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

In [2]:
```
df=pd.read_csv(r"C:\Users\venka\OneDrive\Documents\fiat500_VehicleSelection_Da
print(df)
```

```
        ID   model  engine_power  age_in_days      km  previous_owners
0        1  lounge            51          882   25000                1  \
1        2     pop            51         1186   32500                1
2        3   sport            74         4658  142228                1
3        4  lounge            51         2739  160000                1
4        5     pop            73         3074  106880                1
...    ...     ...           ...          ...     ...              ...
1533  1534   sport            51         3712  115280                1
1534  1535  lounge            74         3835  112000                1
1535  1536     pop            51         2223   60457                1
1536  1537  lounge            51         2557   80750                1
1537  1538     pop            51         1766   54276                1

            lat        lon  price
0     44.907242   8.611560   8900
1     45.666359  12.241890   8800
2     45.503300  11.417840   4200
3     40.633171  17.634609   6000
4     41.903221  12.495650   5700
...         ...        ...    ...
1533  45.069679   7.704920   5200
1534  45.845692   8.666870   4600
1535  45.481541   9.413480   7500
1536  45.000702   7.682270   5990
1537  40.323410  17.568270   7900

[1538 rows x 9 columns]
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
```

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.tail()
```

Out[6]:

| | Km | Price |
|---|---|---|
| **1533** | 115280 | 5200 |
| **1534** | 112000 | 4600 |
| **1535** | 60457 | 7500 |
| **1536** | 80750 | 5990 |
| **1537** | 54276 | 7900 |

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.describe()
```

Out[8]:

|  | 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 [9]:
```
df.shape
```

Out[9]:
```
(1538, 2)
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Km       0
Price    0
dtype: int64
```

In [11]:
```
x=np.array(df['Km']).reshape(-1,1)
y=np.array(df['Price']).reshape(-1,1)
```

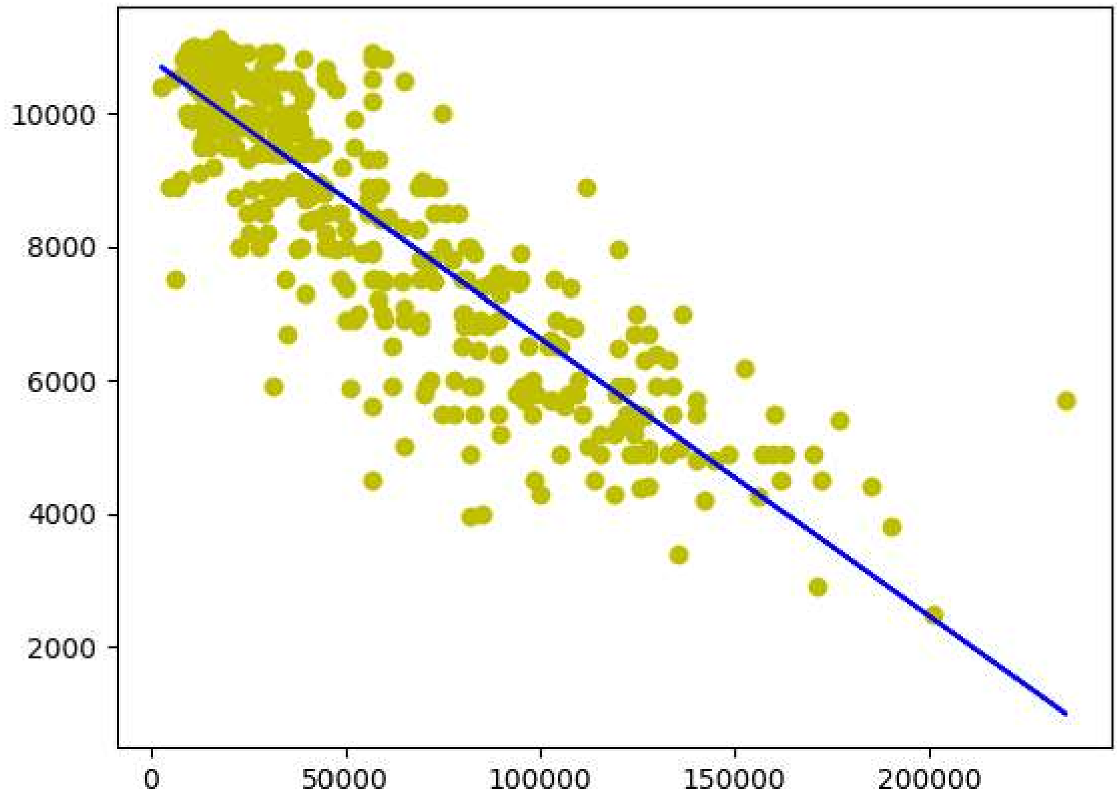
In [12]:
```
df.dropna(inplace=True)
```

In [13]:
```
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)
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

```
0.7522535193571669
```

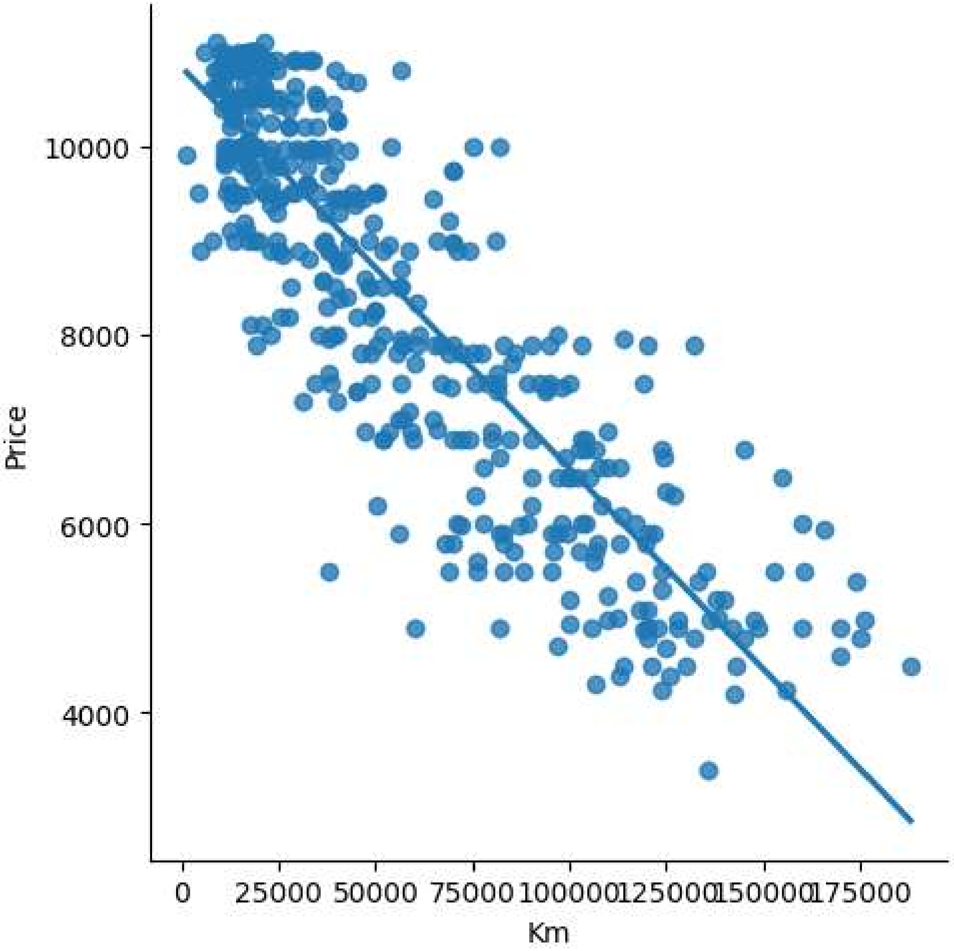
In [14]:

```
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 [15]:
```
udf=df[:][:500]
sns.lmplot(x="Km",y="Price",data=udf,order=1,ci=None)
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x21141ef4110>
```

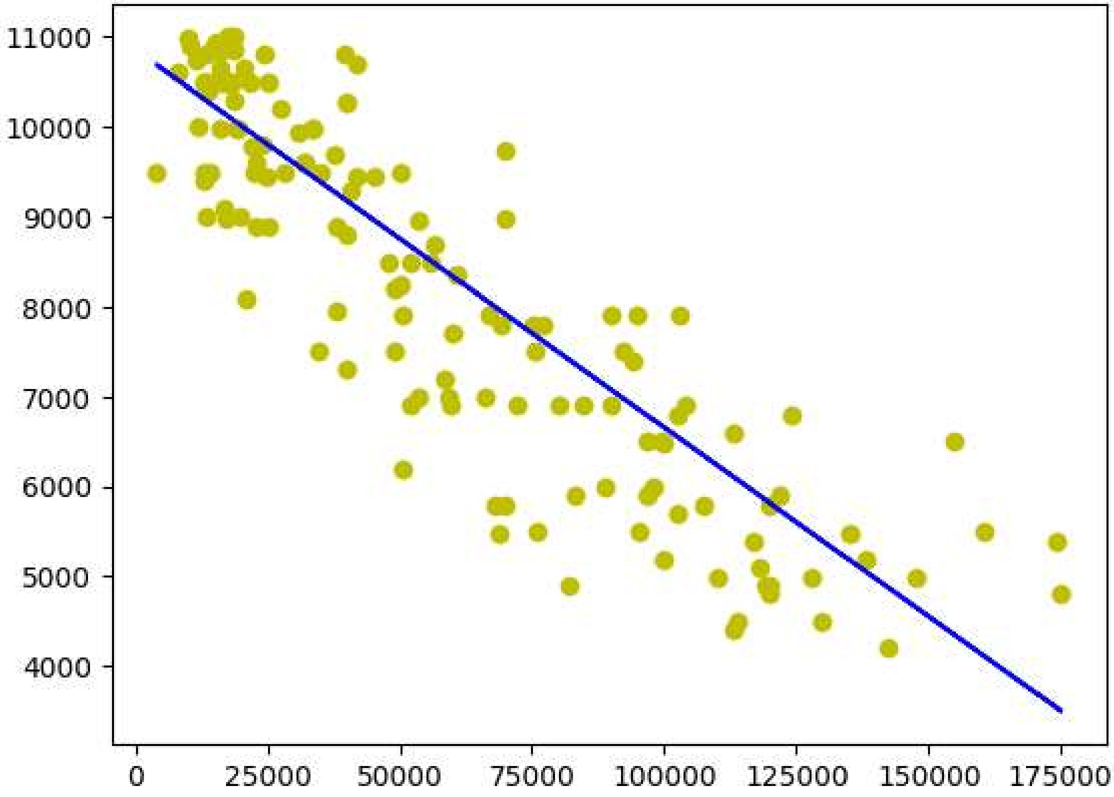


In [16]:
```
udf.fillna(method='ffill',inplace=True)
X=np.array(udf['Km']).reshape(-1,1)
y=np.array(udf['Price']).reshape(-1,1)
udf.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
regr.fit(X_train,y_train)
```

Out[16]:
```
LinearRegression()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

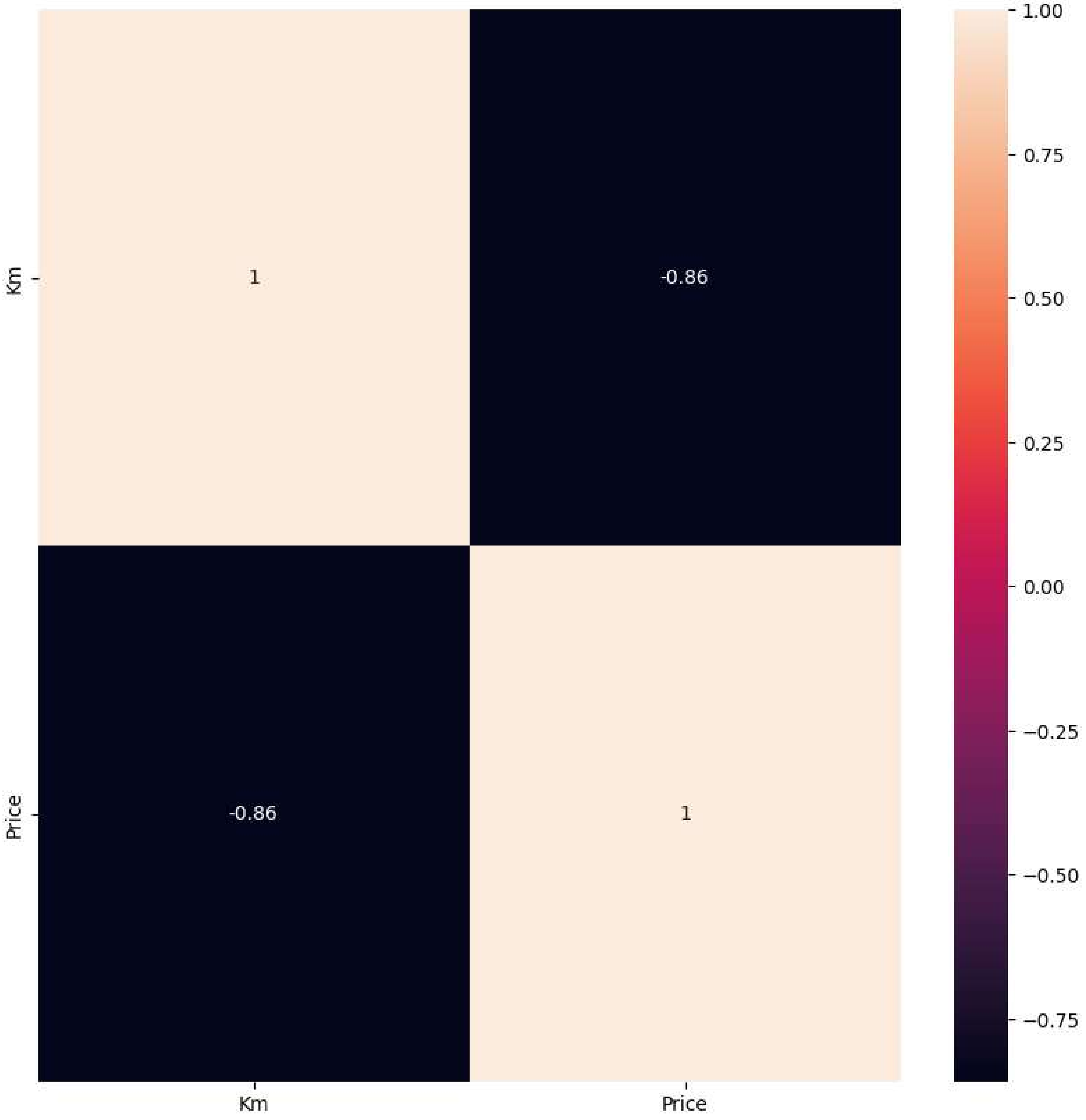
In [17]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



In [18]:
```
from sklearn.linear_model import Ridge,Lasso,RidgeCV,LassoCV
```

In [19]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

Out[19]:
```
<Axes: >
```



In [20]:

```
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
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 [21]:

```
#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 [22]:

```
#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 [23]:

```
#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))
```

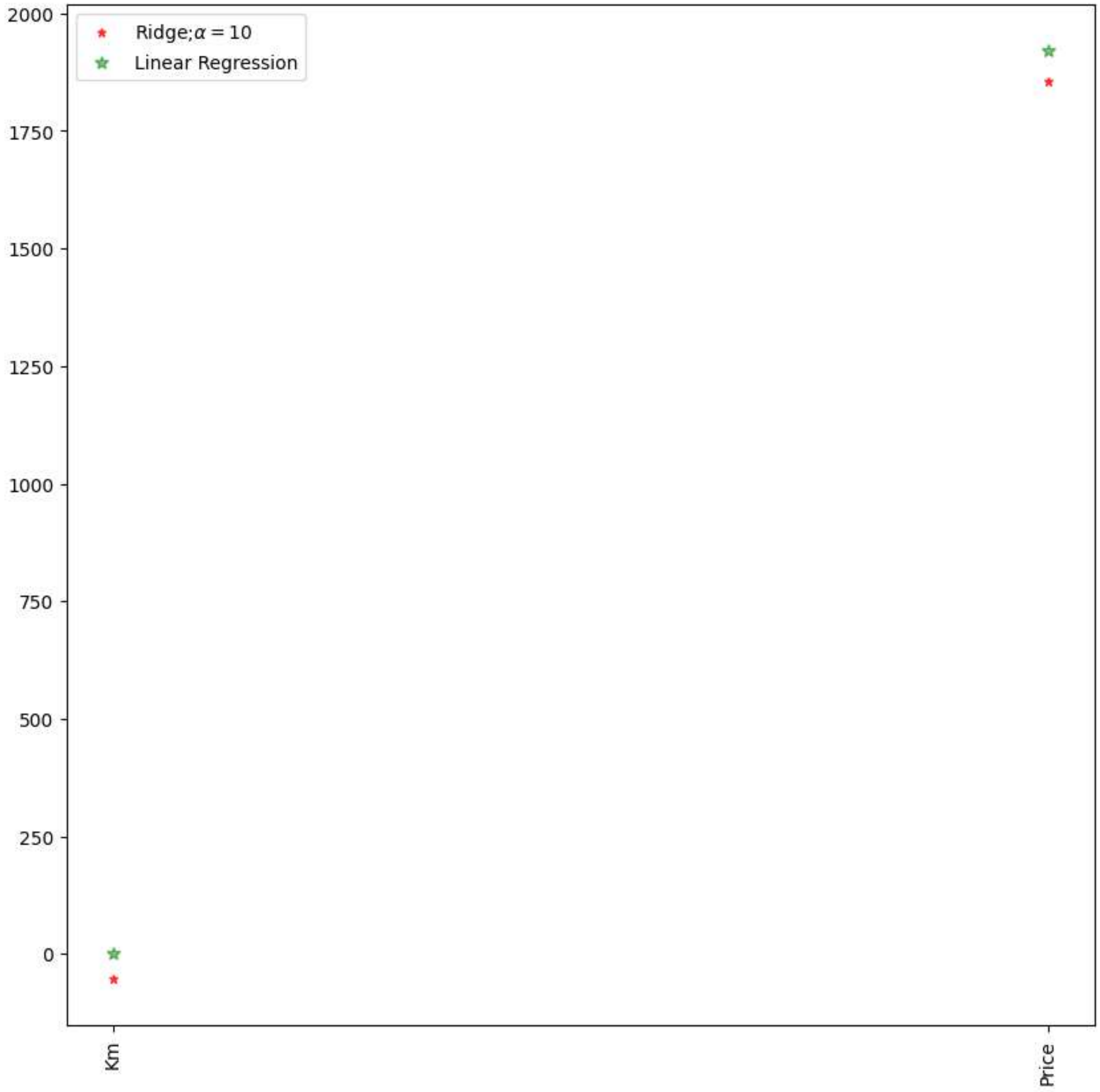
```
0.999999999999966
0.9999999999999674
```

In [24]:

```
#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_tr
#score
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.9999999877496772
0.9999999874481674
```

In [36]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',marker
plt.plot(features,regr.coef_,alpha=0.5,linestyle='none',marker='*',markersize
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

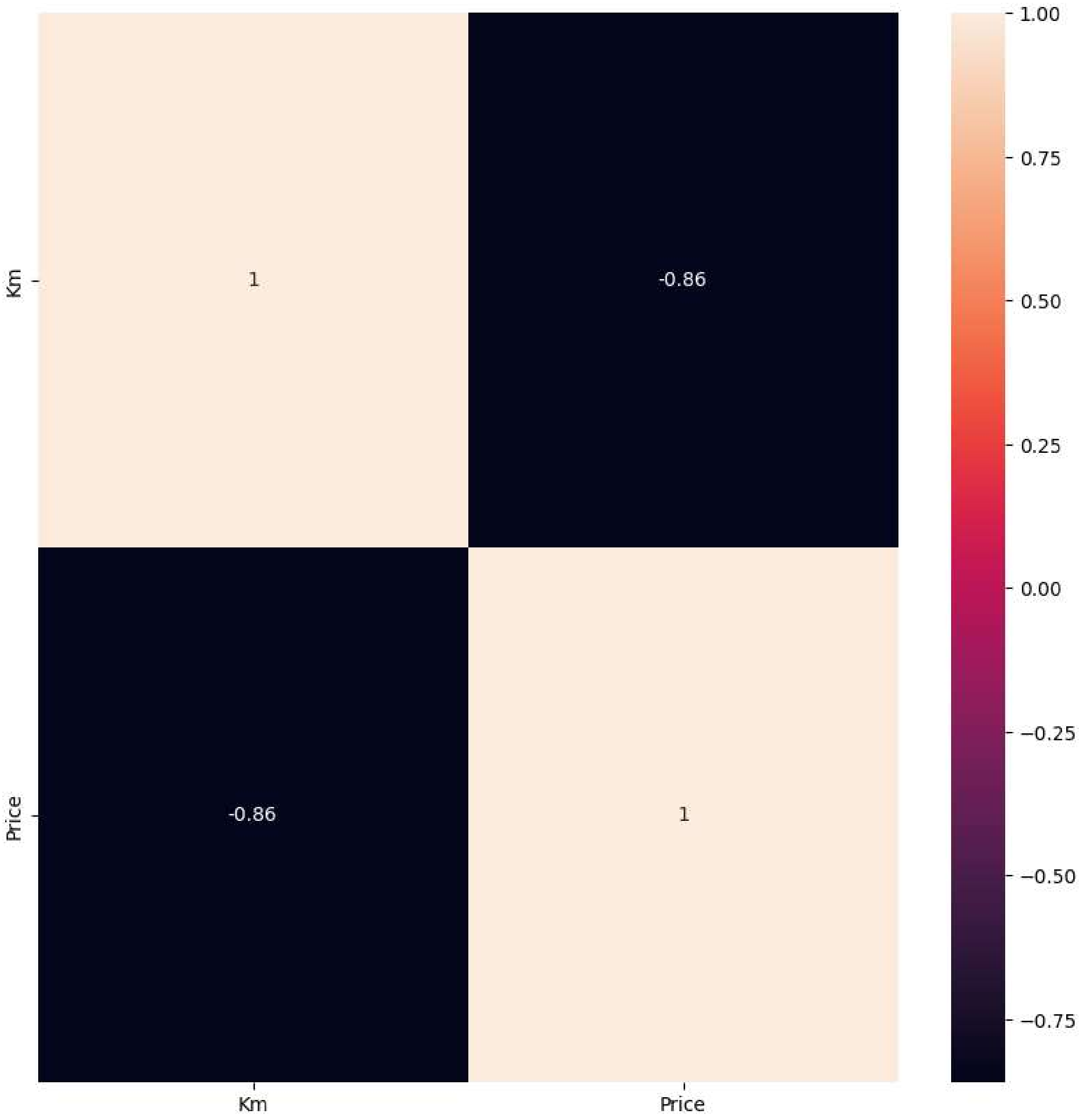


In [26]:

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

Out[26]:

```
<Axes: >
```



In [27]:

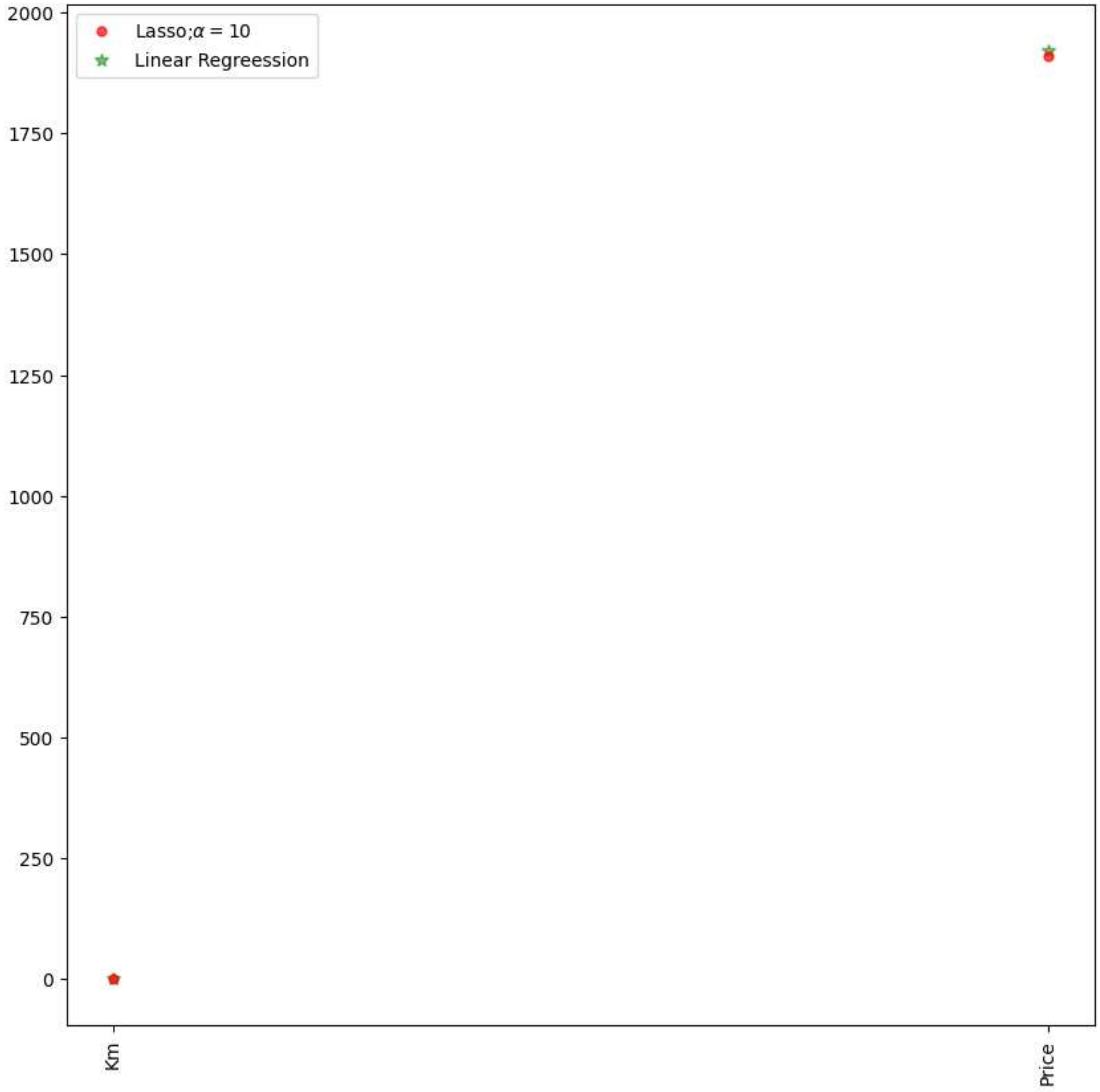
```
#lasso regression model
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
#train and test score for ridge regression
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nLasso model:\n")
print("The train score for lasso model is {}".format(train_score_lasso))
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 [35]:
```
plt.figure(figsize=(10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='o',marker
plt.plot(features,regr.coef_,alpha=0.5,linestyle='none',marker='*',markersize
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

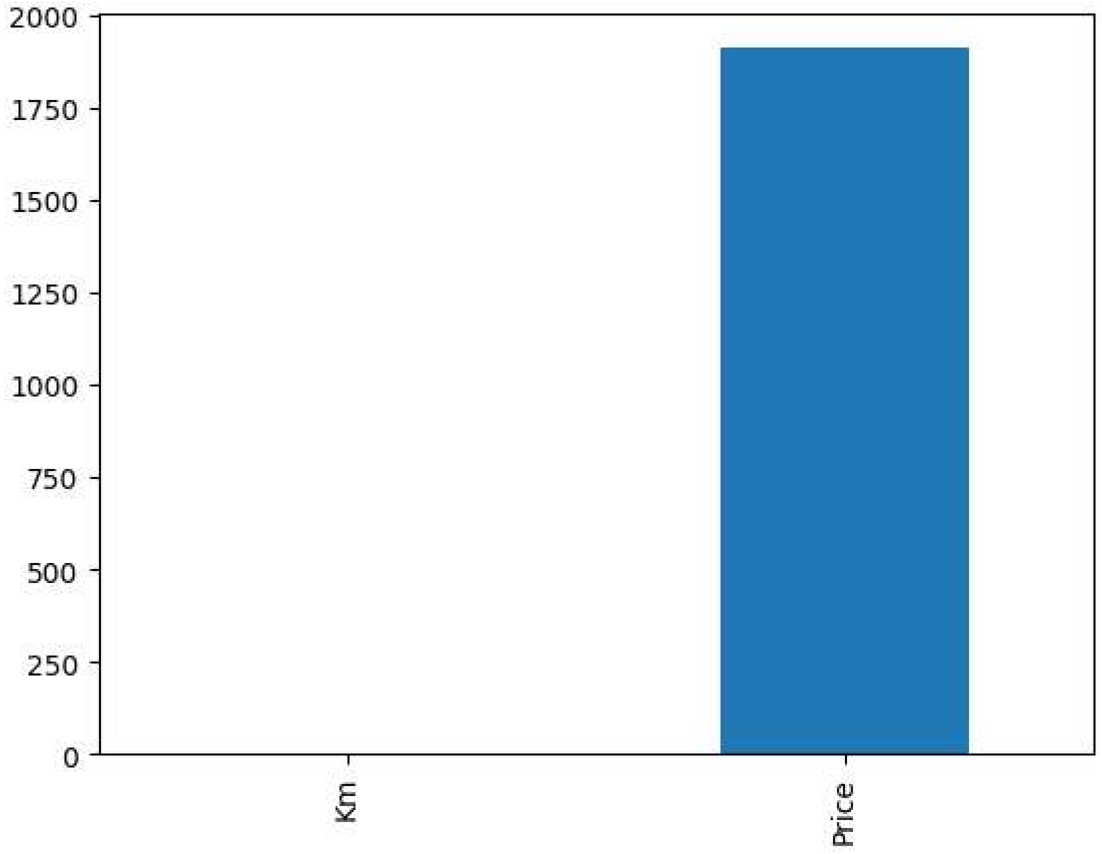


In [29]:

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

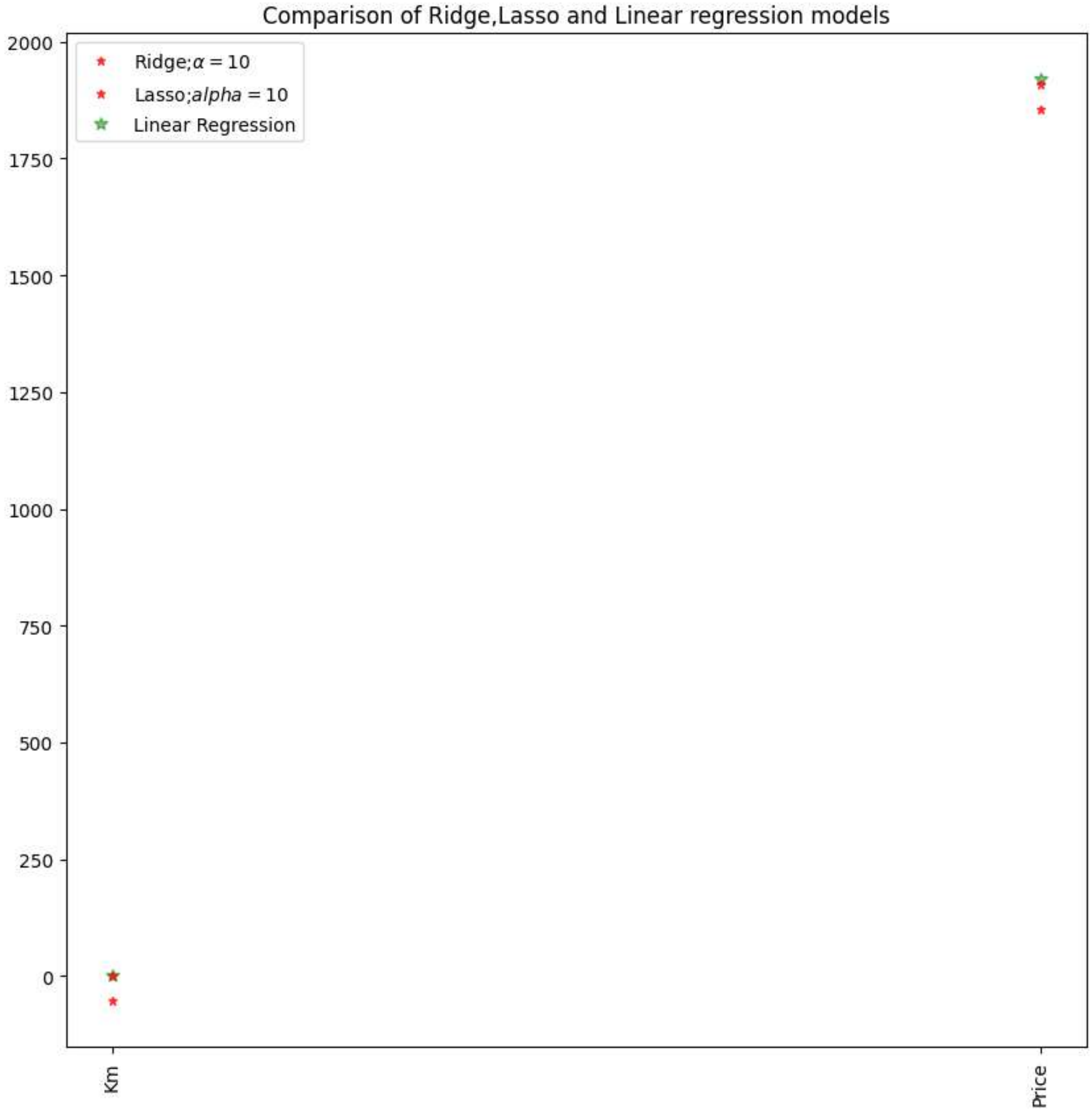
Out[29]:

```
<Axes: >
```



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='*',marker
#add plot for lasso regression
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',marker
#add plot for linear model
plt.plot(features,regr.coef_,alpha=0.5,linestyle='none',marker='*',markersize
#rotate axis
plt.xticks(rotation=90)
plt.legend()
plt.title("Comparison of Ridge,Lasso and Linear regression models")
plt.show()
```



```
In [ ]:
```