90>




```
sns.lineplot(data=df,x='carheight',y='price')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0c11541150>




```
df.drop(['car_ID','symboling'],axis=1,inplace=True)
```

Feature Selection

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler

X = df.iloc[:,:-1]
y = df['price']

X_norm = MinMaxScaler()
X_norm.fit_transform(X)
```

```
array([[0.05830904, 0.41343284, 0.31666667, ..., 0.34693878, 0.22222222,
        0.28947368],
       [0.05830904, 0.41343284, 0.31666667, ..., 0.34693878, 0.22222222,
        0.28947368],
       [0.2303207 , 0.44925373, 0.43333333, ..., 0.34693878, 0.16666667,
        0.26315789],
       ...,
       [0.65597668, 0.7119403 , 0.71666667, ..., 0.55102041, 0.13888889,
        0.18421053],
       [0.65597668, 0.7119403 , 0.71666667, ..., 0.26530612, 0.36111111,
        0.28947368],
       [0.65597668, 0.7119403 , 0.71666667, ..., 0.51020408, 0.16666667,
        0.23684211]])
```

```
Chi = SelectKBest(k=3)
Chi_features = Chi.fit_transform(X,y)


Chi_support = Chi.get_support()
Chi_feat = X.loc[:,Chi_support].columns.tolist()

print("Original feature number ", X.shape[1])
print(Chi_feat)
```

```
Original feature number  13
['carwidth', 'curbweight', 'enginesize']
```

Correlation features

```
corr = df.corr()
sns.heatmap(corr, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c1043de10>
```


```
corr_target = abs(corr['price'])
relevant_feat = corr_target[corr_target > 0.8]
print("Original feature :", df.shape[1])
print(relevant_feat)
```

```
Original feature : 14
curbweight    0.835305
enginesize    0.874145
horsepower    0.808139
price         1.000000
Name: price, dtype: float64
```

```
from sklearn.feature_selection import mutual_info_classif
res = mutual_info_classif(X.astype(int),y.astype(int))
feature_importance = pd.Series(res,df.columns[0:len(df.columns)-1])
feature_importance.plot(kind='bar')
print(res)
```

```
[1.96850238 1.44607998 1.72890834 1.68922    1.51407408 1.16566588
 1.03103398 0.8256772  0.8745418  2.14490143 1.83133307 1.82432366
 1.76869883]
```

Backward Elimination

```
df.shape
```

```
(205, 14)
```

```
import statsmodels.api as sm
X_data = df[['curbweight','enginesize','horsepower']]
X1 = np.append(arr=np.ones((205,1)).astype(int), values = X_data, axis = 1)
X_opt = X1[:,[0,1,2,3]]
```

```
pd.DataFrame(X_data)
```

	curbweight	enginesize	horsepower
0	2548	130	111
1	2548	130	111
2	2823	152	154
3	2337	109	102
4	2824	136	115
...	...	...	...
200	2952	141	114
201	3049	141	160
202	3012	173	134
203	3217	145	106
204	3062	141	114

205 rows × 3 columns

```
pd.DataFrame(X_opt)
```

	0	1	2	3
0	1	2548	130	111
1	1	2548	130	111
2	1	2823	152	154
3	1	2337	109	102
4	1	2824	136	115
...	...	...	...	...
200	1	2952	141	114
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
regressor_OLS = sm.OLS(endog=y,exog=X_opt).fit()
regressor_OLS.summary()
```

	OLS Regression Results		
Dep. Variable:	price	**R-squared:**	0.814
Model:	OLS	**Adj. R-squared:**	0.811
Method:	Least Squares	**F-statistic:**	292.9
Date:	Fri, 27 May 2022	**Prob (F-statistic):**	4.36e-73
Time:	22:00:32	**Log-Likelihood:**	-1960.2

In the above output, The features P_value is not greater than 0.5

Train and Test set

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_data,y,
test_size=0.25,
random_state=0)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(153, 3)
(52, 3)
(153,)
(52,)
```

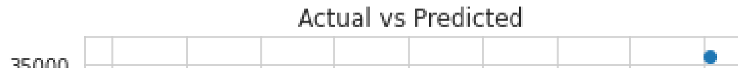
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

```
plt.scatter(y_test,y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```




```
print("Training score is ",model.score(X_train,y_train))
print("Testing score is ",model.score(X_test,y_test))
```

```
Training score is  0.8115088706121217
Testing score is  0.8138284013856698
```

```
from sklearn.metrics import r2_score
print(r2_score(y_test,y_pred))
```

```
0.8138284013856698
```