In [2]:

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

In [3]:

```
df=pd.read_csv(r"C:\Users\shaik\Downloads\fiat500_VehicleSelection_Dataset (2).csv")
df
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.611
1	2	pop	51	1186	32500	1	45.666359	12.241
2	3	sport	74	4658	142228	1	45.503300	11.417
3	4	lounge	51	2739	160000	1	40.633171	17.634
4	5	pop	73	3074	106880	1	41.903221	12.495
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704
1534	1535	lounge	74	3835	112000	1	45.845692	8.666
1535	1536	pop	51	2223	60457	1	45.481541	9.413
1536	1537	lounge	51	2557	80750	1	45.000702	7.682
1537	1538	pop	51	1766	54276	1	40.323410	17.568

1538 rows × 9 columns

In [4]:

```
df=df[['km','price']]
df.columns=['Km','Price']
```

In [5]:

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

Out[5]:

	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 [6]:

```
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 [7]:

```
df.describe()
```

Out[7]:

	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 [8]:

```
df.columns
```

Out[8]:

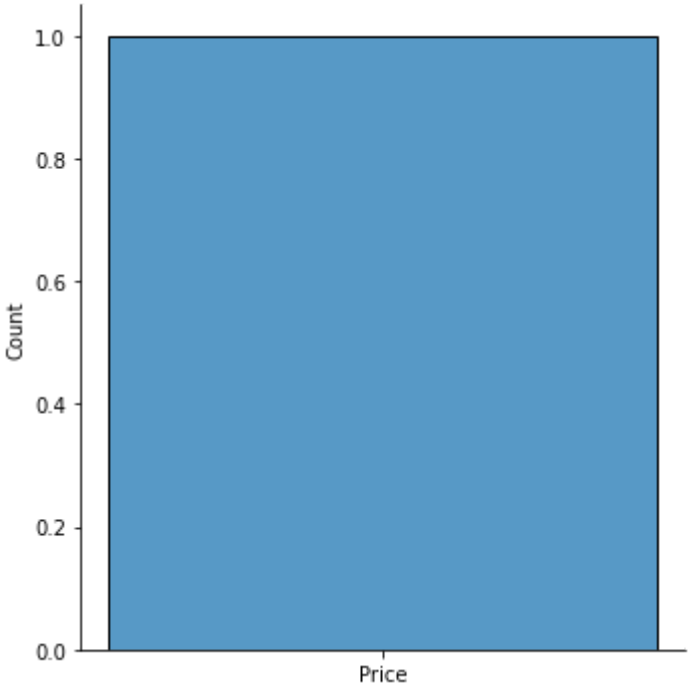
```
Index(['Km', 'Price'], dtype='object')
```

In [9]:

```
sns.displot(['Price'])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1aa8e81f580>
```

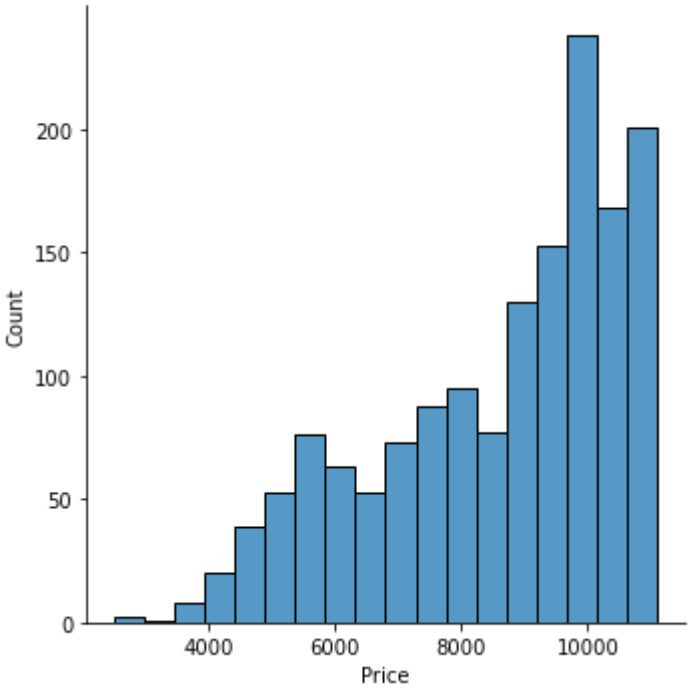


In [11]:

```
sns.displot(df['Price'])
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1aa898677f0>
```



In [13]:

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

In [15]:

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

```
C:\Users\shaik\AppData\Local\Temp\ipykernel_11660\1379821321.py:1: Setting
WithCopyWarning:
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-doc
s/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.dropna(inplace=True)
```

In [16]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=101)
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
```

```
[10749.70401206]
```

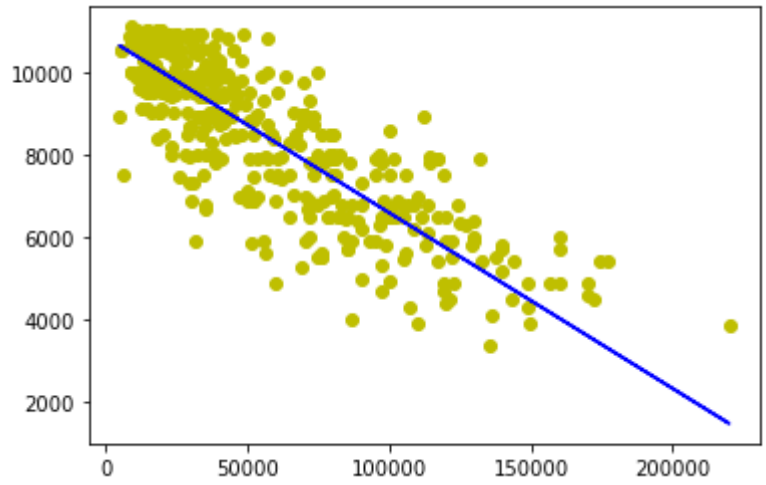
In [19]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

```
0.6922995403446177
```

In [20]:

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

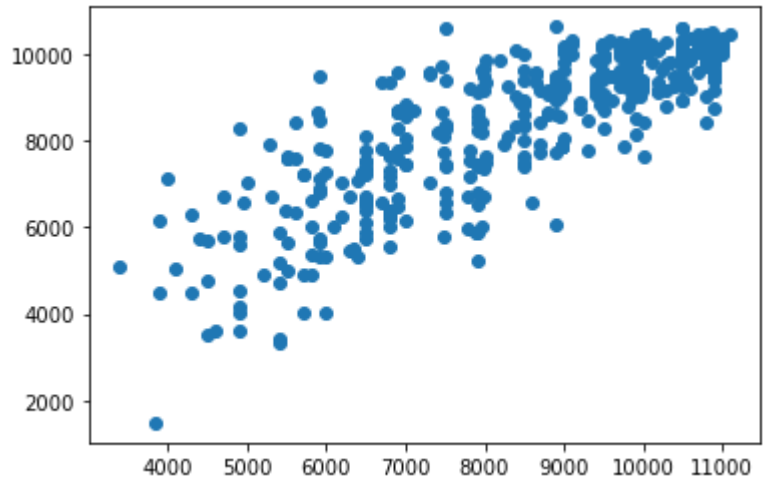


In [21]:

```
predictions=regr.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x1aa92cdf580>
```

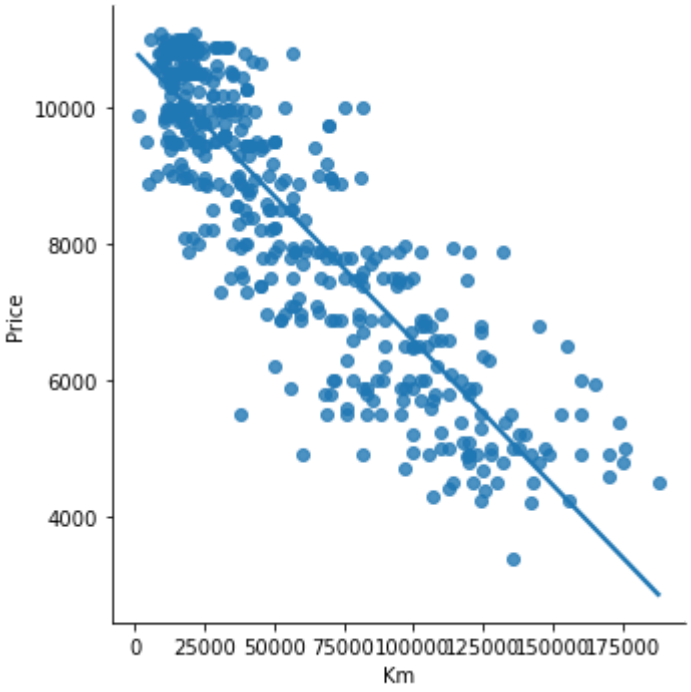


In [23]:

```
udf=df[:][:500]
sns.lmplot(x="Km",y="Price",data=udf,order=1,ci=None)
```

Out[23]:

```
<seaborn.axisgrid.FacetGrid at 0x1aa94709460>
```



In [24]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('MAE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 800.2442548738708
MSE: 1070052.9876072728
MAE: 1034.4336554884865
```

In [25]:

```
#accuracy
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

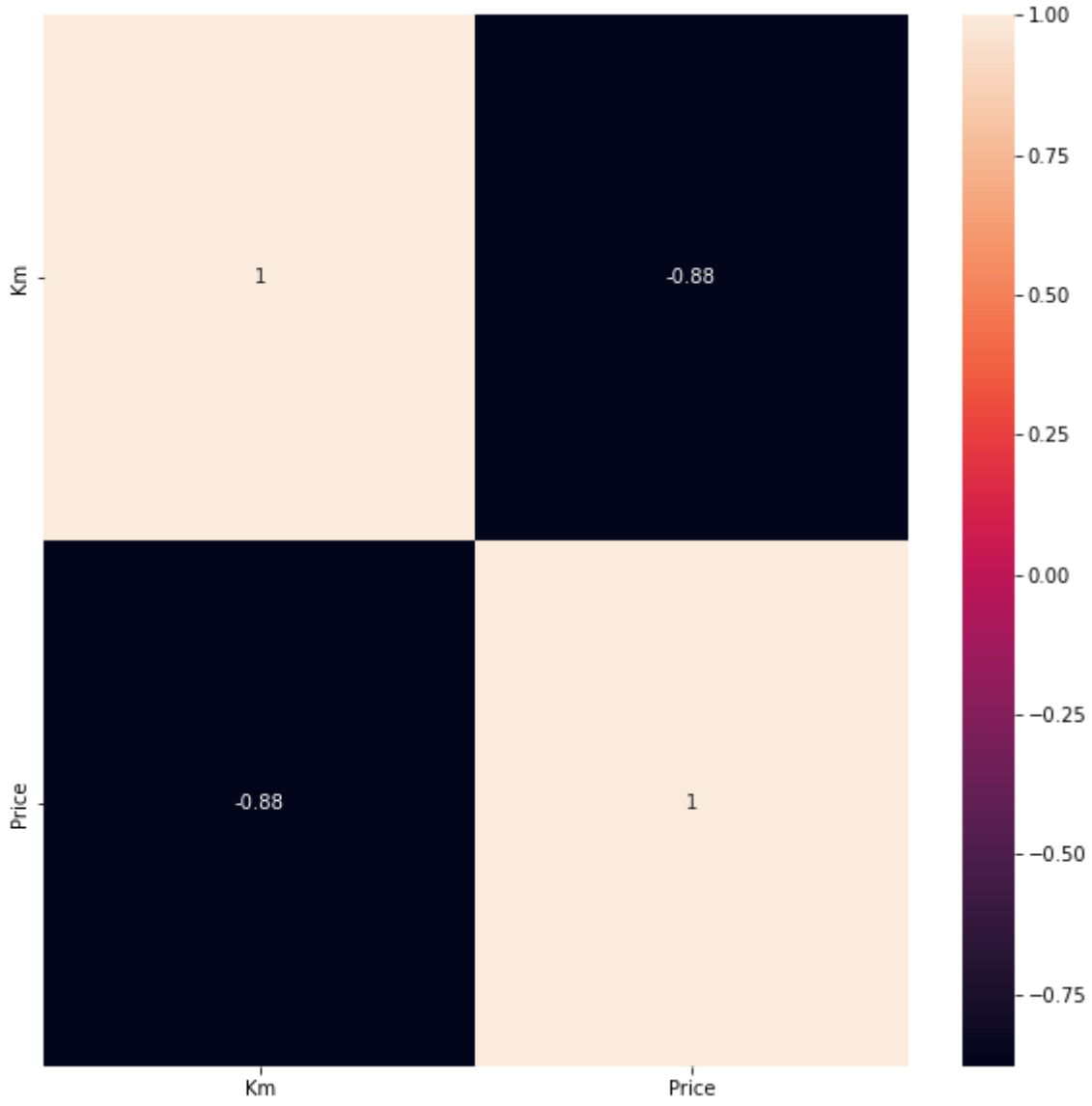
```
0.6922995403446177
```

In [27]:

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

Out[27]:

<AxesSubplot:>

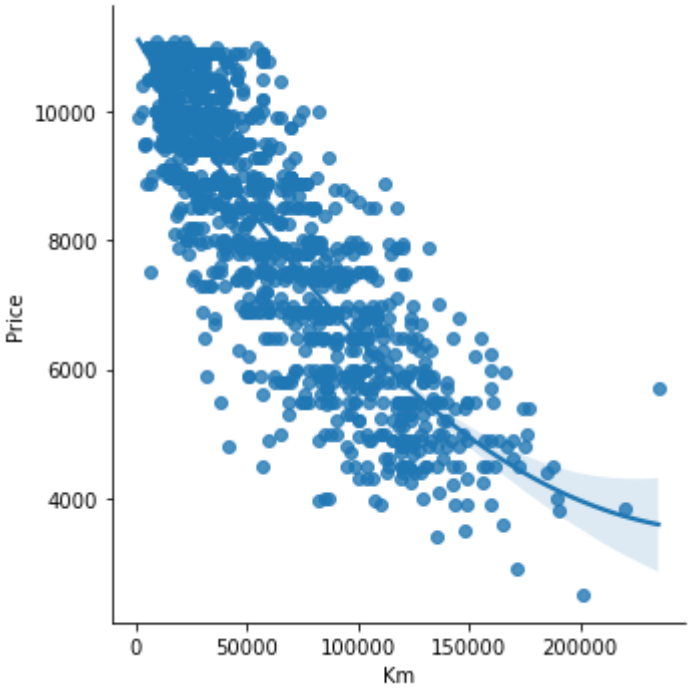


In [31]:

```
sns.lmplot(x="Km",y="Price",data=df,order=2)
```

Out[31]:

```
<seaborn.axisgrid.FacetGrid at 0x1aa94861fa0>
```



In [35]:

```
udf.fillna(method='ffill',inplace=True)
x=np.array(df['Km']).reshape(-1,1)
y=np.array(df['Price']).reshape(-1,1)
udf.dropna(inplace=True)
```

In [36]:

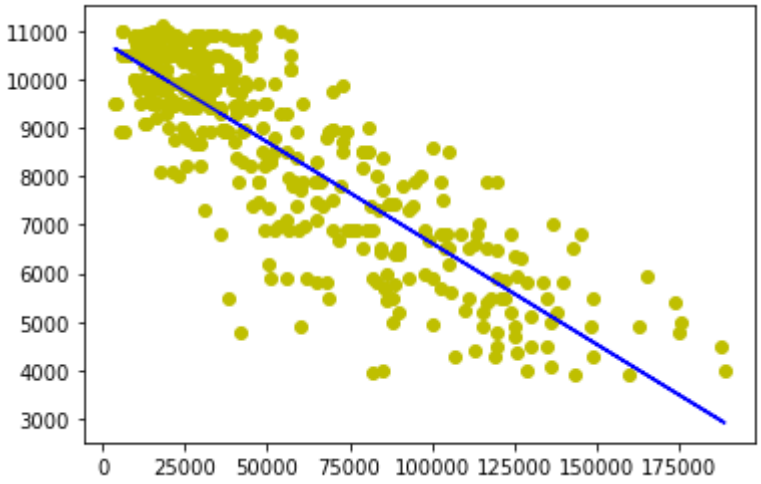
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[36]:

```
LinearRegression()
```

In [37]:

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

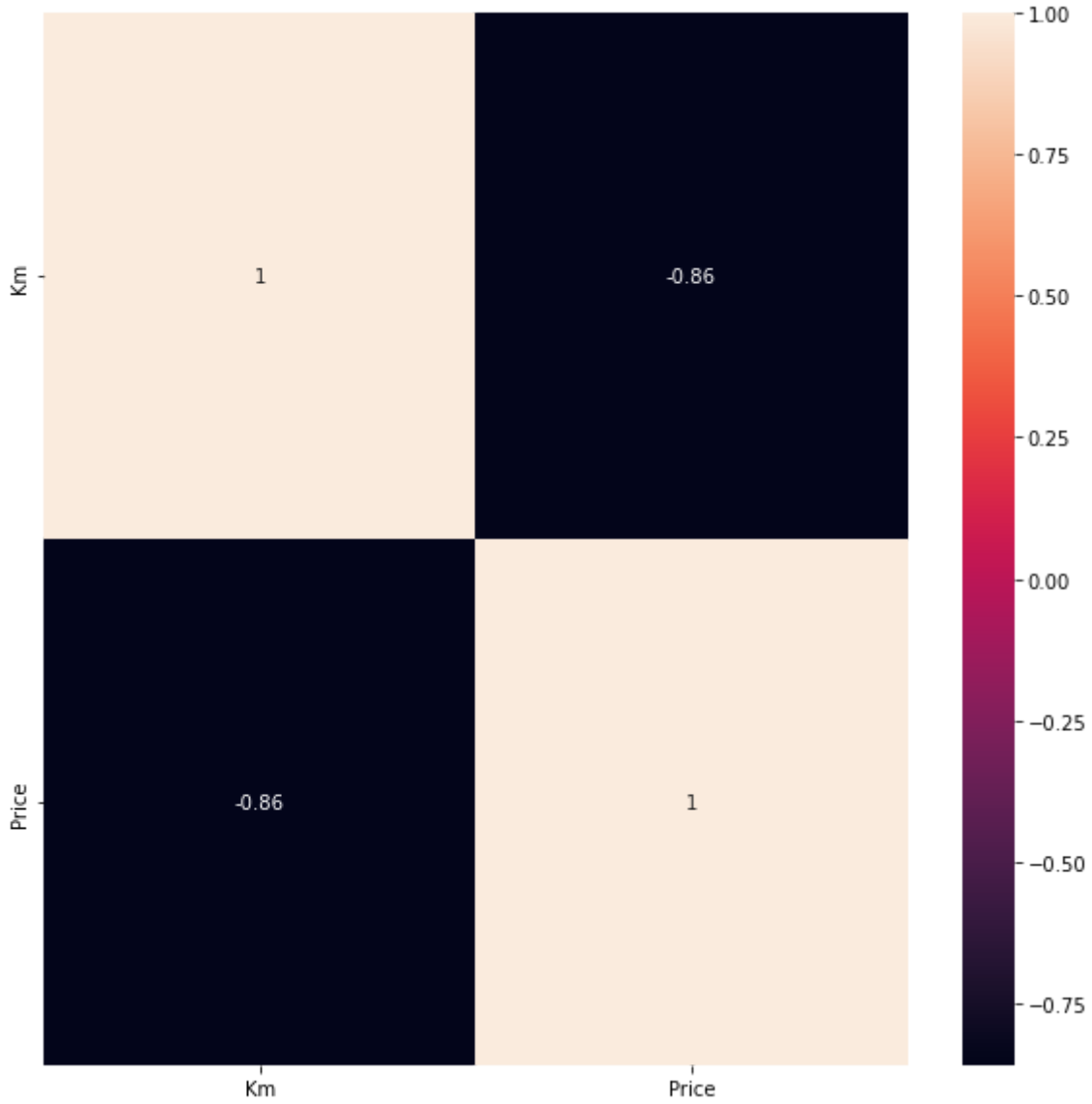


In [38]:

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

Out[38]:

<AxesSubplot:>



In [40]:

```
from sklearn.preprocessing import StandardScaler
features=df.columns[0:2]
target=df.columns[-1]
X=df[features].values
y=df[target].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=17)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
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 [41]:

```
#Linear regression model
regr=LinearRegression()
regr.fit(X_train,y_train)
actual=y_test #actual value
train_score_regr=regr.score(X_train,y_train)
test_score_regr=regr.score(X_test,y_test)
print("\nLinear model:\n")
print("The train score for Linear model is {}".format(train_score_regr))
print("The test score for Linear model is {}".format(test_score_regr))
```

```
Linear model:

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

In [42]:

```
#ridge regression model
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test score for 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 0.9997095924476731
The test score for ridge model is 0.9997198323998524
```

In [43]:

```
#using the linear cv model for ridge regression
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(ridge_cv.score(X_train,y_train))
print(ridge_cv.score(X_test,y_test))
```

```
0.9999999999999676
0.9999999999999686
```

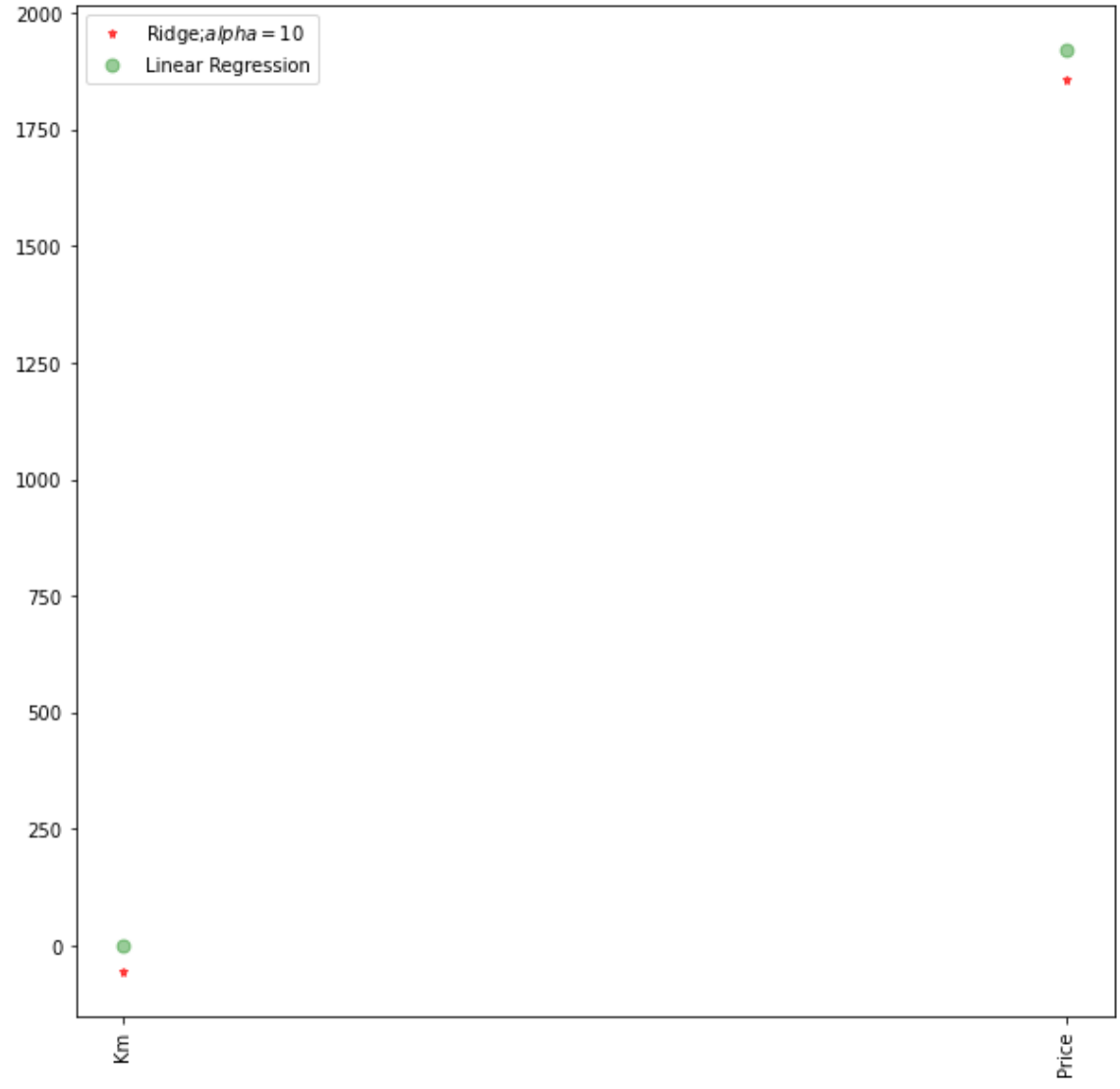
In [44]:

```
#using the linear cv model for lasso regression
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_train,y_train)
#score
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.9999999877496772
0.9999999874481674
```

In [48]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',
         markersize=5,color='red',label=r'Ridge;$alpha=10$',zorder=7)
plt.plot(features,regr.coef_,alpha=0.4,linestyle='none',marker='o',
         markersize=7,color='green',label='Linear Regression',zorder=7)
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [45]:

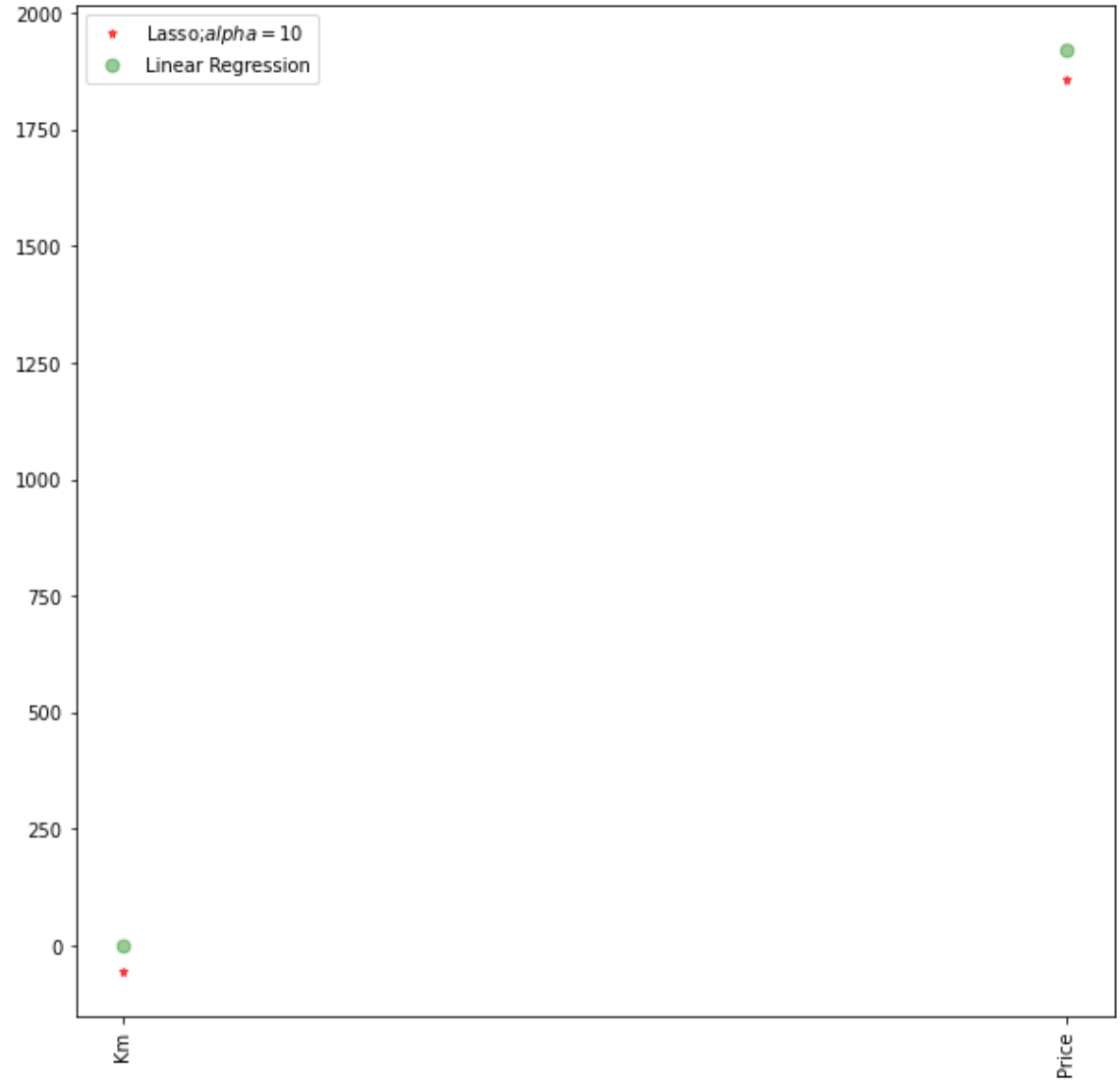
```
#Lasso regression model
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
#train and test score for ridge regression
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nLasso model:\n")
print("The train score for lasso model is {}".format(train_score_lasso))
print("The test score for lasso model is {}".format(test_score_lasso))
```

```
Lasso model:

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

In [49]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',
         markersize=5,color='red',label=r'Lasso;$alpha=10$',zorder=7)
plt.plot(features,regr.coef_,alpha=0.4,linestyle='none',marker='o',
         markersize=7,color='green',label='Linear Regression',zorder=7)
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

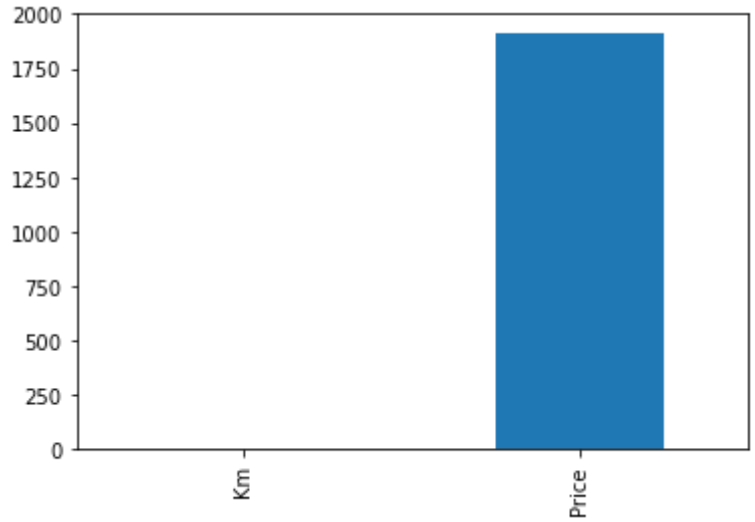


In [51]:

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

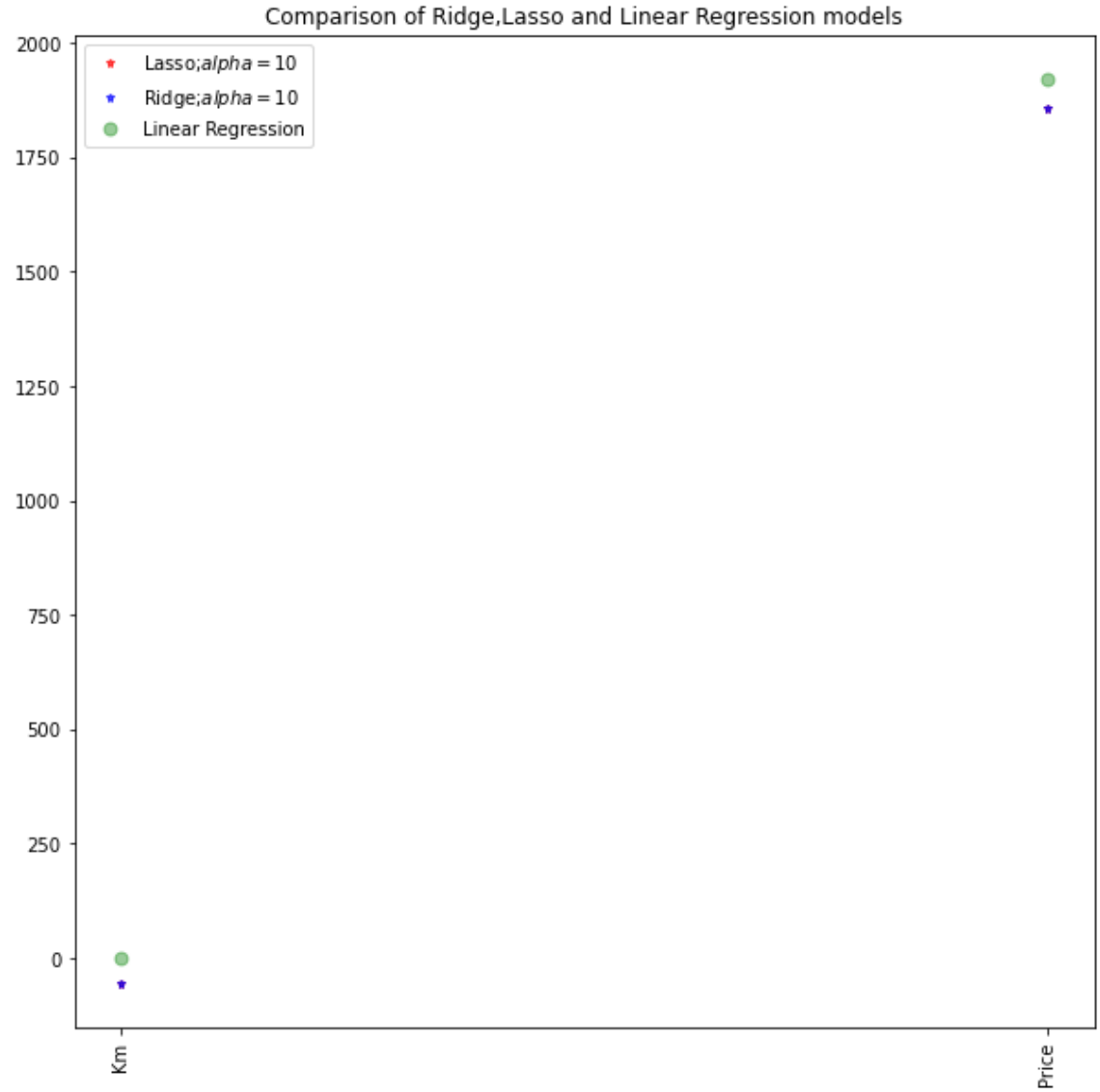
Out[51]:

<AxesSubplot:>



In [53]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',
         markersize=5,color='red',label=r'Lasso;$alpha=10$',zorder=7)
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',
         markersize=5,color='blue',label=r'Ridge;$alpha=10$',zorder=7)
plt.plot(features,regr.coef_,alpha=0.4,linestyle='none',marker='o',
         markersize=7,color='green',label='Linear Regression',zorder=7)
plt.xticks(rotation=90)
plt.legend()
plt.title("Comparison of Ridge,Lasso and Linear Regression models")
plt.show()
```



In [57]:

```
#Elastic Net
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
y_pred_elastic=regr.predict(X_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
[-1.57759052e-05  9.99719867e-01]
3.24479152506683
Mean Squared Error on test set 77316211.5587428
```