Used Cars in Egypt

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

In [ ]:

In [ ]:

Read and explore data

In [2]:
```
cars_v1 = pd.read_csv('cars.csv')
```

In [4]:
```
cars_v1.head()
```

Out[4]:

	Unnamed: 0	Brand	Model	Body	Color	Year	Fuel	Kilometers	Engine	Transmission	Price
0	5337	Hyundai	Accent	Sedan	Black	2007	Benzine	140000 to 159999	1600 CC	Automatic	140.0
1	5338	Hyundai	Accent	Sedan	Silver	2005	Benzine	180000 to 199999	1000 - 1300 CC	Manual	78.0
2	5339	Hyundai	Accent	Sedan	Gray	1999	Benzine	140000 to 159999	1400 - 1500 CC	Manual	70.0
3	5340	Hyundai	Accent	Sedan	Blue- Navy Blue	2009	Benzine	140000 to 159999	1600 CC	Automatic	150.0
4	5341	Hyundai	Accent	Sedan	Silver	2000	Benzine	10000 to 19999	1000 - 1300 CC	Manual	75.0

In [6]:
```
cars_v1.describe(include='all')
```

Out[6]:

	Unnamed: 0	Brand	Model	Body	Color	Year	Fuel	Kilometers	Engine	Transn
count	14741.000000	14741	14741	14741	14741	14741.000000	14741	14741	14741	
unique	NaN	3	18	3	14	NaN	2	16	3	
top	NaN	Hyundai	128	Sedan	White	NaN	Benzine	More than 200000	1600 CC	N
freq	NaN	5692	2425	13453	2614	NaN	14200	2505	6762	
mean	8934.846754	NaN	NaN	NaN	NaN	2005.456821	NaN	NaN	NaN	
std	4922.065495	NaN	NaN	NaN	NaN	12.655566	NaN	NaN	NaN	
min	812.000000	NaN	NaN	NaN	NaN	1970.000000	NaN	NaN	NaN	
25%	4497.000000	NaN	NaN	NaN	NaN	1998.000000	NaN	NaN	NaN	
50%	8182.000000	NaN	NaN	NaN	NaN	2010.000000	NaN	NaN	NaN	
75%	13373.000000	NaN	NaN	NaN	NaN	2015.000000	NaN	NaN	NaN	
max	17058.000000	NaN	NaN	NaN	NaN	2022.000000	NaN	NaN	NaN	

In [10]:

```python
for col in cars_v1.columns:
    print("Col is ", col)
    print(cars_v1[col].value_counts())
```

```
Col is  Unnamed: 0
5337     1
2931     1
2933     1
2934     1
2935     1
        ..
16284    1
16285    1
16286    1
16287    1
14213    1
Name: Unnamed: 0, Length: 14741, dtype: int64
Col is  Brand
Hyundai      5692
Fiat         5033
Chevrolet    4016
Name: Brand, dtype: int64
Col is  Model
128        2425
Verna      1903
Elantra    1529
Lanos      1342
Accent     1272
Optra      1252
Shahin     1142
Aveo        994
131         572
Cruze       428
Uno         350
Avante      282
Tipo        274
Punto       270
Matrix      268
Tucson      182
I10         166
Excel        90
Name: Model, dtype: int64
Col is  Body
Sedan        13453
Hatchback     1106
SUV            182
Name: Body, dtype: int64
Col is  Color
White              2614
Black              2032
Silver             1952
Gray               1670
Red                1538
Blue- Navy Blue    1406
Other Color        1134
Burgundy           1061
Green               456
Gold                374
Beige               152
Brown               140
Yellow              134
Orange               78
Name: Color, dtype: int64
Col is  Year
```

```
2013    850
2010    763
2011    728
2015    727
2012    693
2017    690
2014    622
2019    607
2009    591
2018    574
2016    562
2008    485
2021    388
2006    348
2020    346
2007    300
2001    260
1999    241
2000    231
2002    230
1998    219
2005    218
1987    212
2003    211
1990    211
1979    209
1982    198
2004    185
1997    183
1996    173
1980    172
1988    160
1983    160
1985    152
1981    145
1984    145
1993    144
1994    144
1986    143
1991    140
1977    130
2022    118
1989    115
1978    114
1995    114
1976    111
1992     90
1975     69
1974     56
1972     22
1973     21
1971     20
1970      1
Name: Year, dtype: int64
Col is  Fuel
Benzine        14200
Natural Gas      541
Name: Fuel, dtype: int64
Col is  Kilometers
More than 200000    2505
```

```
10000 to 19999      1666
180000 to 199999    1349
100000 to 119999    1192
0 to 9999           1088
140000 to 159999    1064
120000 to 139999    1005
90000 to 99999       996
160000 to 179999     760
20000 to 29999       612
80000 to 89999       560
50000 to 59999       436
60000 to 69999       402
40000 to 49999       372
30000 to 39999       370
70000 to 79999       364
Name: Kilometers, dtype: int64
Col is  Engine
1600 CC           6762
1400 - 1500 CC    4356
1000 - 1300 CC    3623
Name: Engine, dtype: int64
Col is  Transmission
Manual       9862
Automatic    4879
Name: Transmission, dtype: int64
Col is  Price
115.0    254
23.0     234
138.0    209
161.4    195
185.6    195
        ...
122.5      1
68.5       1
202.0      1
111.1      1
46.6       1
Name: Price, Length: 631, dtype: int64
Col is  Gov
Cairo             4458
Giza              2412
Alexandria        1636
Sharqia            851
Qalyubia           806
Gharbia            630
Dakahlia           590
Monufia            444
Ismailia           360
Suez               308
Fayoum             276
Beheira            246
Minya              236
Asyut              216
Damietta           210
Beni Suef          186
Kafr al-Sheikh     174
Sohag              158
Red Sea            132
Port Said          128
Qena               100
```

```
South Sinai         56
Luxor               42
Aswan               42
Matruh              36
New Valley           8
Name: Gov, dtype: int64
```

In [ ]:
```python
cars_v1[['x', 'y']] = cars_v1['Kilometers'].str.extractall('(\d+)').unstack().loc[:,0]
```

In [19]:
```python
cars_v1['x']=cars_v1['x'].astype(int)
```

In [54]:
```python
cars_v1['y'].value_counts()
```

Out[54]:
```
19999     1666
199999    1349
119999    1192
9999      1088
159999    1064
139999    1005
99999      996
179999     760
29999      612
89999      560
59999      436
69999      402
49999      372
39999      370
79999      364
Name: y, dtype: int64
```

In [56]:
```python
cars_v1['y']=cars_v1['y'].astype(float)
```

In [ ]:
```python

```

In [57]:
```python
cars_v1.dtypes
```

Out[57]:
```
Unnamed: 0        int64
Brand            object
Model            object
Body             object
Color            object
Year              int64
Fuel             object
Kilometers       object
Engine           object
Transmission     object
Price           float64
Gov              object
x                 int32
y               float64
KM_Adj          float64
dtype: object
```

In [58]:
```python
cars_v1['KM_Adj'] = cars_v1[['x', 'y']].mean(axis=1)
```

In [61]:
```python
cars_v1.drop(['Kilometers','x','y'],axis=1, inplace= True)
```

In [62]:
```python
cars_v1.head()
```

Out[62]:

	Unnamed: 0	Brand	Model	Body	Color	Year	Fuel	Engine	Transmission	Price	Gov	KM_A
0	5337	Hyundai	Accent	Sedan	Black	2007	Benzine	1600 CC	Automatic	140.0	Giza	149999
1	5338	Hyundai	Accent	Sedan	Silver	2005	Benzine	1000 - 1300 CC	Manual	78.0	Qena	189999
2	5339	Hyundai	Accent	Sedan	Gray	1999	Benzine	1400 - 1500 CC	Manual	70.0	Giza	149999
3	5340	Hyundai	Accent	Sedan	Blue- Navy Blue	2009	Benzine	1600 CC	Automatic	150.0	Cairo	149999
4	5341	Hyundai	Accent	Sedan	Silver	2000	Benzine	1000 - 1300 CC	Manual	75.0	Giza	1499

In [ ]:

In [63]:
```
cars_v1[['x', 'y']] = cars_v1['Engine'].str.extractall('(\d+)').unstack().loc[:,0]
```

In [68]:
```
cars_v1['x'] = cars_v1['x'].astype(int)
```

In [70]:
```
cars_v1['y'] = cars_v1['y'].astype(float)
```

In [77]:
```
cars_v1.loc[cars_v1['y'].isnull() , 'y'] = cars_v1['x']
```

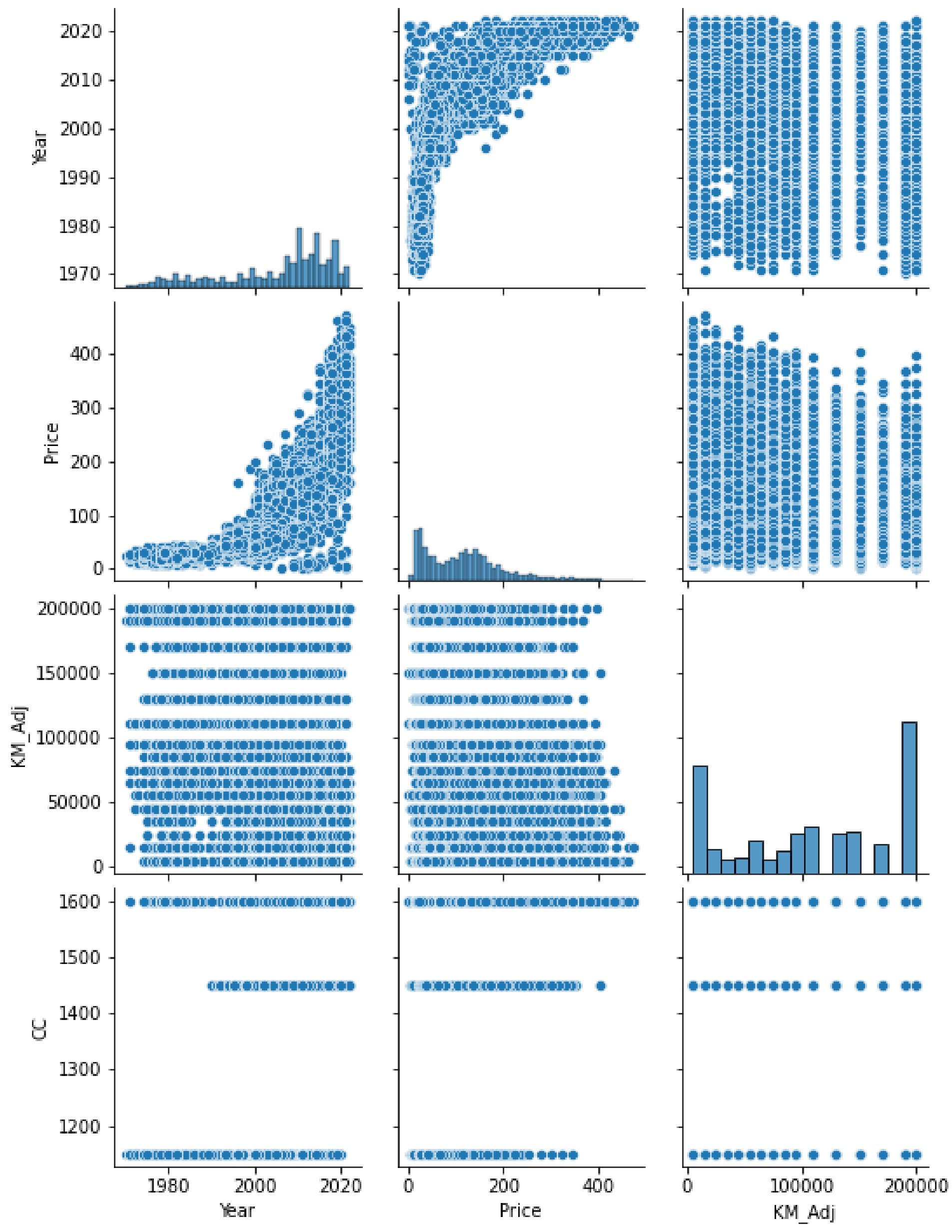
In [79]:
```
cars_v1['CC'] = cars_v1[['x', 'y']].mean(axis=1)
```

In [81]:
```
cars_v1.drop(['x','y'],axis =1 , inplace = True)
```

In [84]:
```
cars_v1.drop(['Unnamed: 0'],axis =1 , inplace = True)
```

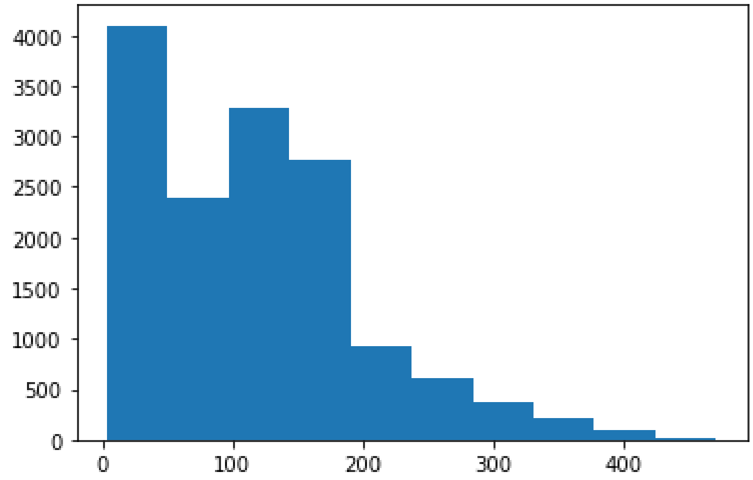
In [85]:
```
sns.pairplot(cars_v1)
```

Out[85]:
```
<seaborn.axisgrid.PairGrid at 0x18c03217fa0>
```



In [86]:
```python
plt.hist(cars_v1['Price'])
```

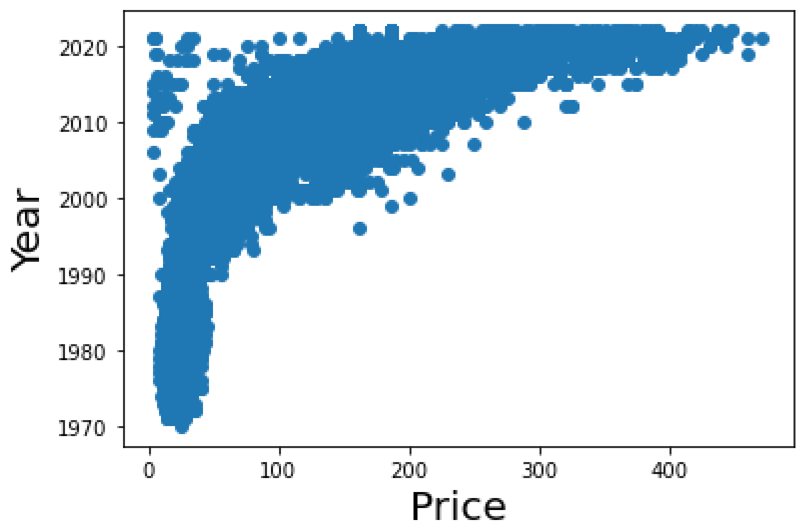
Out[86]:
```
(array([4098., 2394., 3274., 2762.,  921.,  604.,  371.,  204.,   95.,
          18.]),
 array([  3.  ,  49.85,  96.7 , 143.55, 190.4 , 237.25, 284.1 , 330.95,
        377.8 , 424.65, 471.5 ]),
 <BarContainer object of 10 artists>)
```



Starting the regression

In [88]:
```
y= cars_v1['Price']
x1 = cars_v1['Year']
```

In [89]:
```
plt.scatter(y,x1)
plt.xlabel('Price',fontsize= 20)
plt.ylabel('Year',fontsize= 20)
plt.show()
```



In [90]:
```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[90]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.600
Model:	OLS	**Adj. R-squared:**	0.599
Method:	Least Squares	**F-statistic:**	2.206e+04
Date:	Thu, 10 Nov 2022	**Prob (F-statistic):**	0.00
Time:	12:48:49	**Log-Likelihood:**	-79165.
No. Observations:	14741	**AIC:**	1.583e+05
Df Residuals:	14739	**BIC:**	1.584e+05
Df Model:	1		
Covariance Type:	nonrobust		

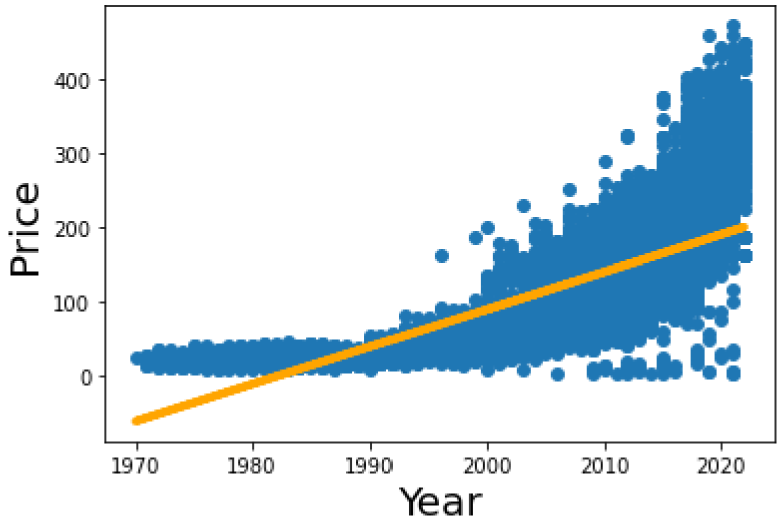
	coef	std err	t	P>|t|	[0.025	0.975]
const	-9968.2243	67.894	-146.821	0.000	-1.01e+04	-9835.144
Year	5.0287	0.034	148.541	0.000	4.962	5.095

Omnibus:	3085.001	**Durbin-Watson:**	0.913
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	7142.836
Skew:	1.183	**Prob(JB):**	0.00
Kurtosis:	5.455	**Cond. No.**	3.18e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.18e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [91]:

```
plt.scatter(x1,y)
y_hat = -9968.2243 + 5.0287* cars_v1['Year']
fig = plt.plot(x1, y_hat, lw=4, c= 'orange')
plt.ylabel('Price', fontsize= 20)
plt.xlabel('Year', fontsize= 20)
plt.show()
```



In [92]:
```
y= cars_v1['Price']
x1 = cars_v1[['Year', 'KM_Adj']]
```

In [93]:
```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[93]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.601
Model:	OLS	**Adj. R-squared:**	0.601
Method:	Least Squares	**F-statistic:**	1.110e+04
Date:	Thu, 10 Nov 2022	**Prob (F-statistic):**	0.00
Time:	13:07:33	**Log-Likelihood:**	-79137.
No. Observations:	14741	**AIC:**	1.583e+05
Df Residuals:	14738	**BIC:**	1.583e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-9842.2165	69.780	-141.045	0.000	-9978.995	-9705.438
Year	4.9684	0.035	143.121	0.000	4.900	5.036
KM_Adj	-4.757e-05	6.29e-06	-7.567	0.000	-5.99e-05	-3.52e-05

Omnibus:	2957.006	**Durbin-Watson:**	0.914
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6662.742
Skew:	1.148	**Prob(JB):**	0.00
Kurtosis:	5.361	**Cond. No.**	2.11e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.11e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

Test OLS Assumptions

In [94]:
```
cars_v1['cairo'] = cars_v1['Gov']
```

In [95]:
```
cars_v1.loc[cars_v1['cairo'] == 'Cairo' , 'cairo'] = 'Cairo'
```

In [96]:
```
cars_v1.loc[cars_v1['cairo'] != 'Cairo' , 'cairo'] = 'Not_Cairo'
```

In [97]:
```
cars_v1['cairo'].value_counts()
```

Out[97]:
```
Not_Cairo    10283
Cairo         4458
Name: cairo, dtype: int64
```

In [98]:
```
cars_v1['cairo'] = cars_v1['cairo'].map({'Cairo':1,"Not_Cairo":0})
```

In [99]:
```
cars_v1['cairo'].value_counts()
```

Out[99]:
```
0    10283
1     4458
Name: cairo, dtype: int64
```

In [100]:
```
y= cars_v1['Price']
x1 = cars_v1[['Year', 'cairo']]
```

In [101]:
```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[101]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.600
Model:	OLS	**Adj. R-squared:**	0.599
Method:	Least Squares	**F-statistic:**	1.103e+04
Date:	Thu, 10 Nov 2022	**Prob (F-statistic):**	0.00
Time:	13:49:43	**Log-Likelihood:**	-79165.
No. Observations:	14741	**AIC:**	1.583e+05
Df Residuals:	14738	**BIC:**	1.584e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-9967.9346	67.902	-146.799	0.000	-1.01e+04	-9834.838
Year	5.0285	0.034	148.508	0.000	4.962	5.095
cairo	0.2941	0.933	0.315	0.753	-1.535	2.123

Omnibus:	3084.447	**Durbin-Watson:**	0.913
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	7142.306
Skew:	1.183	**Prob(JB):**	0.00
Kurtosis:	5.455	**Cond. No.**	3.18e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.18e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Not Significant, Try another Factor

In [103]:
```
cars_v1['CC'].value_counts()
```

Out[103]:
```
1600.0    6762
1450.0    4356
1150.0    3623
Name: CC, dtype: int64
```

In [104]:
```
cars_v1['new_CC'] = cars_v1['CC'].map({1600.0:1, 1450.0:0 ,1150.0:0 })
```

In [106]:
```
cars_v1['new_CC'].value_counts()
```

Out[106]:
```
0    7979
1    6762
Name: new_CC, dtype: int64
```

In [107]:
```
y= cars_v1['Price']
x1 = cars_v1[['Year', 'new_CC']]
```

In [108]:
```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[108]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.662
Model:	OLS	**Adj. R-squared:**	0.662
Method:	Least Squares	**F-statistic:**	1.441e+04
Date:	Thu, 10 Nov 2022	**Prob (F-statistic):**	0.00
Time:	13:53:26	**Log-Likelihood:**	-77923.
No. Observations:	14741	**AIC:**	1.559e+05
Df Residuals:	14738	**BIC:**	1.559e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-8630.5232	67.498	-127.863	0.000	-8762.828	-8498.218
Year	4.3515	0.034	129.001	0.000	4.285	4.418
new_CC	44.5645	0.857	52.020	0.000	42.885	46.244

Omnibus:	3154.650	**Durbin-Watson:**	1.077
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8964.914
Skew:	1.128	**Prob(JB):**	0.00
Kurtosis:	6.083	**Cond. No.**	3.44e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.44e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Car CC is significant, maybe without combining is better, but now we are trying the binary

In [109]:
```
new_data =pd.DataFrame({'const':1, 'Year':[1990, 2015],'new_CC':[0,1]})
new_data = new_data[['const','Year','new_CC']]
new_data
```

Out[109]:

	const	Year	new_CC
0	1	1990	0
1	1	2015	1

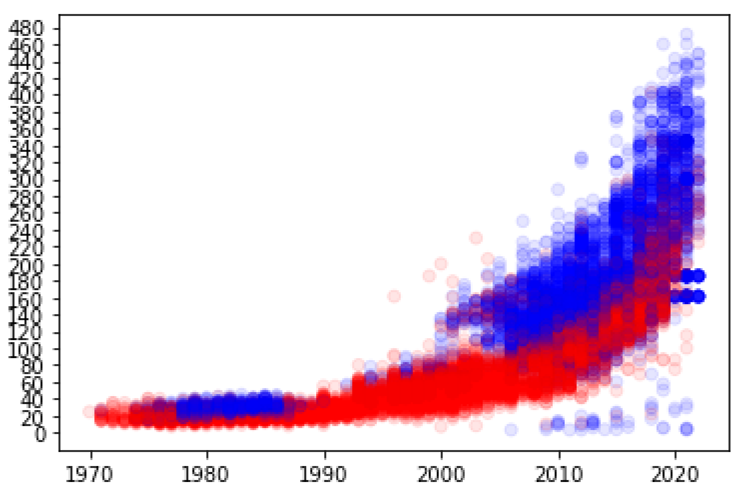
In [111]:
```
predections = results.predict(new_data)
predections
```

Out[111]:
```
0     28.882579
1    182.233613
dtype: float64
```

In [124]:
```
def pltcolor(lst):
    cols=[]
    for l in lst:
        if l==0:
            cols.append('red')
        elif l==1:
            cols.append('blue')
    return cols
# Create the colors list using the function above
cols=pltcolor(cars_v1['new_CC'])
```

In [128]:
```
plt.yticks(np.arange(0, 500, step=20))
plt.scatter(cars_v1['Year'],cars_v1['Price'],alpha = .1, c=cols)
```

Out[128]:
```
<matplotlib.collections.PathCollection at 0x18c0b764ac0>
```



Using SKLearn

In [129]:
```
from sklearn.linear_model import LinearRegression
```

In [130]:
```python
x = cars_v1['Year']
y = cars_v1['Price']
```

In [132]:
```python
x.shape, y.shape
```

Out[132]:
```
((14741,), (14741,))
```

In [137]:
```python
x_matrix = x.values.reshape(-1,1)
```

In [138]:
```python
reg = LinearRegression()
```

In [140]:
```python
reg.fit(x_matrix,y)
```

Out[140]:
```
LinearRegression()
```

In [141]:
```python
reg.score(x_matrix, y)
```

Out[141]:
```
0.5995221119507614
```

In [143]:
```python
reg.coef_
```

Out[143]:
```
array([5.02868432])
```

In [144]:
```python
reg.intercept_
```

Out[144]:
```
-9968.224279816577
```

In [153]:
```python
reg.predict([[2023]])
```

Out[153]:
```
array([204.80409602])
```

In [154]:
```python
new_data = pd.DataFrame(data= [2000,2003,2005],columns =['Year'])
new_data
```

Out[154]:

	Year
0	2000
1	2003
2	2005

In [155]:
```python
reg.predict(new_data)
```

```
D:\Data_Science\Anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fea
ture names, but LinearRegression was fitted without feature names
  warnings.warn(
```

Out[155]:
```
array([ 89.1443567 , 104.23040966, 114.28777829])
```

In [156]:
```python
new_data['Predectied_Price']= reg.predict(new_data)
new_data
```

```
D:\Data_Science\Anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fea
ture names, but LinearRegression was fitted without feature names
  warnings.warn(
```

Out[156]:

	Year	Predectied_Price
0	2000	89.144357
1	2003	104.230410
2	2005	114.287778

In [157]:

```
x = cars_v1[['Year','new_CC']]
y = cars_v1['Price']
```

In [158]:

```
reg= LinearRegression()
reg.fit(x,y)
```

Out[158]:

```
LinearRegression()
```

In [159]:

```
reg.coef_
```

Out[159]:

```
array([ 4.3514602 , 44.56452889])
```

In [160]:

```
reg.intercept_
```

Out[160]:

```
-8630.523223118082
```

Formula for adjusted R^2

$R^2_{adj.} = 1 - (1 - R^2) * \frac{n-1}{n-p-1}$

In [164]:

```
r2 = reg.score(x,y)

n = x.shape[0]

p = x.shape[1]

adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[164]:

```
0.6616009314286118
```

Feature selection

In [166]:

```
from sklearn.feature_selection import f_regression
```

In [168]:

```
f_regression(x,y)
# 0 : f statistics
# 1 : p value
p_values = f_regression(x,y)[1]
p_values.round(3)
# P Value should be less than 5% to be significant
```

Out[168]:
```
array([0., 0.])
```

Standrization

In [169]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [170]:
```
scaler.fit(x)
```

Out[170]:
```
StandardScaler()
```

In [179]:
```
x_scalled = scaler.transform(x)
```

Regression with Scalled features

In [180]:
```
reg= LinearRegression()
reg.fit(x_scalled,y)
```

Out[180]:
```
LinearRegression()
```

In [182]:
```
reg.coef_
```

Out[182]:
```
array([55.06832355, 22.20619705])
```

In [183]:
```
reg.intercept_
```

Out[183]:
```
116.5849874499691
```

In [186]:
```
r2 = reg.score(x_scalled,y)
n = x_scalled.shape[0]
p = x_scalled.shape[1]

adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[186]:
```
0.6616009314286119
```

In [189]:
```
reg_summary = pd.DataFrame([['Intercept'],['Year'],["new_CC"]],columns=['features'])
reg_summary['weight'] = reg.intercept_, reg.coef_[0],reg.coef_[1]
```

In [190]:
```
reg_summary
```

Out[190]:

	features	weight
0	Intercept	116.584987
1	Year	55.068324
2	new_CC	22.206197

In [197]:
```
new_data2 =pd.DataFrame({'Year':[1990, 2015,1990,2015],'new_CC':[0,1,1,0]})
new_data2 = new_data2[['Year','new_CC']]
```

```
new_data2
```

Out[197]:

	Year	new_CC
0	1990	0
1	2015	1
2	1990	1
3	2015	0

In [198]:
```
reg.predict(new_data2)
```

```
D:\Data_Science\Anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fea
ture names, but LinearRegression was fitted without feature names
  warnings.warn(
```

Out[198]:
```
array([109702.54885003, 111101.46313581, 109724.75504708, 111079.25693875])
```

In [199]:
```
new_data2_scalled= scaler.transform(new_data2)
```

In [200]:
```
reg.predict(new_data2_scalled)
```

Out[200]:
```
array([ 28.88257921, 182.23361315,  73.44710809, 137.66908426])
```

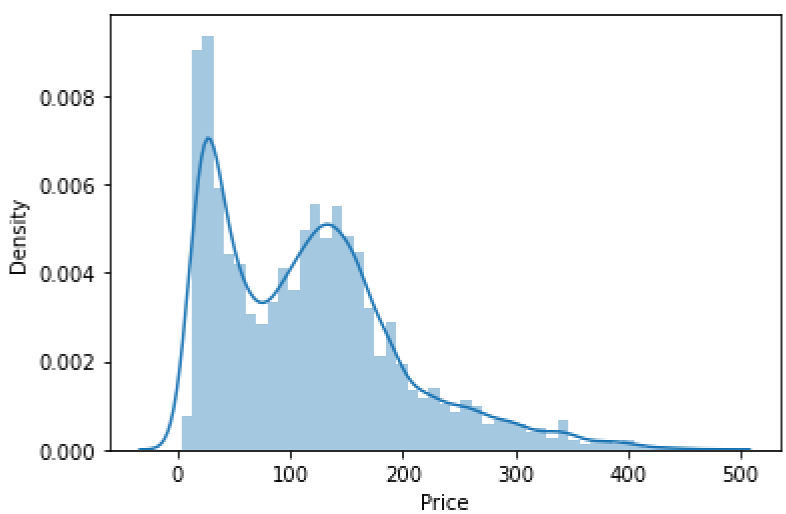
In [ ]:
```

```

In [217]:
```
sns.distplot(cars_v1['Price'])
```

```
D:\Data_Science\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[217]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```

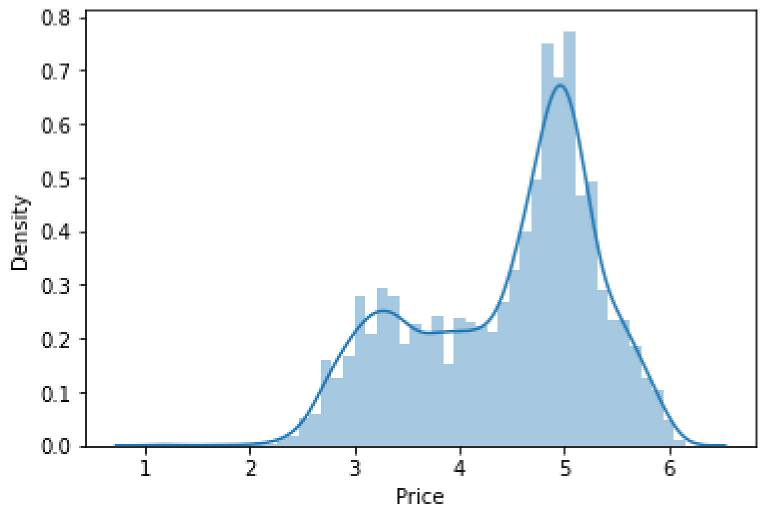


In [218]:
```
price_log = np.log(cars_v1['Price'])
```

In [219]:
```
sns.distplot(price_log)
```

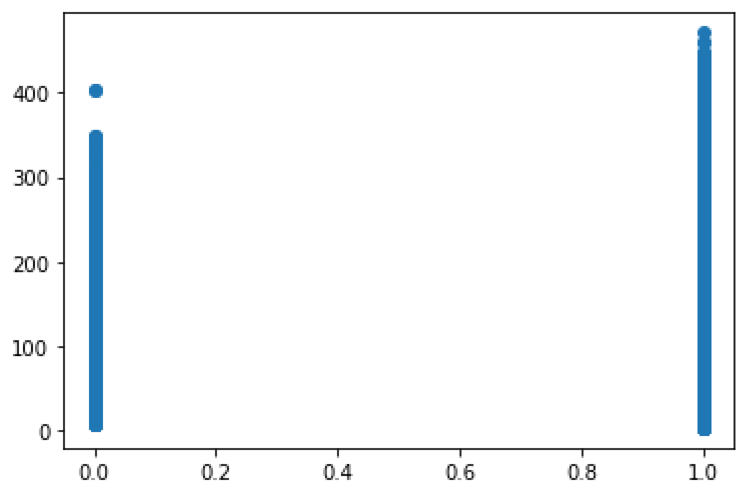
```
D:\Data_Science\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[219]: <AxesSubplot:xlabel='Price', ylabel='Density'>



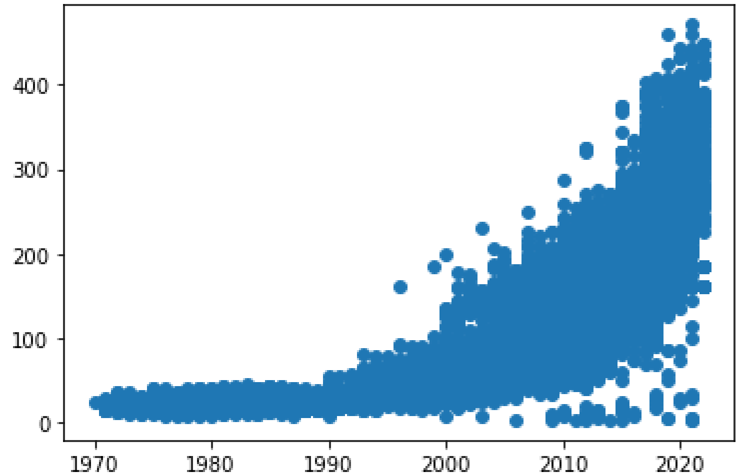
In [228]:
```
plt.scatter(cars_v1['new_CC'],cars_v1['Price'])
```

Out[228]: <matplotlib.collections.PathCollection at 0x18c0e1abb20>



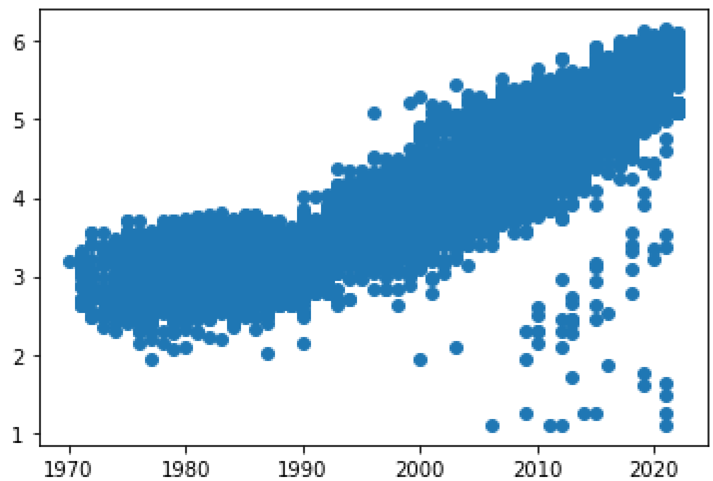
In [231]:
```
plt.scatter(cars_v1['Year'],cars_v1['Price'])
```

Out[231]: <matplotlib.collections.PathCollection at 0x18c0e2d37c0>



In [230]:
```
plt.scatter(cars_v1['Year'],price_log)
```

Out[230]:
```
<matplotlib.collections.PathCollection at 0x18c0e18e130>
```



Multicollinearity

In [232]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [234]:
```
cars_v1.columns.values
```

Out[234]:
```
array(['Brand', 'Model', 'Body', 'Color', 'Year', 'Fuel', 'Engine',
       'Transmission', 'Price', 'Gov', 'KM_Adj', 'CC', 'cairo', 'new_CC'],
      dtype=object)
```

In [241]:
```
variables = cars_v1[['Year', 'Price','KM_Adj', 'new_CC']]
```

In [242]:
```
vif = pd.DataFrame()
```

In [243]:
```
vif['VIF'] = [variance_inflation_factor(variables.values,i) for i in range(variables .sl
```

In [244]:
```
vif['features'] = variables.columns
```

In [246]:
```
#values between 1 and 5 are good
# 10 is not acceptable
```

```
vif
```

Out[246]:

	VIF	features
0	6.782124	Year
1	4.369353	Price
2	3.582898	KM_Adj
3	2.564681	new_CC

In [251]:
```
cars_v1.describe(include='all')
```

Out[251]:

	Brand	Model	Body	Color	Year	Fuel	Engine	Transmission	Price	Go
count	14741	14741	14741	14741	14741.000000	14741	14741	14741	14741.000000	1474
unique	3	18	3	14	NaN	2	3	2	NaN	2
top	Hyundai	128	Sedan	White	NaN	Benzine	1600 CC	Manual	NaN	Cair
freq	5692	2425	13453	2614	NaN	14200	6762	9862	NaN	445
mean	NaN	NaN	NaN	NaN	2005.456821	NaN	NaN	NaN	116.584987	Na
std	NaN	NaN	NaN	NaN	12.655566	NaN	NaN	NaN	82.192718	Na
min	NaN	NaN	NaN	NaN	1970.000000	NaN	NaN	NaN	3.000000	Na
25%	NaN	NaN	NaN	NaN	1998.000000	NaN	NaN	NaN	43.700000	Na
50%	NaN	NaN	NaN	NaN	2010.000000	NaN	NaN	NaN	110.000000	Na
75%	NaN	NaN	NaN	NaN	2015.000000	NaN	NaN	NaN	161.000000	Na
max	NaN	NaN	NaN	NaN	2022.000000	NaN	NaN	NaN	471.500000	Na

Get Dummies

In [252]:
```
cars_dummies = pd.get_dummies(cars_v1,prefix=['Brand', 'Body'], columns=['Brand', 'Body
```

In [258]:
```
cars_dummies.drop(['cairo','new_CC'],axis= 1, inplace= True)
```

In [260]:
```
cars_dummies['log_price']= np.log(cars_dummies['Price'])
```

In [262]:
```
cars_dummies.columns
```

Out[262]:
```
Index(['Model', 'Color', 'Year', 'Fuel', 'Engine', 'Transmission', 'Price',
       'Gov', 'KM_Adj', 'CC', 'Brand_Fiat', 'Brand_Hyundai', 'Body_SUV',
       'Body_Sedan', 'log_price'],
      dtype='object')
```

Linear Regression

In [264]:
```
targets = cars_dummies['log_price']
```

```
inputs = cars_dummies.drop(['Model', 'Color','Fuel', 'Engine', 'Transmission', 'Price',
```

In [265]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [266]:
```
scaler.fit(inputs)
```

Out[266]:
```
StandardScaler()
```

In [267]:
```
inputs_scaled = scaler.transform(inputs)
```

In [ ]:

Train Test Split

In [220]:
```
from sklearn.model_selection import train_test_split
```

In [268]:
```
X_train, X_test, y_train, y_test = train_test_split(inputs_scaled, targets, test_size=
```

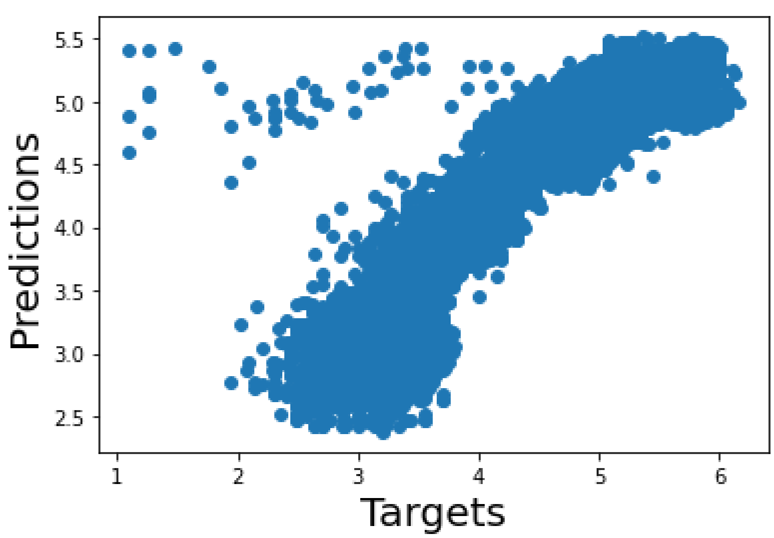
In [269]:
```
reg = LinearRegression()
```

In [270]:
```
reg.fit(X_train,y_train)
```

Out[270]:
```
LinearRegression()
```

In [272]:
```
y_hat = reg.predict(X_train)
```

In [275]:
```
plt.scatter(y_train,y_hat)
plt.xlabel('Targets', fontsize= 20)
plt.ylabel('Predictions', fontsize= 20)
plt.show()
```

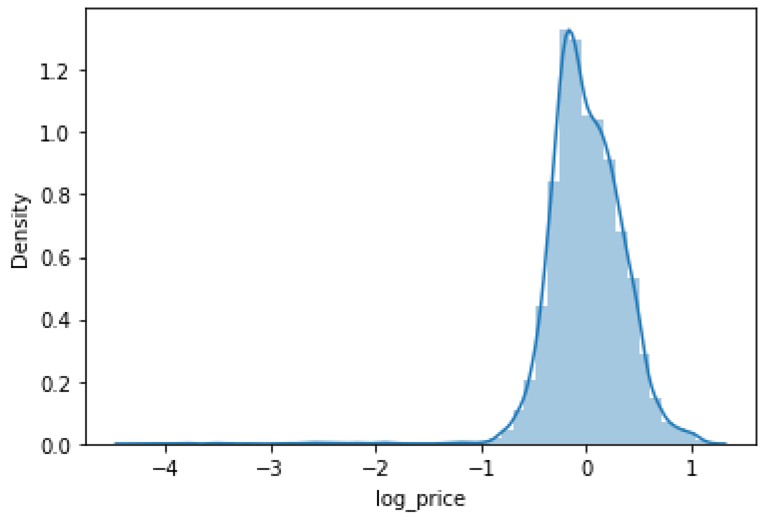


In [276]:
```
sns.distplot(y_train - y_hat)

#Diff are normal with mean of zero
```

```
D:\Data_Science\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[276]: <AxesSubplot:xlabel='log_price', ylabel='Density'>



In [278]:
```
reg.score(X_train, y_train)
```

Out[278]: 0.827772794127398

In [281]:
```
reg.intercept_
```

Out[281]: 4.449310558911252

In [282]:
```
reg.coef_
```

Out[282]:
```
array([ 0.63839467, -0.02108024, -0.15173085,  0.04714382, -0.03700554,
       -0.05535697])
```

In [283]:
```
reg_sum = pd.DataFrame(inputs.columns.values, columns =['features'])
reg_sum['weights'] = reg.coef_
reg_sum
```

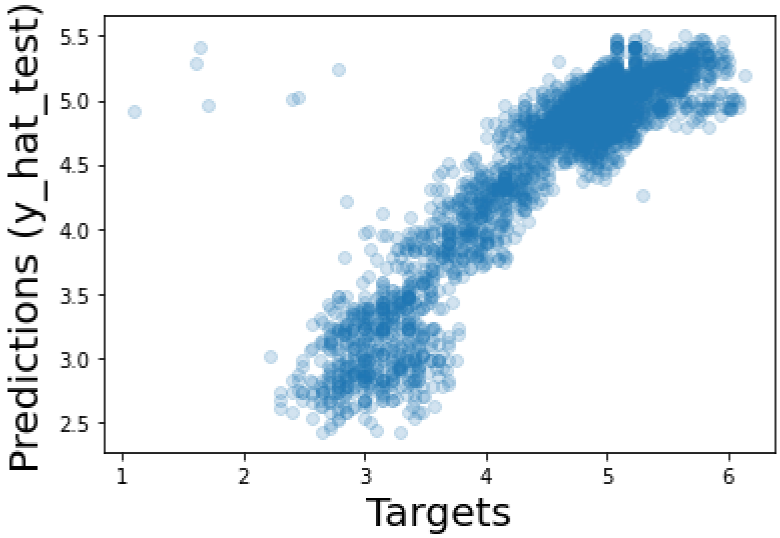
Out[283]:

	features	weights
0	Year	0.638395
1	KM_Adj	-0.021080
2	Brand_Fiat	-0.151731
3	Brand_Hyundai	0.047144
4	Body_SUV	-0.037006
5	Body_Sedan	-0.055357

Testing

In [285]:
```
y_hat_test = reg.predict(X_test)
```

In [287]:
```
plt.scatter(y_test,y_hat_test, alpha =.2)
plt.xlabel('Targets', fontsize= 20)
plt.ylabel('Predictions (y_hat_test)', fontsize= 20)
plt.show()
```



In [299]:
```
df_pf = pd.DataFrame(np.exp(y_hat_test), columns =['Prediction'])
df_pf
```

Out[299]:

	Prediction
0	194.433879
1	179.646652
2	173.143321
3	95.367831
4	48.374679
...	...
2944	106.130016
2945	101.903494
2946	132.192594
2947	51.685648
2948	194.231250

2949 rows × 1 columns

In [300]:
```
y_test= y_test.reset_index(drop=True)
```

In [301]:
```
df_pf['Target'] = np.exp(y_test)
```

In [302]:
```
df_pf.head()
```

Out[302]:

	Prediction	Target
0	194.433879	258.8
1	179.646652	212.8
2	173.143321	149.5
3	95.367831	130.0
4	48.374679	43.0

In [303]:

```
df_pf['Residual']= df_pf['Target'] - df_pf['Prediction']
```

In [304]:

```
df_pf
```

Out[304]:

	Prediction	Target	Residual
0	194.433879	258.8	64.366121
1	179.646652	212.8	33.153348
2	173.143321	149.5	-23.643321
3	95.367831	130.0	34.632169
4	48.374679	43.0	-5.374679
...	...	...	...
2944	106.130016	135.0	28.869984
2945	101.903494	140.0	38.096506
2946	132.192594	85.0	-47.192594
2947	51.685648	46.0	-5.685648
2948	194.231250	264.5	70.268750

2949 rows × 3 columns

In [319]:

```
df_pf['Diff%'] = np.absolute(df_pf['Residual']/df_pf['Target']*100).round(decimals=2)
df_pf
```

Out[319]:

	Prediction	Target	Residual	Diff%
0	194.433879	258.8	64.366121	24.87
1	179.646652	212.8	33.153348	15.58
2	173.143321	149.5	-23.643321	15.81
3	95.367831	130.0	34.632169	26.64
4	48.374679	43.0	-5.374679	12.50
...	...	...	...	...
2944	106.130016	135.0	28.869984	21.39
2945	101.903494	140.0	38.096506	27.21
2946	132.192594	85.0	-47.192594	55.52
2947	51.685648	46.0	-5.685648	12.36
2948	194.231250	264.5	70.268750	26.57

2949 rows × 4 columns

In [320]:
```
df_pf.describe()
```

Out[320]:

	Prediction	Target	Residual	Diff%
count	2949.000000	2949.000000	2949.000000	2949.000000
mean	109.970049	119.555680	9.585631	31.897986
std	58.660083	83.287642	48.077432	147.508638
min	11.273800	3.000000	-220.185765	0.020000
25%	53.996347	46.000000	-16.945430	11.270000
50%	122.323310	115.000000	-0.258770	21.640000
75%	154.416791	161.400000	21.903266	34.860000
max	245.541000	460.000000	302.881023	4478.830000

In [ ]:
```
#pd.options.display.max_rows = 999
#pd.set_options('display.float_format', lambda x:"%.2f%" % x)
```

In [321]:
```
f_regression(X_test,y_test)
```

Out[321]:
```
(array([1.27346179e+04, 1.62972740e+02, 3.82886607e+03, 5.96788555e+02,
        1.75220808e+01, 3.98576639e+00]),
 array([0.00000000e+000, 2.28937348e-036, 0.00000000e+000, 3.50056158e-120,
        2.92302443e-005, 4.59780673e-002]))
```

In [ ]:
```

```

Using All Data

Get Dummies

In [329]:
```
cars_v1.head()
cars_v2= cars_v1.drop(['CC',"cairo",'new_CC'],axis=1)
```

In [330]:
```

```

In [331]:
```
cars_v2['log_price']= np.log(cars_v2['Price'])
```

In [332]:
```
cars_v2.drop(['Price'],axis= 1, inplace= True)
```

In [338]:
```
cars_v2 = pd.get_dummies(cars_v2, drop_first=True)
```

In [ ]:
```

```

In [347]:
```
targets = cars_v2['log_price']
inputs = cars_v2.drop(['log_price'], axis= 1)
```

In [348]:
```
inputs.head()
```

Out[348]:

	Year	KM_Adj	Brand_Fiat	Brand_Hyundai	Model_131	Model_Accent	Model_Avante	Model_Aveo
0	2007	149999.5	0	1	0	1	0	0
1	2005	189999.5	0	1	0	1	0	0
2	1999	149999.5	0	1	0	1	0	0
3	2009	149999.5	0	1	0	1	0	0
4	2000	14999.5	0	1	0	1	0	0

5 rows × 65 columns

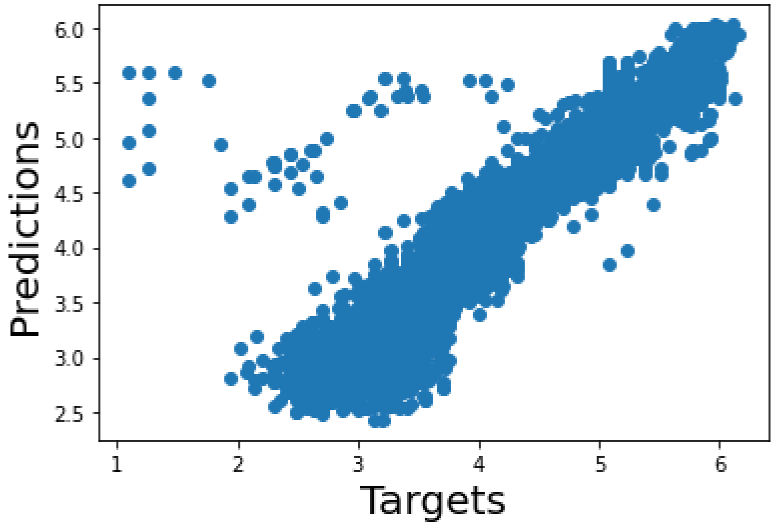
In [349]:
```
scaler = StandardScaler()
scaler.fit(inputs)
inputs_scaled = scaler.transform(inputs)
```

In [350]:
```
X_train, X_test, y_train, y_test = train_test_split(inputs_scaled, targets, test_size=
reg = LinearRegression()
reg.fit(X_train,y_train)
```

Out[350]:
```
LinearRegression()
```

In [351]:
```
y_hat = reg.predict(X_train)
```

In [352]:
```
plt.scatter(y_train,y_hat)
plt.xlabel('Targets', fontsize= 20)
plt.ylabel('Predictions', fontsize= 20)
plt.show()
```



In [353]:
```
reg.score(X_train, y_train)

reg.coef_

reg_sum = pd.DataFrame(inputs.columns.values, columns =['features'])
reg_sum['weights'] = reg.coef_
reg_sum
```

Out[353]:

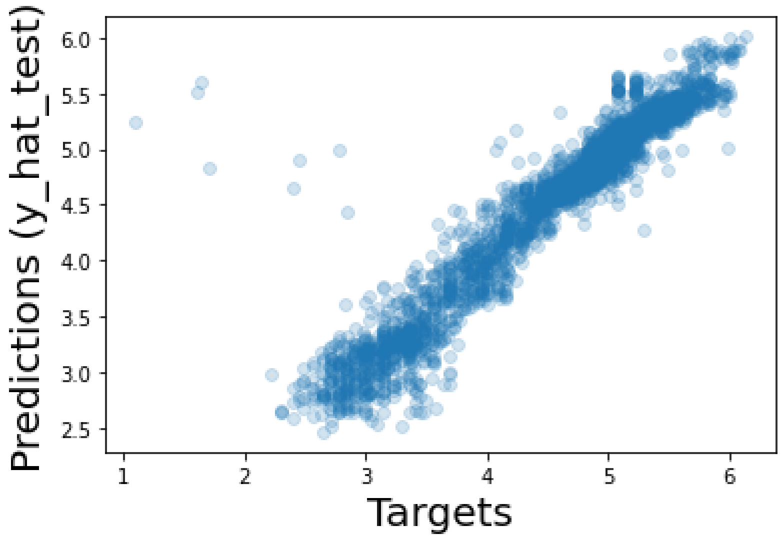
	features	weights
0	Year	4.969610e-01
1	KM_Adj	-1.682569e-03
2	Brand_Fiat	-4.429173e+10
3	Brand_Hyundai	1.400403e+10
4	Model_131	3.452594e-02
...	...	...
60	Gov_Red Sea	7.672787e-03
61	Gov_Sharqia	-1.606464e-03
62	Gov_Sohag	1.432896e-03
63	Gov_South Sinai	1.009703e-03
64	Gov_Suez	-2.652645e-03

65 rows × 2 columns

In [354]:
```
y_hat_test = reg.predict(X_test)
```

In [355]:
```
y_test= y_test.reset_index(drop=True)
```

In [356]:
```
plt.scatter(y_test,y_hat_test, alpha =.2)
plt.xlabel('Targets', fontsize= 20)
plt.ylabel('Predictions (y_hat_test)', fontsize= 20)
plt.show()
```



In [357]:
```python
df_pf = pd.DataFrame(np.exp(y_hat_test), columns =['Prediction'])
df_pf
```

Out[357]:

	Prediction
0	225.329036
1	211.546267
2	129.327479
3	114.999917
4	50.128709
...	...
2944	138.513329
2945	139.306055
2946	94.444913
2947	49.638902
2948	234.142508

2949 rows × 1 columns

In [358]:
```python
df_pf['Target'] = np.exp(y_test)
```

In [359]:
```python
df_pf.head()
```

Out[359]:

	Prediction	Target
0	225.329036	258.8
1	211.546267	212.8
2	129.327479	149.5
3	114.999917	130.0
4	50.128709	43.0

In [360]:
```python
df_pf['Residual']= df_pf['Target'] - df_pf['Prediction']

df_pf['Diff%'] = np.absolute(df_pf['Residual']/df_pf['Target']*100).round(decimals=2)
df_pf
```

Out[360]:

	Prediction	Target	Residual	Diff%
0	225.329036	258.8	33.470964	12.93
1	211.546267	212.8	1.253733	0.59
2	129.327479	149.5	20.172521	13.49
3	114.999917	130.0	15.000083	11.54
4	50.128709	43.0	-7.128709	16.58
...	...	...	...	...
2944	138.513329	135.0	-3.513329	2.60
2945	139.306055	140.0	0.693945	0.50
2946	94.444913	85.0	-9.444913	11.11
2947	49.638902	46.0	-3.638902	7.91
2948	234.142508	264.5	30.357492	11.48

2949 rows × 4 columns

In [373]:
```python
p_value_table =  pd.DataFrame (data = inputs.columns, columns = ['Features' ])
```

In [375]:
```python
p_value_table['weights'] = f_regression(X_test,y_test)[1]
```

In [378]:
```python
p_value_table.sort_values('weights')
```

Out[378]:

	Features	weights
0	Year	0.000000e+00
2	Brand_Fiat	0.000000e+00
39	Transmission_Manual	3.854971e-315
38	Engine_1600 CC	2.967835e-198
3	Brand_Hyundai	3.500562e-120
...	...	...
24	Color_Blue- Navy Blue	8.544123e-01
62	Gov_Sohag	8.564710e-01
52	Gov_Luxor	8.804858e-01
49	Gov_Giza	9.314950e-01
48	Gov_Gharbia	9.540106e-01

65 rows × 2 columns

In [379]:
```
# Gov is not Relevant, should be removed
```

In [381]:
```
reg.score(X_test, y_test)
```

Out[381]:
```
0.9107993978077813
```

In [382]:
```
# r2 for

r2 = reg.score(X_train,y_train)
n = X_train.shape[0]
p = X_train.shape[1]
adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[382]:
```
0.9050039208760138
```

In [383]:
```
r2 = reg.score(X_test,y_test)

n = X_test.shape[0]
p = X_test.shape[1]

adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[383]:
```
0.9087882846816994
```

In [ ]:

In [ ]:

In [ ]: