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\HP\OneDrive\Documents\fiat500_VehicleSelection_Dataset.cs
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.61
1	2	pop	51	1186	32500	1	45.666359	12.24
2	3	sport	74	4658	142228	1	45.503300	11.41
3	4	lounge	51	2739	160000	1	40.633171	17.63
4	5	pop	73	3074	106880	1	41.903221	12.49
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.70
1534	1535	lounge	74	3835	112000	1	45.845692	8.66
1535	1536	pop	51	2223	60457	1	45.481541	9.41
1536	1537	lounge	51	2557	80750	1	45.000702	7.68
1537	1538	pop	51	1766	54276	1	40.323410	17.56

1538 rows × 9 columns

In [3]:

```
df=df[['engine_power','age_in_days']]
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]:

```
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]:

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

In [25]:

```
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

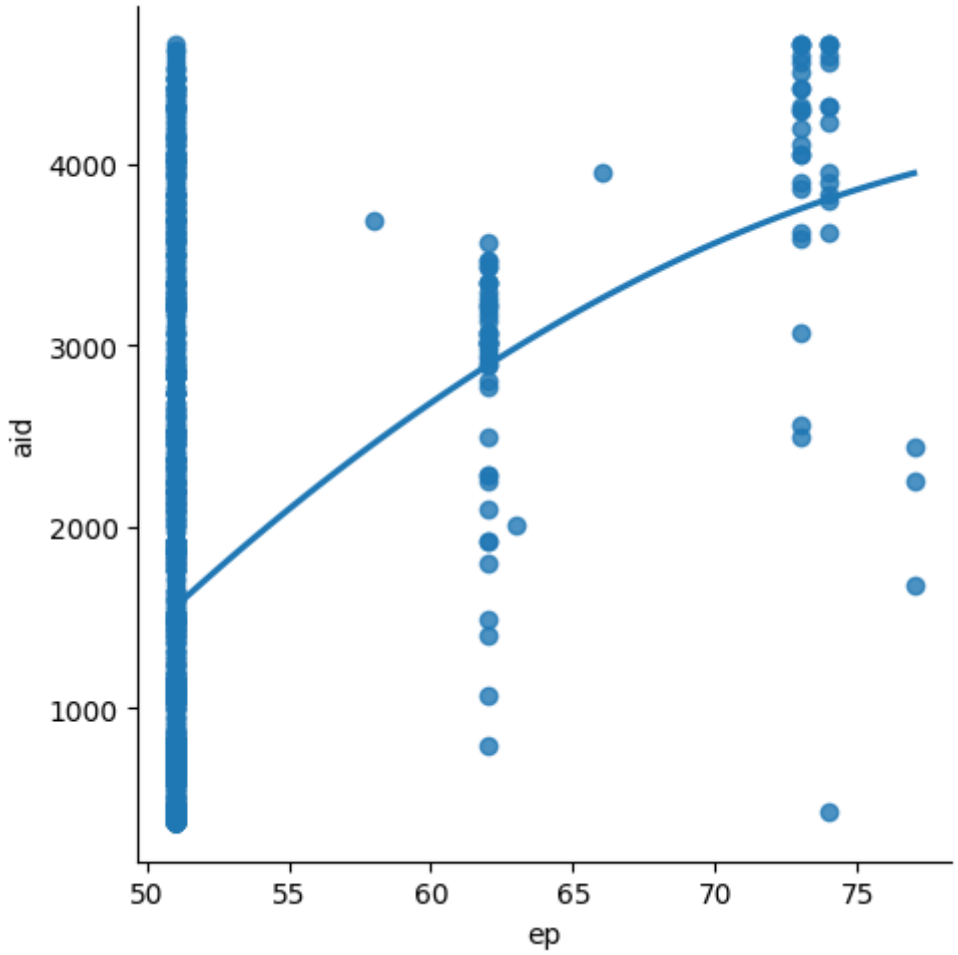
1538 rows × 2 columns

In [26]:

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

Out[26]:

```
<seaborn.axisgrid.FacetGrid at 0x247bfdeb9d0>
```



In [27]:

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

```
1.0
```

In [28]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

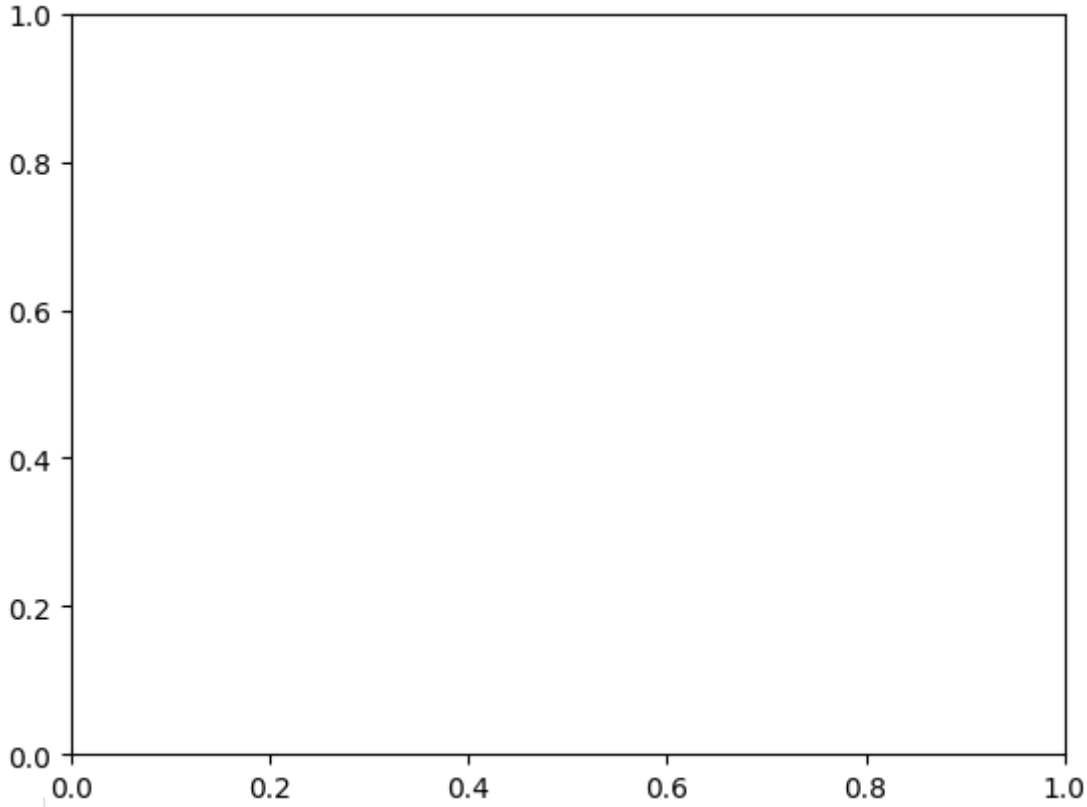
```
---------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
Cell In[28], line 2
      1 y_pred=regr.predict(x_test)
----> 2 plt.scatter(x_test,y_test,color='b')
      3 plt.plot(x_test,y_pred,color='k')
   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

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotl
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\matplotl
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\matplotl
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
```



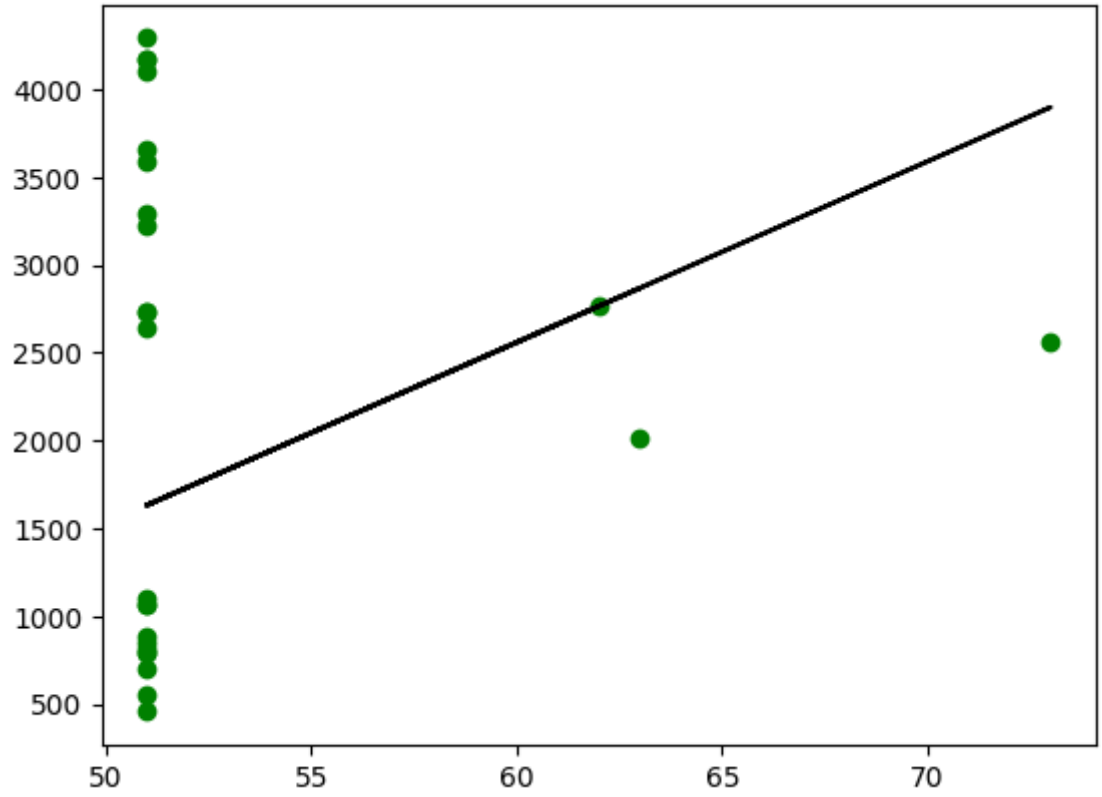

```
df100=df[:][:100]
sns.lmplot(x='ep',y='aid',data=df100,order=1,ci=None)

```

In [29]:

```
df100.fillna(method='ffill',inplace=True)
X=np.array(df100['ep']).reshape(-1,1)
y=np.array(df100['aid']).reshape(-1,1)
df100.dropna(inplace=True)
X_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(x_test,y_test))
print("Regression: ",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
-0.18690025900835017
Regression:  -0.18690025900835017
```



In [30]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
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:  -0.18690025900835017
```

Conclusion:

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

In [31]:

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

```
1.0
1.0
```

In [32]:

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso,LassoCV
ridge = Ridge(alpha=10)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print('\n Ridge method\n')
print('The score of ridge method is {}'.format(train_score_ridge))
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]:

```
lasso = Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_lasso=lasso.score(x_train,y_train)
test_score_lasso=lasso.score(x_test,y_test)
print('\n Lasso method\n')
print('The score of lasso method is {}'.format(train_score_lasso))
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]:

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

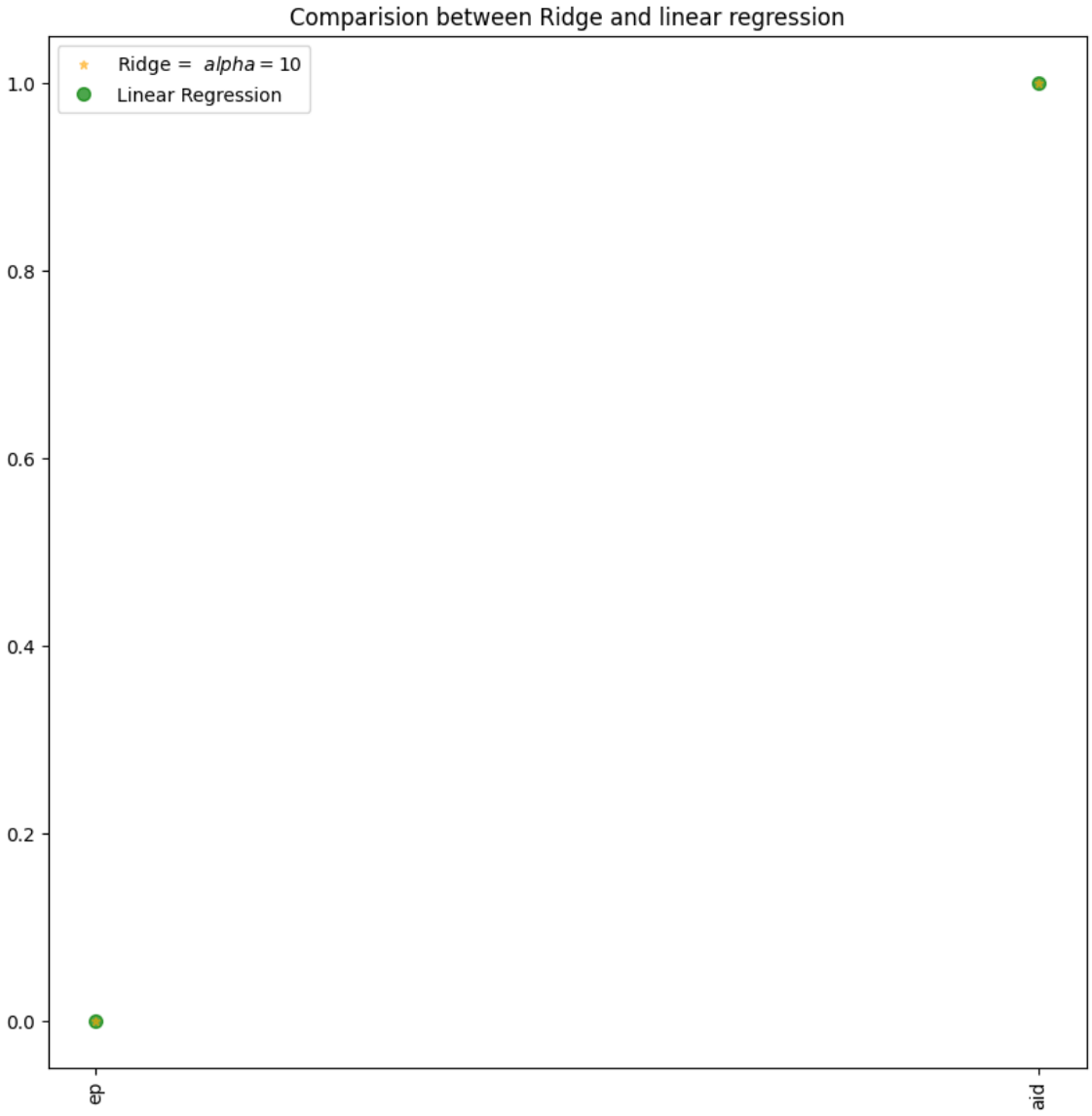
```
ridge_cv=RidgeCV(alphas=[1,2.3,0.2,0.3,0.4,0.5,0.6]).fit(x_train,y_train)
print("\n RidgeCV Method\n")
print("The score of Ridge method is {}".format(ridge_cv.score(x_train,y_train)))
print("The score of Ridge method is {}".format(ridge_cv.score(x_test,y_test)))
```

```
 RidgeCV Method

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

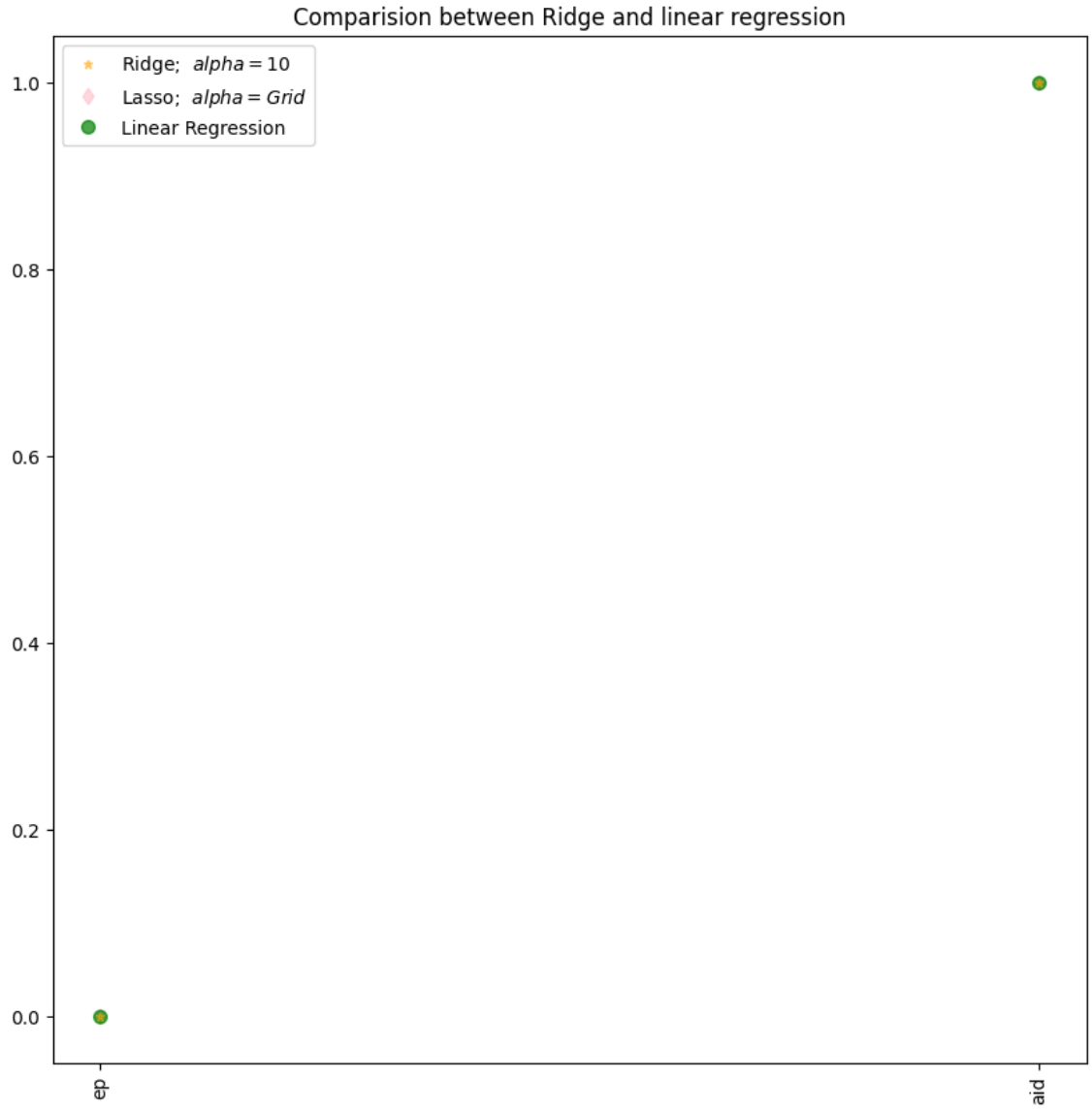
In [36]:

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



In [37]:

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



ElasticNet

In [39]:

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

```
The score of ElasticNet is 0.9999999999996378
The score of ElasticNet is 0.9999999999996378
[0.        0.9999994]
0.0009934970473750582
```

In [42]:

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

```
The mean squeared error is 5.844007028719034e-07
```

In [ ]:

```

```