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]:

```
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]:

```
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]:

```
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 [7]:

```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_22436\4116506308.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [8]:

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

In [9]:

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

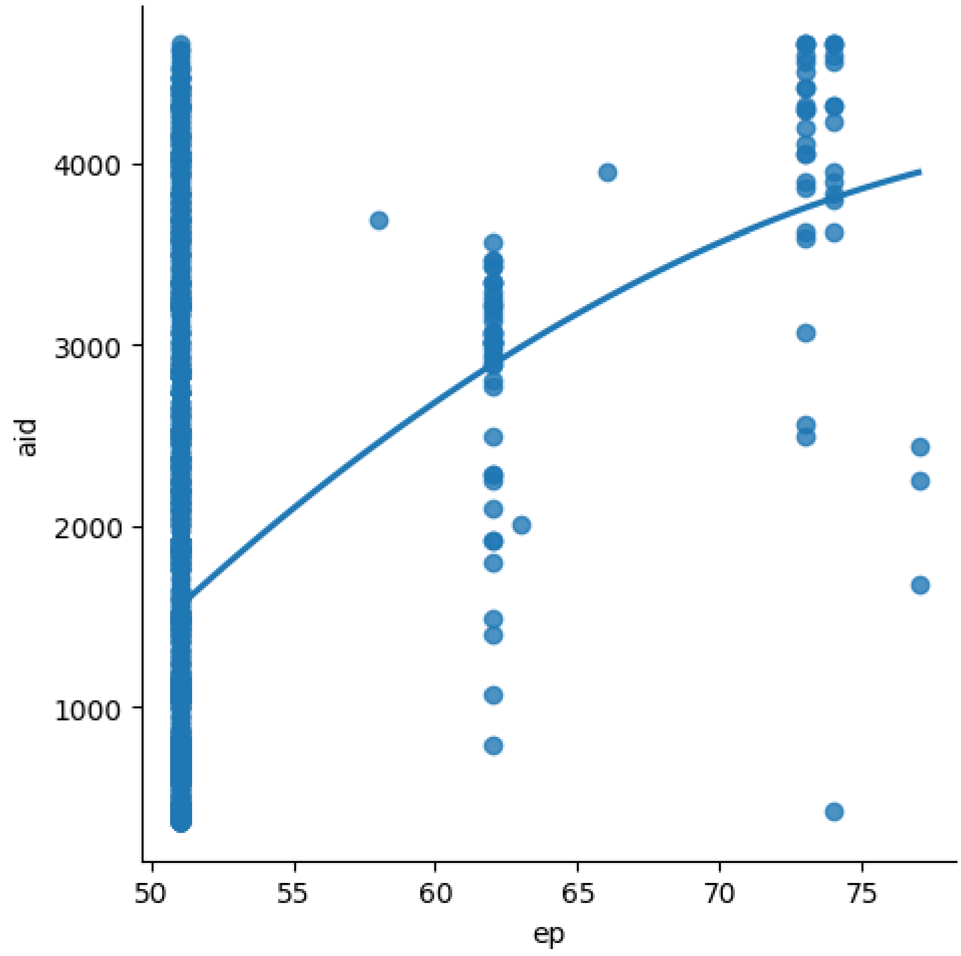
```
C:\Users\HP\AppData\Local\Temp\ipykernel_22436\1379821321.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

In [10]:

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

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x20e4319bb50>
```



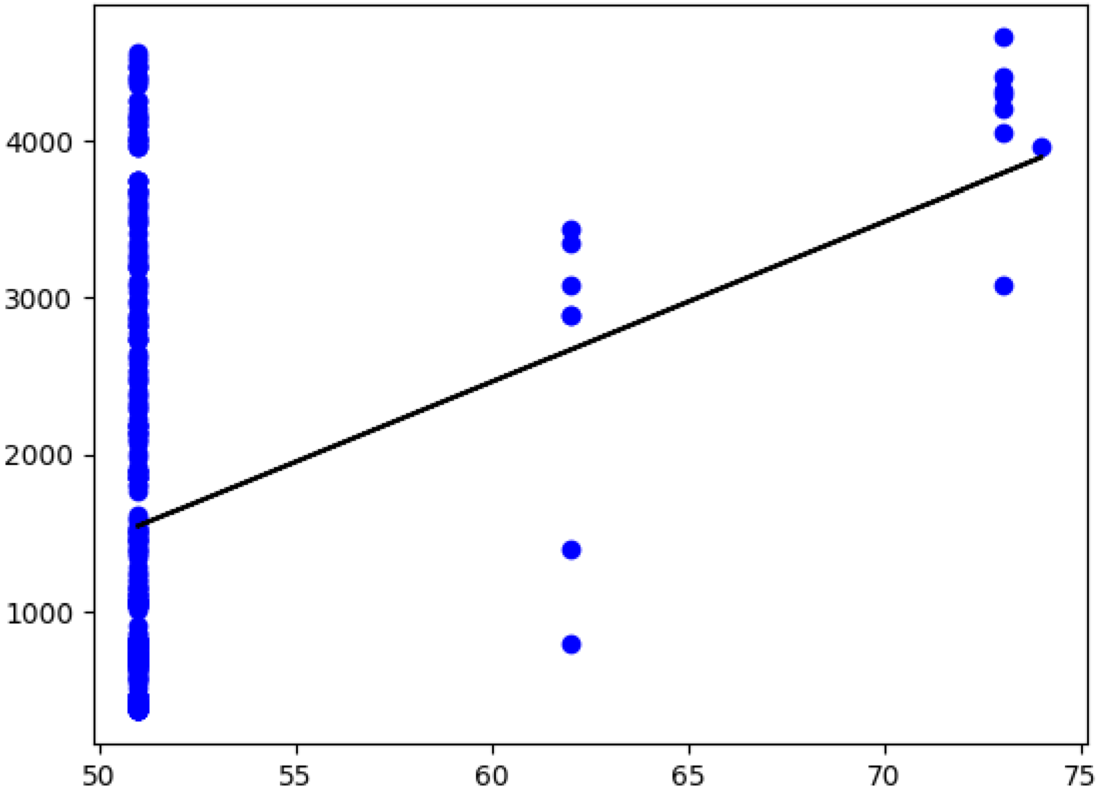
In [11]:

```
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))
```

```
0.08266478450766102
```

In [12]:

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

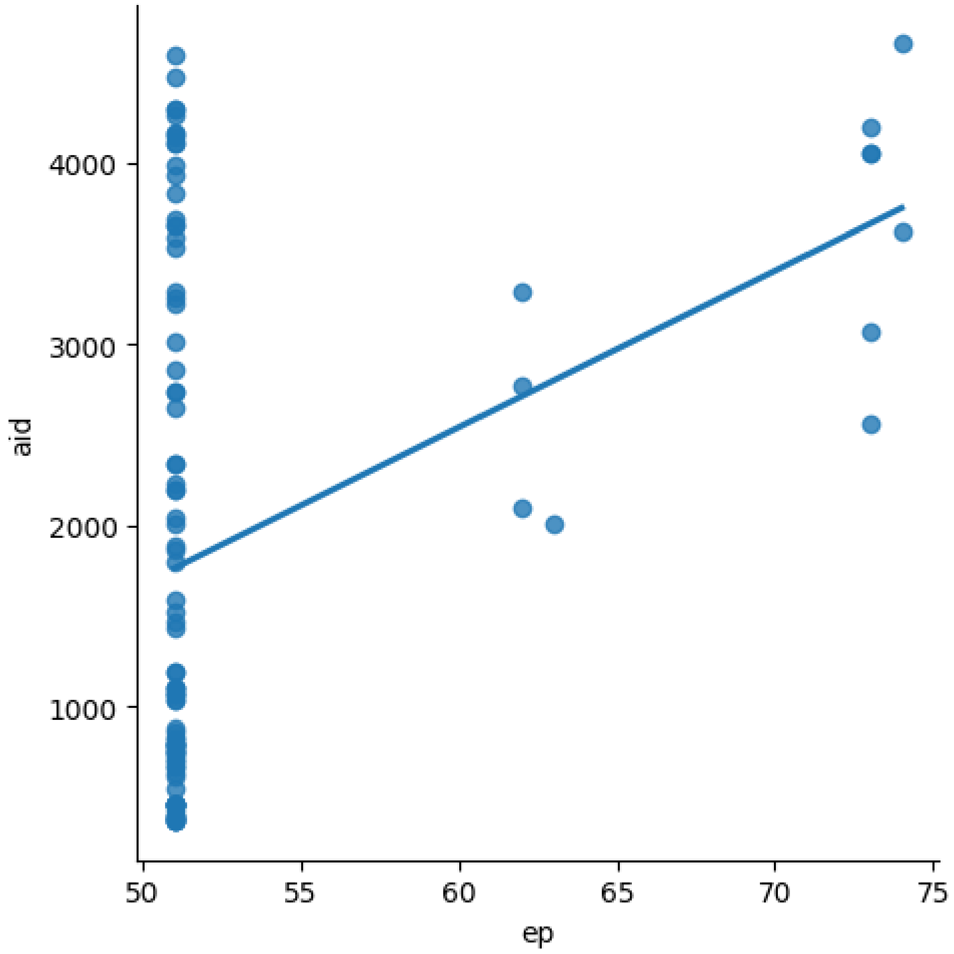


In [13]:

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

```

Out[13]:

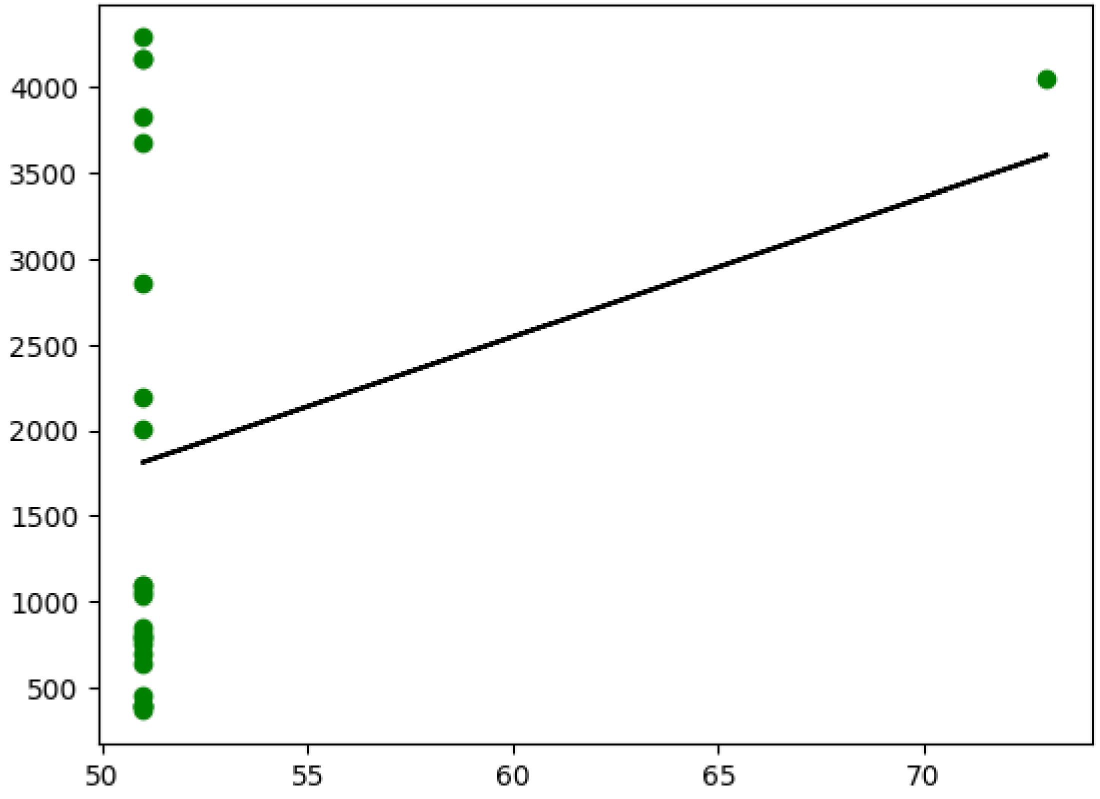
```
<seaborn.axisgrid.FacetGrid at 0x20e28fa1ab0>
```



In [14]:

```
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()
```

```
0.08993389468303326
Regression:  0.08993389468303326
```



In [15]:

```
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:  0.08993389468303326
```

Conclusion:

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

In [16]:

```python
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 [17]:

```python
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 [18]:

```python
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 0.9999999999632855
The score of lasso method is 0.9999999999632833
```

In [19]:

```
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
```

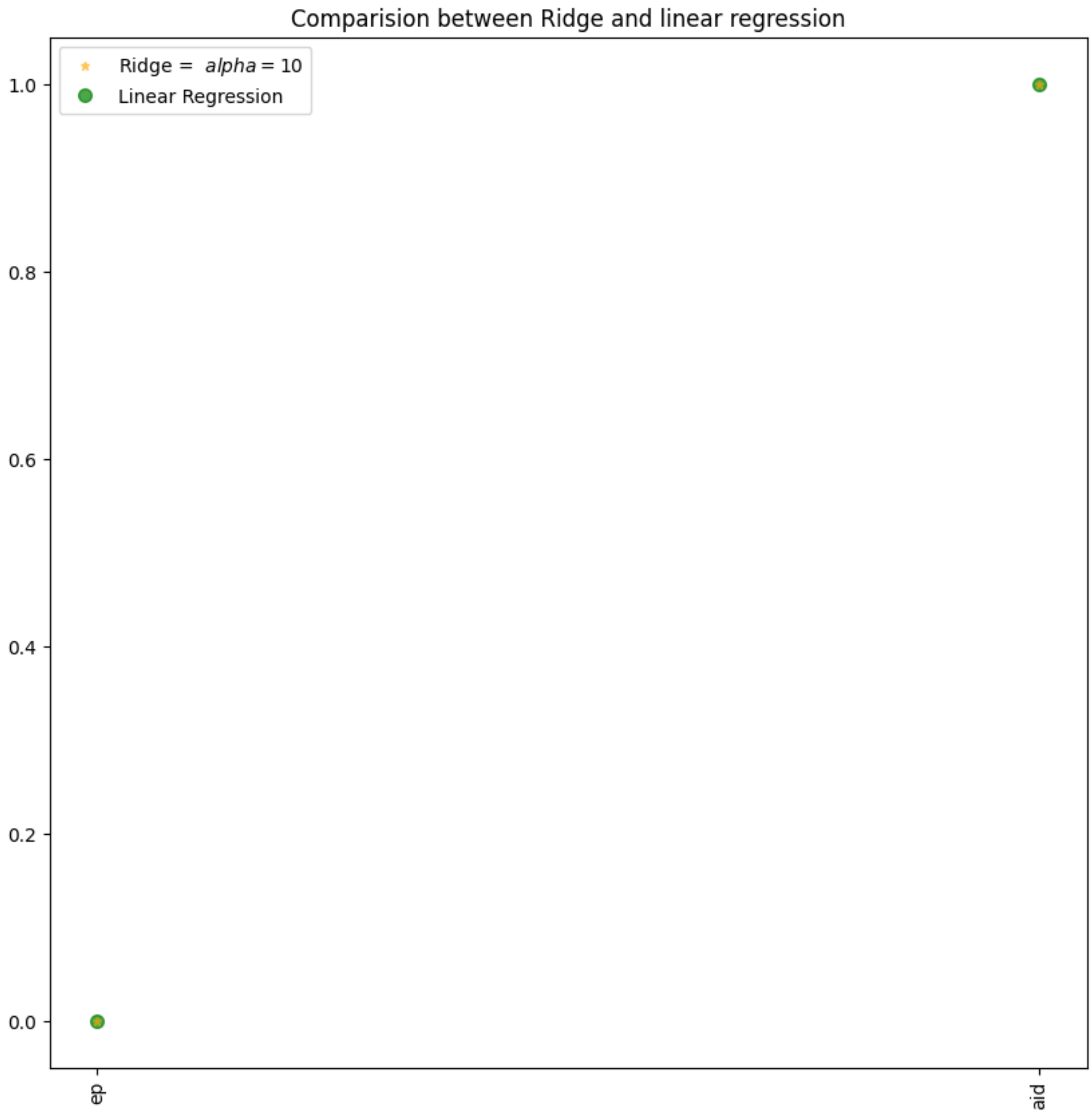
In [20]:

```
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 0.9999999996466192
The score of Ridge method is 0.9999999996465979
```

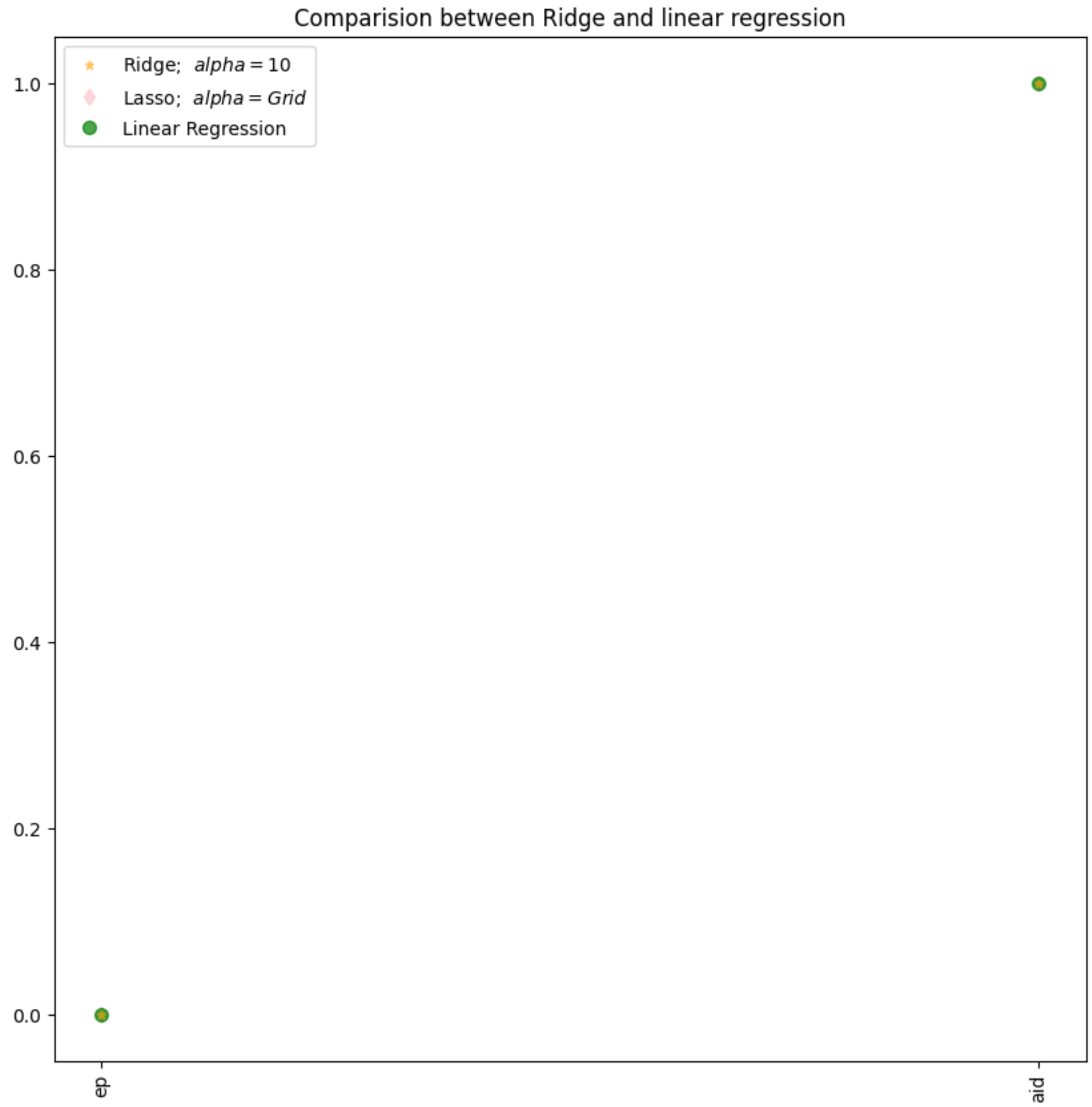
In [21]:

```
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 [22]:

```
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()
```



In [ ]:

```

```