VEHICLE SELECTION USING RIDGE AND LASSO

In [5]:

```
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 [6]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\Downloads\fiat500_VehicleSelection_Dataset (
df
```

Out[6]:

| | 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 [7]:

```
df.head()
```

Out[7]:

| | 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 |

In [8]:

```
df.tail()
```

Out[8]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | lo |
|---|---|---|---|---|---|---|---|---|
| **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 |

In [9]:

```
df.shape
```

Out[9]:

```
(1538, 9)
```

In [10]:

```
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 [11]:

```
df.isna().any()
```

Out[11]:

```
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 [12]:

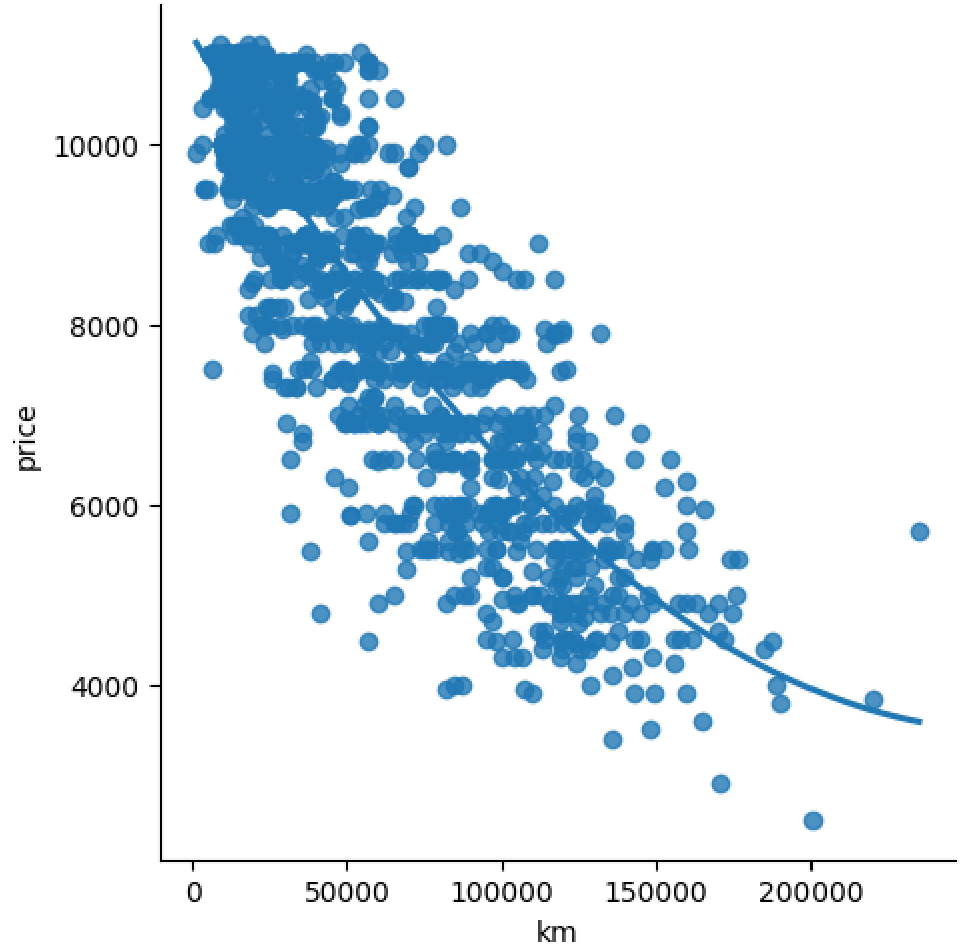
```
df.describe()
```

Out[12]:

| | ID | engine_power | age_in_days | km | previous_owners | l |
|---|---|---|---|---|---|---|
| **count** | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.00000 |
| **mean** | 769.500000 | 51.904421 | 1650.980494 | 53396.011704 | 1.123537 | 43.54136 |
| **std** | 444.126671 | 3.988023 | 1289.522278 | 40046.830723 | 0.416423 | 2.13351 |
| **min** | 1.000000 | 51.000000 | 366.000000 | 1232.000000 | 1.000000 | 36.85583 |
| **25%** | 385.250000 | 51.000000 | 670.000000 | 20006.250000 | 1.000000 | 41.80299 |
| **50%** | 769.500000 | 51.000000 | 1035.000000 | 39031.000000 | 1.000000 | 44.39409 |
| **75%** | 1153.750000 | 51.000000 | 2616.000000 | 79667.750000 | 1.000000 | 45.46796 |
| **max** | 1538.000000 | 77.000000 | 4658.000000 | 235000.000000 | 4.000000 | 46.79561 |

In [13]:

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



In [14]:

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

In [15]:

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

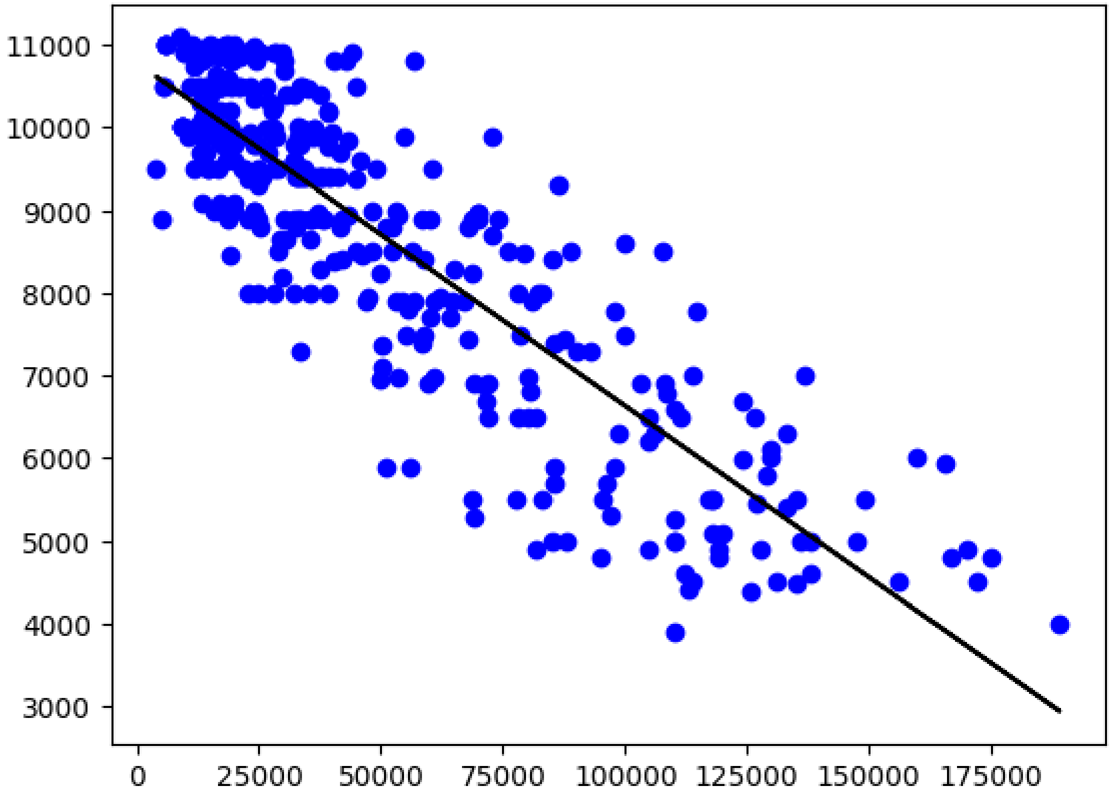
In [16]:

```
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))
```

```
0.7671660418686099
```

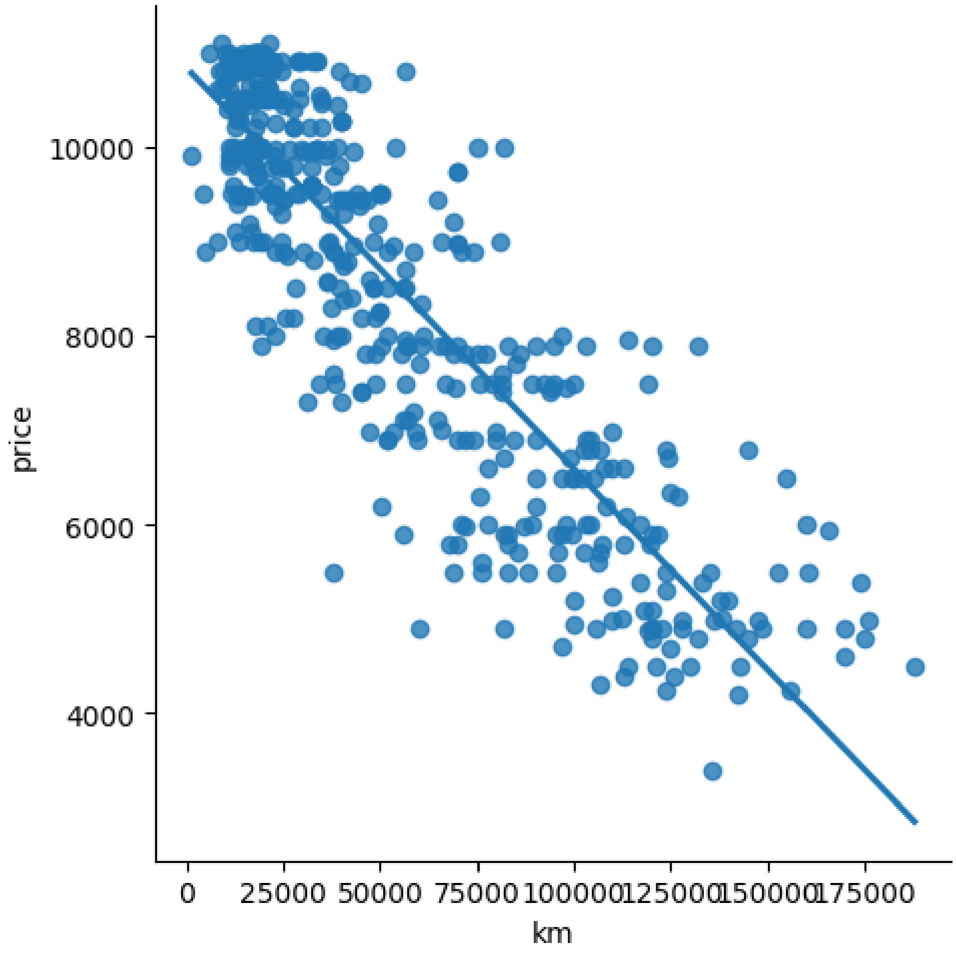
In [17]:

```
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 [18]:

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



In [19]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [20]:

```
#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: 0.7671660418686099
```

RIDGE AND LASSO

In [21]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [22]:

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

In [27]:

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

Out[27]:

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

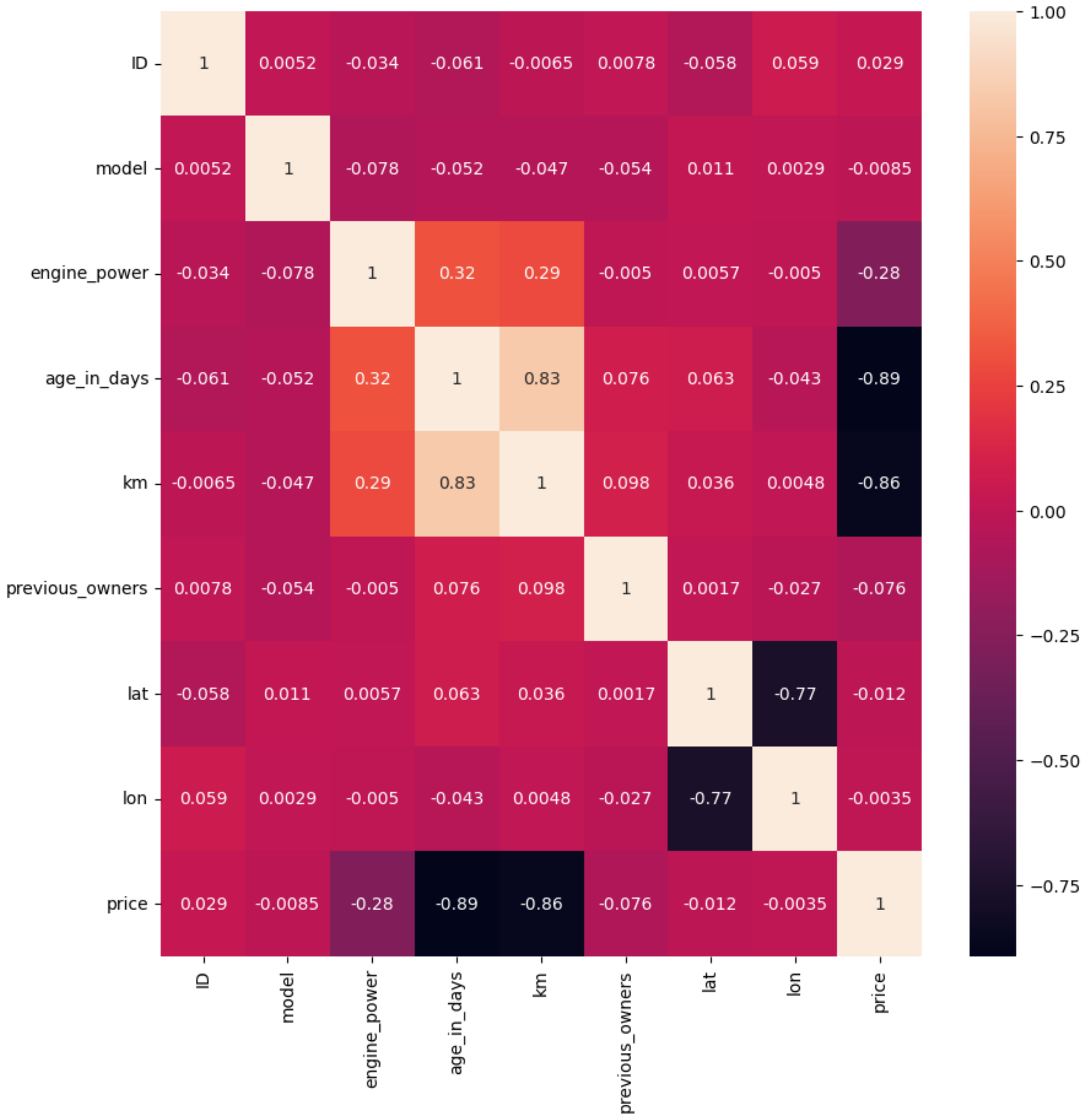
1538 rows × 9 columns

In [28]:

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

Out[28]:

```
<Axes: >
```



In [24]:

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

In [44]:

```
#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=1
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 [45]:

```
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
```

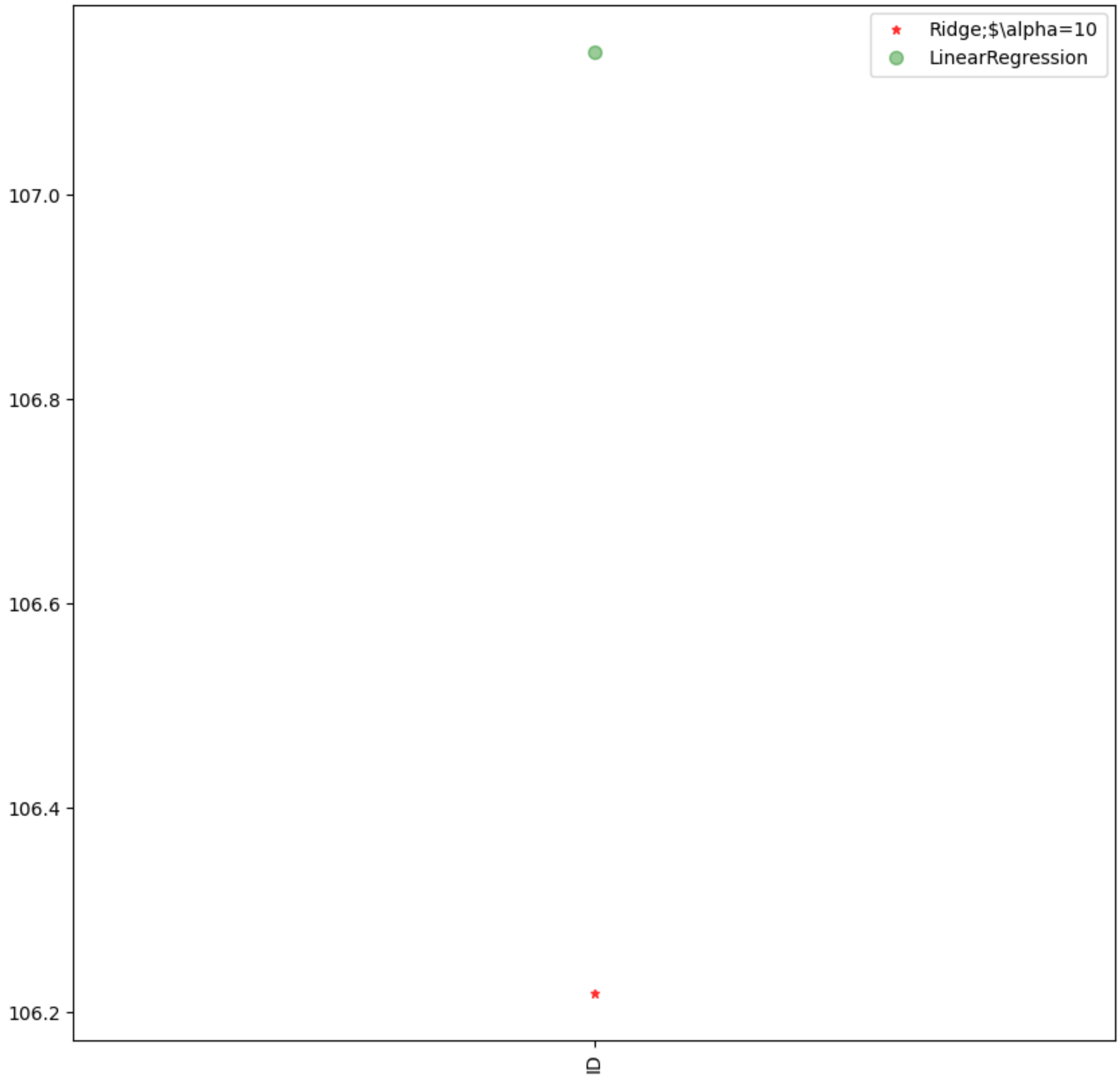
In [46]:

```
#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 [47]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,colo
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='gre
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [48]:

```
#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))
```

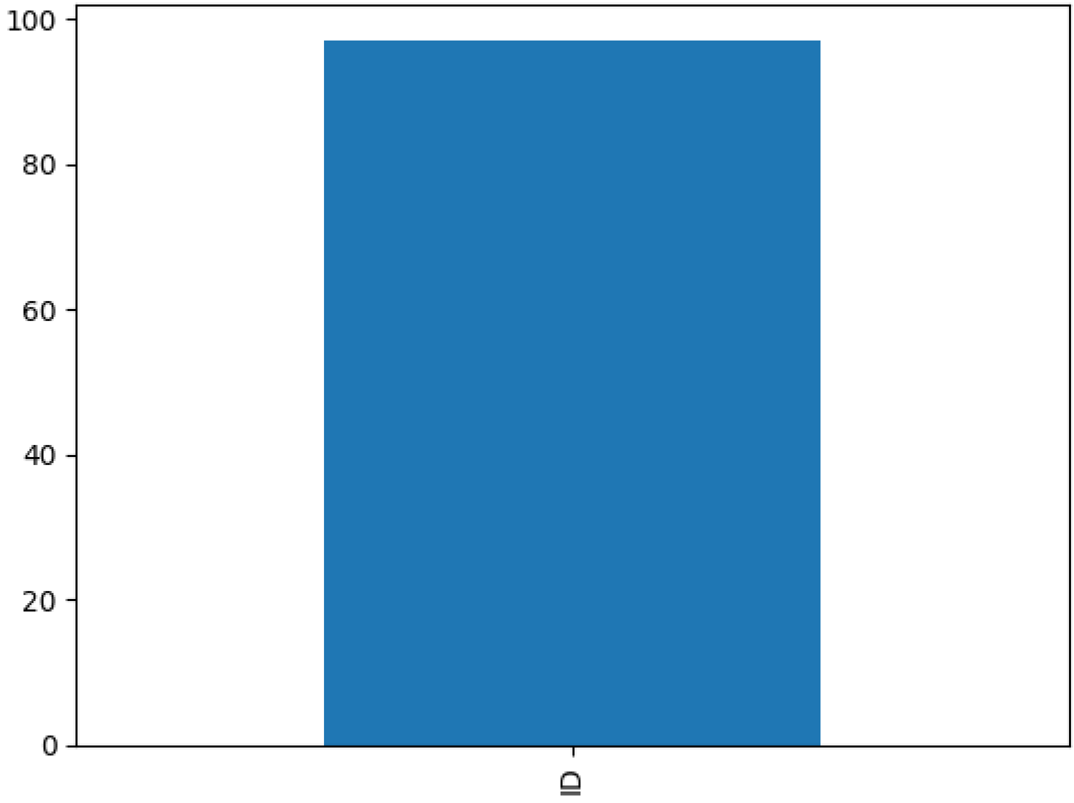
```
Lasso Model: 

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

In [49]:

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

Out[49]:

```
<Axes: >
```



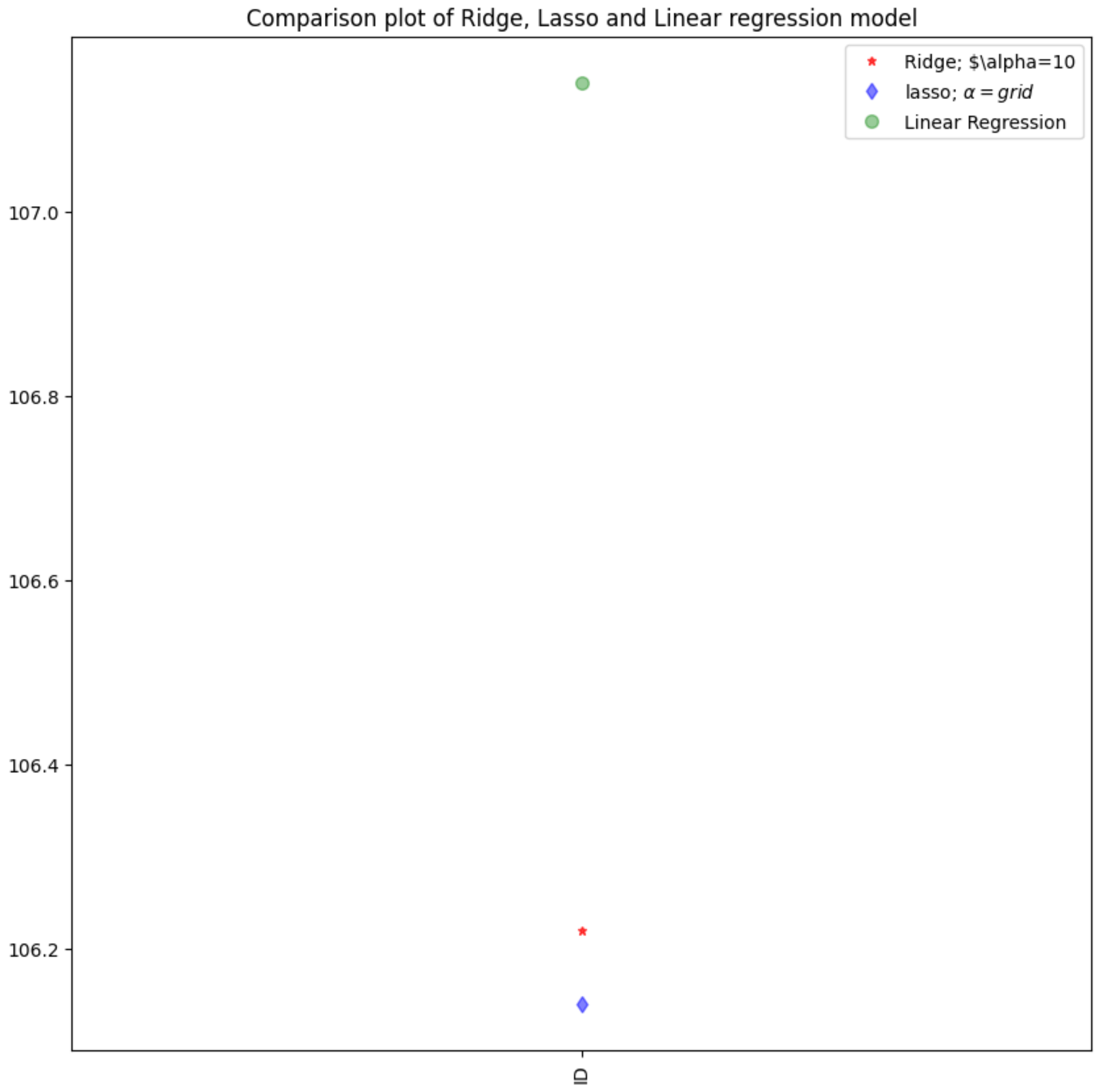
In [50]:

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

```
0.0031025989567363688
-0.008299466692577973
```

In [54]:

```
#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,colo
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
#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

In [52]:

```
#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 -0.008307809466002958
```

Type *Markdown* and LaTeX: $\alpha^2$

In [ ]:

In [ ]:

In [ ]: