

In [24]:

```

1 import numpy as np
2 import pandas as pd
3 import seaborn as sb
4 import matplotlib.pyplot as plt
5 from sklearn import preprocessing ,svm
6 from sklearn.model_selection import train_test_split
7 from sklearn.linear_model import LinearRegression
8 from sklearn.linear_model import Ridge,Lasso
9 from sklearn.preprocessing import StandardScaler
10 dv=pd.read_csv(r"C:\Users\kunam\Downloads\fiat500_VehicleSelection_Dataset.csv")
11 dv.head(10)

```

Out[24]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	lounge	51	882	25000	1	44.907242	8.611561
1	2	pop	51	1186	32500	1	45.666359	12.241891
2	3	sport	74	4658	142228	1	45.503300	11.417841
3	4	lounge	51	2739	160000	1	40.633171	17.634601
4	5	pop	73	3074	106880	1	41.903221	12.495651
5	6	pop	74	3623	70225	1	45.000702	7.682271
6	7	lounge	51	731	11600	1	44.907242	8.611561
7	8	lounge	51	1521	49076	1	41.903221	12.495651
8	9	sport	73	4049	76000	1	45.548000	11.549471
9	10	sport	51	3653	89000	1	45.438301	10.991701

In [14]:

```
1 dv.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

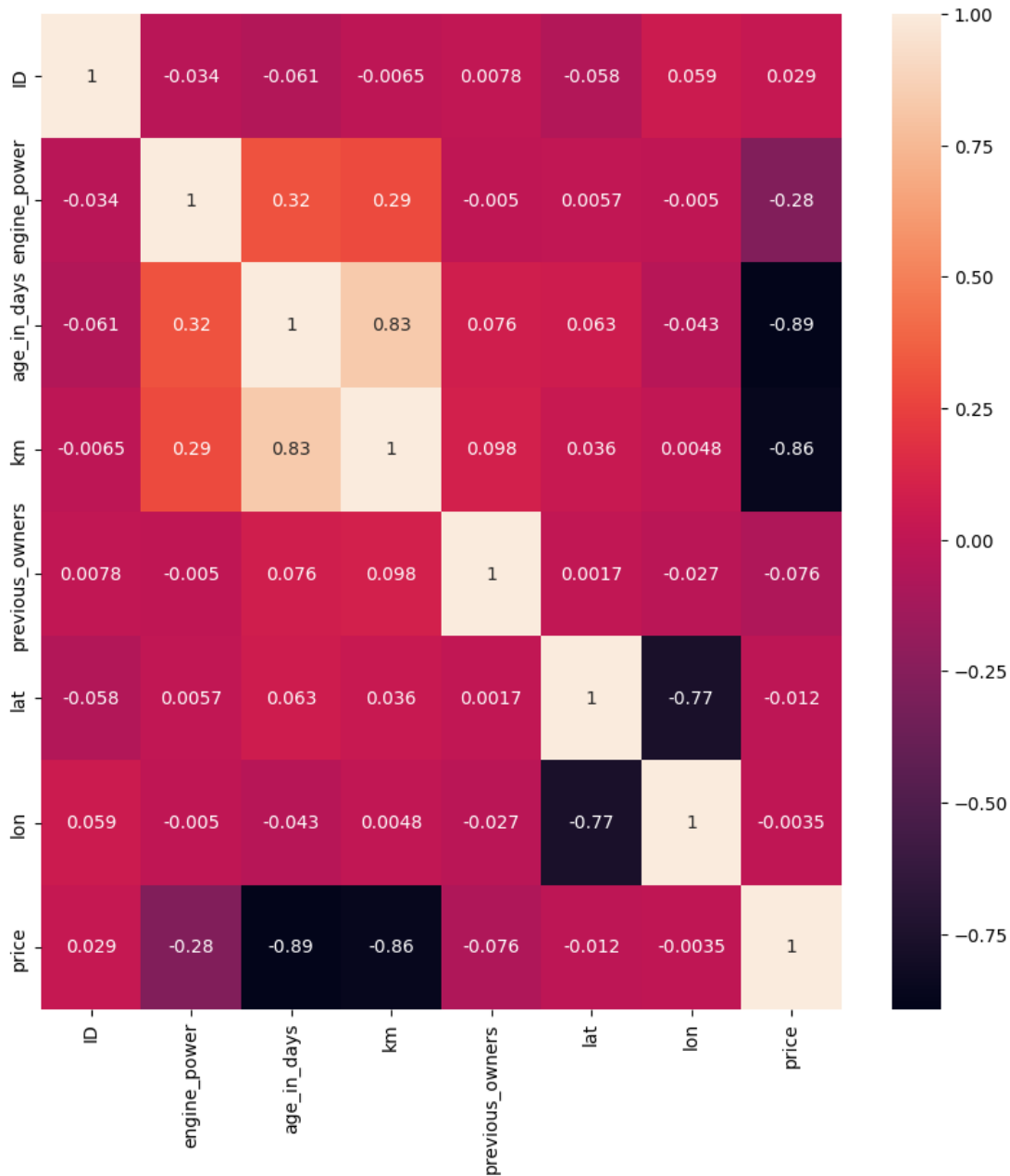
```

In [15]:

```
1 dat=dv
2 dat.drop(columns=["model"], inplace = True)
3 plt.figure(figsize = (10, 10))
4 sb.heatmap(dv.corr(), annot = True)
```

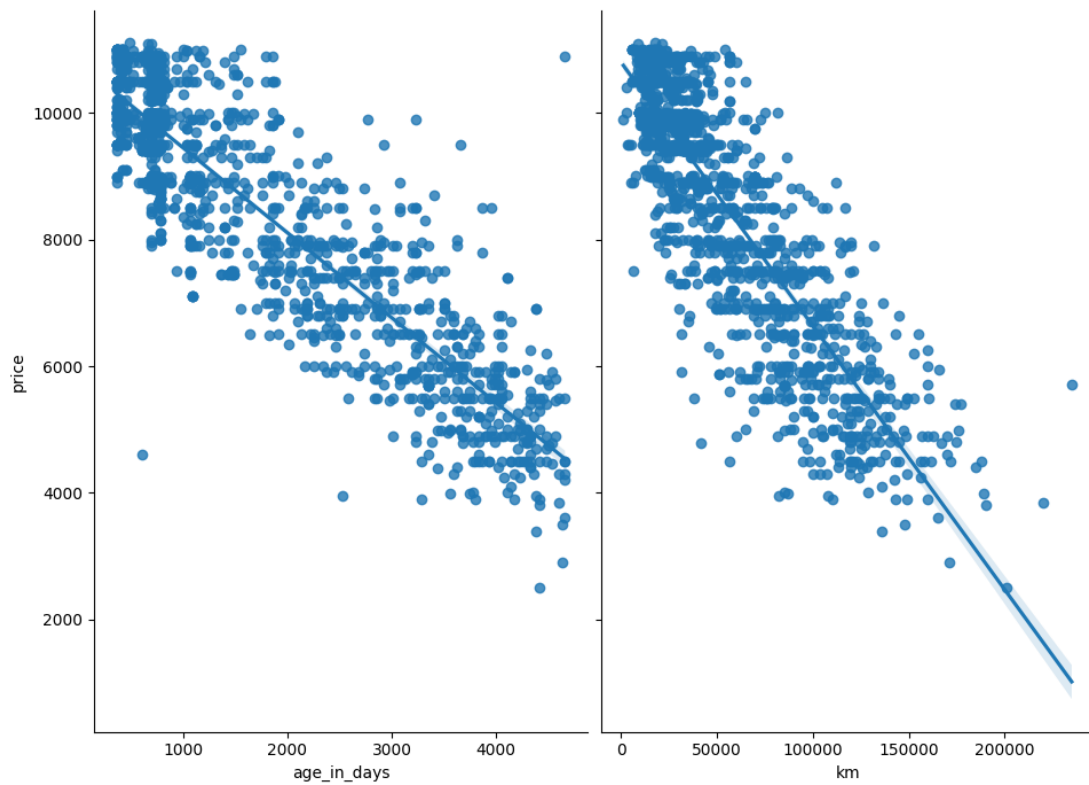
Out[15]:

&lt;Axes: &gt;



In [25]:

```
1 sb.pairplot(dv,x_vars=['age_in_days','km'],y_vars='price',height=7,aspect=0.7,kind='scatter')
2 #sb.pairplot(dv)
3 dv.price = np.log(dv.price)
```



In [26]:

```
1 dv.columns
```

Out[26]:

```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',  
      'lat', 'lon', 'price'],  
      dtype='object')
```

In [27]:

```
1 features = dv.columns[3:5]
2 target = dv.columns[-1]
3 #X and y values
4 x = dv[features].values
5 y = dv[target].values
6 #split
7 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_s
8 print("The dimension of X_train is {}".format(x_train.shape))
9 print("The dimension of X_test is {}".format(x_test.shape))
10 #Scale features
11 scaler = StandardScaler()
12 x_train = scaler.fit_transform(x_train)
13 x_test = scaler.transform(x_test)
```

The dimension of X\_train is (1076, 2)

The dimension of X\_test is (462, 2)

In [28]:

```
1 #Model
2 lr = LinearRegression()
3 #Fit model
4 lr.fit(x_train, y_train)
5 #actual
6 actual = y_test
7 train_score_lr = lr.score(x_train, y_train)
8 test_score_lr = lr.score(x_test, y_test)
9 print("\nLinear Regression Model:\n")
10 print("The train score for lr model is {}".format(train_score_lr))
11 print("The test score for lr model is {}".format(test_score_lr))
```

Linear Regression Model:

The train score for lr model is 0.8407959504511326

The test score for lr model is 0.8396374461789744

## Ridge Regression

In [29]:

```
1 #Ridge Regression Model
2 ridgeReg = Ridge(alpha=10)
3 ridgeReg.fit(x_train,y_train)
4 #train and test scorefor ridge regression
5 train_score_ridge = ridgeReg.score(x_train, y_train)
6 test_score_ridge = ridgeReg.score(x_test, y_test)
7 print("\nRidge Model:\n")
8 print("The train score for ridge model is {}".format(train_score_ridge))
9 print("The test score for ridge model is {}".format(test_score_ridge))
```

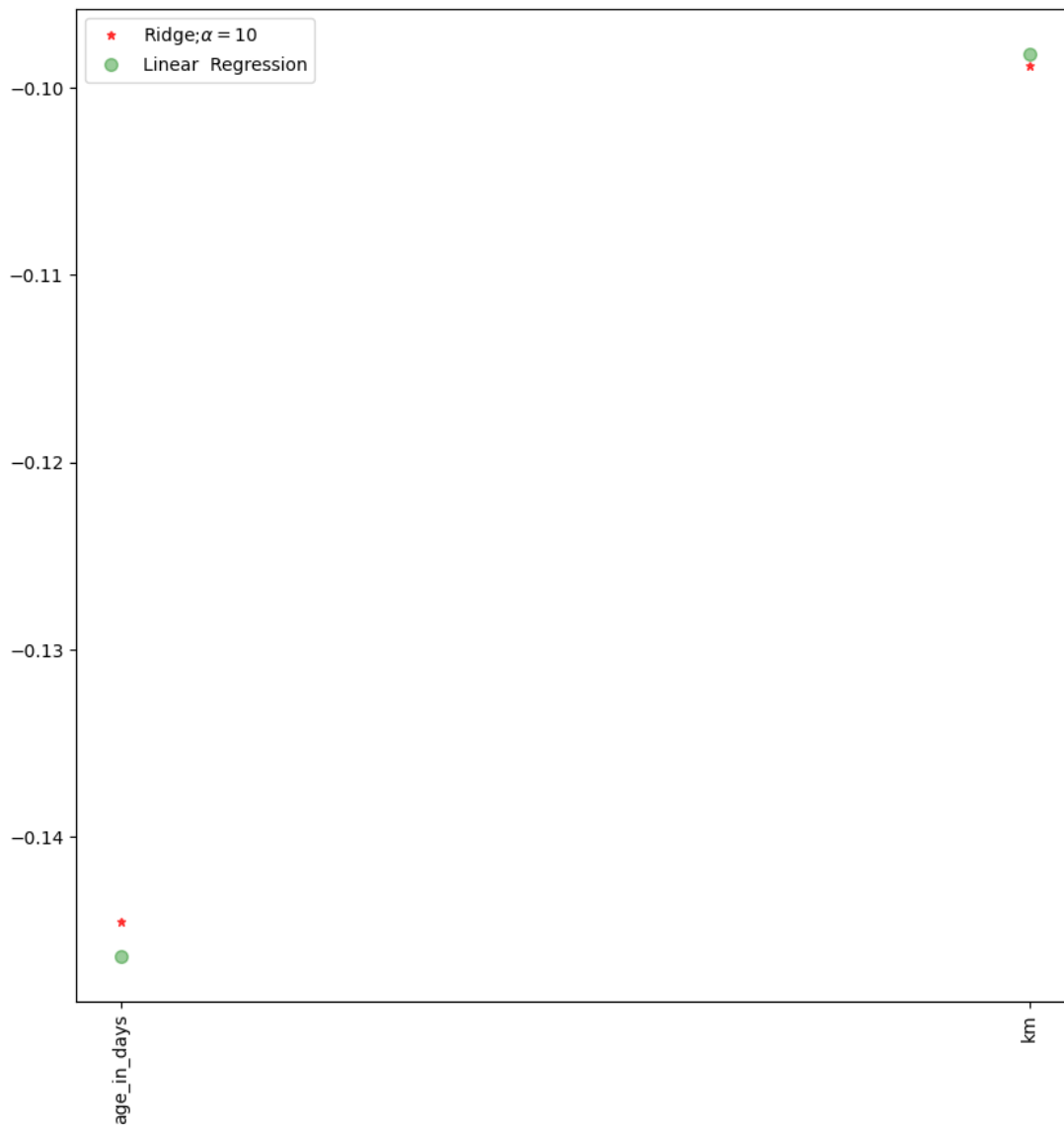
Ridge Model:

The train score for ridge model is 0.8407665046629869

The test score for ridge model is 0.8395909167380576

In [30]:

```
1 plt.figure(figsize=(10,10))
2 plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=10,color='red')
3 plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green')
4 plt.xticks(rotation=90)
5 plt.legend()
6 plt.show()
```



## Lasso Regression

In [31]:

```
1 print("\nLasso Model: \n")
2 lasso = Lasso(alpha = 10)
3 lasso.fit(x_train,y_train)
4 train_score_ls =lasso.score(x_train,y_train)
5 test_score_ls =lasso.score(x_test,y_test)
6 print("The train score for ls model is {}".format(train_score_ls))
7 print("The test score for ls model is {}".format(test_score_ls))
```

Lasso Model:

The train score for ls model is 0.0

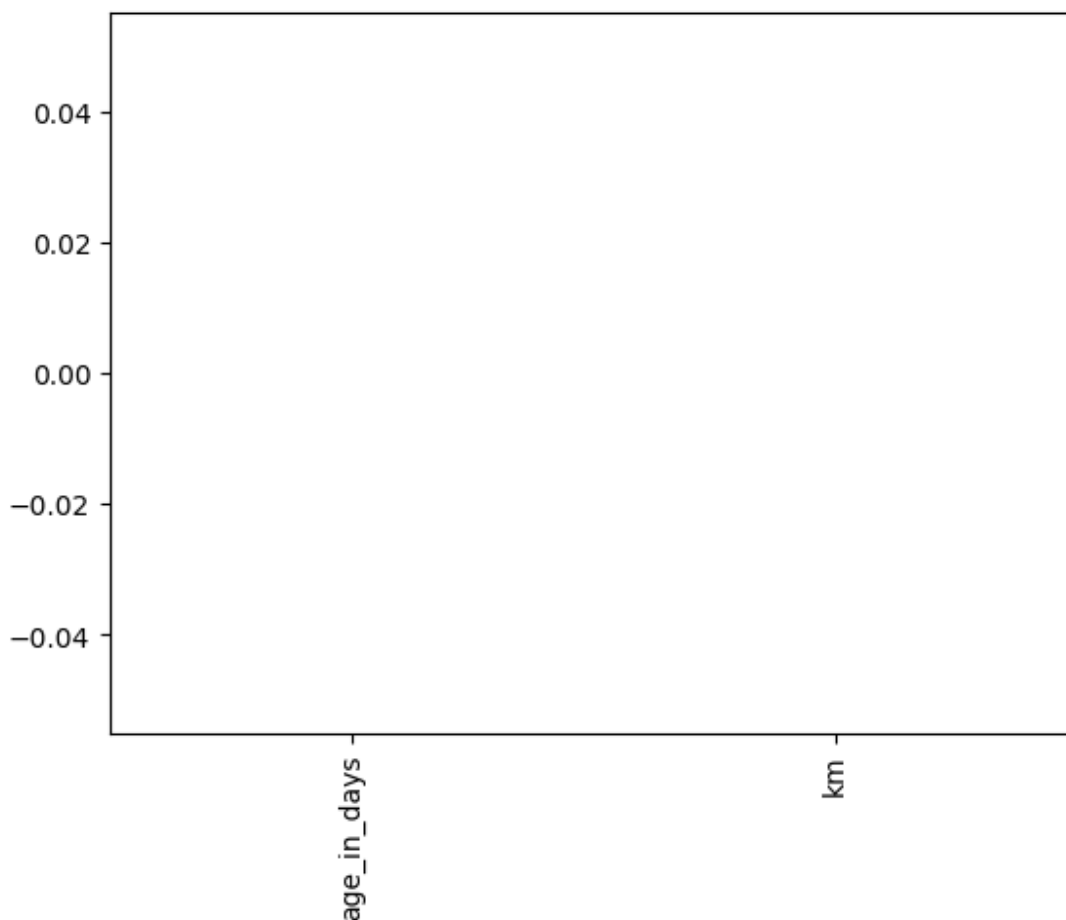
The test score for ls model is -0.0004974348027177999

In [32]:

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

Out[32]:

&lt;Axes: &gt;



In [33]:

```
1 #Using the linear cv model
2 from sklearn.linear_model import LassoCV
3 #Lasso Cross validation
4 lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit
5 #score
6 print(lasso_cv.score(x_train, y_train))
7 print(lasso_cv.score(x_test, y_test))
```

0.8407957935001775

0.8396376947295177

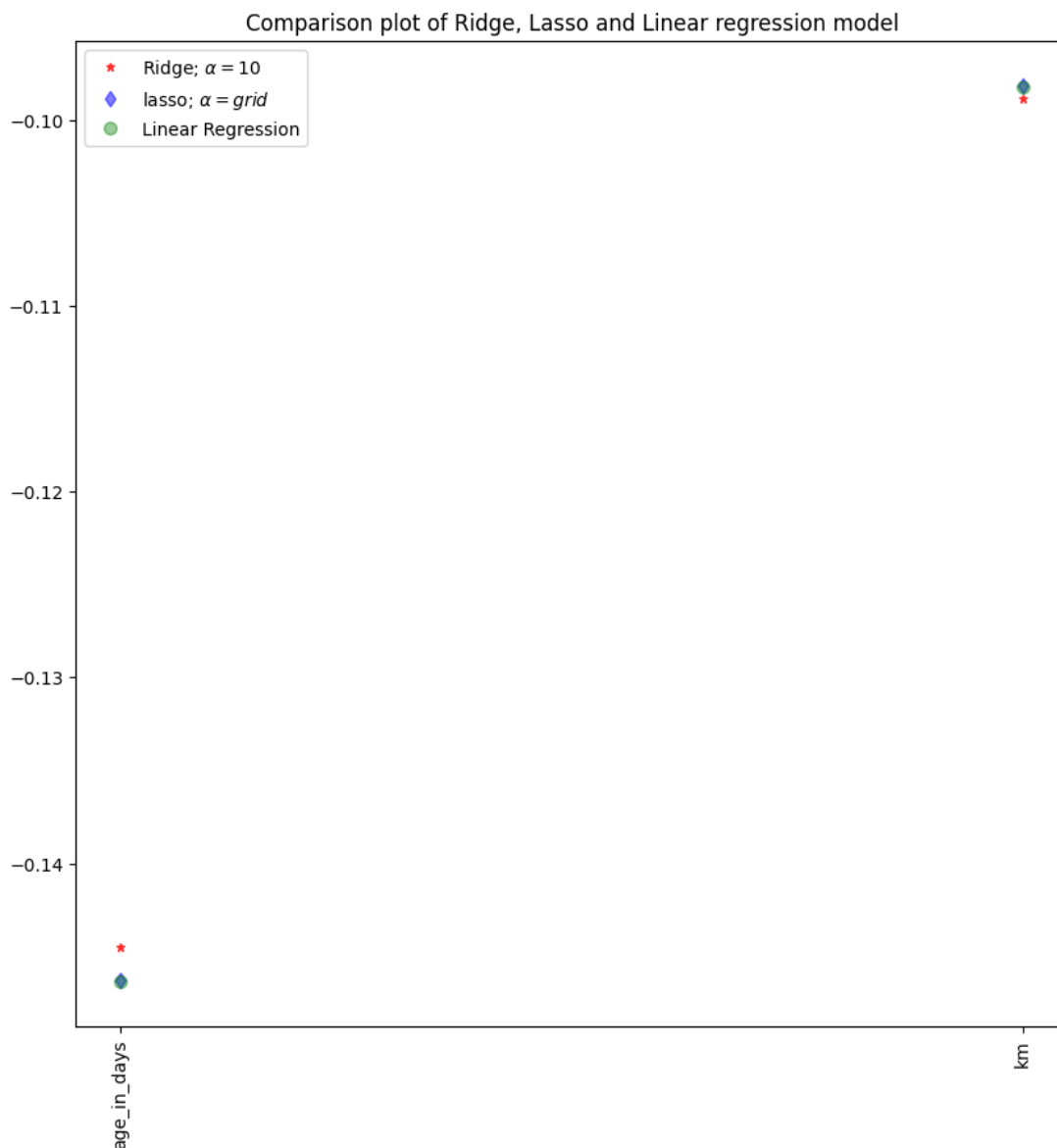


In [34]:

```

1 plt.figure(figsize = (10, 10))
2 #add plot for ridge regression
3 plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize:
4 #add plot for lasso regression
5 plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color=
6 #add plot for linear model
7 plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,colo
8 #rotate axis
9 plt.xticks(rotation = 90)
10 plt.legend()
11 plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
12 plt.show()

```



In [35]:

```

1 from sklearn.linear_model import RidgeCV
2 #Ridge Cross validation
3 ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(x_train, y_train)
4 #score
5 print("The train score for ridge model is {}".format(ridge_cv.score(x_train, y_train)))
6 print("The train score for ridge model is {}".format(ridge_cv.score(x_test, y_test)))

```

The train score for ridge model is 0.8407665046629889

The train score for ridge model is 0.8395909167380597

In [36]:

```

1 from sklearn.linear_model import ElasticNet
2 regr=ElasticNet()
3 regr.fit(x,y)
4 print(regr.coef_)
5 print(regr.intercept_)

```

```

[-1.14013910e-04 -2.51291433e-06]
9.34861331834344

```

In [37]:

```

1 y_pred_elastic=regr.predict(x_train)
2 mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
3 print(mean_squared_error)

```

0.1679531699267824

In [ ]:

1

In [ ]:

1

In [ ]:

1

In [ ]:

1

In [ ]:

1

## ElasticNet Regression on Vehical selection

In [38]:

```
1 from sklearn.linear_model import ElasticNet
2 regr=ElasticNet()
3 regr.fit(x,y)
4 print(regr.coef_)
5 print(regr.intercept_)
```

```
[-1.14013910e-04 -2.51291433e-06]
9.34861331834344
```

In [39]:

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

```
0.1679531699267824
```