In [176]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
print(data)
```

```
        ID   model  engine_power  age_in_days      km  previous_owners  \
0        1  lounge            51          882   25000                1
1        2     pop            51         1186   32500                1
2        3   sport            74         4658  142228                1
3        4  lounge            51         2739  160000                1
4        5     pop            73         3074  106880                1
...    ...     ...           ...          ...     ...              ...
1533  1534   sport            51         3712  115280                1
1534  1535  lounge            74         3835  112000                1
1535  1536     pop            51         2223   60457                1
1536  1537  lounge            51         2557   80750                1
1537  1538     pop            51         1766   54276                1

            lat        lon  price
0     44.907242   8.611560   8900
1     45.666359  12.241890   8800
2     45.503300  11.417840   4200
3     40.633171  17.634609   6000
4     41.903221  12.495650   5700
...         ...        ...    ...
1533  45.069679   7.704920   5200
1534  45.845692   8.666870   4600
1535  45.481541   9.413480   7500
1536  45.000702   7.682270   5990
1537  40.323410  17.568270   7900

[1538 rows x 9 columns]
```

In [177]:
```
data.head(10)
```

Out[177]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
5	6	pop	74	3623	70225	1	45.000702	7.682270	7900
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
8	9	sport	73	4049	76000	1	45.548000	11.549470	5600
9	10	sport	51	3653	89000	1	45.438301	10.991700	6000

In [178]:
```
data.tail()
```

Out[178]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [179]:
```
data.shape
```

Out[179]: (1538, 9)

In [180]:
```
data.columns
```

Out[180]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [181]:
```
data.describe()
```

Out[181]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [182]:
```
data.isna().sum()
```

Out[182]:
```
ID                 0
model              0
engine_power       0
age_in_days        0
km                 0
previous_owners    0
lat                0
lon                0
price              0
dtype: int64
```

remove unwanted columns

In [183]:
```
data1=data.drop(columns=['ID','lon','lat'])
```

In [184]:
```
data1
```

Out[184]:

	model	engine_power	age_in_days	km	previous_owners	price
0	lounge	51	882	25000	1	8900
1	pop	51	1186	32500	1	8800
2	sport	74	4658	142228	1	4200
3	lounge	51	2739	160000	1	6000
4	pop	73	3074	106880	1	5700
...	...	...	...	...	...	...
1533	sport	51	3712	115280	1	5200
1534	lounge	74	3835	112000	1	4600
1535	pop	51	2223	60457	1	7500
1536	lounge	51	2557	80750	1	5990
1537	pop	51	1766	54276	1	7900

1538 rows × 6 columns

convert the strings into numbers on column using get_dummies()

In [185]:
```
data1=pd.get_dummies(data1)
```

In [186]:
```
data1
```

Out[186]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
1	51	1186	32500	1	8800	0	1	0
2	74	4658	142228	1	4200	0	0	1
3	51	2739	160000	1	6000	1	0	0
4	73	3074	106880	1	5700	0	1	0
...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0	1
1534	74	3835	112000	1	4600	1	0	0
1535	51	2223	60457	1	7500	0	1	0
1536	51	2557	80750	1	5990	1	0	0
1537	51	1766	54276	1	7900	0	1	0

1538 rows × 8 columns

In [187]:
```
data2=data1.loc[(data1.model_lounge==1)]
```

In [188]:
```
data2
```

Out[188]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8900	1	0	0
3	51	2739	160000	1	6000	1	0	0
6	51	731	11600	1	10750	1	0	0
7	51	1521	49076	1	9190	1	0	0
11	51	366	17500	1	10990	1	0	0
...	...	...	...	...	...	...	...	...
1528	51	2861	126000	1	5500	1	0	0
1529	51	731	22551	1	9900	1	0	0
1530	51	670	29000	1	10800	1	0	0
1534	74	3835	112000	1	4600	1	0	0
1536	51	2557	80750	1	5990	1	0	0

1094 rows × 8 columns

In [189]:
```
y=data2['price']
x=data2.drop(['price','model_pop','model_sport'],axis=1)
```

In [190]:
```
x
```

Out[190]:

	engine_power	age_in_days	km	previous_owners	model_lounge
0	51	882	25000	1	1
3	51	2739	160000	1	1
6	51	731	11600	1	1
7	51	1521	49076	1	1
11	51	366	17500	1	1
...	...	...	...	...	...
1528	51	2861	126000	1	1
1529	51	731	22551	1	1
1530	51	670	29000	1	1
1534	74	3835	112000	1	1
1536	51	2557	80750	1	1

1094 rows × 5 columns

In [191]:
```
y
```

Out[191]:
```
0        8900
3        6000
6       10750
7        9190
11      10990
        ...
1528     5500
1529     9900
1530    10800
1534     4600
1536     5990
Name: price, Length: 1094, dtype: int64
```

splitting the data into training set and testing set

In [192]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [193]:
```
x_train
```

Out[193]:

	engine_power	age_in_days	km	previous_owners	model_lounge
441	51	762	36448	1	1
701	51	701	27100	1	1
695	51	3197	51083	1	1
1415	51	670	33000	1	1
404	51	456	14000	1	1
...	...	...	...	...	...
459	51	397	15628	1	1
654	51	3227	95554	1	1
189	51	1431	81900	1	1
1455	51	701	33942	1	1
1218	51	882	25000	1	1

732 rows × 5 columns

In [194]:
```
y_train
```

Out[194]:
```
441      8980
701     10300
695      5880
1415    10490
404      9499
        ...  
459     10850
654      5900
189     10000
1455     9400
1218     8900
Name: price, Length: 732, dtype: int64
```

In [195]:
```
x_test
```

Out[195]:

	engine_power	age_in_days	km	previous_owners	model_lounge
676	51	762	18609	1	1
215	51	701	25000	1	1
146	51	4018	152900	1	1
1319	51	731	20025	1	1
1041	51	640	38231	1	1
...	...	...	...	...	...
757	51	4018	102841	1	1
167	51	397	15341	1	1
156	51	1858	35304	1	1
1145	51	456	14970	1	1
1393	51	609	32665	2	1

362 rows × 5 columns

In [196]:
```
y_test
```

Out[196]:
```
676     10250
215      9790
146      5500
1319     9900
1041     8900
        ...
757      6000
167     10950
156      8000
1145    10700
1393     9400
Name: price, Length: 362, dtype: int64
```

Ridge Regression

In [197]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
alpha=[1e-15,1e-10,1e-4,1e-3,1e-2,15,10,20,30]
ridge=Ridge()                                         #creating an object for Ridge
parameters={'alpha':alpha}
ridge_regressor=GridSearchCV(ridge,parameters)
ridge_regressor.fit(x_train,y_train)                  #training and fitting
```

Out[197]:
```
▸ GridSearchCV
▸ estimator: Ridge
  ▸ Ridge
```

In [198]:
```
ridge_regressor.best_params_
```

Out[198]:
```
{'alpha': 30}
```

In [199]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred=ridge.predict(x_test)
```

In [200]:
```
Ridge_Error=mean_squared_error(y_pred,y_test)
Ridge_Error
```

Out[200]:
```
519771.8129989745
```

In [201]:
```
from sklearn.metrics import r2_score #to know the efficiency of the predicted price
r2_score(y_test,y_pred)
```

Out[201]:
```
0.8373030813683994
```

In [202]:
```
results=pd.DataFrame(columns=['Actual','Predicted'])
results['Actual']=y_test
results['Predicted']=y_pred
```

In [203]:
```
results
```

Out[203]:

	Actual	Predicted
676	10250	10045.347779
215	9790	9989.171535
146	5500	4769.099603
1319	9900	10048.683238
1041	8900	9813.944798
...	...	...
757	6000	5640.378648
167	10950	10431.681162
156	8000	8765.506865
1145	10700	10384.884273
1393	9400	9929.721685

362 rows × 2 columns

In [204]:
```
results["Difference"]=results.apply(lambda x:x.Actual-x.Predicted,axis=1)
```

In [210]:
```
results["id"]=results.index
```

In [211]:
```
results
```

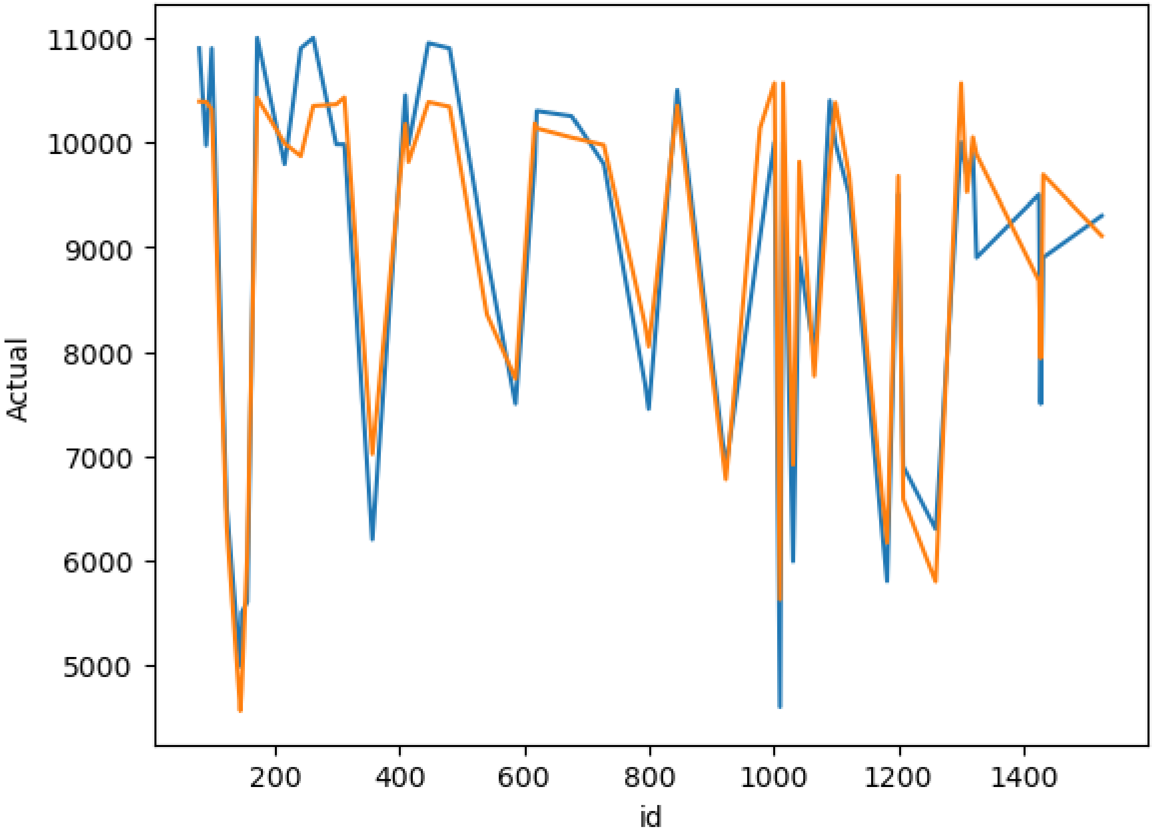
Out[211]:

	Actual	Predicted	Difference	id
676	10250	10045.347779	204.652221	676
215	9790	9989.171535	-199.171535	215
146	5500	4769.099603	730.900397	146
1319	9900	10048.683238	-148.683238	1319
1041	8900	9813.944798	-913.944798	1041
...	...	...	...	...
757	6000	5640.378648	359.621352	757
167	10950	10431.681162	518.318838	167
156	8000	8765.506865	-765.506865	156
1145	10700	10384.884273	315.115727	1145
1393	9400	9929.721685	-529.721685	1393

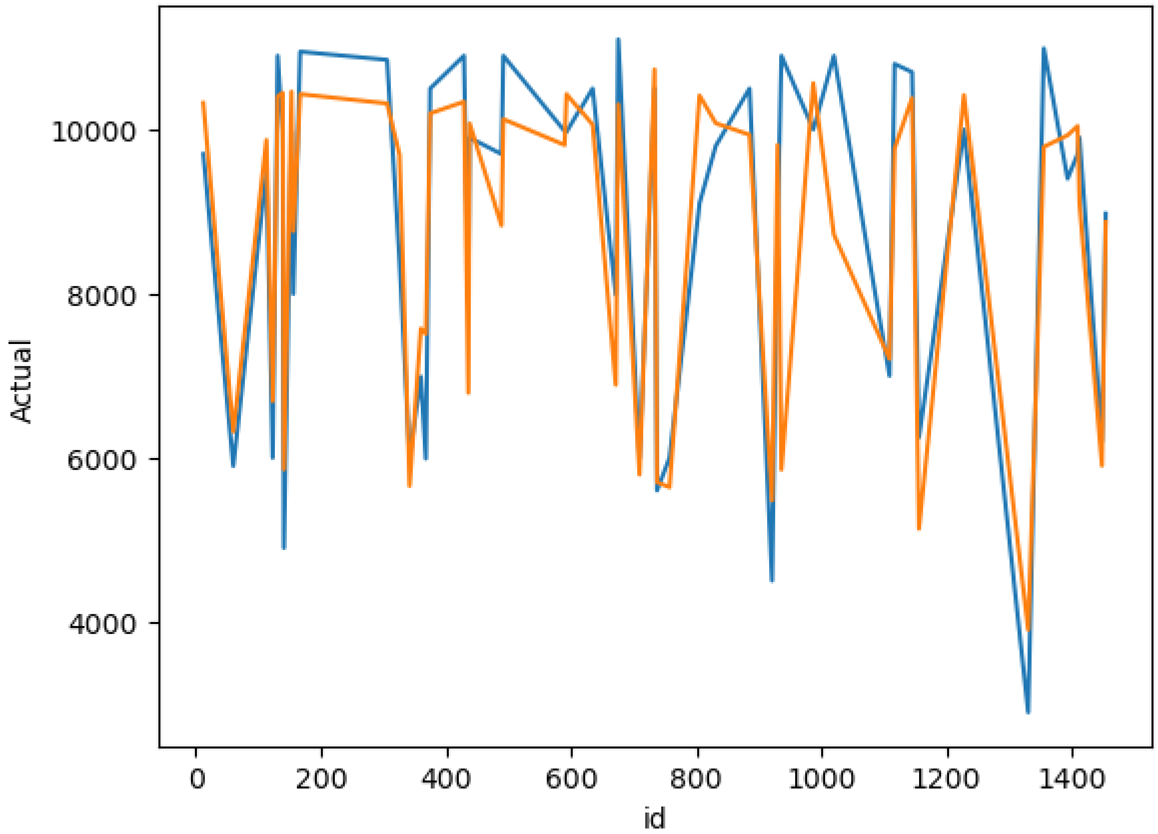
362 rows × 4 columns

Plot the data

In [226]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='id',y='Actual',data=results.head(50))
sns.lineplot(x='id',y='Predicted',data=results.head(50))
plt.show()
```



In [215]:
```
sns.lineplot(x='id',y='Actual',data=results.tail(50))
sns.lineplot(x='id',y='Predicted',data=results.tail(50))
plt.show()
```



In [ ]: