In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler

```

In [2]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\datascience\vehicle dataset.csv")
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [3]:

```
df=df[["engine_power","km"]]

```

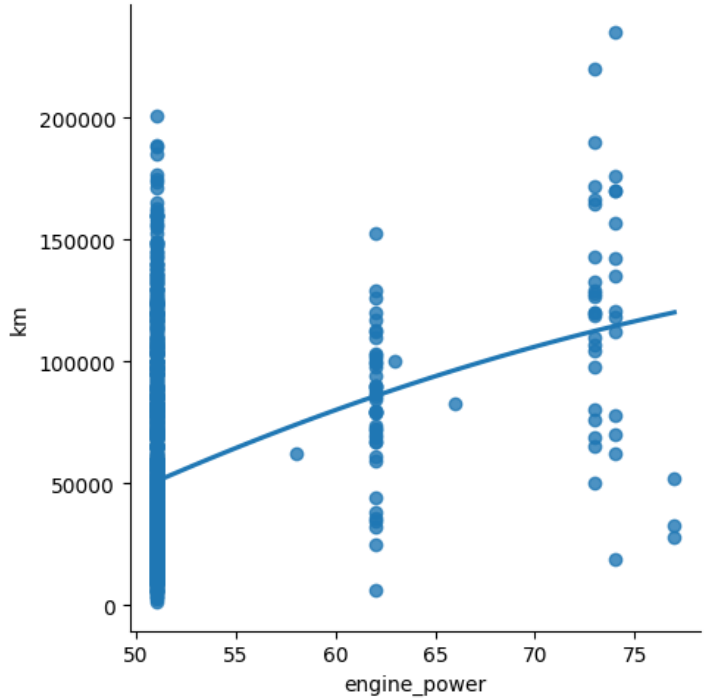
In [4]:

```
sb.lmplot(x="engine_power",y="km",data=df,order=2,ci=None)

```

Out[4]:

```
<seaborn.axisgrid.FacetGrid at 0x2dbaba9fc50>
```



In [5]:

```
df.describe()

```

Out[5]:

	engine_power	km
count	1538.000000	1538.000000
mean	51.904421	53396.011704
std	3.988023	40046.830723
min	51.000000	1232.000000
25%	51.000000	20006.250000
50%	51.000000	39031.000000
75%	51.000000	79667.750000
max	77.000000	235000.000000

In [6]:

```
df.fillna(method="ffill",inplace=True)
```

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_5492\1844562654.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method="ffill",inplace=True)
```

In [7]:

```
x=np.array(df['engine_power']).reshape(-1,1)
```

In [8]:

```
y=np.array(df['km']).reshape(-1,1)
```

In [*]:

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

In [13]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [16]:

```
regr=LinearRegression()
```

In [25]:

```
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
print(regr.score(x_train,y_train))

```

```
0.07471804568445017
0.08341371834104194
```

In [23]:

```
df500=df[:][:500]
sb.lmplot(x="engine_power",y="Km",data=df500,order=1,ci=None)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[23], line 2
      1 df500=df[:][:500]
----> 2 sb.lmplot(x="engine_power",y="Km",data=df500,order=1,ci=None)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\regression.py:595, in lmplot(data, x,
y, hue, col, row, palette, col_wrap, height, aspect, markers, sharex, sharey, hue_order, col_order, row_order,
legend, legend_out, x_estimator, x_bins, x_ci, scatter, fit_reg, ci, n_boot, units, seed, order, logistic, low
ess, robust, logx, x_partial, y_partial, truncate, x_jitter, y_jitter, scatter_kws, line_kws, facet_kws)
    593 need_cols = [x, y, hue, col, row, units, x_partial, y_partial]
    594 cols = np.unique([a for a in need_cols if a is not None]).tolist()
--> 595 data = data[cols]
    597 # Initialize the grid
    598 facets = FacetGrid(
    599     data, row=row, col=col, hue=hue,
    600     palette=palette,
   (...)
    603     **facet_kws,
    604 )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py:3767, in DataFrame.__get
item__(self, key)
   3765     if is_iterator(key):
   3766         key = list(key)
-> 3767     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   3769 # take() does not accept boolean indexers
   3770 if getattr(indexer, "dtype", None) == bool:

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes\base.py:5876, in Index._g
et_indexer_strict(self, key, axis_name)
   5873 else:
   5874     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 5876 self._raise_if_missing(keyarr, indexer, axis_name)
   5878 keyarr = self.take(indexer)
   5879 if isinstance(key, Index):
   5880     # GH 42790 - Preserve name from an Index

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes\base.py:5938, in Index._r
aise_if_missing(self, key, indexer, axis_name)
   5935     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   5937 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 5938 raise KeyError(f"{not_found} not in index")

KeyError: "['Km'] not in index"
```

In [21]:

```
df500.fillna(method="ffill",inplace=True)
x=np.array(df500["engine_power"]).reshape(-1,1)
y=np.array(df500["km"]).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
a=LinearRegression()
a.fit(x_train,y_train)
print(a.score(x_test,y_test))
y_pred=a.predict(x_test)
plt.scatter(x_test,y_test,color="b")
plt.plot(x_test,y_pred,color="k")
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[21], line 1
----> 1 df500.fillna(method="ffill",inplace=True)
      2 x=np.array(df500["engine_power"]).reshape(-1,1)
      3 y=np.array(df500["km"]).reshape(-1,1)

NameError: name 'df500' is not defined
```

In [34]:

```
features=df.columns[0:8]
target=df.columns[-1]
```

In [35]:

```
x=df[features].values
y=df[target].values
```

In [36]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
print(x_train.shape)
print(x_test.shape)
```

```
(1076, 2)
(462, 2)
```

In [37]:

```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

In [52]:

```
regr=LinearRegression()
regr.fit(x_train,y_train)
train_score_a=regr.score(x_train, y_train)
test_score_a=regr.score(x_test,y_test)
print("\nLinear Regression Model:\n")
print(train_score_a)
print(test_score_a)
```

```
Linear Regression Model:

1.0
1.0
```

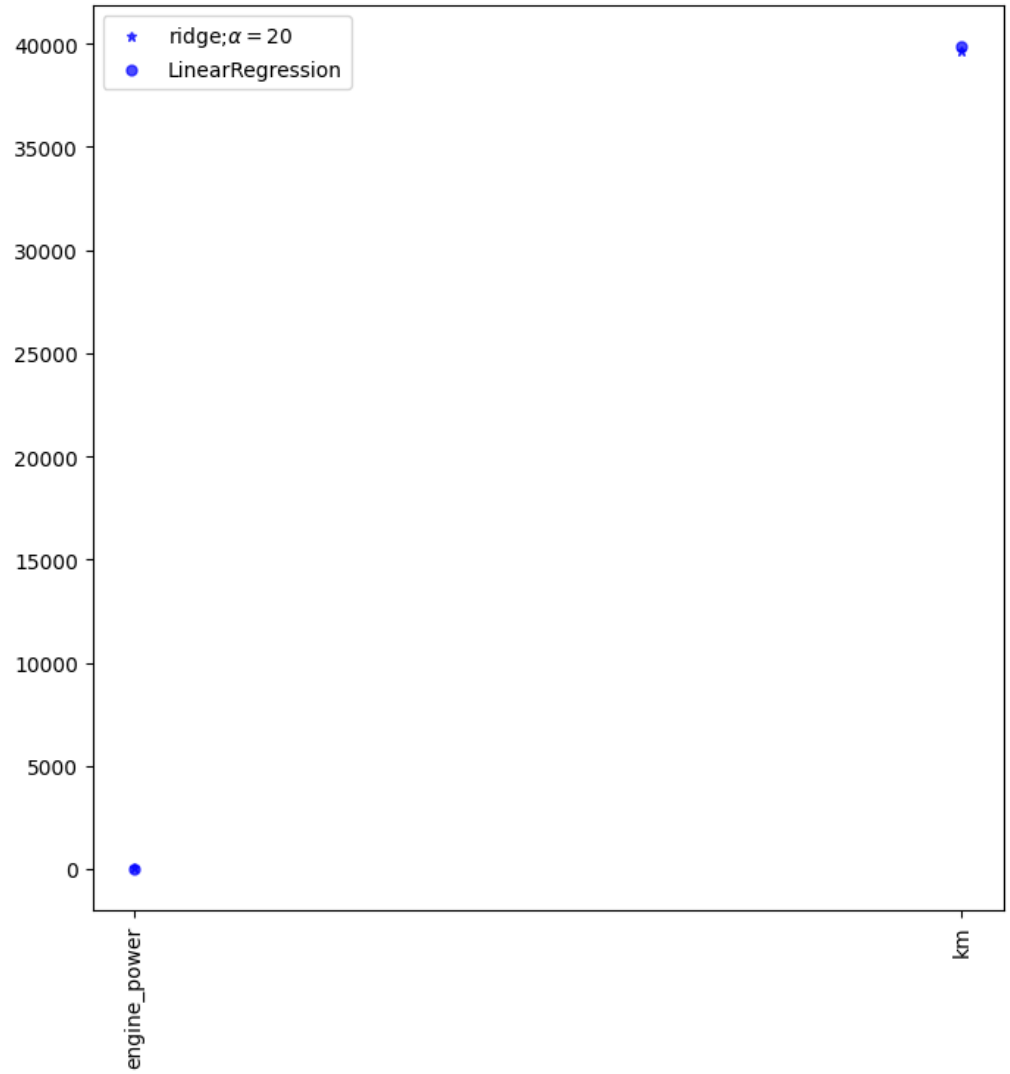
In [55]:

```
.#ridge Regression#
r=Ridge(alpha=100)
r.fit(x_train,y_train)
train_score_ridge=r.score(x_train,y_train)
test_score_ridge=r.score(x_test,y_test)
print(train_score_ridge)
print(test_score_ridge)
```

```
  Cell In[55], line 1
    .#ridge Regression#
    ^
SyntaxError: invalid syntax
```

In [56]:

```
figure(figsize=(8,8))
plot(features,r.coef_,linestyle="None",alpha=0.7,markersize=5,color="blue",label=r"ridge;$\alpha=20$",marker="*",zorder=5)
plot(features,regr.coef_,alpha=0.7,linestyle="None",markersize=5,color="blue",label=r"LinearRegression",marker="o",zorder=5)
xticks(rotation=90)
legend()
show()
```



In [50]:

```
#Lasso Regression#
l=Lasso(alpha=10)
l.fit(x_train,y_train)
train_lasso=l.score(x_train,y_train)
test_lasso=l.score(x_test,y_test)
print(train_lasso)
print(test_lasso)
```

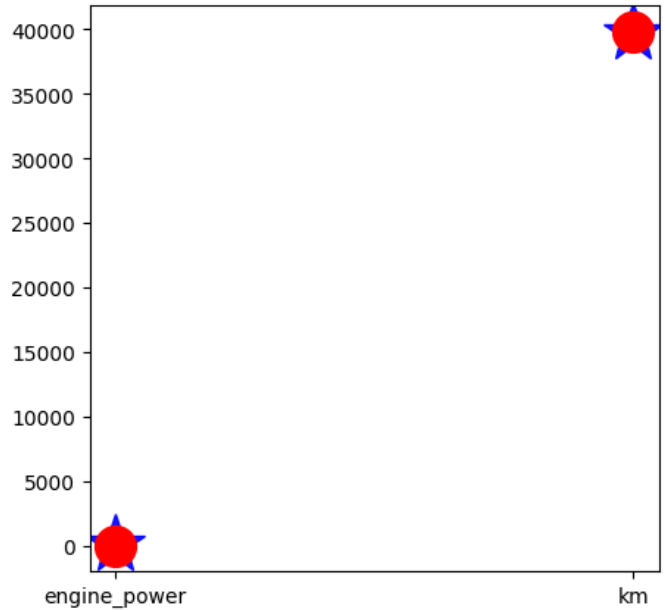
```
0.9999999370548741
0.9999999370294943
```

In [64]:

```
plt.figure(figsize=(5,5))
plt.plot(features,r.coef_,alpha=0.9,marker="*",markersize="30",label=r"ridge;$\alpha=30$",color="blue",linestyle="None")
plt.plot(features,l.coef_,alpha=1,marker="o",markersize=20,label="LinearRegression",color="red",zorder=10,linestyle="none")
```

Out[64]:

```
[<matplotlib.lines.Line2D at 0x2dbb66a5f10>]
```



In [66]:

```
ridge_cv=RidgeCV(alphas=[0.0001,1.2,0.009,0.076]).fit(x_train,y_train)
```

In [67]:

```
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.9999999999999901
0.9999999999999898
```

In [69]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0. 1.]
3.331616608193144e-05
```

In [ ]:

```

```