In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error
from sklearn.model_selection import train_test_split
import statsmodels.formula.api as smf

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
toyota_data = pd.read_csv('ToyotaCorolla.csv',encoding='latin1')
toyota_data.head()
```

Out[2]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	...	C
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	Diesel	90	1	...	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	Diesel	90	1	...	
2	3	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	Diesel	90	1	...	
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000	Diesel	90	0	...	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	Diesel	90	0	...	

5 rows × 38 columns

In [3]:
```
toyota_data.columns
```

Out[3]:
```
Index(['Id', 'Model', 'Price', 'Age_08_04', 'Mfg_Month', 'Mfg_Year', 'KM',
       'Fuel_Type', 'HP', 'Met_Color', 'Color', 'Automatic', 'cc', 'Doors',
       'Cylinders', 'Gears', 'Quarterly_Tax', 'Weight', 'Mfr_Guarantee',
       'BOVAG_Guarantee', 'Guarantee_Period', 'ABS', 'Airbag_1', 'Airbag_2',
       'Airco', 'Automatic_airco', 'Boardcomputer', 'CD_Player',
       'Central_Lock', 'Powered_Windows', 'Power_Steering', 'Radio',
       'Mistlamps', 'Sport_Model', 'Backseat_Divider', 'Metallic_Rim',
       'Radio_cassette', 'Tow_Bar'],
      dtype='object')
```

In [51]:
```
toyota_data1=toyota_data[['Price','Age_08_04','KM','HP','cc','Doors','Gears','Qua
toyota_data1
```

Out[51]:

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Quarterly_Tax	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

In [52]:
```
toyota_data1= toyota_data1.rename({'Age_08_04':'Age','cc':'CC','Quarterly_Tax':'Q
toyota_data1
```

Out[52]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

In [6]:
```
toyota_data1.shape
```

Out[6]:
```
(1436, 9)
```

In [7]:
```
toyota_data1.isna().sum()
```

Out[7]:
```
Price     0
Age       0
KM        0
HP        0
CC        0
Doors     0
Gears     0
QT        0
Weight    0
dtype: int64
```

In [8]:
```
toyota_data1.dtypes
```

Out[8]:
```
Price     int64
Age       int64
KM        int64
HP        int64
CC        int64
Doors     int64
Gears     int64
QT        int64
Weight    int64
dtype: object
```

In [9]:
```
toyota_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Price   1436 non-null   int64
 1   Age     1436 non-null   int64
 2   KM      1436 non-null   int64
 3   HP      1436 non-null   int64
 4   CC      1436 non-null   int64
 5   Doors   1436 non-null   int64
 6   Gears   1436 non-null   int64
 7   QT      1436 non-null   int64
 8   Weight  1436 non-null   int64
dtypes: int64(9)
memory usage: 101.1 KB
```

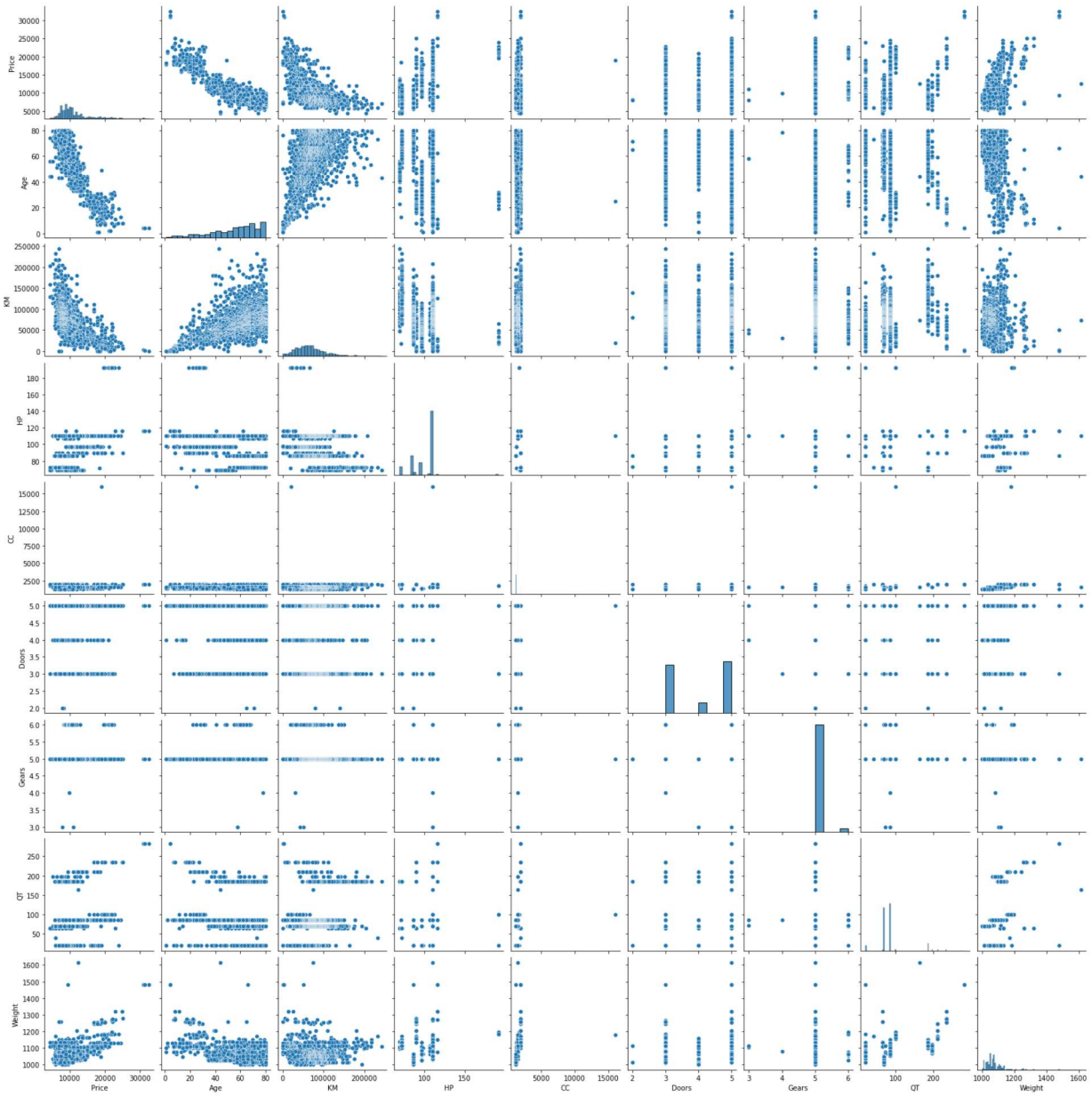
In [10]:
```
toyota_data1.describe(include='all')
```

Out[10]:

	Price	Age	KM	HP	CC	Doors	Ge
count	1436.000000	1436.000000	1436.000000	1436.000000	1436.00000	1436.000000	1436.000
mean	10730.824513	55.947075	68533.259749	101.502089	1576.85585	4.033426	5.026
std	3626.964585	18.599988	37506.448872	14.981080	424.38677	0.952677	0.188
min	4350.000000	1.000000	1.000000	69.000000	1300.00000	2.000000	3.000
25%	8450.000000	44.000000	43000.000000	90.000000	1400.00000	3.000000	5.000
50%	9900.000000	61.000000	63389.500000	110.000000	1600.00000	4.000000	5.000
75%	11950.000000	70.000000	87020.750000	110.000000	1600.00000	5.000000	5.000
max	32500.000000	80.000000	243000.000000	192.000000	16000.00000	5.000000	6.000

In [11]:
```
plt.figure(figsize=(10,8))
sns.pairplot(toyota_data1)
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```

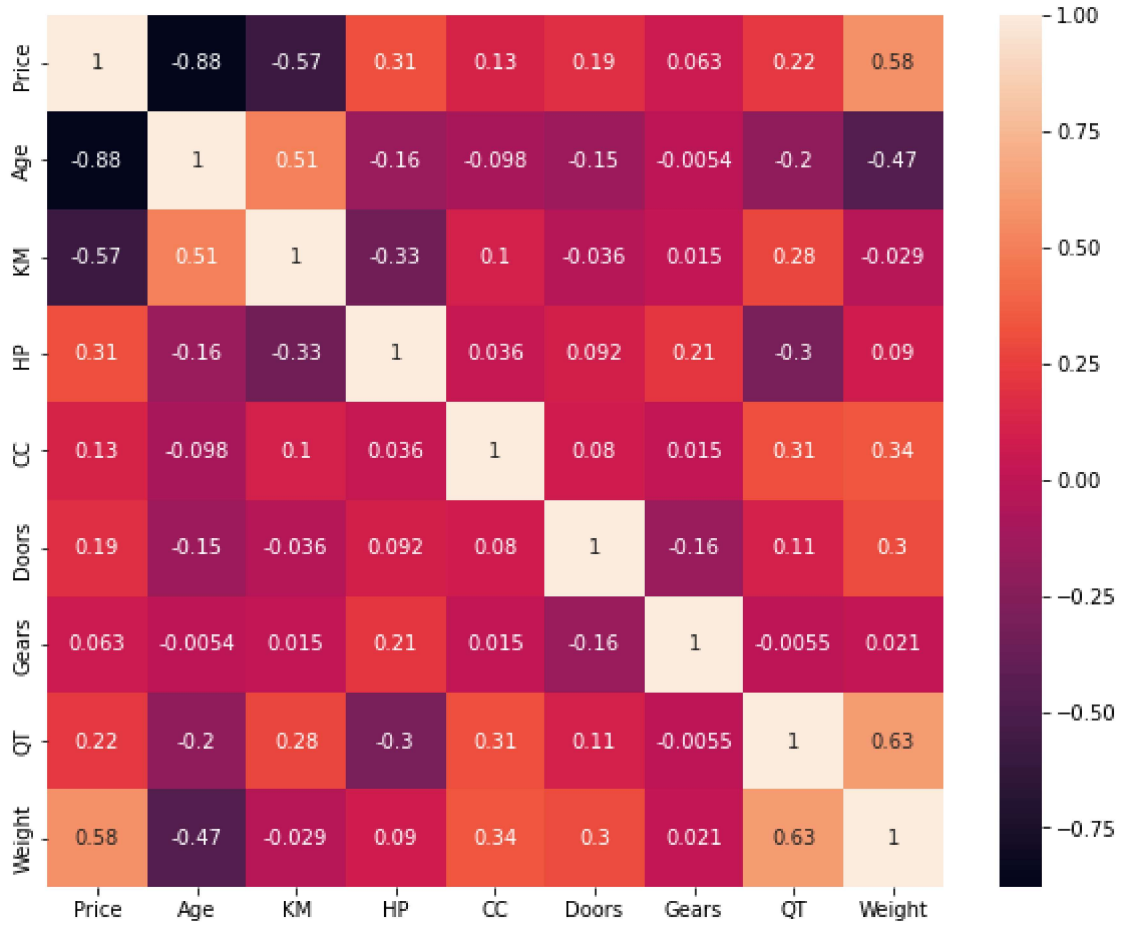


In [12]:
```
corr_matrix = toyota_data1.corr()
corr_matrix
```

Out[12]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	W
Price	1.000000	-0.876590	-0.569960	0.314990	0.126389	0.185326	0.063104	0.219197	0.58
Age	-0.876590	1.000000	0.505672	-0.156622	-0.098084	-0.148359	-0.005364	-0.198431	-0.47
KM	-0.569960	0.505672	1.000000	-0.333538	0.102683	-0.036197	0.015023	0.278165	-0.02
HP	0.314990	-0.156622	-0.333538	1.000000	0.035856	0.092424	0.209477	-0.298432	0.08
CC	0.126389	-0.098084	0.102683	0.035856	1.000000	0.079903	0.014629	0.306996	0.33
Doors	0.185326	-0.148359	-0.036197	0.092424	0.079903	1.000000	-0.160141	0.109363	0.30
Gears	0.063104	-0.005364	0.015023	0.209477	0.014629	-0.160141	1.000000	-0.005452	0.02
QT	0.219197	-0.198431	0.278165	-0.298432	0.306996	0.109363	-0.005452	1.000000	0.62
Weight	0.581198	-0.470253	-0.028598	0.089614	0.335637	0.302618	0.020613	0.626134	1.00

In [13]:
```
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix,annot=True)
plt.show()
```



In [ ]:
```
x = toyota_data1.drop('Price',axis=1)
y = toyota_data1[['Price']]
```

In [15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,shuffle=True,
```

In [16]:
```
x_train.shape,x_test.shape
```

Out[16]:
```
((1148, 8), (288, 8))
```

In [17]:
```
linear_model = LinearRegression()
linear_model.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
linear_model.coef_
```

Out[18]:
```
array([[-1.20859866e+02, -2.14109713e-02,  3.15722985e+01,
        -1.27693132e-01,  1.20005776e+01,  5.14603172e+02,
         3.43610273e+00,  1.80917460e+01]])
```

In [19]:
```
linear_model.intercept_
```

Out[19]:
```
array([-6356.88165821])
```

In [20]:
```
y_pred_train = linear_model.predict(x_train)
y_pred_train
```

Out[20]:
```
array([[ 7895.35460929],
       [11297.67906071],
       [13379.52087103],
       ...,
       [ 7457.09725548],
       [ 8151.14408175],
       [ 8182.17162762]])
```

In [21]:
```
error = y_train - y_pred_train
error
```

Out[21]:

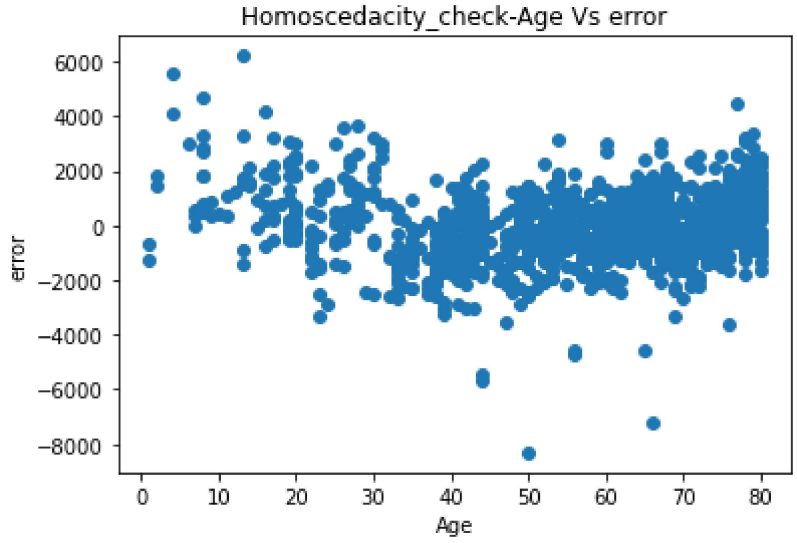
	Price
740	354.645391
1035	1202.320939
329	-1684.520871
1178	-340.932787
730	1599.697052
...	...
432	-371.347831
1283	415.357801
1265	1292.902745
1277	648.855918
1414	-682.171628

1148 rows × 1 columns

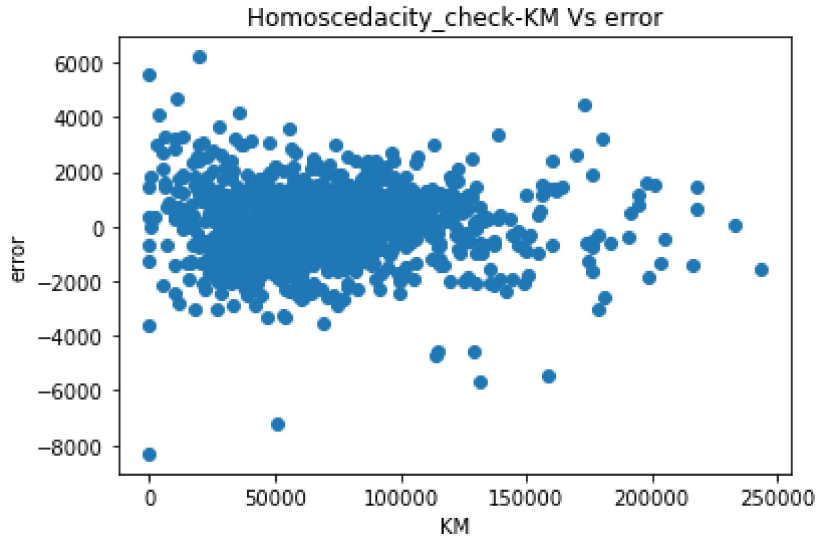
In [22]:
```
mean_absolute_percentage_error(y_train,y_pred_train)
```

Out[22]:
```
0.1009074908217336
```

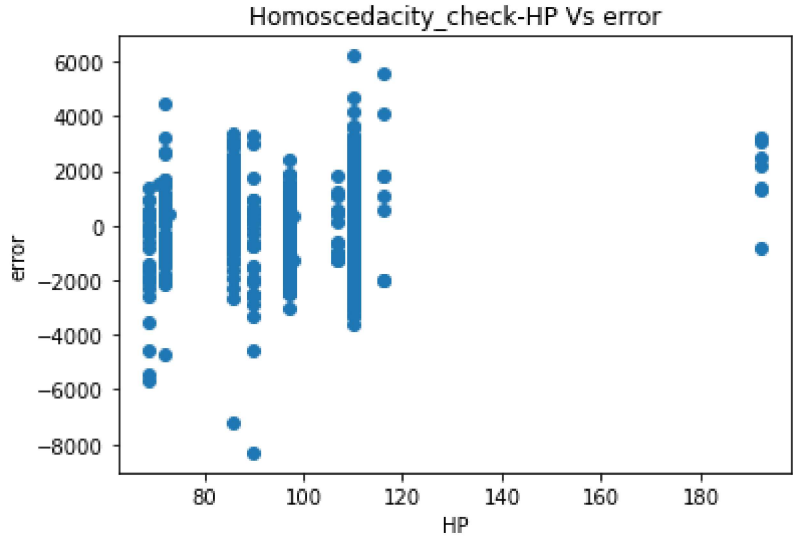
In [23]:
```
plt.scatter(x=x_train['Age'],y=error)
plt.title('Homoscedacity_check-Age Vs error')
plt.xlabel('Age')
plt.ylabel('error')
plt.show()
```



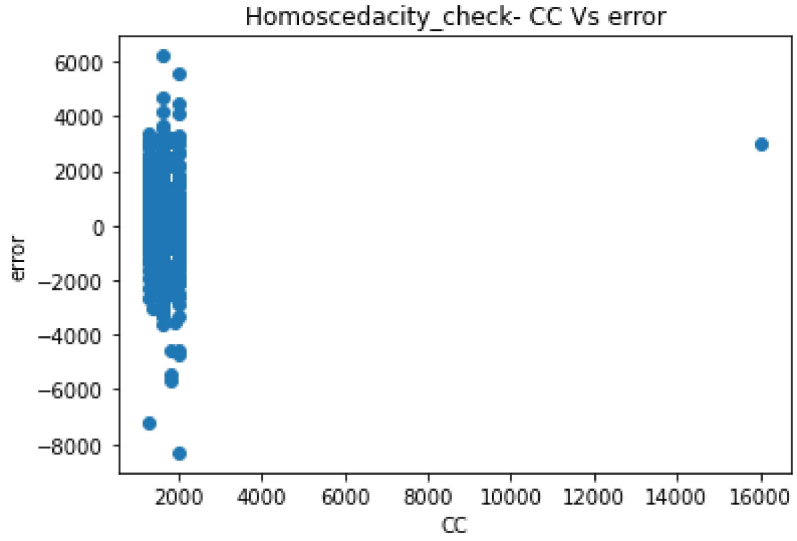
In [24]:
```
plt.scatter(x=x_train['KM'],y=error)
plt.title('Homoscedacity_check-KM Vs error')
plt.xlabel('KM')
plt.ylabel('error')
plt.show()
```



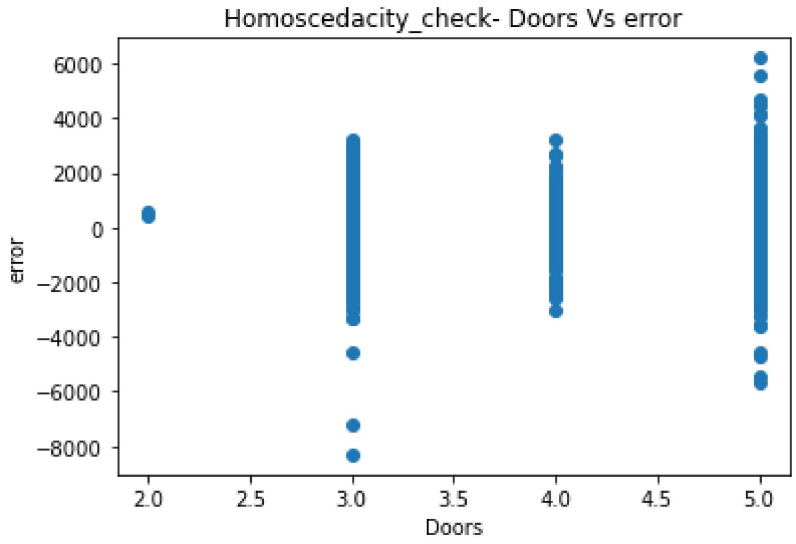
In [25]:
```
plt.scatter(x=x_train['HP'],y=error)
plt.title('Homoscedacity_check-HP Vs error')
plt.xlabel('HP')
plt.ylabel('error')
plt.show()
```



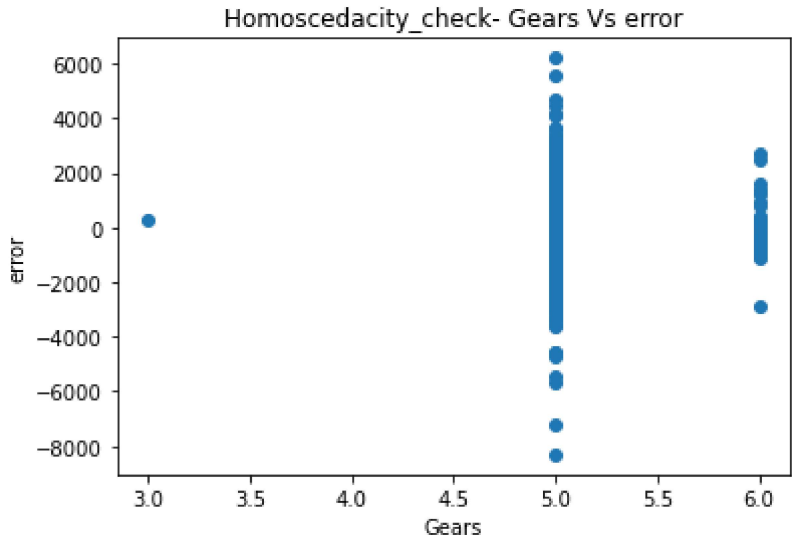
In [26]:
```
plt.scatter(x=x_train['CC'],y=error)
plt.title('Homoscedacity_check- CC Vs error')
plt.xlabel('CC')
plt.ylabel('error')
plt.show()
```



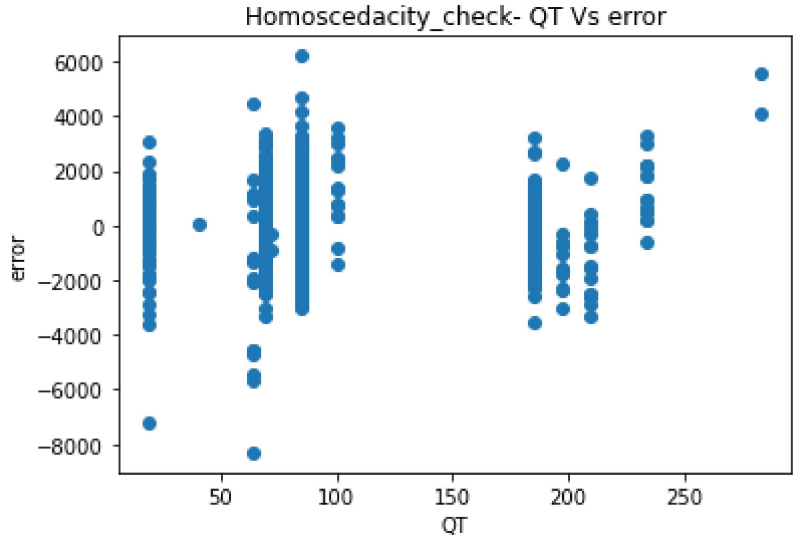
In [27]:
```
plt.scatter(x=x_train['Doors'],y=error)
plt.title('Homoscedacity_check- Doors Vs error')
plt.xlabel('Doors')
plt.ylabel('error')
plt.show()
```



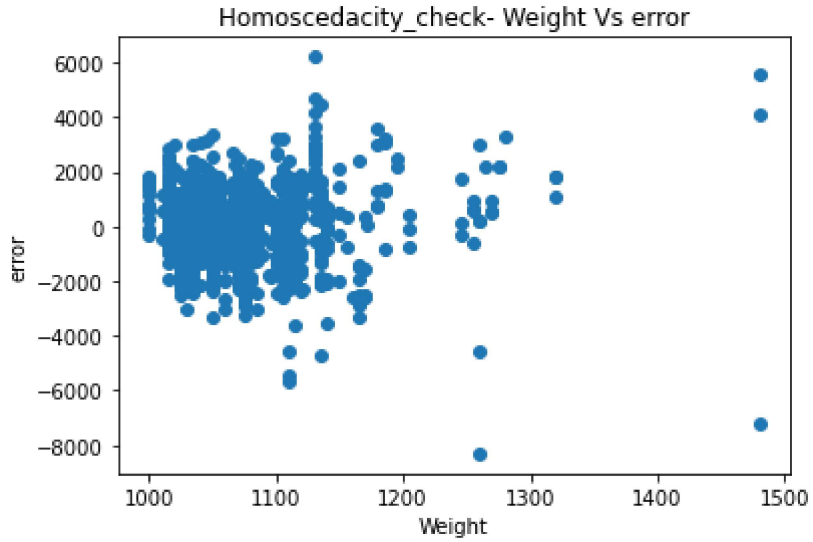
In [28]:
```
plt.scatter(x=x_train['Gears'],y=error)
plt.title('Homoscedacity_check- Gears Vs error')
plt.xlabel('Gears')
plt.ylabel('error')
plt.show()
```



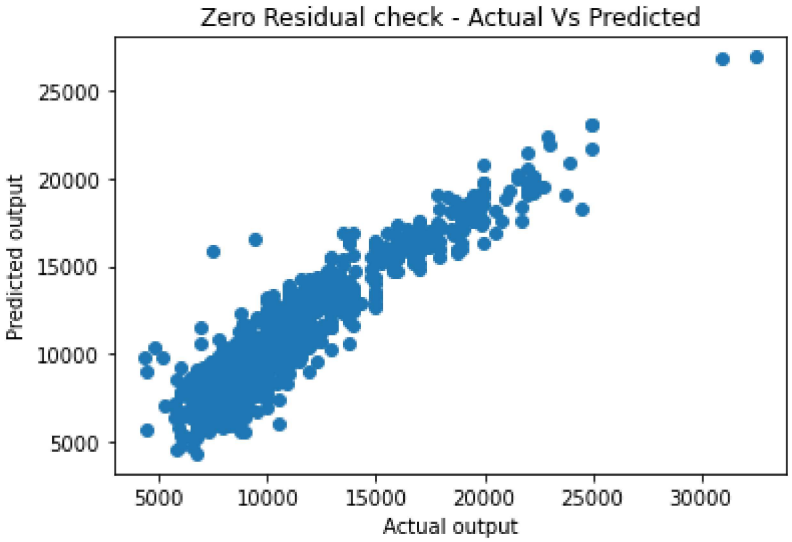
In [29]:
```
plt.scatter(x=x_train['QT'],y=error)
plt.title('Homoscedacity_check- QT Vs error')
plt.xlabel('QT')
plt.ylabel('error')
plt.show()
```



In [30]:
```
plt.scatter(x=x_train['Weight'],y=error)
plt.title('Homoscedacity_check- Weight Vs error')
plt.xlabel('Weight')
plt.ylabel('error')
plt.show()
```



In [31]:
```
plt.scatter(x=y_train,y=y_pred_train)
plt.title('Zero Residual check - Actual Vs Predicted')
plt.xlabel('Actual output')
plt.ylabel('Predicted output')
plt.show()
```



In [32]:
```
y_pred_test = linear_model.predict(x_test)
```

In [33]:
```
mean_absolute_percentage_error(y_test,y_pred_test)
```

Out[33]: 0.09788035676954028

In [34]:
```
x.head()
```

Out[34]:

	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	23	46986	90	2000	3	5	210	1165
1	23	72937	90	2000	3	5	210	1165
2	24	41711	90	2000	3	5	210	1165
3	26	48000	90	2000	3	5	210	1165
4	30	38500	90	2000	3	5	210	1170

In [35]:
```
model = smf.ols('Price~Age',toyota_data1).fit()
print('Rsquare       :',round(model.rsquared,4))
print('Adj Rsquared  :',round(model.rsquared_adj,4))
print('AIC           :',round(model.aic,4))
print('BIC           :',round(model.bic,4))
```

```
Rsquare       : 0.7684
Adj Rsquared  : 0.7682
AIC           : 25516.9706
BIC           : 25527.5098
```

In [36]:
```
model_2= smf.ols('Price~Age+KM',toyota_data1).fit()
print('Rsquare       :',round(model_2.rsquared,4))
print('Adj Rsquared  :',round(model_2.rsquared_adj,4))
print('AIC           :',round(model_2.aic,4))
print('BIC           :',round(model_2.bic,4))
```

```
Rsquare       : 0.79
Adj Rsquared  : 0.7897
AIC           : 25378.6096
BIC           : 25394.4185
```

In [37]:
```
model_3 = smf.ols('Price~Age+KM+HP',toyota_data1).fit()
print('Rsquare       :',round(model_3.rsquared,4))
print('Adj Rsquared  :',round(model_3.rsquared_adj,4))
print('AIC           :',round(model_3.aic,4))
print('BIC           :',round(model_3.bic,4))
```

```
Rsquare       : 0.8103
Adj Rsquared  : 0.8099
AIC           : 25234.4118
BIC           : 25255.4902
```

In [38]:
```
model_4= smf.ols('Price~Age+KM+HP+CC',toyota_data1).fit()
print('Rsquare       :',round(model_4.rsquared,4))
print('Adj Rsquared  :',round(model_4.rsquared_adj,4))
print('AIC           :',round(model_4.aic,4))
print('BIC           :',round(model_4.bic,4))
```

```
Rsquare       : 0.8135
Adj Rsquared  : 0.813
AIC           : 25211.8665
BIC           : 25238.2146
```

In [39]:
```
model_5= smf.ols('Price~Age+KM+HP+CC+Doors',toyota_data1).fit()
print('Rsquare       :',round(model_5.rsquared,4))
print('Adj Rsquared  :',round(model_5.rsquared_adj,4))
print('AIC           :',round(model_5.aic,4))
print('BIC           :',round(model_5.bic,4))
```

```
Rsquare       : 0.8158
Adj Rsquared  : 0.8151
AIC           : 25196.5421
BIC           : 25228.1598
```

In [40]:
```
model_6 = smf.ols('Price~Age+KM+HP+CC+Doors+Gears',toyota_data1).fit()
print('Rsquare       :',round(model_6.rsquared,4))
print('Adj Rsquared  :',round(model_6.rsquared_adj,4))
print('AIC           :',round(model_6.aic,4))
print('BIC           :',round(model_6.bic,4))
```

```
Rsquare       : 0.8173
Adj Rsquared  : 0.8166
AIC           : 25186.1133
BIC           : 25223.0007
```

In [41]:
```
model_7 = smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT',toyota_data1).fit()
print('Rsquare       :',round(model_7.rsquared,4))
print('Adj Rsquared  :',round(model_7.rsquared_adj,4))
print('AIC           :',round(model_7.aic,4))
print('BIC           :',round(model_7.bic,4))
```

```
Rsquare       : 0.8397
Adj Rsquared  : 0.8389
AIC           : 25000.7486
BIC           : 25042.9055
```

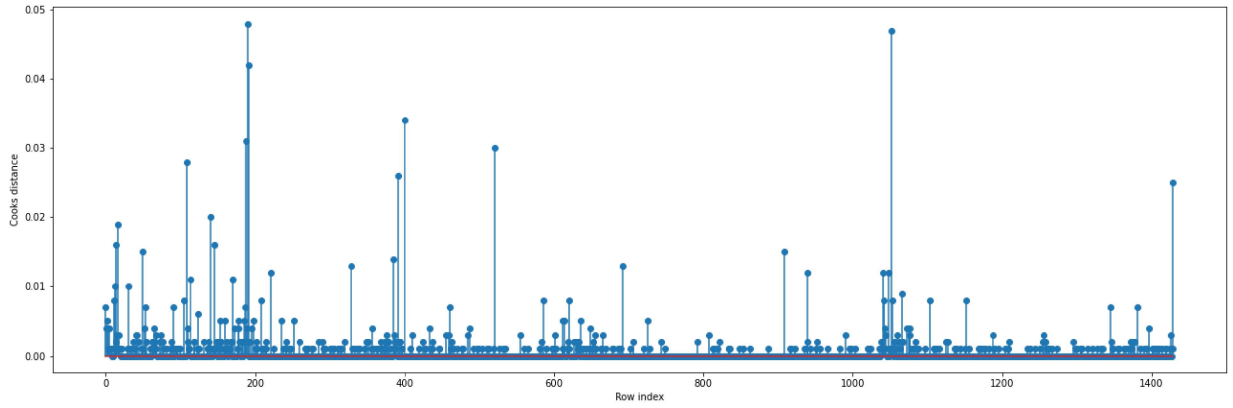
In [79]:
```
model_8= smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT+Weight',toyota_data1).fit()
print('Rsquare       :',round(model_8.rsquared,4))
print('Adj Rsquared  :',round(model_8.rsquared_adj,4))
print('AIC           :',round(model_8.aic,4))
print('BIC           :',round(model_8.bic,4))
```

```
Rsquare       : 0.8921
Adj Rsquared  : 0.8915
AIC           : 24329.6021
BIC           : 24376.991
```

In [80]:
```
(c,_)=model_8.get_influence().cooks_distance
c
```

Out[80]:
```
array([7.32566266e-03, 3.79983935e-03, 5.24624456e-03, ...,
       1.70171379e-05, 1.22456680e-03, 2.49161529e-02])
```

In [81]:
```
plt.figure(figsize=(22,7))
plt.stem(np.arange(len(toyota_data1)),np.round(c,3))
plt.xlabel('Row index')
plt.ylabel('Cooks distance')
plt.show()
```



In [77]:
```
np.argmax(c),np.max(c)
```

Out[77]: (986, 0.11657027651089945)

In [78]:
```
toyota_data1=toyota_data1.drop(toyota_data1.index[986],axis=0).reset_index(drop=T
toyota_data1
```

Out[78]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1425	7500	69	20544	86	1300	3	5	69	1025
1426	10845	72	19000	86	1300	3	5	69	1015
1427	8500	71	17016	86	1300	3	5	69	1015
1428	7250	70	16916	86	1300	3	5	69	1015
1429	6950	76	1	110	1600	5	5	19	1114

1430 rows × 9 columns

In [82]:
```
x = toyota_data1.drop('Price',axis=1)
y = toyota_data1[['Price']]
```

In [83]:
```
std_scalar = StandardScaler()
std_scalar = std_scalar.fit_transform(x)
x_scaled=pd.DataFrame(std_scalar,columns=x.columns)
x_scaled
```

Out[83]:

	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	-1.771663	-0.576972	-0.767903	2.323164	-1.085866	-0.154487	2.989051	1.910318
1	-1.771663	0.115474	-0.767903	2.323164	-1.085866	-0.154487	2.989051	1.910318
2	-1.717921	-0.717724	-0.767903	2.323164	-1.085866	-0.154487	2.989051	1.910318
3	-1.610439	-0.549916	-0.767903	2.323164	-1.085866	-0.154487	2.989051	1.910318
4	-1.395473	-0.803402	-0.767903	2.323164	-1.085866	-0.154487	2.989051	2.012396
...	...	...	...	...	...	...	...	...
1425	0.700442	-1.282519	-1.034740	-1.426805	-1.085866	-0.154487	-0.441635	-0.947871
1426	0.861667	-1.323717	-1.034740	-1.426805	-1.085866	-0.154487	-0.441635	-1.152027
1427	0.807925	-1.376656	-1.034740	-1.426805	-1.085866	-0.154487	-0.441635	-1.152027
1428	0.754184	-1.379324	-1.034740	-1.426805	-1.085866	-0.154487	-0.441635	-1.152027
1429	1.076632	-1.830664	0.566282	0.180324	1.013916	-0.154487	-1.658190	0.869121

1430 rows × 8 columns

In [84]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size=0.20,shuff]
```

In [86]:
```
x_train.shape,y_train.shape
```

Out[86]:
```
((1144, 8), (1144, 1))
```

In [89]:
```
linear_model1 = LinearRegression()
linear_model1.fit(x_train,y_train)
```

Out[89]:
```
LinearRegression()
```

In [90]:
```
y_pred_train1 = linear_model.predict(x_train)
y_pred_train1
```

Out[90]:
```
array([[ 8094.25302318],
       [ 9738.98193096],
       [11676.31878184],
       ...,
       [ 7926.9102457 ],
       [ 8560.33102908],
       [ 9376.9768836 ]])
```

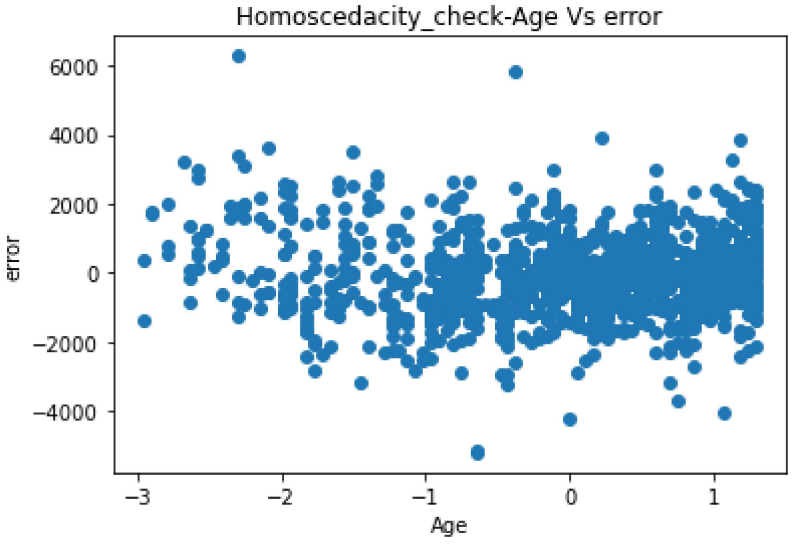
In [91]:
```
error1=y_train-y_pred_train1
error1
```

Out[91]:

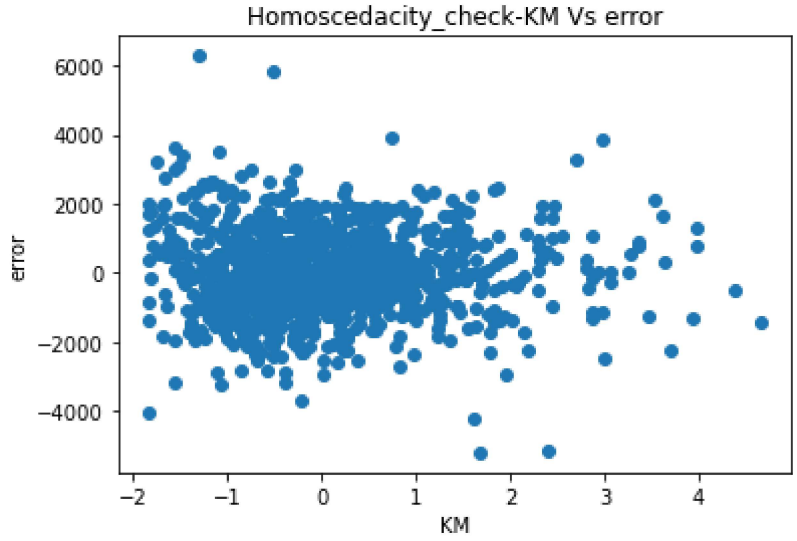
	Price
622	855.746977
702	761.018069
258	273.681218
761	298.549083
535	-1654.631757
...	...
432	275.862226
1283	-442.421403
1265	-781.910246
1277	389.668971
1414	-1226.976884

1144 rows × 1 columns

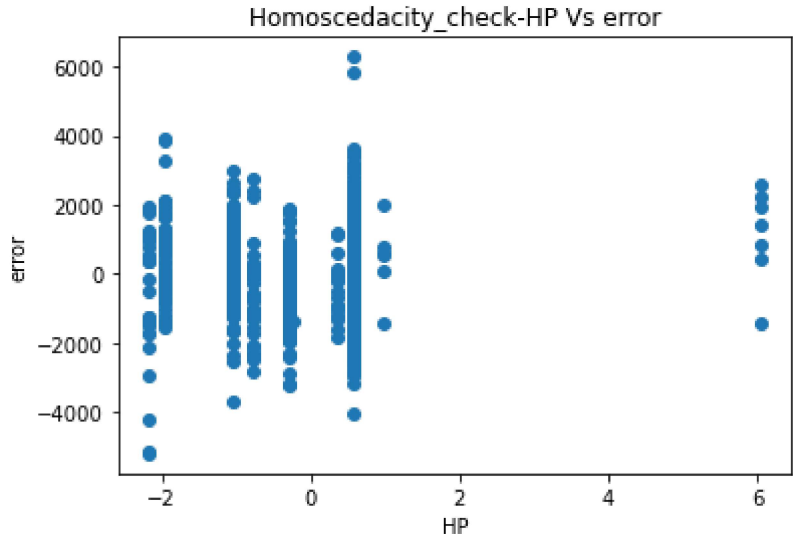
In [105]:
```
plt.scatter(x=x_train['Age'],y=error1)
plt.title('Homoscedacity_check-Age Vs error')
plt.xlabel('Age')
plt.ylabel('error')
plt.show()
```



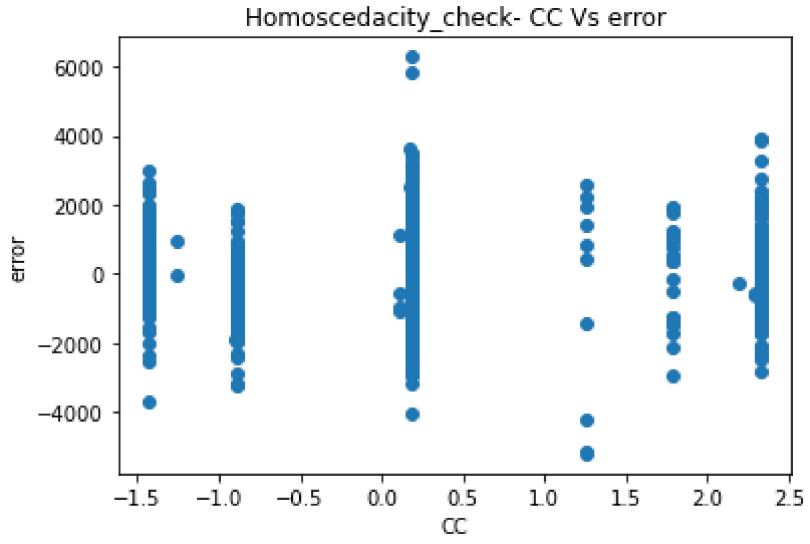
In [106]:
```
plt.scatter(x=x_train['KM'],y=error1)
plt.title('Homoscedacity_check-KM Vs error')
plt.xlabel('KM')
plt.ylabel('error')
plt.show()
```



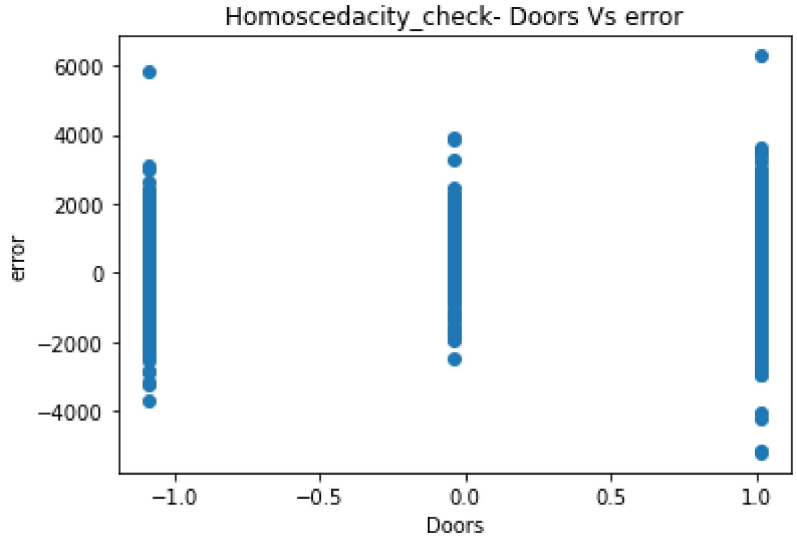
In [107]:
```
plt.scatter(x=x_train['HP'],y=error1)
plt.title('Homoscedacity_check-HP Vs error')
plt.xlabel('HP')
plt.ylabel('error')
plt.show()
```



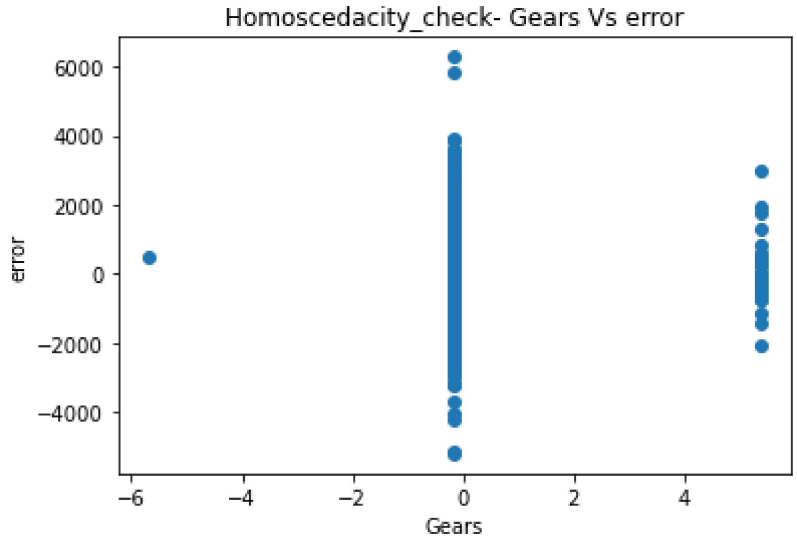
In [108]:
```
plt.scatter(x=x_train['CC'],y=error1)
plt.title('Homoscedacity_check- CC Vs error')
plt.xlabel('CC')
plt.ylabel('error')
plt.show()
```



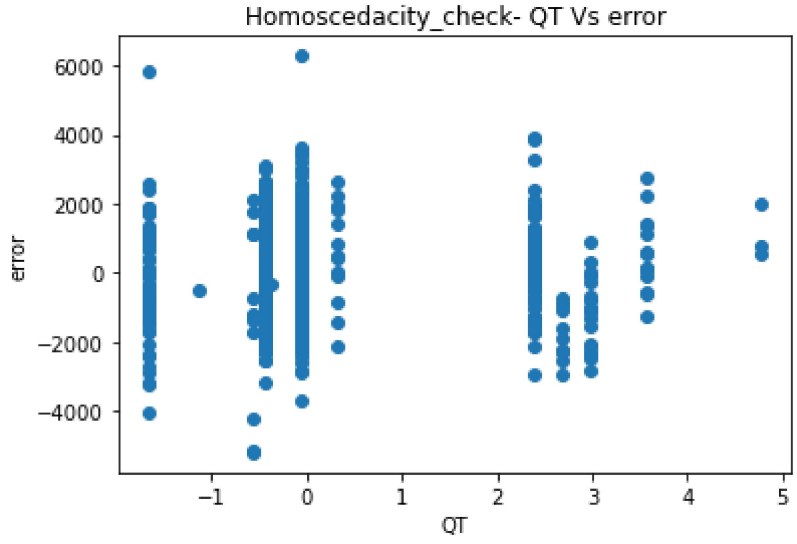
In [109]:
```
plt.scatter(x=x_train['Doors'],y=error1)
plt.title('Homoscedacity_check- Doors Vs error')
plt.xlabel('Doors')
plt.ylabel('error')
plt.show()
```



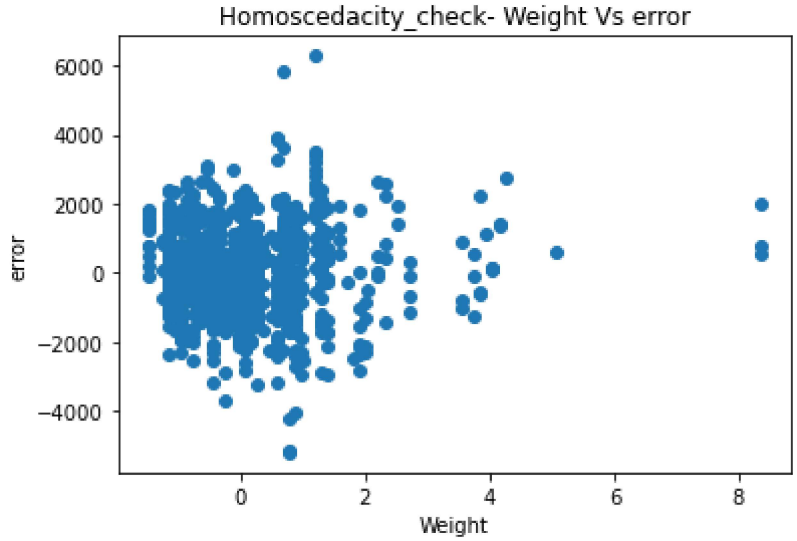
In [110]:
```
plt.scatter(x=x_train['Gears'],y=error1)
plt.title('Homoscedacity_check- Gears Vs error')
plt.xlabel('Gears')
plt.ylabel('error')
plt.show()
```



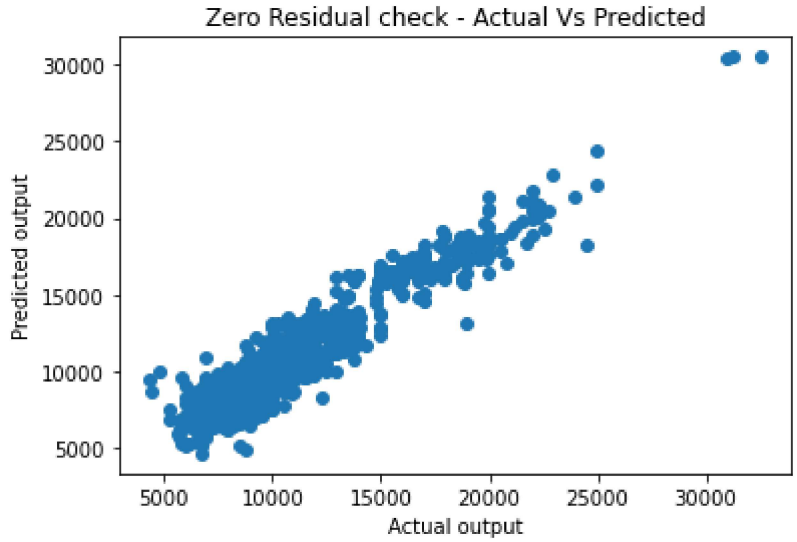
In [111]:
```
plt.scatter(x=x_train['QT'],y=error1)
plt.title('Homoscedacity_check- QT Vs error')
plt.xlabel('QT')
plt.ylabel('error')
plt.show()
```



In [112]:
```
plt.scatter(x=x_train['Weight'],y=error1)
plt.title('Homoscedacity_check- Weight Vs error')
plt.xlabel('Weight')
plt.ylabel('error')
plt.show()
```



In [113]:
```
plt.scatter(x=y_train,y=y_pred_train1)
plt.title('Zero Residual check - Actual Vs Predicted')
plt.xlabel('Actual output')
plt.ylabel('Predicted output')
plt.show()
```



In [92]:
```
mean_absolute_percentage_error(y_train,y_pred_train1)
```

Out[92]:
```
0.09110884753969503
```

In [93]:
```
x_scaled['Price']=y
```

In [94]:
```
model = smf.ols('Price~Age',x_scaled).fit()
print('Rsquare       :',round(model.rsquared,4))
print('Adj Rsquared  :',round(model.rsquared_adj,4))
print('AIC           :',round(model.aic,4))
print('BIC           :',round(model.bic,4))
```

```
Rsquare       : 0.7691
Adj Rsquared  : 0.769
AIC           : 25403.9402
BIC           : 25414.4711
```

In [95]:
```
model_2= smf.ols('Price~Age+KM',x_scaled).fit()
print('Rsquare       :',round(model_2.rsquared,4))
print('Adj Rsquared  :',round(model_2.rsquared_adj,4))
print('AIC           :',round(model_2.aic,4))
print('BIC           :',round(model_2.bic,4))
```

```
Rsquare       : 0.7913
Adj Rsquared  : 0.791
AIC           : 25261.9117
BIC           : 25277.708
```

In [97]:
```
model_3 = smf.ols('Price~Age+KM+HP',x_scaled).fit()
print('Rsquare       :',round(model_3.rsquared,4))
print('Adj Rsquared  :',round(model_3.rsquared_adj,4))
print('AIC           :',round(model_3.aic,4))
print('BIC           :',round(model_3.bic,4))
```

```
Rsquare       : 0.8113
Adj Rsquared  : 0.8109
AIC           : 25119.3335
BIC           : 25140.3952
```

In [98]:
```
model_4= smf.ols('Price~Age+KM+HP+CC',x_scaled).fit()
print('Rsquare       :',round(model_4.rsquared,4))
print('Adj Rsquared  :',round(model_4.rsquared_adj,4))
print('AIC           :',round(model_4.aic,4))
print('BIC           :',round(model_4.bic,4))
```

```
Rsquare       : 0.8224
Adj Rsquared  : 0.8219
AIC           : 25035.1979
BIC           : 25061.525
```

In [99]:
```
model_5= smf.ols('Price~Age+KM+HP+CC+Doors',x_scaled).fit()
print('Rsquare       :',round(model_5.rsquared,4))
print('Adj Rsquared  :',round(model_5.rsquared_adj,4))
print('AIC           :',round(model_5.aic,4))
print('BIC           :',round(model_5.bic,4))
```

```
Rsquare       : 0.8239
Adj Rsquared  : 0.8233
AIC           : 25024.3426
BIC           : 25055.9352
```

In [100]:
```
model_6 = smf.ols('Price~Age+KM+HP+CC+Doors+Gears',x_scaled).fit()
print('Rsquare       :',round(model_6.rsquared,4))
print('Adj Rsquared  :',round(model_6.rsquared_adj,4))
print('AIC           :',round(model_6.aic,4))
print('BIC           :',round(model_6.bic,4))
```

```
Rsquare       : 0.8251
Adj Rsquared  : 0.8243
AIC           : 25017.1632
BIC           : 25054.0212
```

In [101]:
```
model_7 = smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT',x_scaled).fit()
print('Rsquare       :',round(model_7.rsquared,4))
print('Adj Rsquared  :',round(model_7.rsquared_adj,4))
print('AIC           :',round(model_7.aic,4))
print('BIC           :',round(model_7.bic,4))
```

```
Rsquare       : 0.8405
Adj Rsquared  : 0.8398
AIC           : 24886.703
BIC           : 24928.8264
```

In [103]:
```
model_8= smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT+Weight',x_scaled).fit()
print('Rsquare       :',round(model_8.rsquared,4))
print('Adj Rsquared  :',round(model_8.rsquared_adj,4))
print('AIC           :',round(model_8.aic,4))
print('BIC           :',round(model_8.bic,4))
```

```
Rsquare       : 0.8921
Adj Rsquared  : 0.8915
AIC           : 24329.6021
BIC           : 24376.991
```

In [ ]:
```
```