In [1]:
```
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]:
```
df=pd.read_csv(r"C:\Users\Jayadeep\Downloads\fiat500_VehicleSelection_Dataset (1).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.head()
```

Out[3]:

	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

In [4]:
```
df.tail()
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [5]:
```
df.shape
```

Out[5]: (1538, 9)

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

Out[6]:
```
<bound method NDFrame.describe of          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 [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

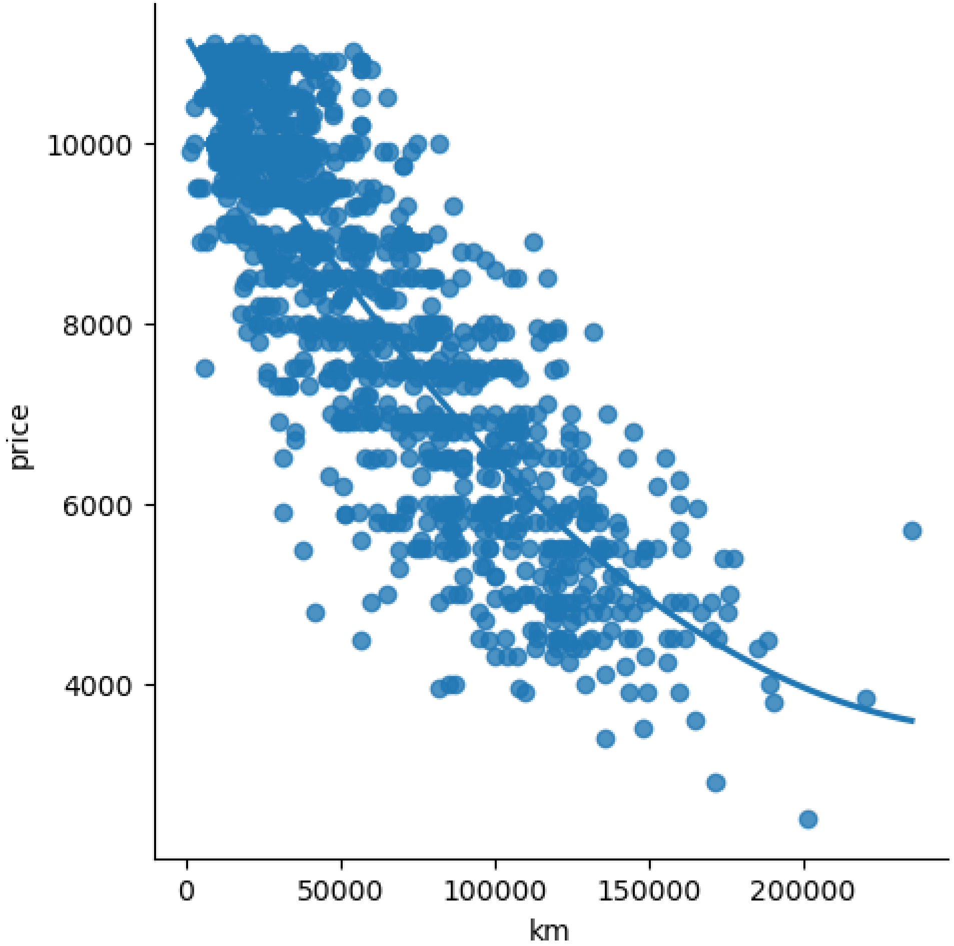
In [8]:
```
df.isna().any()
```

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]:
```
df.isna().any()
```

Out[9]:
```
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 [10]:
```
sns.lmplot(x='km',y='price',data=df,order=2,ci=None)
plt.show()
```



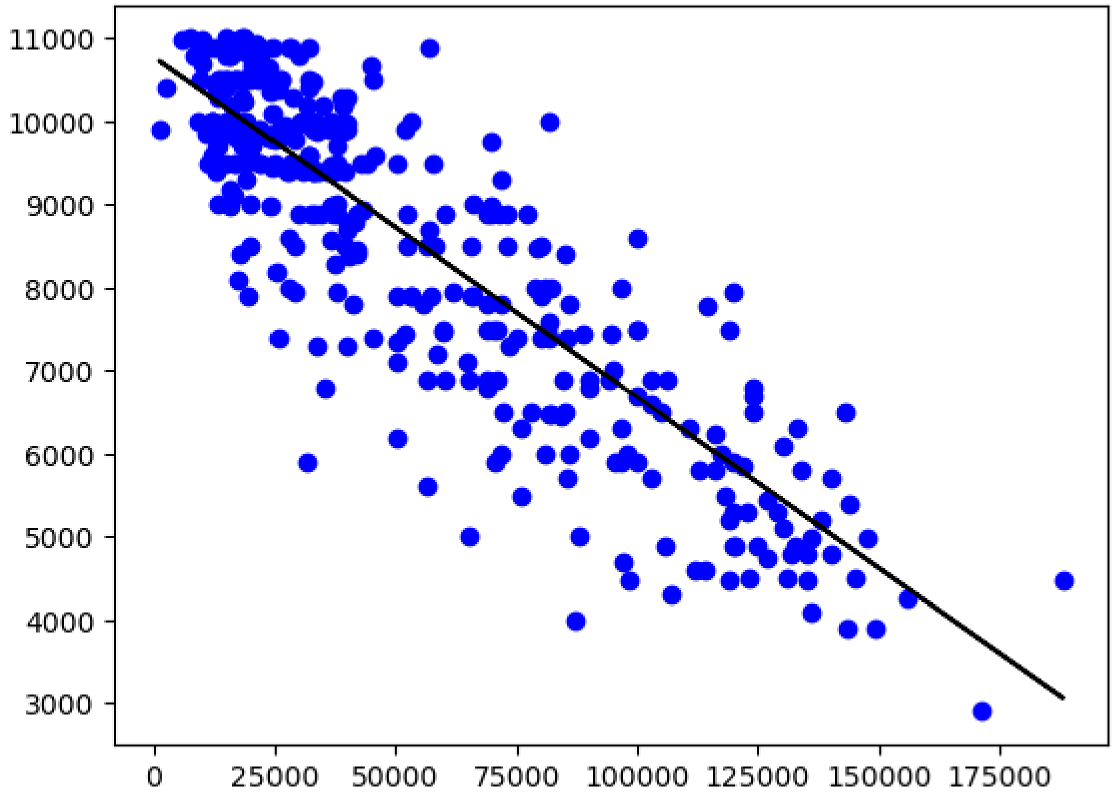
In [22]:
```
x=np.array(df['km']).reshape(-1,1)
y=np.array(df['price']).reshape(-1,1)
```

In [23]:
```
df.dropna(inplace=True)
```

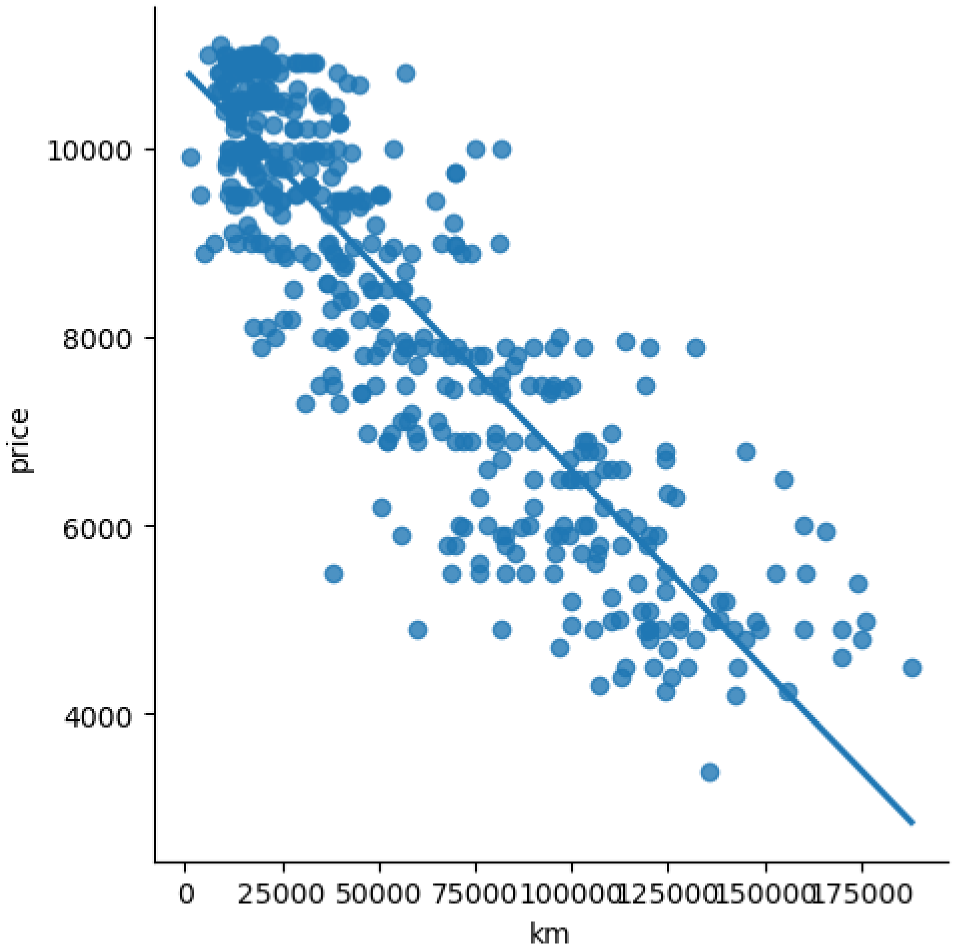
In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
#splitting data into train and test
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.7703956212905387
```

In [25]:
```
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 [26]:
```
df500=df[:][:500]
sns.lmplot(x="km",y="price",data=df500,order=1,ci=None)
plt.show()
```



In [27]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

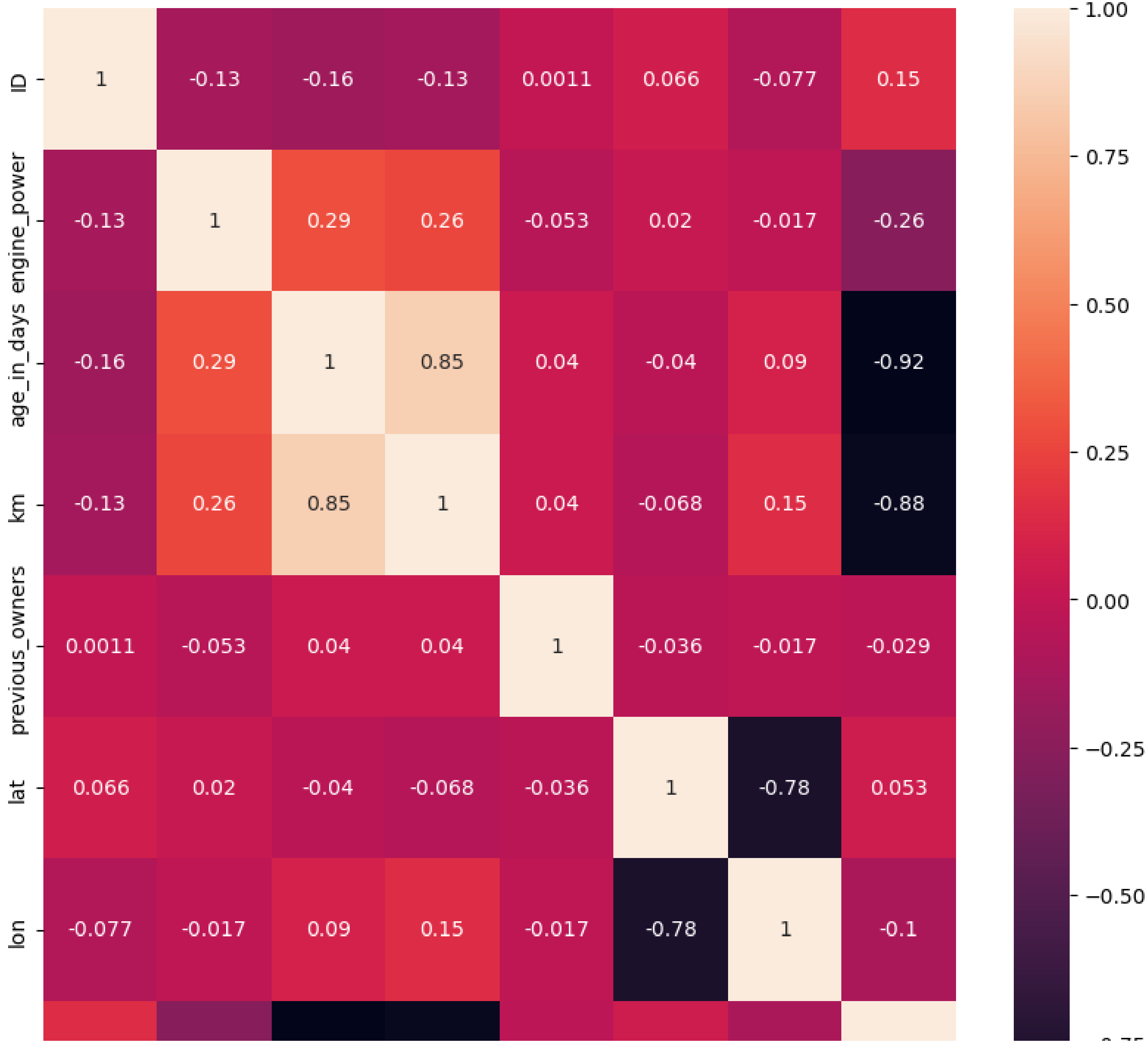
In [28]:
```
#train model
model=LinearRegression()
model.fit(x_train,y_train)
#Evaluation the model on the test set
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

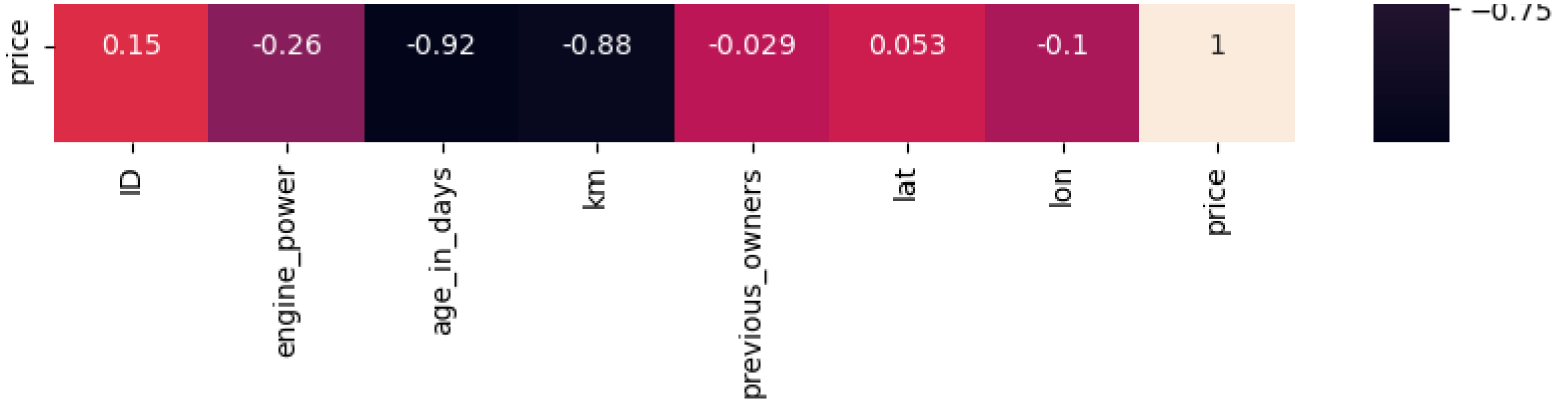
```
R2 score: 0.7703956212905387
```

Ridge and Lasso

In [29]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [30]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df500.corr(), annot = True)
plt.show()
```


ID
engine_power
age_in_days
km
previous_owners
lat
lon
1 -0.13 -0.16 -0.13 0.0011 0.066 -0.077 0.15
-0.13 1 0.29 0.26 -0.053 0.02 -0.017 -0.26
-0.16 0.29 1 0.85 0.04 -0.04 0.09 -0.92
-0.13 0.26 0.85 1 0.04 -0.068 0.15 -0.88
0.0011 -0.053 0.04 0.04 1 -0.036 -0.017 -0.029
0.066 0.02 -0.04 -0.068 -0.036 1 -0.78 0.053
-0.077 -0.017 0.09 0.15 -0.017 -0.78 1 -0.1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50



In [37]:
```
features=df.columns[0:1]
target=df.columns[-1]
```

In [38]:
```
#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=0.25, 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))
#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 [39]:
```
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 0.003102869264770 88
The test score for lr model is -0.008405634316406507
```

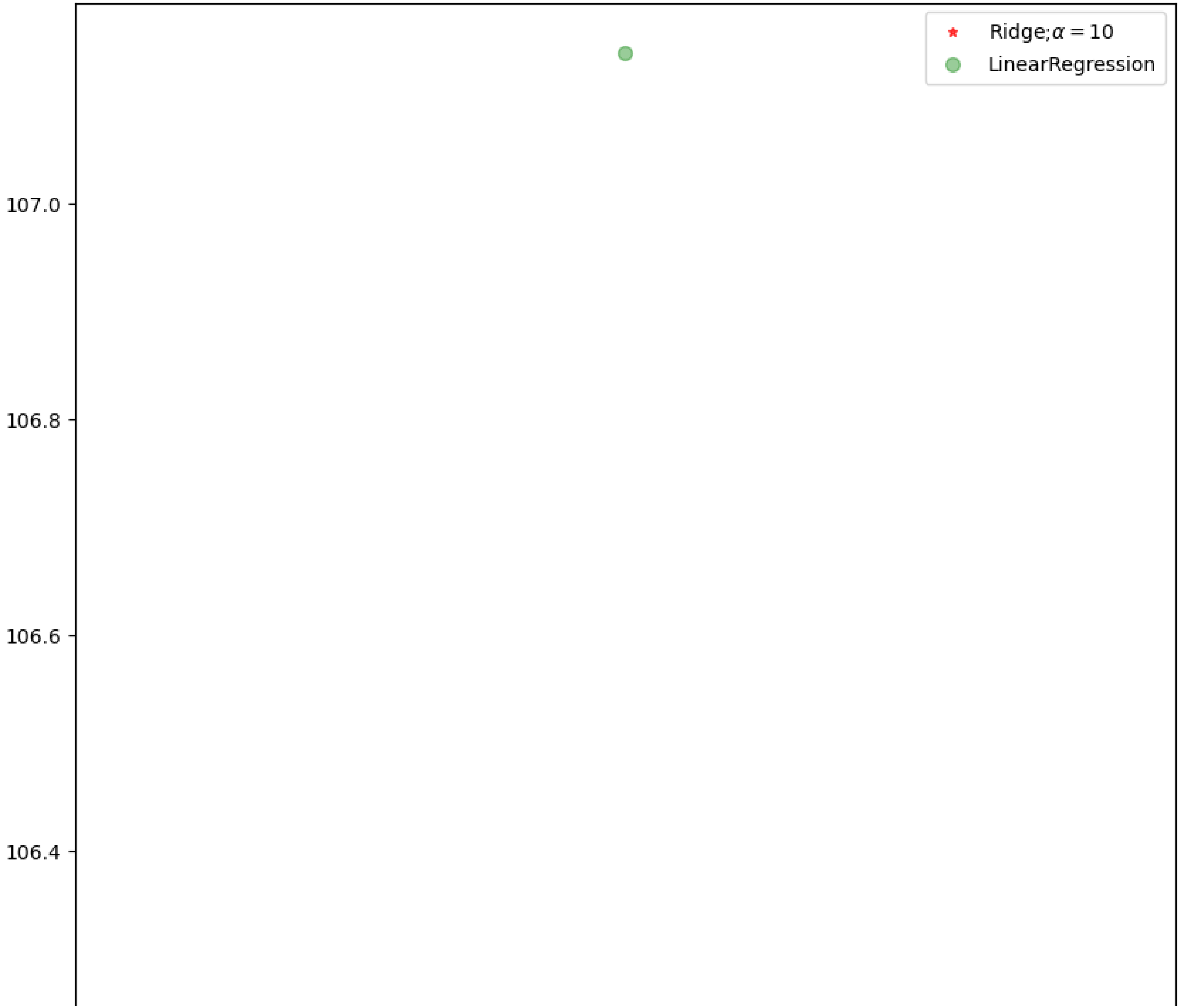
In [40]:
```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

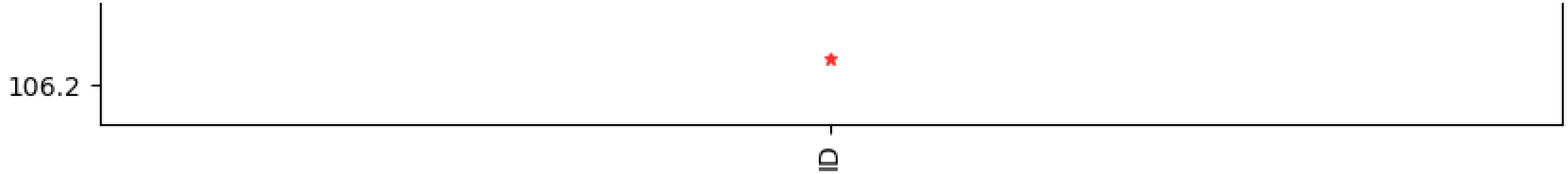
```
Ridge Model:

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

In [41]:
```
plt.figure(figsize=(10,10))
```

Out[41]:
```
<Figure size 1000x1000 with 0 Axes>
```

In [42]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge;$\alpha = 10$
LinearRegression
107.0
106.8
106.6
106.4



In [43]:
```
#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 0.0030758384613109 87
The test score for ls model is -0.007367578602064828
```

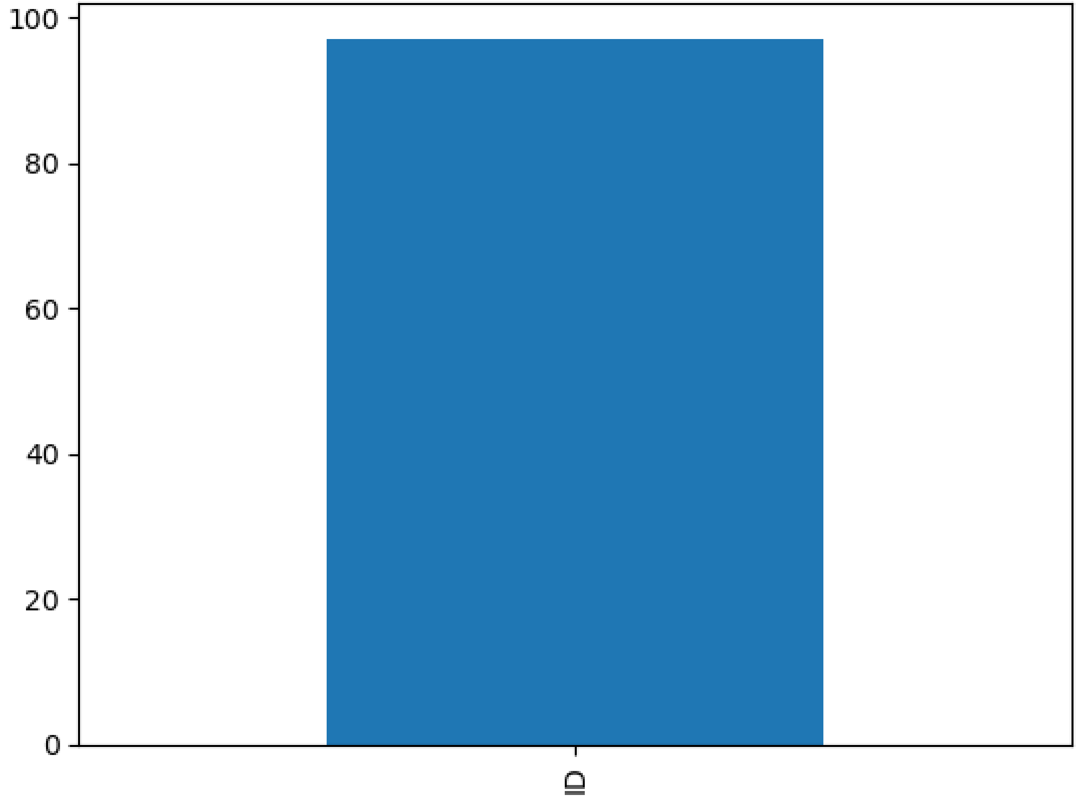
In [44]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[44]: <AxesSubplot:>

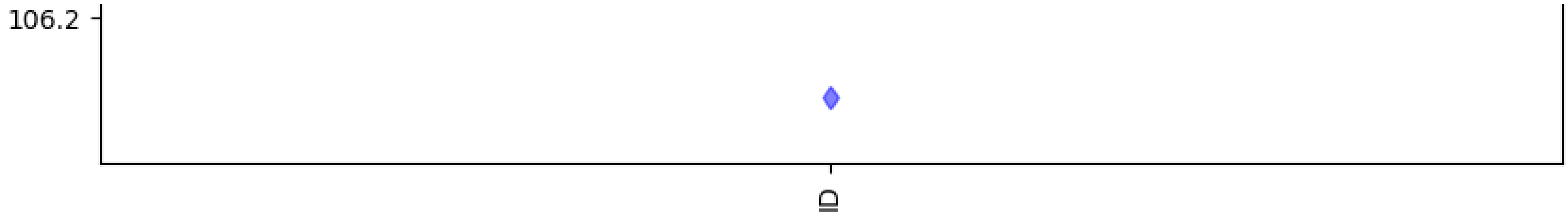
In [45]:
```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_train, y_train)
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.0031025989567363688
-0.008299466692577973
```

In [46]:
```
#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,color='red',label=r'Ridge; $\alpha
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```




Comparison plot of Ridge, Lasso and Linear regression model
Ridge; $\alpha = 10$
lasso; $\alpha = grid$
Linear Regression
107.0
106.8
106.6
106.4



In [47]:
```python
#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_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test)))
```

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

Elastic

In [48]:
```python
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.12455754]
8480.156871173602
```

In [49]:
```python
y_pred_elastic=regr.predict(X_train)
```

In [50]:
```python
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
3708273.194830543
```

In [ ]: