In [279]:
```
import pandas as pd
```

In [280]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [281]:
```
data.describe()
```

Out[281]:
```
                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 [282]:
```
data.head(10)
```

Out[282]:

	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 [283]:
```
data1=data.drop(['lat','ID'],axis=1)
```

In [284]:
```
data1
```

Out[284]:

	model	engine_power	age_in_days	km	previous_owners	lon	price
0	lounge	51	882	25000	1	8.611560	8900
1	pop	51	1186	32500	1	12.241890	8800
2	sport	74	4658	142228	1	11.417840	4200
3	lounge	51	2739	160000	1	17.634609	6000
4	pop	73	3074	106880	1	12.495650	5700
...	...	...	...	...	...	...	...
1533	sport	51	3712	115280	1	7.704920	5200
1534	lounge	74	3835	112000	1	8.666870	4600
1535	pop	51	2223	60457	1	9.413480	7500
1536	lounge	51	2557	80750	1	7.682270	5990
1537	pop	51	1766	54276	1	17.568270	7900

1538 rows × 7 columns

In [285]:
```
data1=pd.get_dummies(data1)
```

In [286]:
```
data1.shape
```

Out[286]:
```
(1538, 9)
```

In [287]:
```
data1
```

Out[287]:

	engine_power	age_in_days	km	previous_owners	lon	price	model_lounge	model_pop	model_sport
0	51	882	25000	1	8.611560	8900	1	0	0
1	51	1186	32500	1	12.241890	8800	0	1	0
2	74	4658	142228	1	11.417840	4200	0	0	1
3	51	2739	160000	1	17.634609	6000	1	0	0
4	73	3074	106880	1	12.495650	5700	0	1	0
...	...	...	...	...	...	...	...	...	...
1533	51	3712	115280	1	7.704920	5200	0	0	1
1534	74	3835	112000	1	8.666870	4600	1	0	0
1535	51	2223	60457	1	9.413480	7500	0	1	0
1536	51	2557	80750	1	7.682270	5990	1	0	0
1537	51	1766	54276	1	17.568270	7900	0	1	0

1538 rows × 9 columns

In [288]:
```
y=data1['price']
```

In [289]:
```
x=data1.drop('price',axis=1)
```

In [290]:

```
y
```

Out[290]:

```
0       8900
1       8800
2       4200
3       6000
4       5700
        ... 
1533    5200
1534    4600
1535    7500
1536    5990
1537    7900
Name: price, Length: 1538, dtype: int64
```

In [291]:

```
#!pip3 install scikit_learn
```

In [292]:

```
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 [293]:

```
from sklearn.model_selection import GridSearchCV
```

In [294]:

```
from sklearn.linear_model import Ridge
```

In [295]:

```
x_train.head(5)
```

Out[295]:

	engine_power	age_in_days	km	previous_owners	lon	model_lounge	model_pop	model_sport
527	51	425	13111	1	7.58602	1	0	0
129	51	1127	21400	1	7.54592	1	0	0
602	51	2039	57039	1	14.52835	0	1	0
331	51	1155	40700	1	12.54016	1	0	0
323	51	425	16783	1	12.49565	1	0	0

In [296]:

```
y_test.head(10)
```

Out[296]:

```
481      7900
76       7900
1502     9400
669      8500
1409     9700
1414     9900
1089     9900
1507     9950
970     10700
1198     8999
Name: price, dtype: int64
```

In [297]:

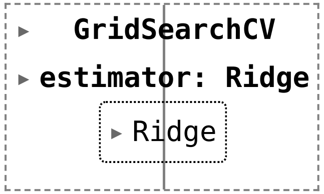
```
alpha=[1e-15,1e-10,1e-10,1e-8,1e-4,1e-3,1e-2,1,5,10,20,30]

ridge = Ridge()
parameters = {'alpha':alpha}
ridge_regressor = GridSearchCV(ridge,parameters)
ridge_regressor.fit(x_train,y_train)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.35498e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=8.73659e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=6.91502e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.08137e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.01088e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.5803e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.24144e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=6.91502e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.08137e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.01088e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.5803e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.24144e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=6.92757e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.09091e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.02113e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.57413e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.23284e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=6.92769e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.09098e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.02123e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.57406e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: I
ll-conditioned matrix (rcond=7.23274e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[297]:



```
▸ GridSearchCV
▸ estimator: Ridge
    ▸ Ridge
```

```
In [298]: ridge_regressor.best_params_

Out[298]: {'alpha': 30}
```

```
In [299]: ridge=Ridge(alpha=30)
          ridge.fit(x_train,y_train)
          y_pred_ridge=ridge.predict(x_test)
```

```
In [300]: from sklearn.metrics import mean_squared_error
          Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
          Ridge_Error

Out[300]: 575383.1771434229
```

```
In [301]: from sklearn.metrics import r2_score
          r2_score(y_test,y_pred_ridge)

Out[301]: 0.8433238795940019
```

In [302]:
```
Results=pd.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=y_pred_ridge
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(15)
```

Out[302]:

	index	price	predicted	ID
0	481	7900	5747.926902	0
1	76	7900	7208.382565	1
2	1502	9400	9818.102659	2
3	669	8500	9769.459155	3
4	1409	9700	10067.159538	4
5	1414	9900	9637.878391	5
6	1089	9900	9645.621130	6
7	1507	9950	10153.777414	7
8	970	10700	9844.599721	8
9	1198	8999	9305.396258	9
10	1088	9890	10479.065427	10
11	576	7990	7772.362188	11
12	965	7380	7666.138173	12
13	1488	6800	6523.634742	13
14	1432	8900	9616.023410	14

In [303]:
```
#extract column syntax.

data2=x.loc[:,"model_lounge"]
```

In [304]:

```
data2
```

Out[304]:

```
0       1
1       0
2       0
3       1
4       0
       ..
1533    0
1534    1
1535    0
1536    1
1537    0
Name: model_lounge, Length: 1538, dtype: uint8
```

In [305]:

```
data2=pd.get_dummies(data2)
```

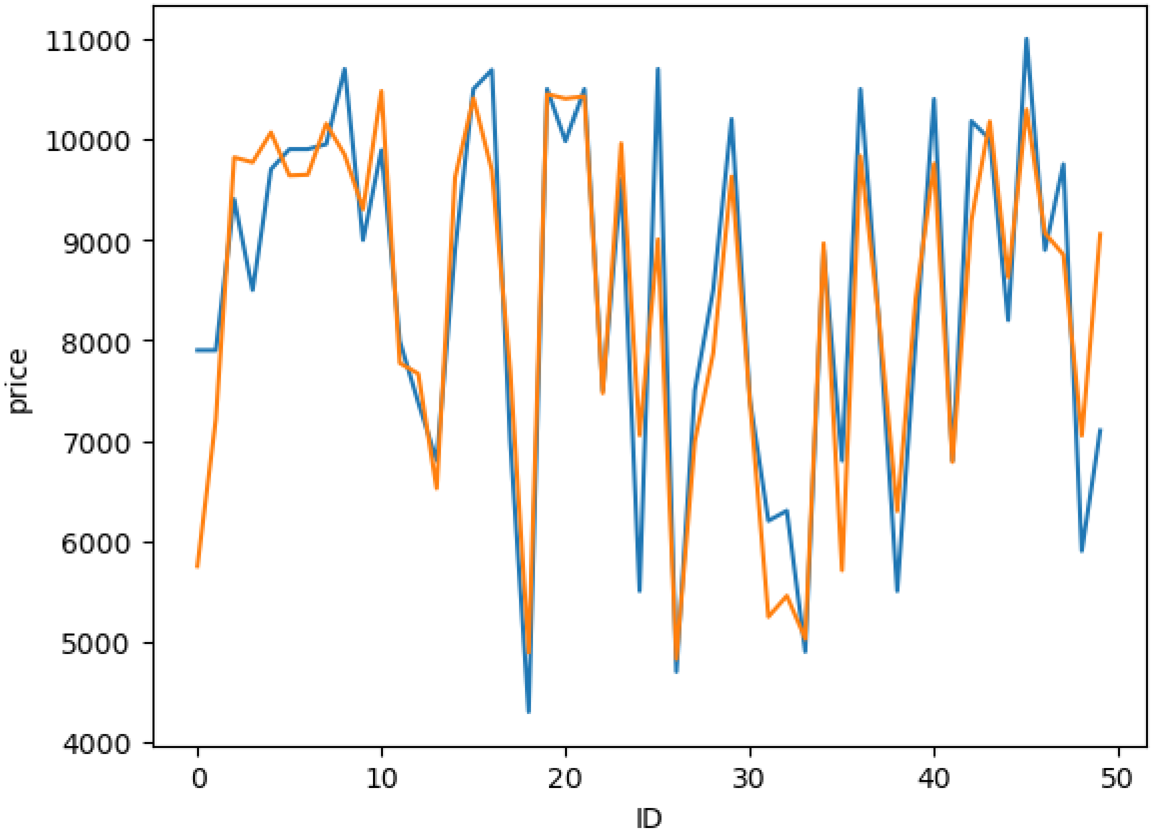
```
In [306]: data2
```

```
Out[306]:
            0   1
        0   0   1
        1   1   0
        2   1   0
        3   0   1
        4   1   0
      ...  ... ...
     1533   1   0
     1534   0   1
     1535   1   0
     1536   0   1
     1537   1   0

1538 rows × 2 columns
```

In [307]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='price',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[307]:
```
[]
```



In [308]:

```
sns.lineplot(x='ID',y='Actual',data=Results.tail(10))
sns.lineplot(x='ID',y='predicted',data=Results.tail(10))
plt.plot()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[308], line 1
----> 1 sns.lineplot(x='ID',y='Actual',data=Results.tail(10))
      2 sns.lineplot(x='ID',y='predicted',data=Results.tail(10))
      3 plt.plot()

File ~/anaconda3/lib/python3.10/site-packages/seaborn/relational.py:618, in lineplot(data, x, y, hue, size, style, units, palette, hue_order, hue_norm, sizes, size_order, size_norm, dashes, markers, style_order, estimator, errorbar, n_boot, seed, orient, sort, err_style, err_kws, legend, ci, ax, **kwargs)
    615 errorbar = _deprecate_ci(errorbar, ci)
    617 variables = _LinePlotter.get_semantics(locals())
--> 618 p = _LinePlotter(
    619     data=data, variables=variables,
    620     estimator=estimator, n_boot=n_boot, seed=seed, errorbar=errorbar,
    621     sort=sort, orient=orient, err_style=err_style, err_kws=err_kws,
    622     legend=legend,
    623 )
    625 p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
    626 p.map_size(sizes=sizes, order=size_order, norm=size_norm)

File ~/anaconda3/lib/python3.10/site-packages/seaborn/relational.py:365, in _LinePlotter.__init__(self, data, variables, estimator, n_boot, seed, errorbar, sort, orient, err_style, err_kws, legend)
    351 def __init__(
    352     self, *,
    353     data=None, variables={},
   (...)
    359     # the kind of plot to draw, but for the time being we need to set
    360     # this information so the SizeMapping can use it
    361     self._default_size_range = (
    362         np.r_[.5, 2] * mpl.rcParams["lines.linewidth"]
    363     )
--> 365     super().__init__(data=data, variables=variables)
    367     self.estimator = estimator
    368     self.errorbar = errorbar

File ~/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:640, in VectorPlotter.__init__(self, dat
a, variables)
    635 # var_ordered is relevant only for categorical axis variables, and may
    636 # be better handled by an internal axis information object that tracks
    637 # such information and is set up by the scale_* methods. The analogous
    638 # information for numeric axes would be information about log scales.
    639 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 640 self.assign_variables(data, variables)
    642 for var, cls in self._semantic_mappings.items():
    643 
    644     # Create the mapping function
    645     map_func = partial(cls.map, plotter=self)

File ~/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:701, in VectorPlotter.assign_variables(se
lf, data, variables)
    699 else:
    700     self.input_format = "long"
--> 701     plot_data, variables = self._assign_variables_longform(
    702         data, **variables,
    703     )
    705 self.plot_data = plot_data
    706 self.variables = variables

File ~/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:938, in VectorPlotter._assign_variables_l
ongform(self, data, **kwargs)
    933 elif isinstance(val, (str, bytes)):
    934 
    935     # This looks like a column name but we don't know what it means!
    937     err = f"Could not interpret value `{val}` for parameter `{key}`"
--> 938     raise ValueError(err)
    940 else:
    941 
    942     # Otherwise, assume the value is itself data
    943 
    944     # Raise when data object is present and a vector can't matched
    945     if isinstance(data, pd.DataFrame) and not isinstance(val, pd.Series):

ValueError: Could not interpret value `Actual` for parameter `y`
```

```
In [ ]:
```