In [1]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```python
df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\fiat500_VehicleSelection_Da
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.61
1	2	pop	51	1186	32500	1	45.666359	12.24
2	3	sport	74	4658	142228	1	45.503300	11.41
3	4	lounge	51	2739	160000	1	40.633171	17.63
4	5	pop	73	3074	106880	1	41.903221	12.49
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.70
1534	1535	lounge	74	3835	112000	1	45.845692	8.66
1535	1536	pop	51	2223	60457	1	45.481541	9.41
1536	1537	lounge	51	2557	80750	1	45.000702	7.68
1537	1538	pop	51	1766	54276	1	40.323410	17.56

1538 rows × 9 columns

In [3]:

```python
df=df[['km','price']]
df.columns=['km','price']
```

In [4]:

```
df.head(10)
```

Out[4]:

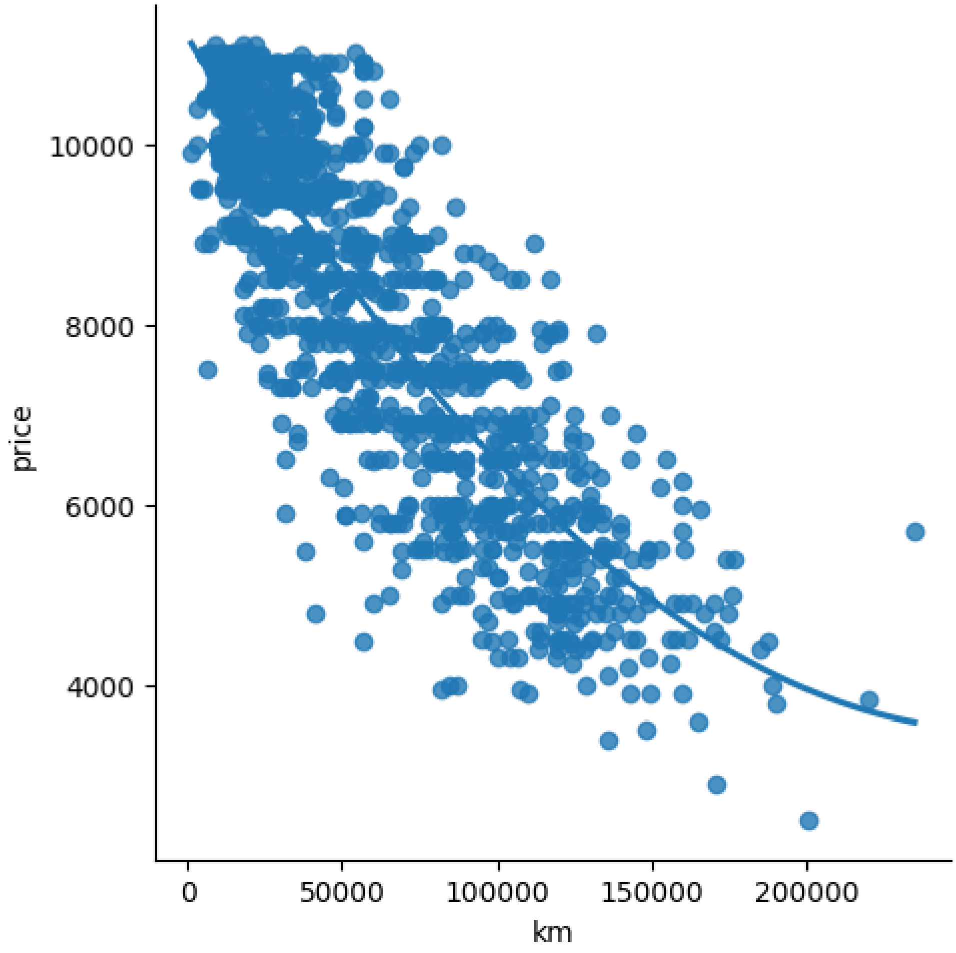
	km	price
0	25000	8900
1	32500	8800
2	142228	4200
3	160000	6000
4	106880	5700
5	70225	7900
6	11600	10750
7	49076	9190
8	76000	5600
9	89000	6000

In [5]:

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

Out[5]:

```
<seaborn.axisgrid.FacetGrid at 0x12889c00090>
```



In [6]:

```
df.describe()
```

Out[6]:

	km	price
count	1538.000000	1538.000000
mean	53396.011704	8576.003901
std	40046.830723	1939.958641
min	1232.000000	2500.000000
25%	20006.250000	7122.500000
50%	39031.000000	9000.000000
75%	79667.750000	10000.000000
max	235000.000000	11100.000000

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   km      1538 non-null   int64
 1   price   1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [8]:

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

```
C:\Users\samit\AppData\Local\Temp\ipykernel_21256\4116506308.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [9]:

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

In [10]:

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

```
C:\Users\samit\AppData\Local\Temp\ipykernel_21256\1379821321.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

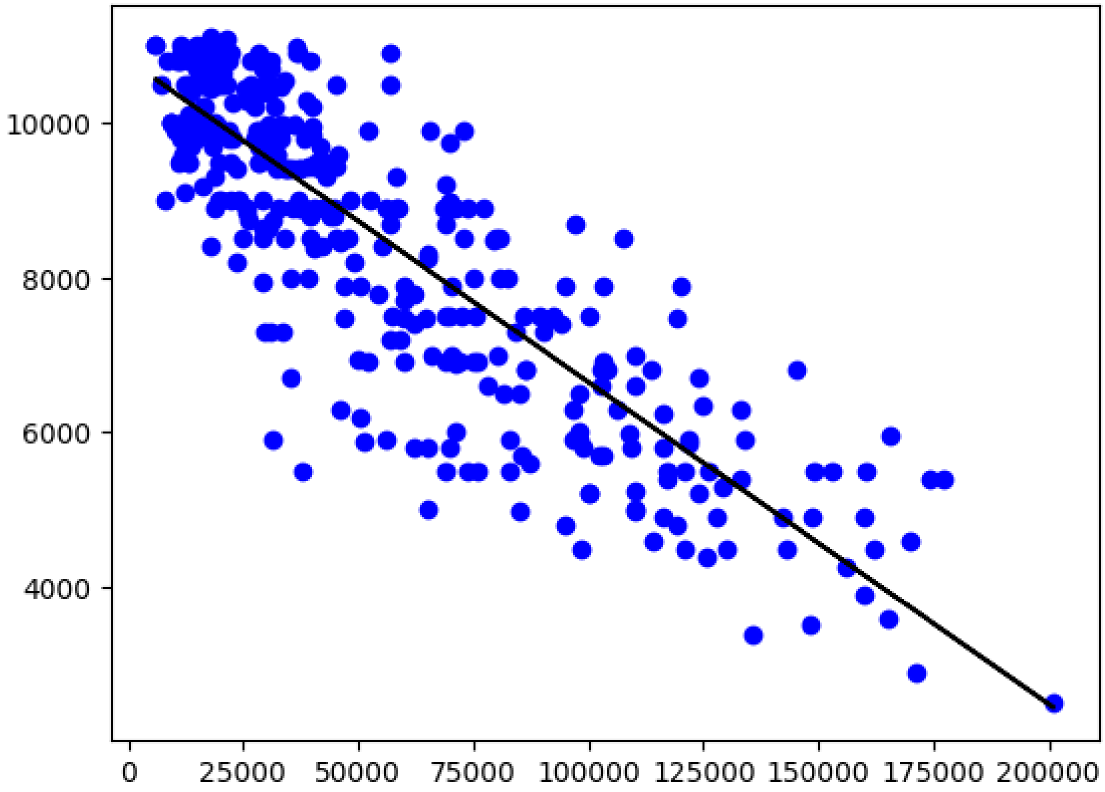
In [11]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.7410844190408512
```

In [12]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

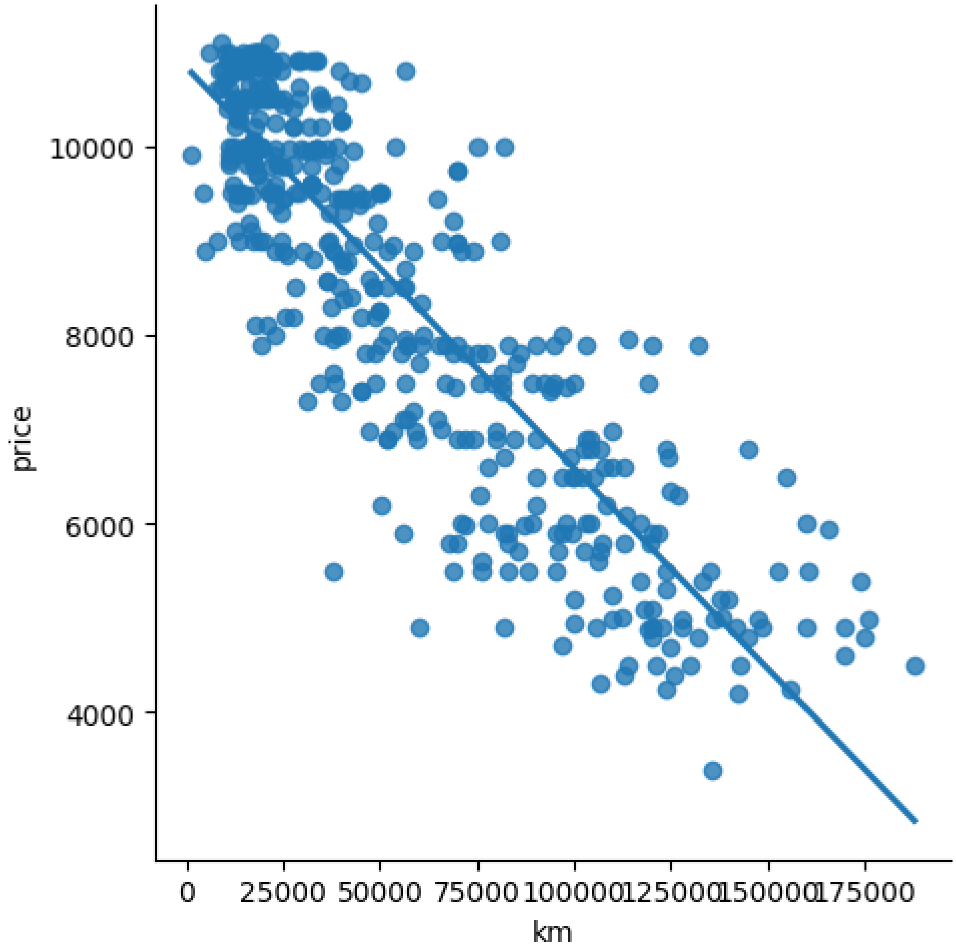


In [13]:

```
df500=df[:][:500]
sns.lmplot(x="km",y="price",data=df500,order=1,ci=None)
```

Out[13]:

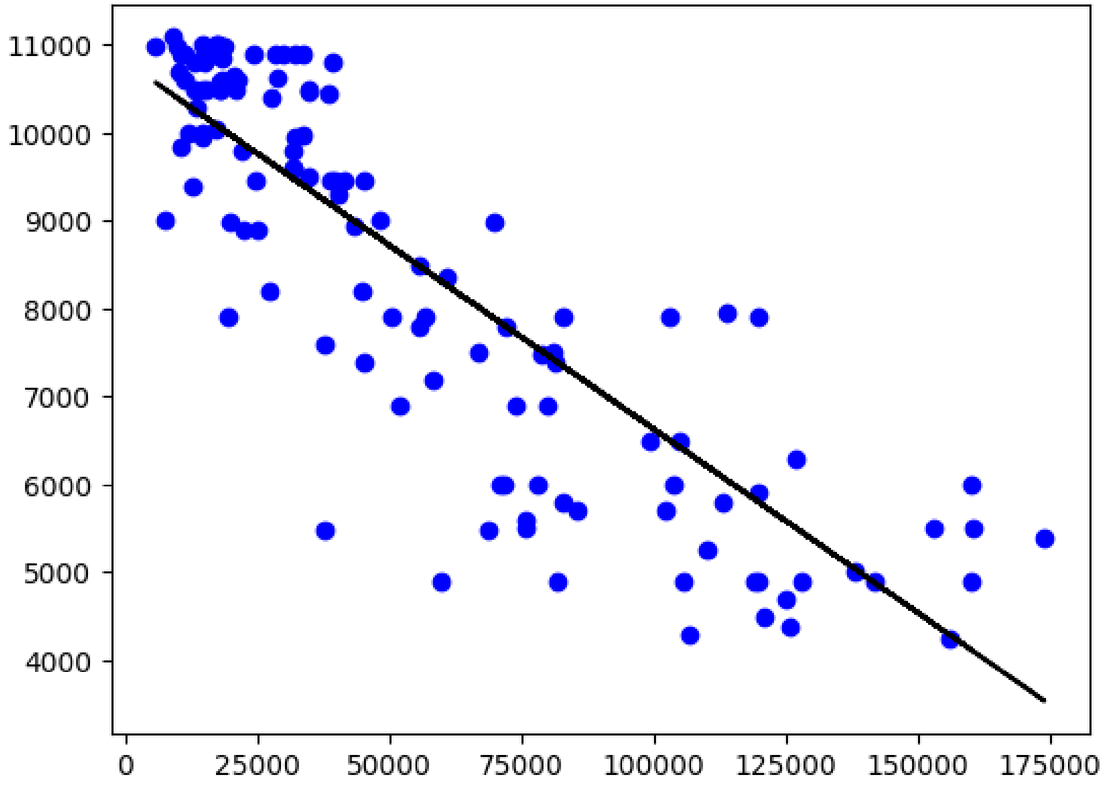
```
<seaborn.axisgrid.FacetGrid at 0x1288bd5f8d0>
```



In [14]:

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

```
Regression: 0.7481969250346543
```



In [15]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.7481969250346543
```

In [26]:

```
features = df.columns[0:2]
target = df.columns[-1]
#X and y values
X = df[features].values
y = df[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
#Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

```

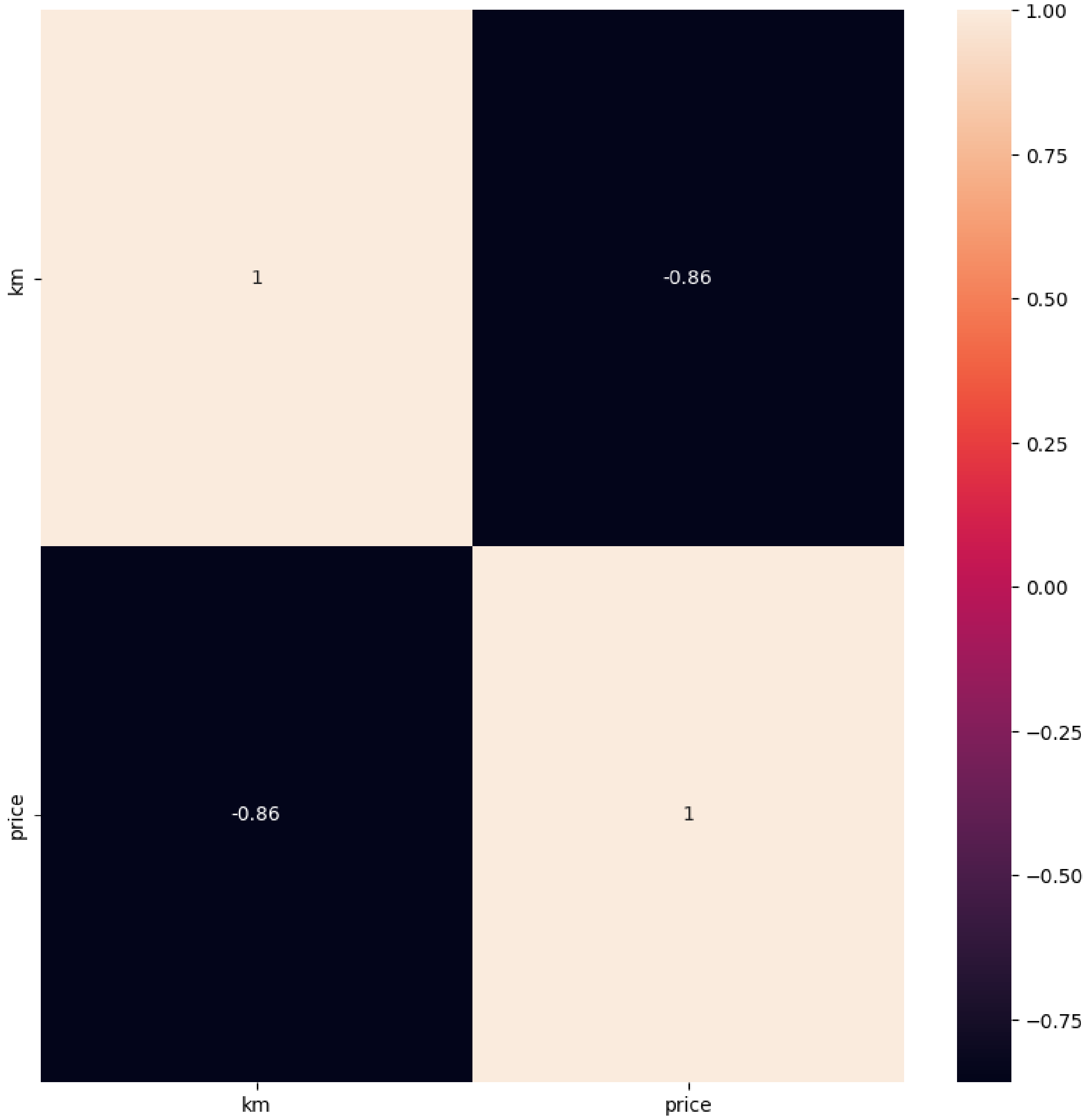
```
The dimension of X_train is (1076, 2)
The dimension of X_test is (462, 2)
```

In [17]:

```
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[17]:

```
<Axes: >
```



In [20]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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 [21]:

```python
#Model
lr = LinearRegression()
#Fit model
lr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

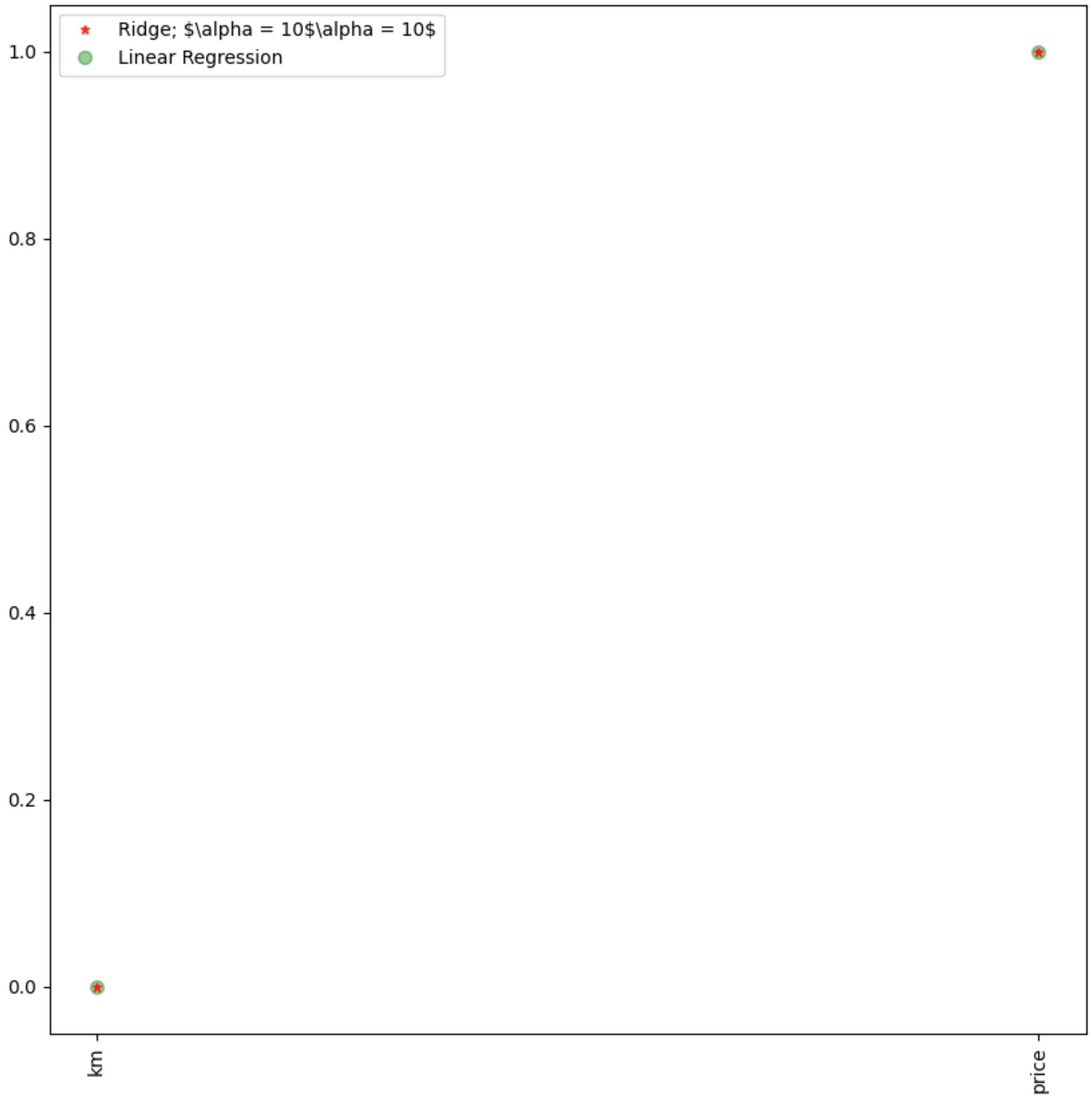
In [22]:

```python
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```

Ridge Model:

The train score for ridge model is 1.0
The test score for ridge model is 1.0
```

In [23]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5
#plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',l
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [24]:

```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

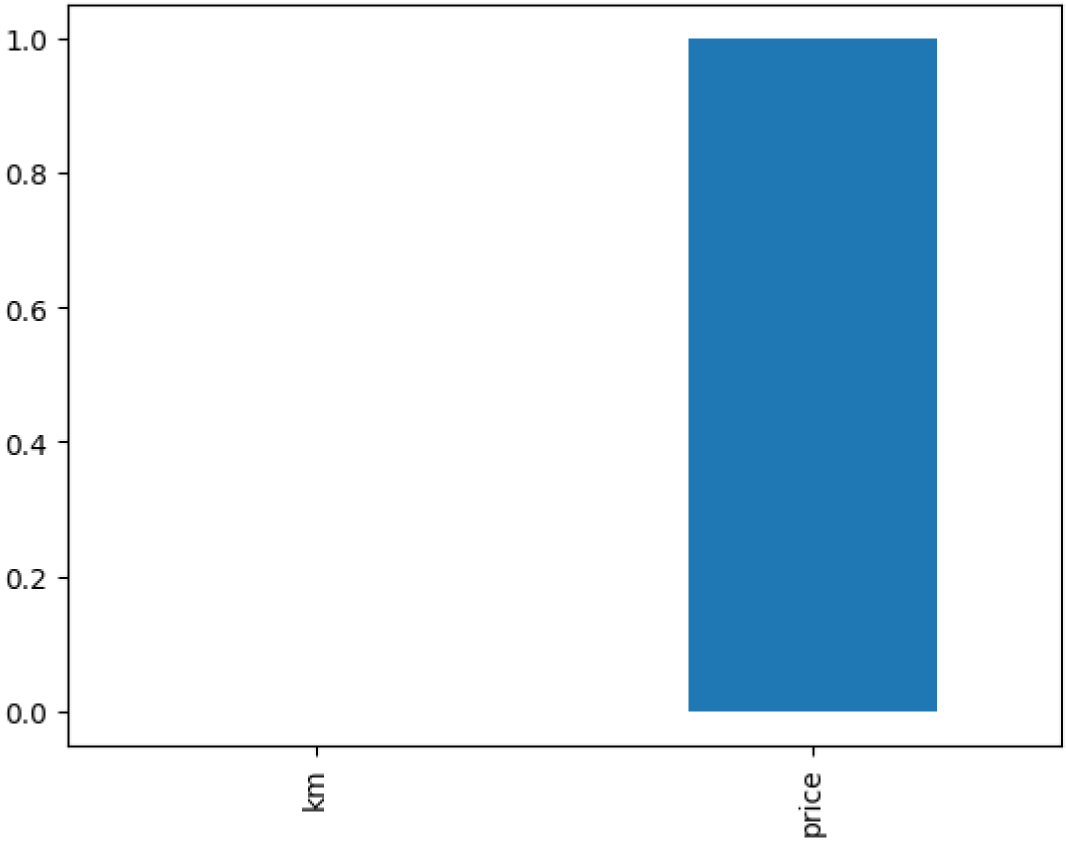
```

Lasso Model: 

The train score for ls model is 0.9999999760460123
The test score for ls model is 0.999999975505097
```

In [25]:

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[25]:

```
<Axes: >
```



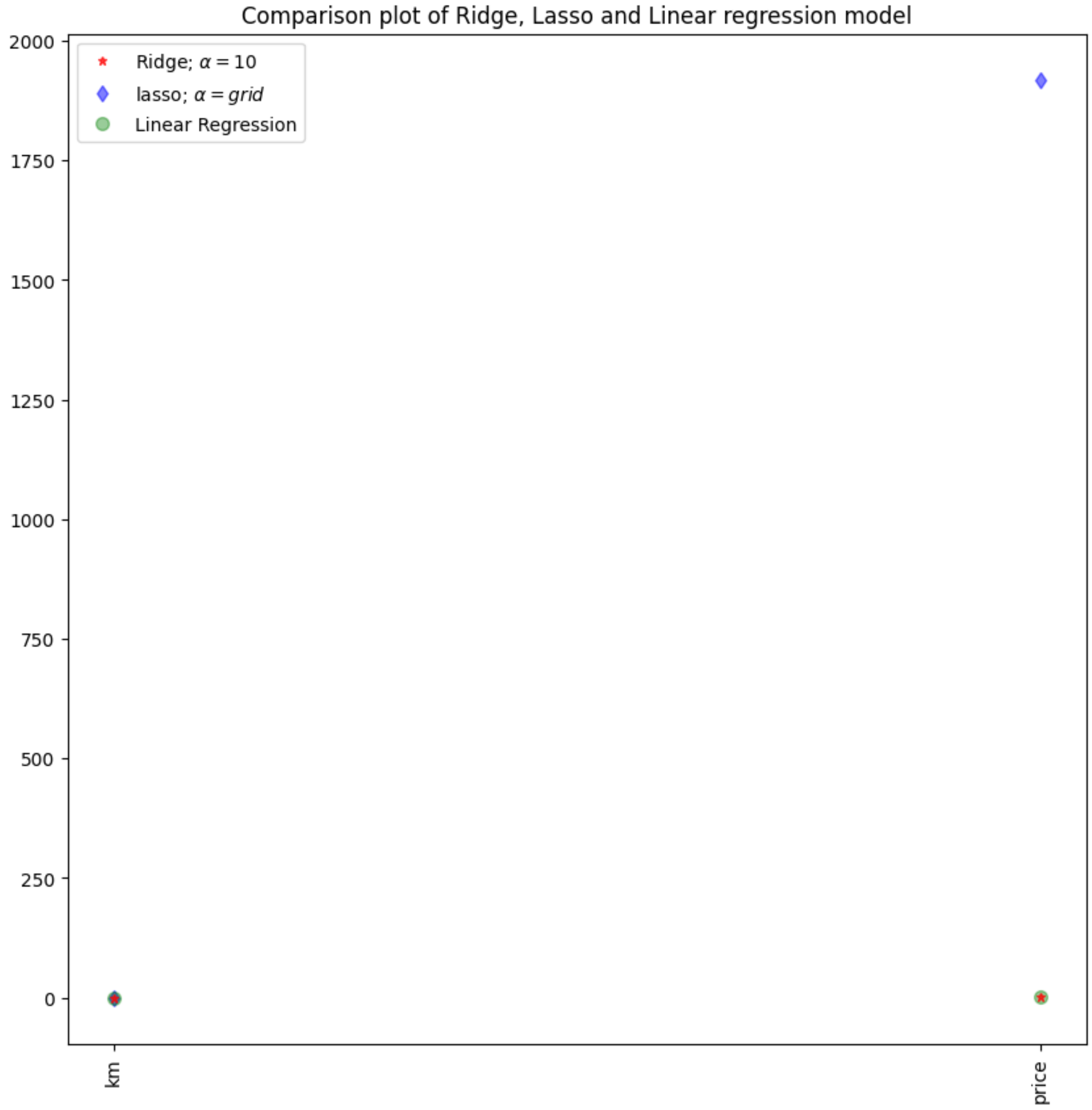
In [27]:

```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999877496772
0.9999999874481674
```

In [28]:

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='b
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()

```



In [29]:

```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

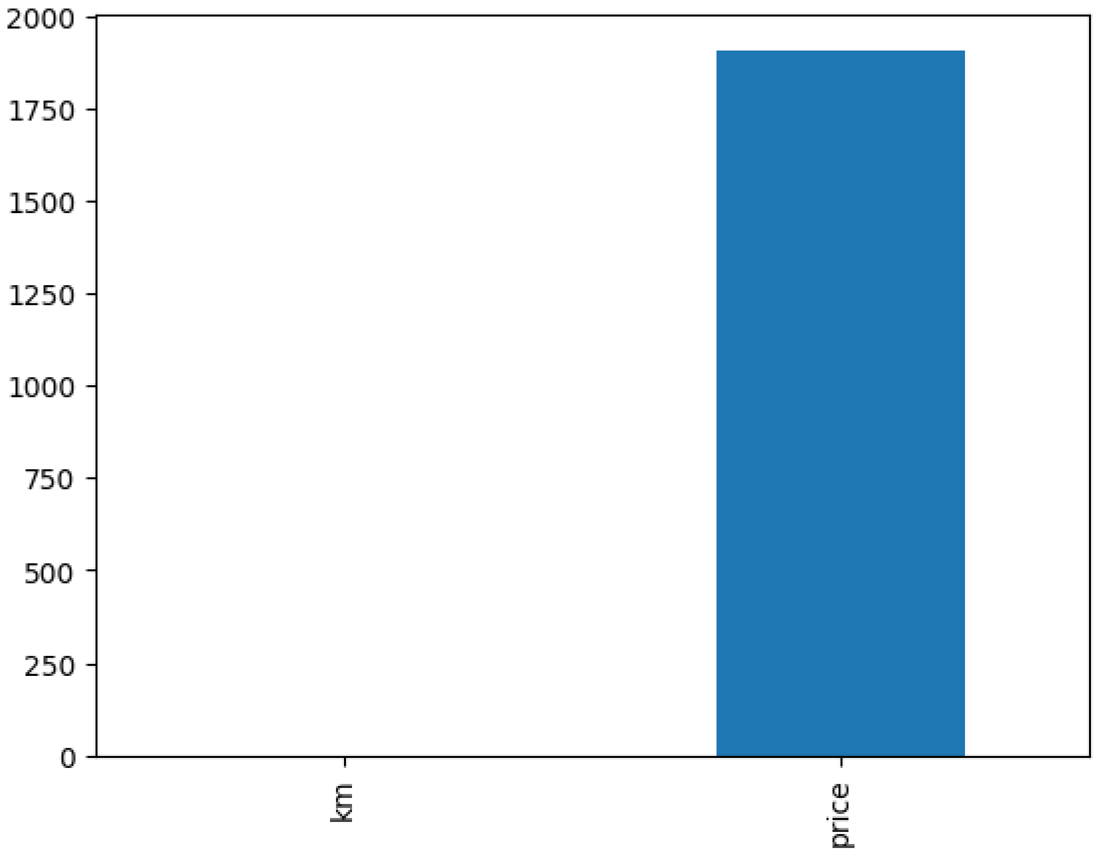
```
Lasso Model: 

The train score for ls model is 0.9999728562194999
The test score for ls model is 0.9999728508562553
```

In [31]:

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[31]:

```
<Axes: >
```



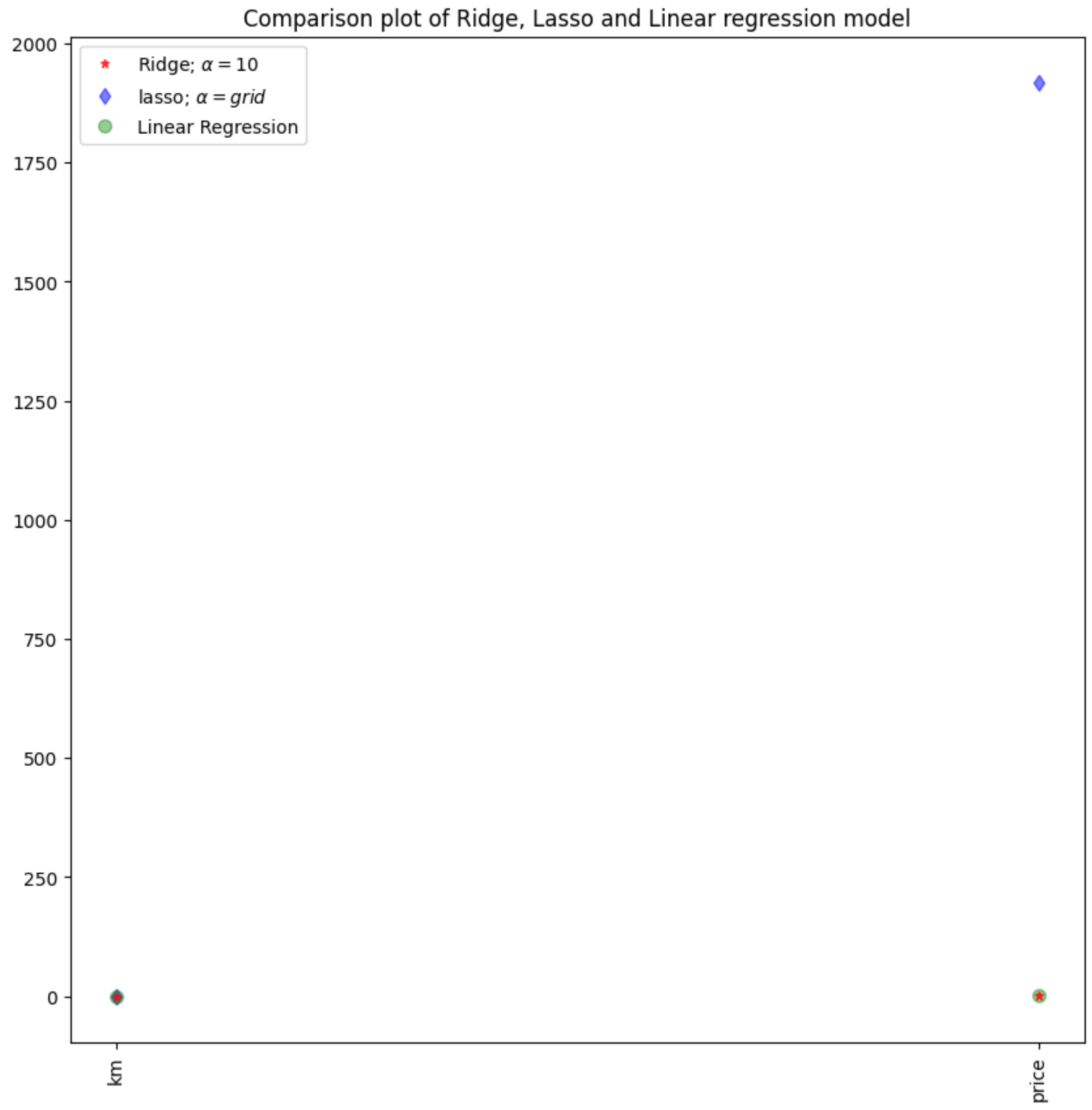
In [30]:

```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999877496772
0.9999999874481674
```

In [32]:

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='b
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()

```



In [33]:

```
#Using the linear CV model
from sklearn.linear_model import RidgeCV
#Ridge Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train, y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y_trai
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test)
```

```
The train score for ridge model is 0.999999999999966
The train score for ridge model is 0.9999999999999674
```

In [34]:

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

```
[-543.34766981  968.58411343]
8584.384758364313
```

In [36]:

```
y_pred_elastic=regr.predict(X_train)
```

In [ ]:

```

```