

In [1]:

```

1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
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

```

In [2]:

```

1 df=pd.read_csv(r"C:\Users\HP\OneDrive\Documents\fiat500_VehicleSelection_Dataset.csv")
2 df

```

Out[2]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | lon |
|------|------|--------|--------------|-------------|--------|-----------------|-----------|-----------|
| 0 | 1 | lounge | 51 | 882 | 25000 | 1 | 44.907242 | 8.611111 |
| 1 | 2 | pop | 51 | 1186 | 32500 | 1 | 45.666359 | 12.244444 |
| 2 | 3 | sport | 74 | 4658 | 142228 | 1 | 45.503300 | 11.411111 |
| 3 | 4 | lounge | 51 | 2739 | 160000 | 1 | 40.633171 | 17.633333 |
| 4 | 5 | pop | 73 | 3074 | 106880 | 1 | 41.903221 | 12.494444 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1533 | 1534 | sport | 51 | 3712 | 115280 | 1 | 45.069679 | 7.700000 |
| 1534 | 1535 | lounge | 74 | 3835 | 112000 | 1 | 45.845692 | 8.666667 |
| 1535 | 1536 | pop | 51 | 2223 | 60457 | 1 | 45.481541 | 9.411111 |
| 1536 | 1537 | lounge | 51 | 2557 | 80750 | 1 | 45.000702 | 7.688889 |
| 1537 | 1538 | pop | 51 | 1766 | 54276 | 1 | 40.323410 | 17.566667 |

1538 rows × 9 columns

In [3]:

```

1 df=df[['engine_power','age_in_days']]
2 df.columns=['ep','aid']

```

In [4]:

```
1 df.describe()
```

Out[4]:

| | ep | aid |
|-------|-------------|-------------|
| count | 1538.000000 | 1538.000000 |
| mean | 51.904421 | 1650.980494 |
| std | 3.988023 | 1289.522278 |
| min | 51.000000 | 366.000000 |
| 25% | 51.000000 | 670.000000 |
| 50% | 51.000000 | 1035.000000 |
| 75% | 51.000000 | 2616.000000 |
| max | 77.000000 | 4658.000000 |

In [5]:

```
1 df.head(10)
```

Out[5]:

| | ep | aid |
|---|----|------|
| 0 | 51 | 882 |
| 1 | 51 | 1186 |
| 2 | 74 | 4658 |
| 3 | 51 | 2739 |
| 4 | 73 | 3074 |
| 5 | 74 | 3623 |
| 6 | 51 | 731 |
| 7 | 51 | 1521 |
| 8 | 73 | 4049 |
| 9 | 51 | 3653 |

In [6]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  -
 0    ep      1538 non-null    int64  
 1    aid      1538 non-null    int64  
dtypes: int64(2)
memory usage: 24.2 KB
```

In [24]:

```
1 df.fillna(method='ffill')
```

Out[24]:

| | ep | aid |
|------|-----|------|
| 0 | 51 | 882 |
| 1 | 51 | 1186 |
| 2 | 74 | 4658 |
| 3 | 51 | 2739 |
| 4 | 73 | 3074 |
| ... | ... | ... |
| 1533 | 51 | 3712 |
| 1534 | 74 | 3835 |
| 1535 | 51 | 2223 |
| 1536 | 51 | 2557 |
| 1537 | 51 | 1766 |

1538 rows × 2 columns

In [8]:

```
1 x=np.array(df['ep']).reshape(-1,1)
2 y=np.array(df['aid']).reshape(-1,1)
```

In [25]:

```
1 df.dropna()
```

Out[25]:

| | ep | aid |
|------|-----|------|
| 0 | 51 | 882 |
| 1 | 51 | 1186 |
| 2 | 74 | 4658 |
| 3 | 51 | 2739 |
| 4 | 73 | 3074 |
| ... | ... | ... |
| 1533 | 51 | 3712 |
| 1534 | 74 | 3835 |
| 1535 | 51 | 2223 |
| 1536 | 51 | 2557 |
| 1537 | 51 | 1766 |

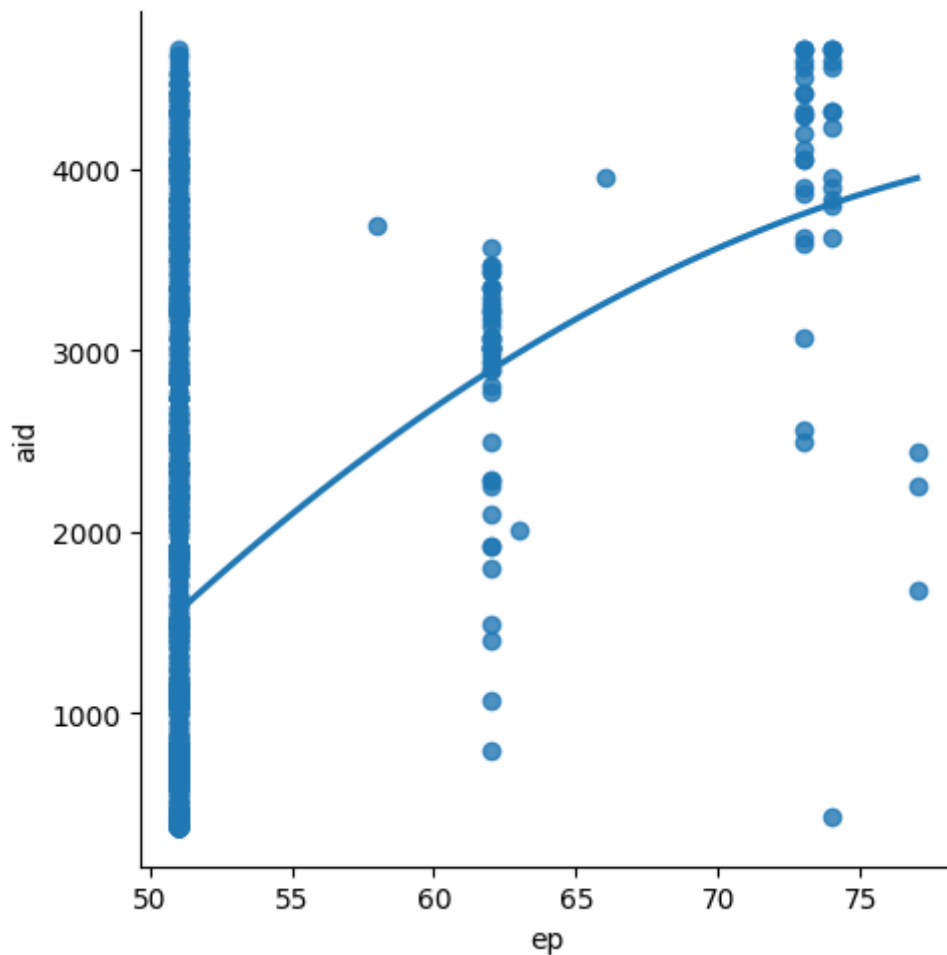
1538 rows × 2 columns

In [26]:

```
1 #Exploring the data scatter_plotting the data scatter
2 sns.lmplot(x = "ep", y = "aid", data = df, order = 2, ci = None)
```

Out[26]:

<seaborn.axisgrid.FacetGrid at 0x247bfdeb9d0>



In [27]:

```
1 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
2 regr=LinearRegression()
3 regr.fit(x_train,y_train)
4 print(regr.score(x_test,y_test))
```

1.0

In [28]:

```

1 y_pred=regr.predict(x_test)
2 plt.scatter(x_test,y_test,color='b')
3 plt.plot(x_test,y_pred,color='k')
4 plt.show()

```

```

-----
--
ValueError                                Traceback (most recent call las
t)
Cell In[28], line 2
      1 y_pred=regr.predict(x_test)
----> 2 plt.scatter(x_test,y_test,color='b')
      3 plt.plot(x_test,y_pred,color='k')
      4 plt.show()

```

```

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\
ib\pyplot.py:2862, in scatter(x, y, s, c, marker, cmap, norm, vmin, vmax,
alpha, linewidths, edgecolors, plotnonfinite, data, **kwargs)

```

```

2857 @_copy_docstring_and_deprecators(Axes.scatter)
2858 def scatter(
2859     x, y, s=None, c=None, marker=None, cmap=None, norm=None,
2860     vmin=None, vmax=None, alpha=None, linewidths=None, *,
2861     edgecolors=None, plotnonfinite=False, data=None, **kwarg
s):
-> 2862     __ret = gca().scatter(
2863         x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
2864         vmin=vmin, vmax=vmax, alpha=alpha, linewidths=linewidths,
2865         edgecolors=edgecolors, plotnonfinite=plotnonfinite,
2866         **({"data": data} if data is not None else {}), **kwargs)
2867     sci(__ret)
2868     return __ret

```

```

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\
ib\__init__.py:1459, in _preprocess_data.<locals>.inner(ax, data, *args,
**kwargs)

```

```

1456 @functools.wraps(func)
1457 def inner(ax, *args, data=None, **kwargs):
1458     if data is None:
-> 1459         return func(ax, *map(sanitize_sequence, args), **kwargs)
1461     bound = new_sig.bind(ax, *args, **kwargs)
1462     auto_label = (bound.arguments.get(label_namer)
1463                   or bound.kwargs.get(label_namer))

```

```

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\
ib\axes\_axes.py:4584, in Axes.scatter(self, x, y, s, c, marker, cmap, no
rm, vmin, vmax, alpha, linewidths, edgecolors, plotnonfinite, **kwargs)

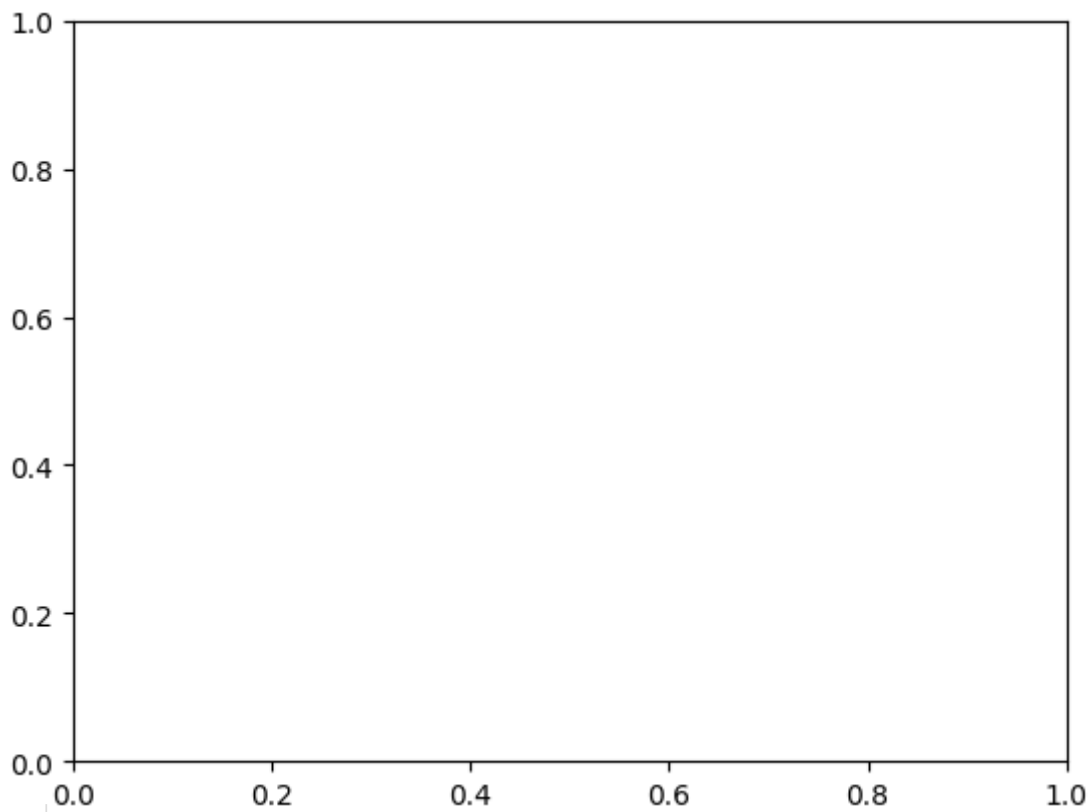
```

```

4582 y = np.ma.ravel(y)
4583 if x.size != y.size:
-> 4584     raise ValueError("x and y must be the same size")
4586 if s is None:
4587     s = (20 if mpl.rcParams['_internal.classic_mode'] else
4588          mpl.rcParams['lines.markersize'] ** 2.0)

```

ValueError: x and y must be the same size



```
1 df100=df[:][:100]
2 sns.lmplot(x='ep',y='aid',data=df100,order=1,ci=None)
3
```

In [29]:

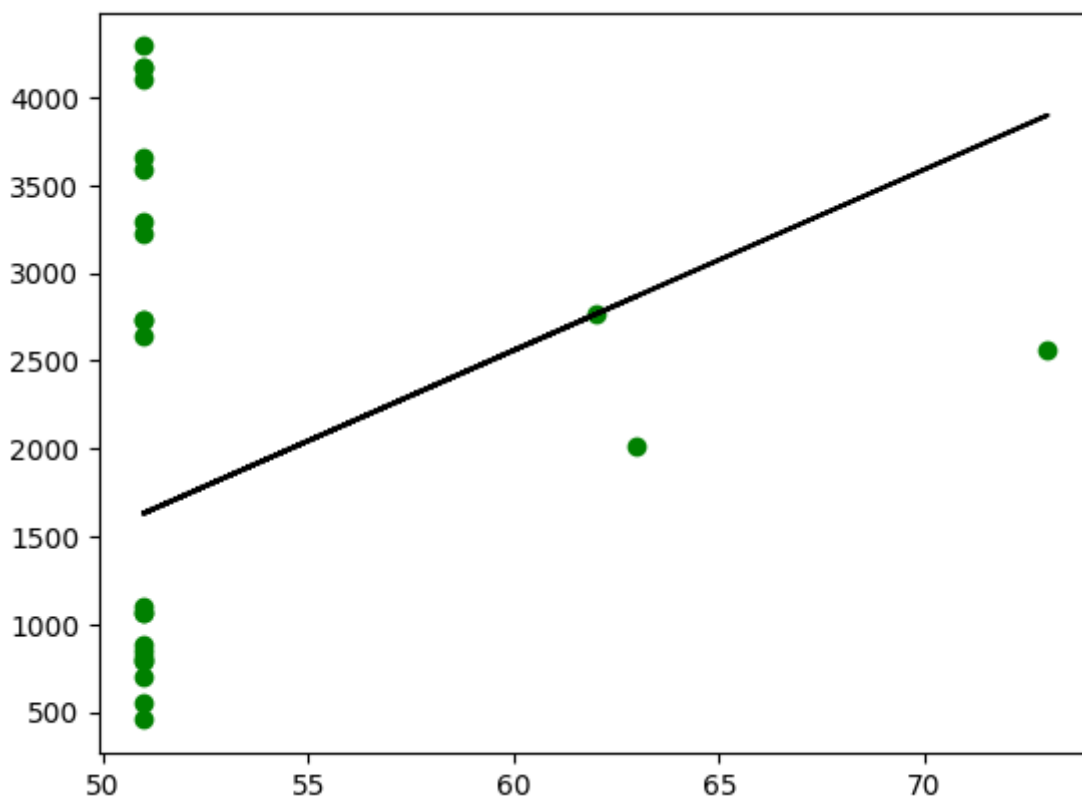
```

1 df100.fillna(method='ffill',inplace=True)
2 X=np.array(df100['ep']).reshape(-1,1)
3 y=np.array(df100['aid']).reshape(-1,1)
4 df100.dropna(inplace=True)
5 X_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
6 regr=LinearRegression()
7 regr.fit(X_train,y_train)
8 print(regr.score(x_test,y_test))
9 print("Regression: ",regr.score(x_test,y_test))
10 y_pred=regr.predict(x_test)
11 plt.scatter(x_test,y_test,color='g')
12 plt.plot(x_test,y_pred,color='k')
13 plt.show()

```

-0.18690025900835017

Regression: -0.18690025900835017



In [30]:

```

1 from sklearn.linear_model import LinearRegression
2 from sklearn.metrics import r2_score
3 model=LinearRegression()
4 model.fit(X_train,y_train)
5 y_pred=model.predict(x_test)
6 r2=r2_score(y_test,y_pred)
7 print("R2_score: ",r2)

```

R2_score: -0.18690025900835017

Conclusion:

Dataset we have taken is poor for linear model but with the smaller data works well with linear model

In [31]:

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.model_selection import train_test_split
3 feature=df.columns[0:3]
4 target=df.columns[-1]
5 x=df[feature].values
6 y=df[target].values
7 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
8 lr = LinearRegression()
9 lr.fit(x_train,y_train)
10 print(lr.score(x_test,y_test))
11 print(lr.score(x_train,y_train))
```

1.0

1.0

In [32]:

```
1 from sklearn.linear_model import Ridge,RidgeCV,Lasso,LassoCV
2 ridge = Ridge(alpha=10)
3 ridge.fit(x_train,y_train)
4 train_score_ridge=ridge.score(x_train,y_train)
5 test_score_ridge=ridge.score(x_test,y_test)
6 print('\n Ridge method\n')
7 print('The score of ridge method is {}'.format(train_score_ridge))
8 print('The score of ridge method is {}'.format(test_score_ridge))
```

Ridge method

The score of ridge method is 1.0

The score of ridge method is 1.0

In [33]:

```
1 lasso = Lasso(alpha=10)
2 lasso.fit(x_train,y_train)
3 train_score_lasso=lasso.score(x_train,y_train)
4 test_score_lasso=lasso.score(x_test,y_test)
5 print('\n Lasso method\n')
6 print('The score of lasso method is {}'.format(train_score_lasso))
7 print('The score of lasso method is {}'.format(test_score_lasso))
```

Lasso method

The score of lasso method is 0.9999999999644759

The score of lasso method is 0.9999999999644752

In [34]:

```
1 lasso_cv= LassoCV(alphas=[0.2,0.03,0.004,0.0001,1,20]).fit(x_train,y_train)
2 train_score_lasso_cv=lasso_cv.score(x_train,y_train)
3 test_score_lasso_cv=lasso_cv.score(x_test,y_test)
4 print('\n LassoCV method\n')
5 print('The score of Lasso method is {}'.format(train_score_lasso_cv))
6 print('The score of Lasso method is {}'.format(test_score_lasso_cv))
```

LassoCV method

The score of Lasso method is 1.0
The score of Lasso method is 1.0

In [35]:

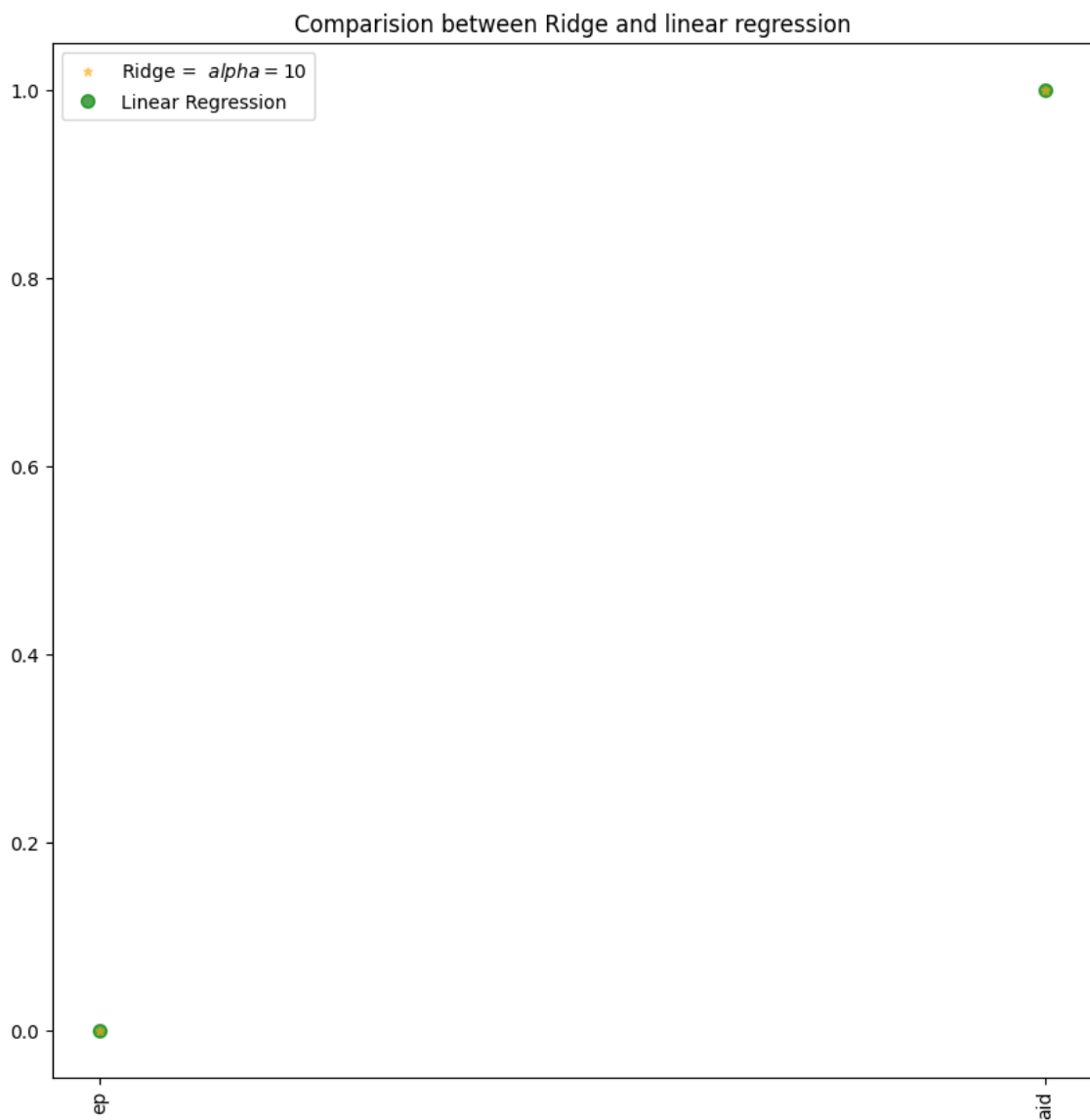
```
1 ridge_cv=RidgeCV(alphas=[1,2.3,0.2,0.3,0.4,0.5,0.6]).fit(x_train,y_train)
2 print("\n RidgeCV Method\n")
3 print("The score of Ridge method is {}".format(ridge_cv.score(x_train,y_train)))
4 print("The score of Ridge method is {}".format(ridge_cv.score(x_test,y_test)))
```

RidgeCV Method

The score of Ridge method is 0.9999999994626445
The score of Ridge method is 0.9999999994626342

In [36]:

```
1 plt.figure(figsize=(10,10))
2 plt.plot(feature,ridge.coef_,alpha=0.5,marker='*',markersize=5,linestyle='None',color='blue')
3 plt.plot(feature,lr.coef_,alpha=0.7,marker='o',markersize=7,color='green',linestyle='None')
4 plt.xticks(rotation=90)
5 plt.title("Comparision between Ridge and linear regression")
6 plt.legend()
7 plt.show()
```

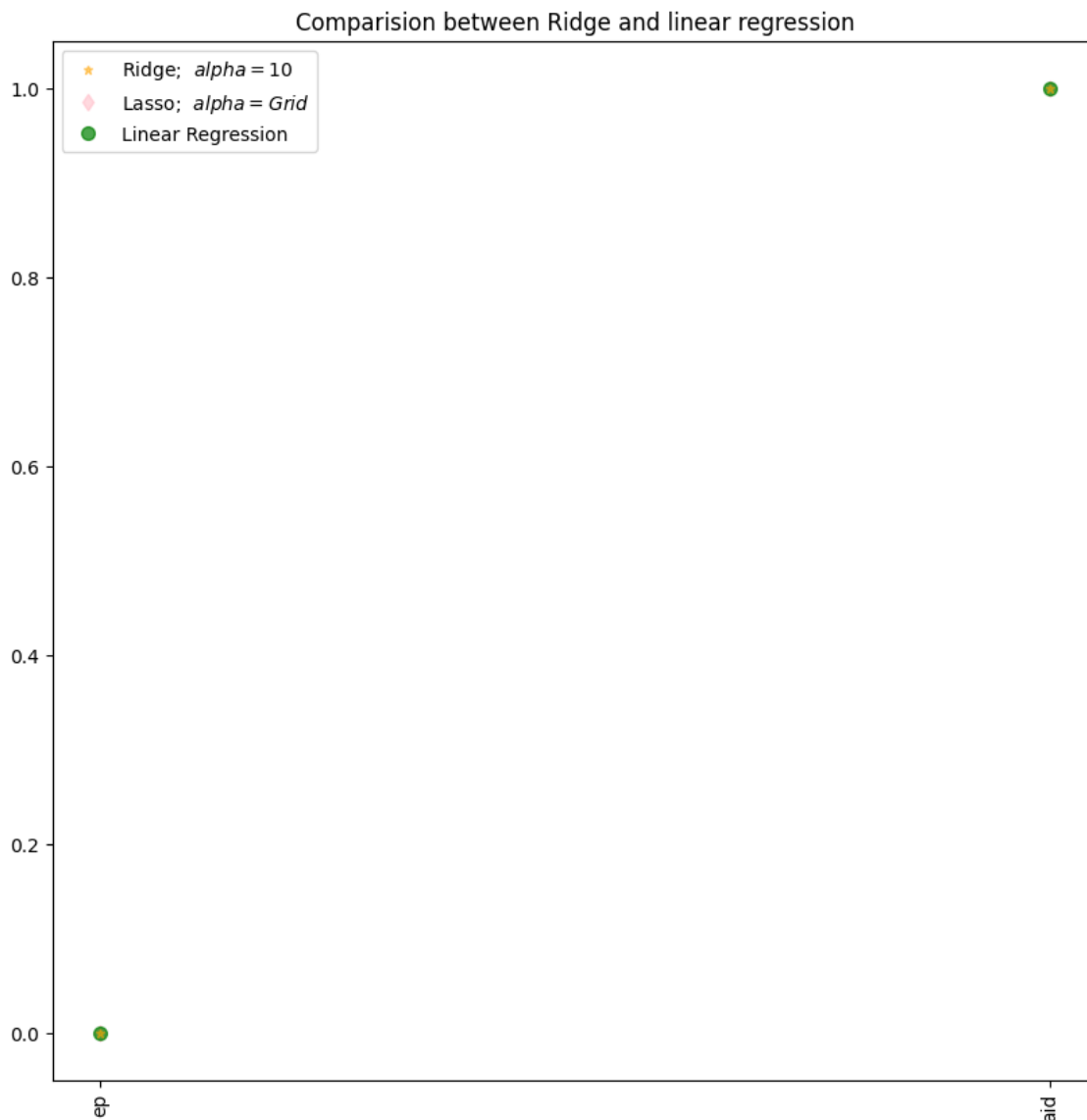


In [37]:

```

1 plt.figure(figsize=(10,10))
2 plt.plot(feature,ridge.coef_,alpha=0.5,marker='*',markersize=5,linestyle='None',col
3 plt.plot(feature,ridge.coef_,alpha=0.6,marker='d',markersize=6,linestyle='None',col
4 plt.plot(feature,lr.coef_,alpha=0.7,marker='o',markersize=7,color='green',linestyle
5 plt.xticks(rotation=90)
6 plt.title("Comparision between Ridge and linear regression")
7 plt.legend()
8 plt.show()

```



ElasticNet

In [39]:

```
1 from sklearn.linear_model import ElasticNet
2 lr=ElasticNet()
3 lr.fit(x,y)
4 print("The score of ElasticNet is {}".format(lr.score(x_train,y_train)))
5 print("The score of ElasticNet is {}".format(lr.score(x_test,y_test)))
6 print(lr.coef_)
7 print(lr.intercept_)
```

The score of ElasticNet is 0.999999999996378
The score of ElasticNet is 0.999999999996378
[0. 0.9999994]
0.0009934970473750582

In [42]:

```
1 y_pred_elastic = lr.predict(x_test)
2 mean_squared_error = np.mean((y_pred_elastic-y_test)**2)
3 print("The mean squared error is {}".format(mean_squared_error))
```

The mean squared error is 5.844007028719034e-07

In []:

```
1
```