In [2]:
```python
import numpy as np
import pandas as pd
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 [3]:
```python
df=pd.read_csv(r"C:\Users\HP\Downloads\fiat500_VehicleSelection_Dataset (2).cs
df
```

Out[3]:

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

1538 rows × 9 columns

In [4]:
```python
df.head()
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	890(
1	2	pop	51	1186	32500	1	45.666359	12.241890	880(
2	3	sport	74	4658	142228	1	45.503300	11.417840	420(
3	4	lounge	51	2739	160000	1	40.633171	17.634609	600(
4	5	pop	73	3074	106880	1	41.903221	12.495650	570(

In [5]:
```
df.tail()
```

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
1533	1534	sport	51	3712	115280	1	45.069679	7.70492
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687
1535	1536	pop	51	2223	60457	1	45.481541	9.41348
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227
1537	1538	pop	51	1766	54276	1	40.323410	17.56827

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

Out[6]:
```
(1538, 9)
```

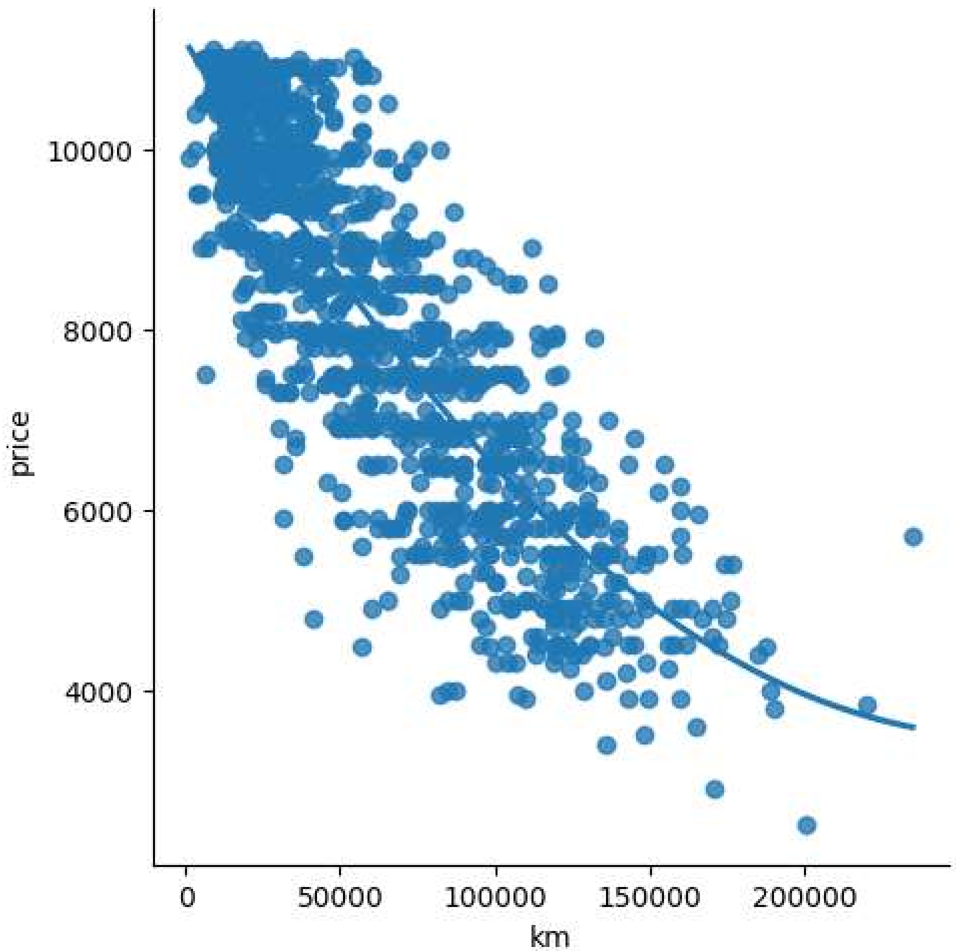
In [7]:
```
df.describe
```

Out[7]:
```
<bound method NDFrame.describe of          ID   model  engine_power  age_in_da
ys      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 [8]:
```
df.isna().any()
```

Out[8]:
```
ID                  False
model               False
engine_power        False
age_in_days         False
km                  False
previous_owners     False
lat                 False
lon                 False
price               False
dtype: bool
```

In [9]:
```
sns.lmplot(x='km',y='price',data=df,order=2,ci=None)
plt.show()
```

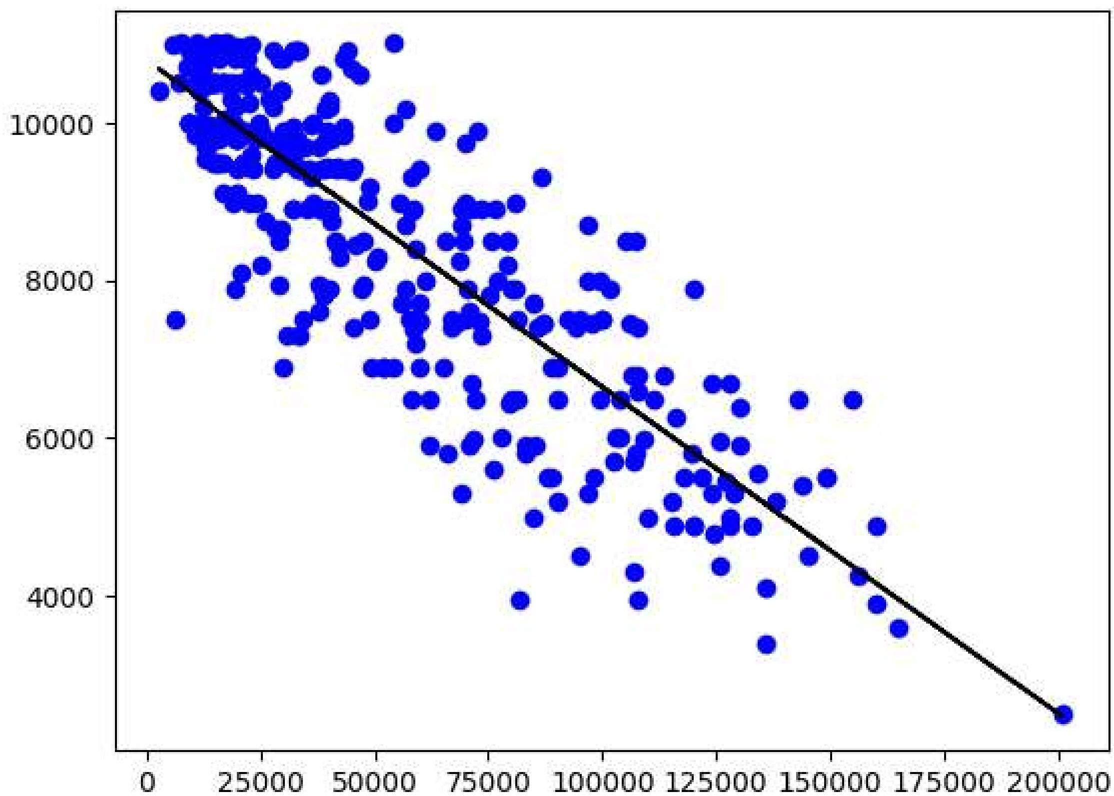


In [10]:
```
x=np.array(df['km']).reshape(-1,1)
y=np.array(df['price']).reshape(-1,1)
```

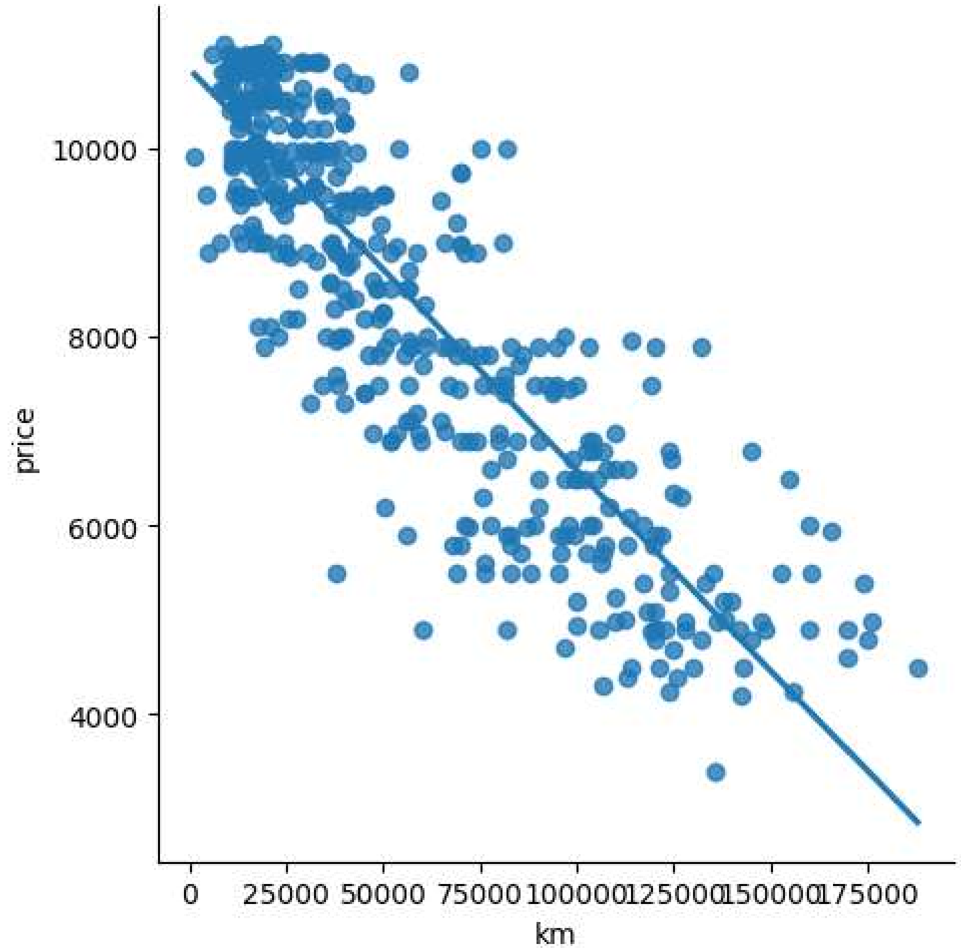
In [11]:
```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
#splitting data into train and test
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.741062312488328
```

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)
plt.show()
```



In [14]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [15]:
```
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: 0.741062312488328
```

In [16]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

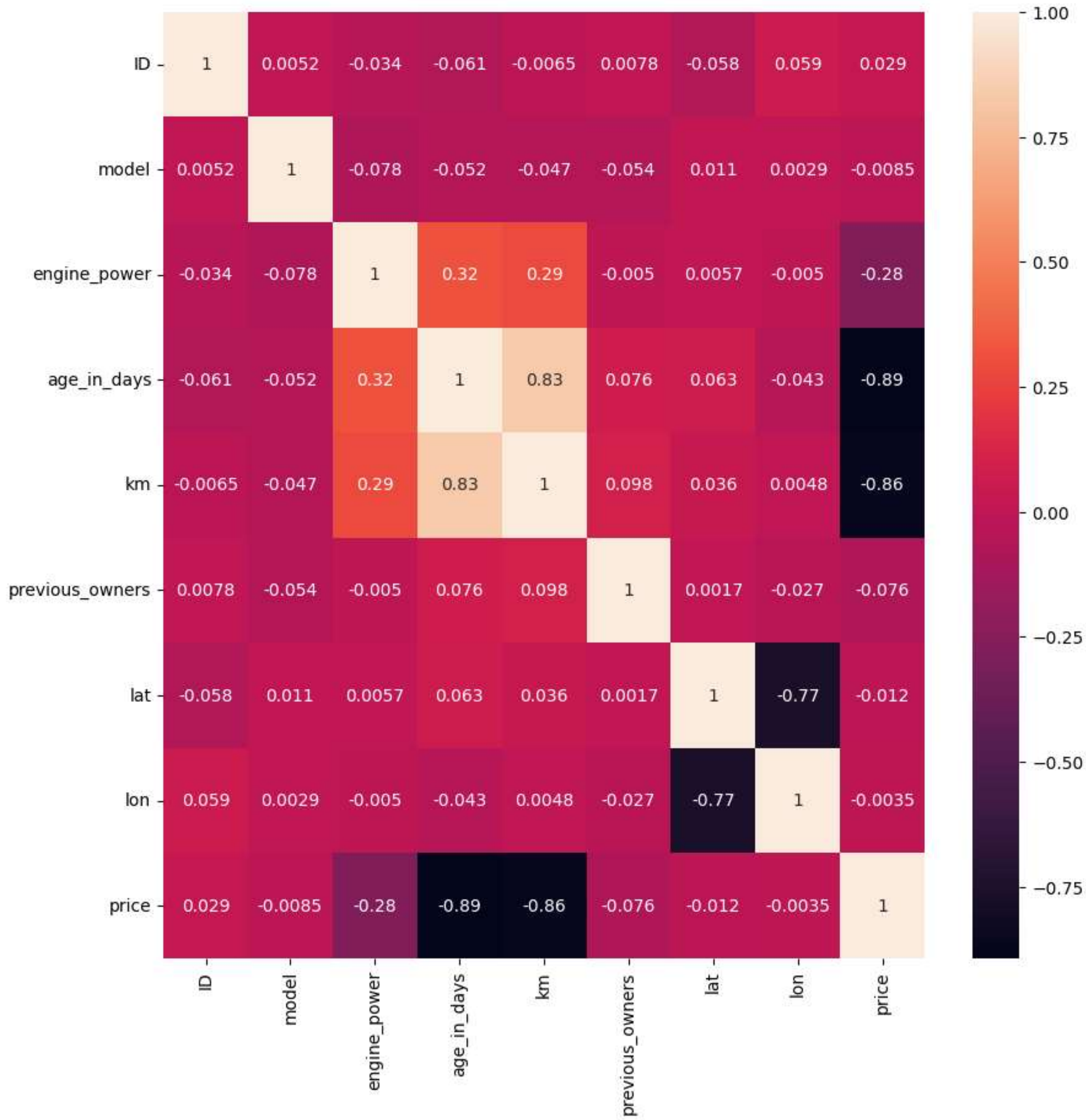
In [19]:
```
converter={"model":{"sport":1,"lounge":2,"pop":3}}
df=df.replace(converter)
df
```

Out[19]:

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

1538 rows × 9 columns

In [20]:
```python
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
plt.show()
```



In [21]:
```
features=df.columns[0:1]
target=df.columns[-1]
X = df[features].values
y = df[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, randc
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 (1153, 1)
The dimension of X_test is (385, 1)
```

In [22]:
```
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 0.00310286926477088
The test score for lr model is -0.008405634316406507
```

In [23]:
```
#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))
```

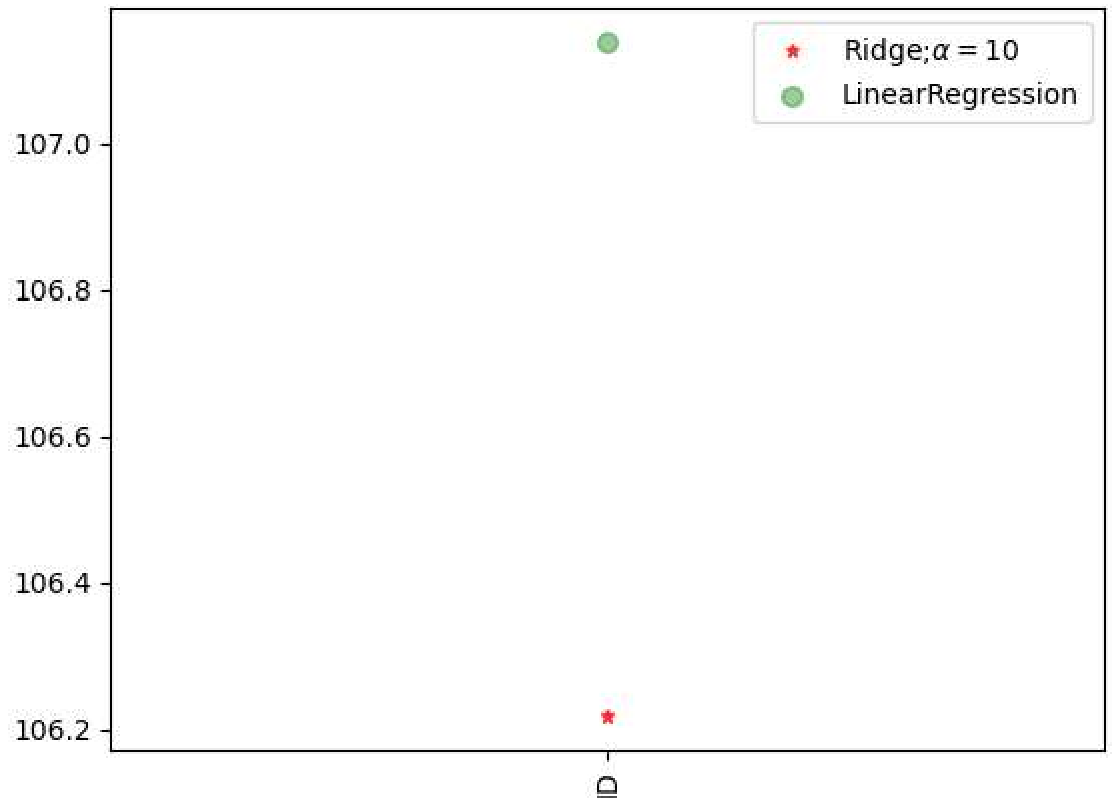
```

Ridge Model:

The train score for ridge model is 0.0031026398591535997
The test score for ridge model is -0.008307809466001403
```

In [24]:
```
plt.figure(figsize=(10,10))
```

Out[24]: <Figure size 1000x1000 with 0 Axes>

<Figure size 1000x1000 with 0 Axes>

In [26]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markers
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

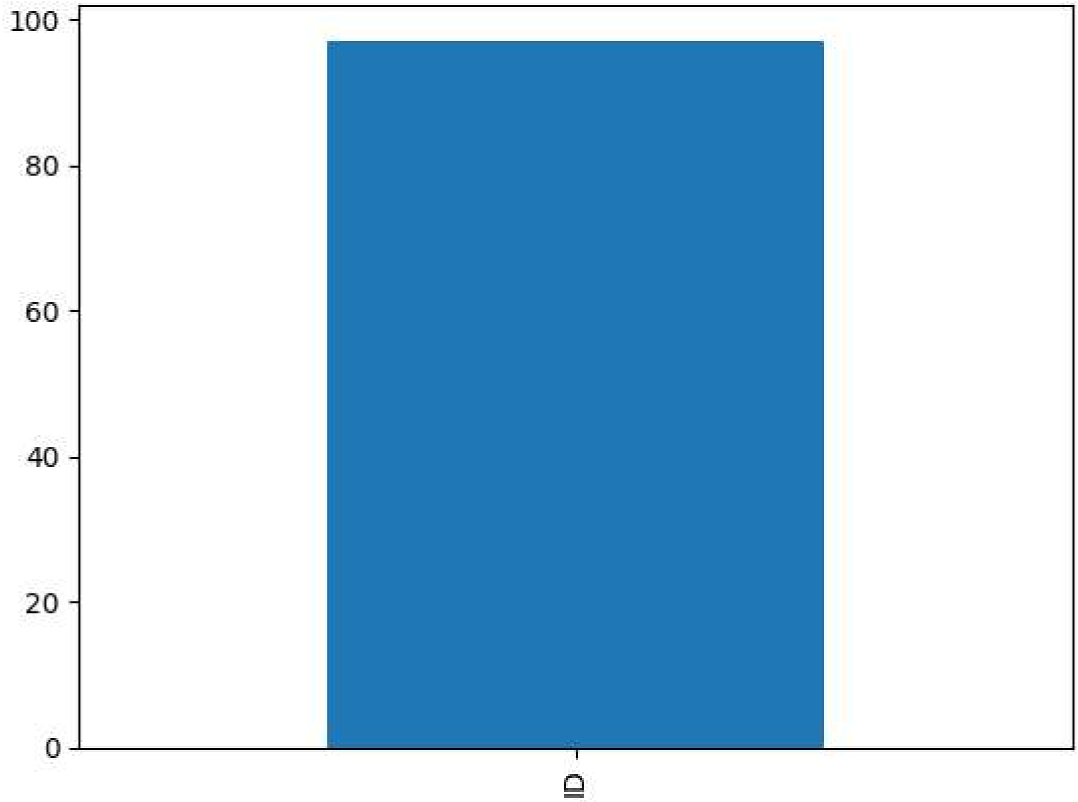


In [27]:
```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for ls model is 0.003075838461310987
The test score for ls model is -0.007367578602064606
```

In [37]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
```

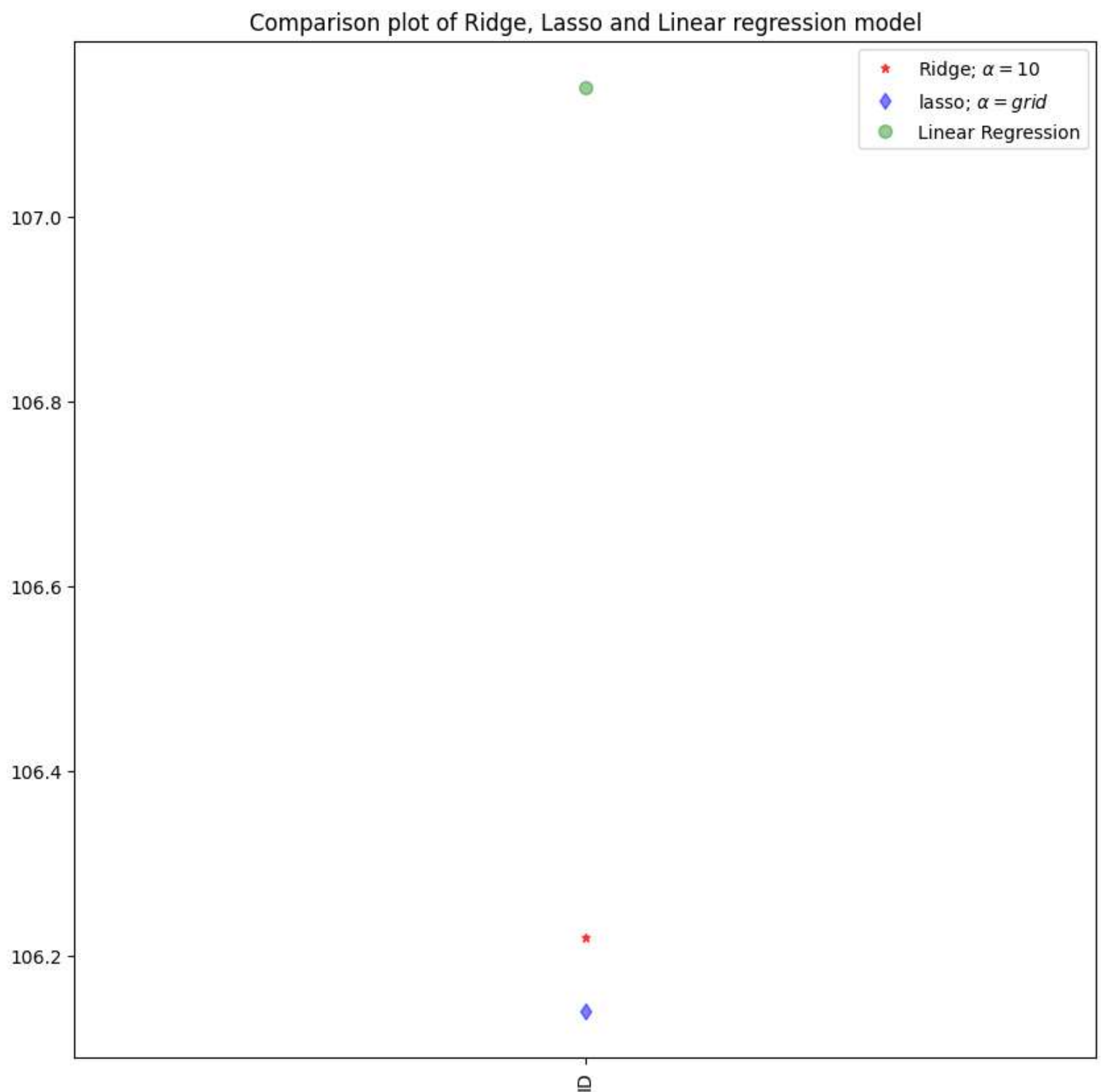
Out[37]: <Axes: >



In [29]:
```
#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).
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.0031025989567363688
-0.008299466692577973
```

In [34]:
```
#plot size

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



In [38]:

```
#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_t
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_
```

```
The train score for ridge model is 0.0031026398591535997
The train score for ridge model is -0.008307809466003402
```

In [39]:

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

```
[0.12455754]
8480.156871173602
```

In [40]:

```
y_pred_elastic=regr.predict(X_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
3708273.194830543
```