

In [27]:

```
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 [28]:

```
a=pd.read_csv(r"C:\Users\lenovo\Downloads\bottle.csv",low_memory=False)  
a
```

Out[28]:

Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
0	1	1	054.0 056.0 19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900
1	1	2	054.0 056.0 19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600
2	1	3	054.0 056.0 19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400
3	1	4	054.0 056.0 19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300
4	1	5	054.0 056.0 19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300
...
864858	34404	864859	093.4 026.4 20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055
864859	34404	864860	093.4 026.4 20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072
864860	34404	864861	093.4 026.4 20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911
864861	34404	864862	093.4 026.4 20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426

Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
864862	34404	864863	20-	15	17.533	33.3880	5.774	24.15297
			1611SR-					
			MX-310-					
			2239-					
864863	34404	864863	093.4	15	17.533	33.3880	5.774	24.15297
			026.4					
864864	34404	864863	09340264-	15	17.533	33.3880	5.774	24.15297
			0015A-3					

864863 rows × 74 columns

In [29]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 864863 entries, 0 to 864862
```

```
Data columns (total 74 columns):
```

#	Column	Non-Null Count	Dtype
0	Cst_Cnt	864863 non-null	int64
1	Btl_Cnt	864863 non-null	int64
2	Sta_ID	864863 non-null	object
3	Depth_ID	864863 non-null	object
4	Depthm	864863 non-null	int64
5	T_degC	853900 non-null	float64
6	Salnty	817509 non-null	float64
7	O2ml_L	696201 non-null	float64
8	STheta	812174 non-null	float64
9	O2Sat	661274 non-null	float64
10	Oxy_μmol/Kg	661268 non-null	float64
11	BtlNum	118667 non-null	float64
12	RecInd	864863 non-null	int64
13	T_prec	853900 non-null	float64
14	T_qual	23127 non-null	float64
15	S_prec	817509 non-null	float64
16	S_qual	74914 non-null	float64
17	P_qual	673755 non-null	float64
18	O_qual	184676 non-null	float64
19	SThtaq	65823 non-null	float64
20	O2Satq	217797 non-null	float64
21	ChlorA	225272 non-null	float64
22	Chlqua	639166 non-null	float64
23	Phaeop	225271 non-null	float64
24	Phaqua	639170 non-null	float64
25	PO4uM	413317 non-null	float64
26	PO4q	451786 non-null	float64
27	SiO3uM	354091 non-null	float64
28	SiO3qu	510866 non-null	float64
29	NO2uM	337576 non-null	float64
30	NO2q	529474 non-null	float64
31	NO3uM	337403 non-null	float64
32	NO3q	529933 non-null	float64
33	NH3uM	64962 non-null	float64
34	NH3q	808299 non-null	float64
35	C14As1	14432 non-null	float64
36	C14A1p	12760 non-null	float64
37	C14A1q	848605 non-null	float64
38	C14As2	14414 non-null	float64
39	C14A2p	12742 non-null	float64
40	C14A2q	848623 non-null	float64
41	DarkAs	22649 non-null	float64
42	DarkAp	20457 non-null	float64
43	DarkAq	840440 non-null	float64
44	MeanAs	22650 non-null	float64
45	MeanAp	20457 non-null	float64
46	MeanAq	840439 non-null	float64
47	IncTim	14437 non-null	object
48	LightP	18651 non-null	float64
49	R_Depth	864863 non-null	int64
50	R_TEMP	853900 non-null	float64
51	R_POTEMP	818816 non-null	float64
52	R_SALINITY	817509 non-null	float64
53	R_SIGMA	812007 non-null	float64
54	R_SVA	812092 non-null	float64
55	R_DYNHT	818206 non-null	float64

```
56  R_O2                696201 non-null float64
57  R_O2Sat            666448 non-null float64
58  R_SIO3             354099 non-null float64
59  R_PO4              413325 non-null float64
60  R_NO3              337411 non-null float64
61  R_NO2              337584 non-null float64
62  R_NH4              64982  non-null float64
63  R_CHLA             225276 non-null float64
64  R_PHAE0            225275 non-null float64
65  R_PRES             864863 non-null int64
66  R_SAMP             122006 non-null float64
67  DIC1               1999  non-null float64
68  DIC2               224   non-null float64
69  TA1                2084  non-null float64
70  TA2                234   non-null float64
71  pH2                10    non-null float64
72  pH1                84    non-null float64
73  DIC Quality Comment 55    non-null object
```

dtypes: float64(64), int64(6), object(4)
memory usage: 488.3+ MB

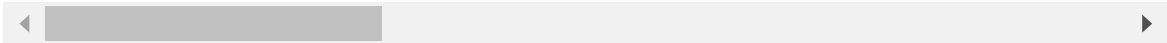
In [30]:

```
a.describe()
```

Out[30]:

	Cst_Cnt	Btl_Cnt	Depthm	T_degC	Salnty	O
count	864863.000000	864863.000000	864863.000000	853900.000000	817509.000000	696201.0
mean	17138.790958	432432.000000	226.831951	10.799677	33.840350	3.3
std	10240.949817	249664.587269	316.050259	4.243825	0.461843	2.0
min	1.000000	1.000000	0.000000	1.440000	28.431000	-0.0
25%	8269.000000	216216.500000	46.000000	7.680000	33.488000	1.3
50%	16848.000000	432432.000000	125.000000	10.060000	33.863000	3.4
75%	26557.000000	648647.500000	300.000000	13.880000	34.196900	5.5
max	34404.000000	864863.000000	5351.000000	31.140000	37.034000	11.1

8 rows × 70 columns



In [31]:

```
a.isna().any()
```

Out[31]:

```
Cst_Cnt          False
Btl_Cnt          False
Sta_ID           False
Depth_ID         False
Depthm           False
...
TA1              True
TA2              True
pH2              True
pH1              True
DIC Quality Comment  True
Length: 74, dtype: bool
```

In [32]:

```
a.isnull().sum()
```

Out[32]:

```
Cst_Cnt          0
Btl_Cnt          0
Sta_ID           0
Depth_ID         0
Depthm           0
...
TA1             862779
TA2             864629
pH2             864853
pH1             864779
DIC Quality Comment  864808
Length: 74, dtype: int64
```

In [33]:

```
a.isnull().sum()
```

Out[33]:

```
Cst_Cnt          0
Btl_Cnt          0
Sta_ID           0
Depth_ID         0
Depthm           0
...
TA1             862779
TA2             864629
pH2             864853
pH1             864779
DIC Quality Comment  864808
Length: 74, dtype: int64
```


In [34]:

```
a=a[['Salnty','T_degC']]  
a.columns=['Sal','Temp']
```

In [35]:

```
a.head(20)
```

Out[35]:

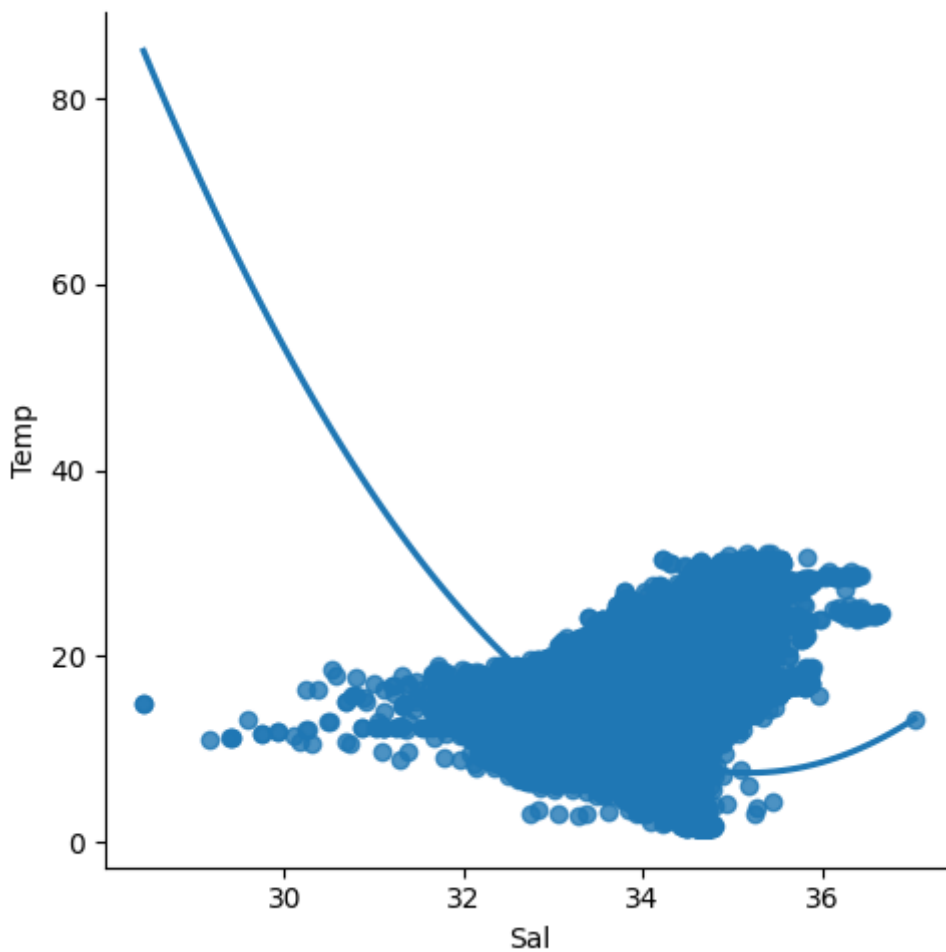
	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45
5	33.431	10.45
6	33.440	10.45
7	33.424	10.24
8	33.420	10.06
9	33.494	9.86
10	33.510	9.83
11	33.580	9.67
12	33.640	9.50
13	33.689	9.32
14	33.847	8.76
15	33.860	8.71
16	33.876	8.53
17	NaN	8.45
18	33.926	8.26
19	33.980	7.96

In [36]:

```
sns.lmplot(x='Sal',y='Temp',data=a,order=2,ci=None)
```

Out[36]:

<seaborn.axisgrid.FacetGrid at 0x2761835b650>



In [37]:

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

C:\Users\lenovo\AppData\Local\Temp\ipykernel_8348\1870026900.py:1: Setting WithCopyWarning:

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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

In [38]:

```

a.fillna(method='ffill',inplace=True)
x=np.array(a['Sal']).reshape(-1,1)
y=np.array(a['Temp']).reshape(-1,1)
a.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("Regression: ",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()

```

C:\Users\lenovo\AppData\Local\Temp\ipykernel_8348\1399910469.py:1: Setting WithCopyWarning:

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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

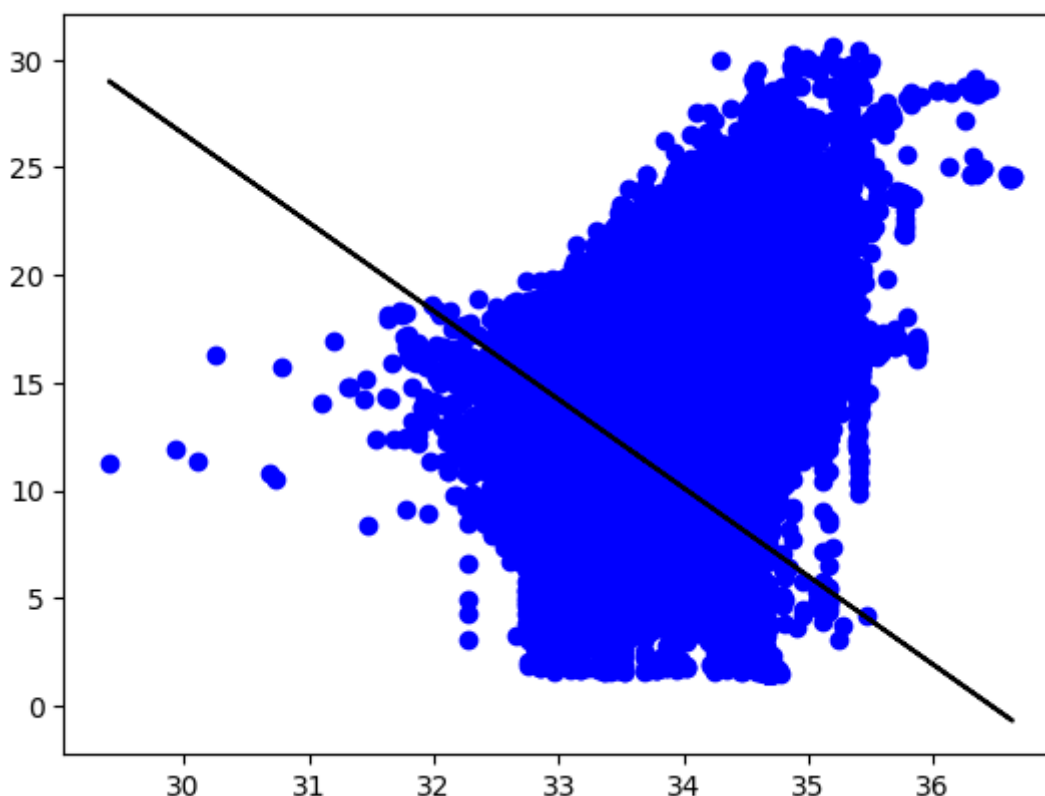
C:\Users\lenovo\AppData\Local\Temp\ipykernel_8348\1399910469.py:4: Setting WithCopyWarning:

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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

Regression: 0.20326068314482237

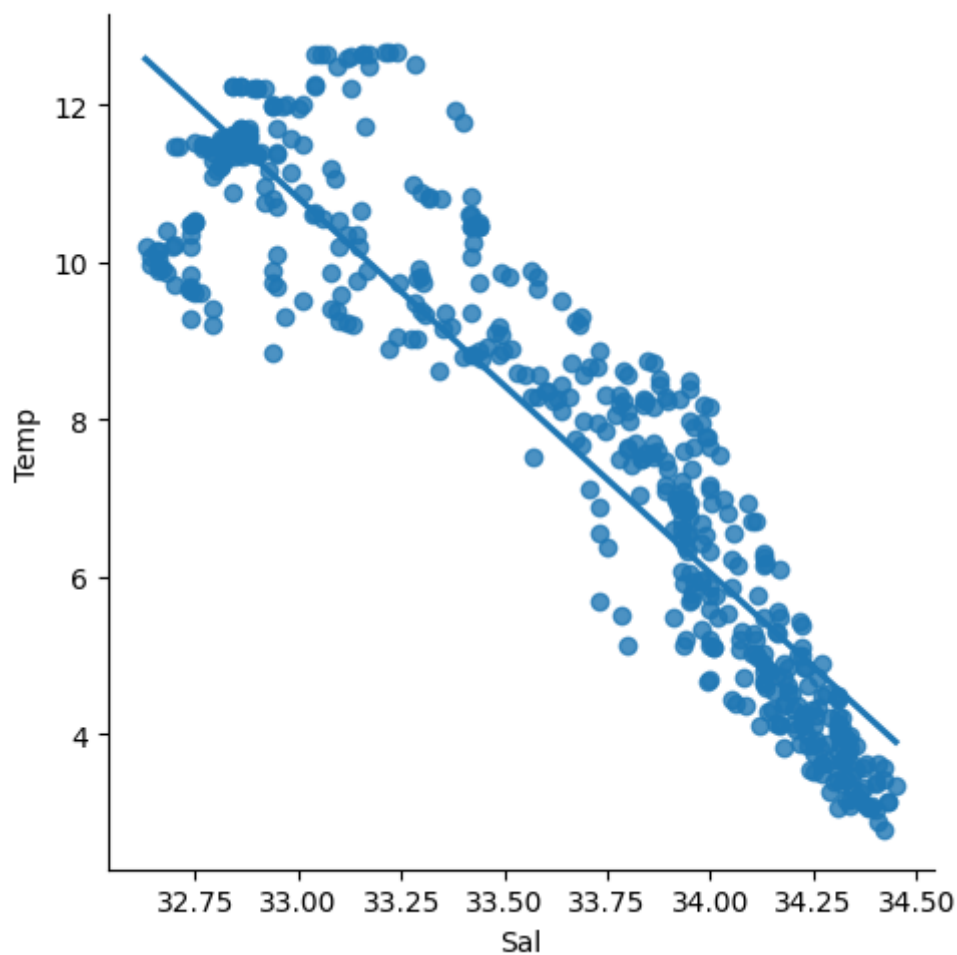


In [39]:

```
a500=a[:, :500]  
sns.lmplot(x='Sal', y='Temp', data=a500, order=1, ci=None)
```

Out[39]:

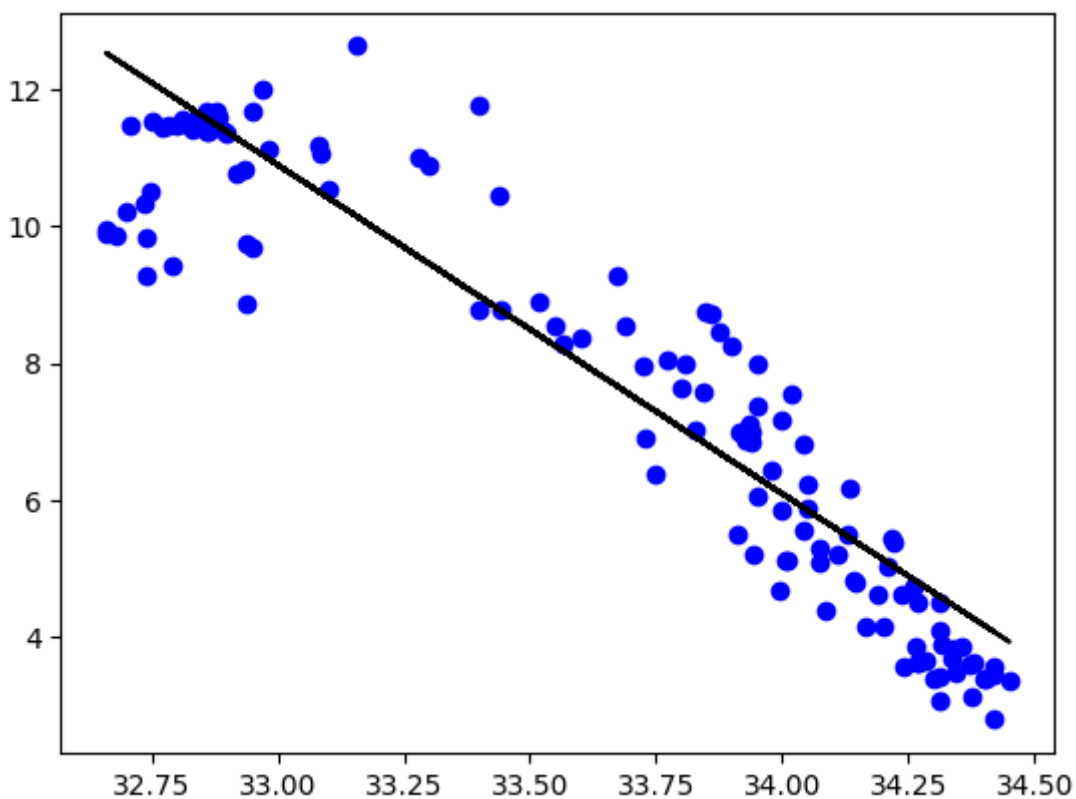
<seaborn.axisgrid.FacetGrid at 0x2761835d390>



In [40]:

```
a500.fillna(method='ffill', inplace=True)
x=np.array(a500['Sal']).reshape(-1,1)
y=np.array(a500['Temp']).reshape(-1,1)
a500.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("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

Regression: 0.8569295828515795



In [41]:

```
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: 0.8569295828515795

In [44]:

```

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
df=pd.read_csv(r"C:\Users\lenovo\Downloads\fiat500_VehicleSelection_Dataset.csv1.csv")
df

```

Out[44]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.611
1	2	pop	51	1186	32500	1	45.666359	12.241
2	3	sport	74	4658	142228	1	45.503300	11.417
3	4	lounge	51	2739	160000	1	40.633171	17.634
4	5	pop	73	3074	106880	1	41.903221	12.495
...
1533	1534	sport	51	3712	115280	1	45.069679	7.704
1534	1535	lounge	74	3835	112000	1	45.845692	8.666
1535	1536	pop	51	2223	60457	1	45.481541	9.413
1536	1537	lounge	51	2557	80750	1	45.000702	7.682
1537	1538	pop	51	1766	54276	1	40.323410	17.568

1538 rows × 9 columns

In [45]:

df.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   ID              1538 non-null   int64
1   model           1538 non-null   object
2   engine_power    1538 non-null   int64
3   age_in_days     1538 non-null   int64
4   km              1538 non-null   int64
5   previous_owners 1538 non-null   int64
6   lat             1538 non-null   float64
7   lon             1538 non-null   float64
8   price           1538 non-null   int64
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB

```

In [46]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                     1538 non-null   int64
1   model                  1538 non-null   object
2   engine_power           1538 non-null   int64
3   age_in_days            1538 non-null   int64
4   km                     1538 non-null   int64
5   previous_owners        1538 non-null   int64
6   lat                    1538 non-null   float64
7   lon                    1538 non-null   float64
8   price                  1538 non-null   int64
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [47]:

```
df.tail()
```

Out[47]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
1533	1534	sport	51	3712	115280	1	45.069679	7.7041
1534	1535	lounge	74	3835	112000	1	45.845692	8.6661
1535	1536	pop	51	2223	60457	1	45.481541	9.4131
1536	1537	lounge	51	2557	80750	1	45.000702	7.6821
1537	1538	pop	51	1766	54276	1	40.323410	17.5681

In [49]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                     1538 non-null   int64
1   model                  1538 non-null   object
2   engine_power           1538 non-null   int64
3   age_in_days            1538 non-null   int64
4   km                     1538 non-null   int64
5   previous_owners        1538 non-null   int64
6   lat                    1538 non-null   float64
7   lon                    1538 non-null   float64
8   price                  1538 non-null   int64
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [50]:

```
df.describe()
```

Out[50]:

	ID	engine_power	age_in_days	km	previous_owners	lat
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.54136
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.13351
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.85583
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.80295
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.39405
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.46796
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.79561

In [51]:

```
df.isna().any()
```

Out[51]:

```
ID                False
model             False
engine_power      False
age_in_days       False
km               False
previous_owners   False
lat              False
lon              False
price            False
dtype: bool
```

In [52]:

```
df.isnull().sum()
```

Out[52]:

```
ID                0
model             0
engine_power      0
age_in_days       0
km               0
previous_owners   0
lat              0
lon              0
price            0
dtype: int64
```


In [53]:

```
df.isnull()
```

Out[53]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...
1533	False	False	False	False	False	False	False	False	False
1534	False	False	False	False	False	False	False	False	False
1535	False	False	False	False	False	False	False	False	False
1536	False	False	False	False	False	False	False	False	False
1537	False	False	False	False	False	False	False	False	False

1538 rows × 9 columns



In [54]:

```
df.loc[:11,["ID","price"]]
```

Out[54]:

	ID	price
0	1	8900
1	2	8800
2	3	4200
3	4	6000
4	5	5700
5	6	7900
6	7	10750
7	8	9190
8	9	5600
9	10	6000
10	11	8950
11	12	10990

In [55]:

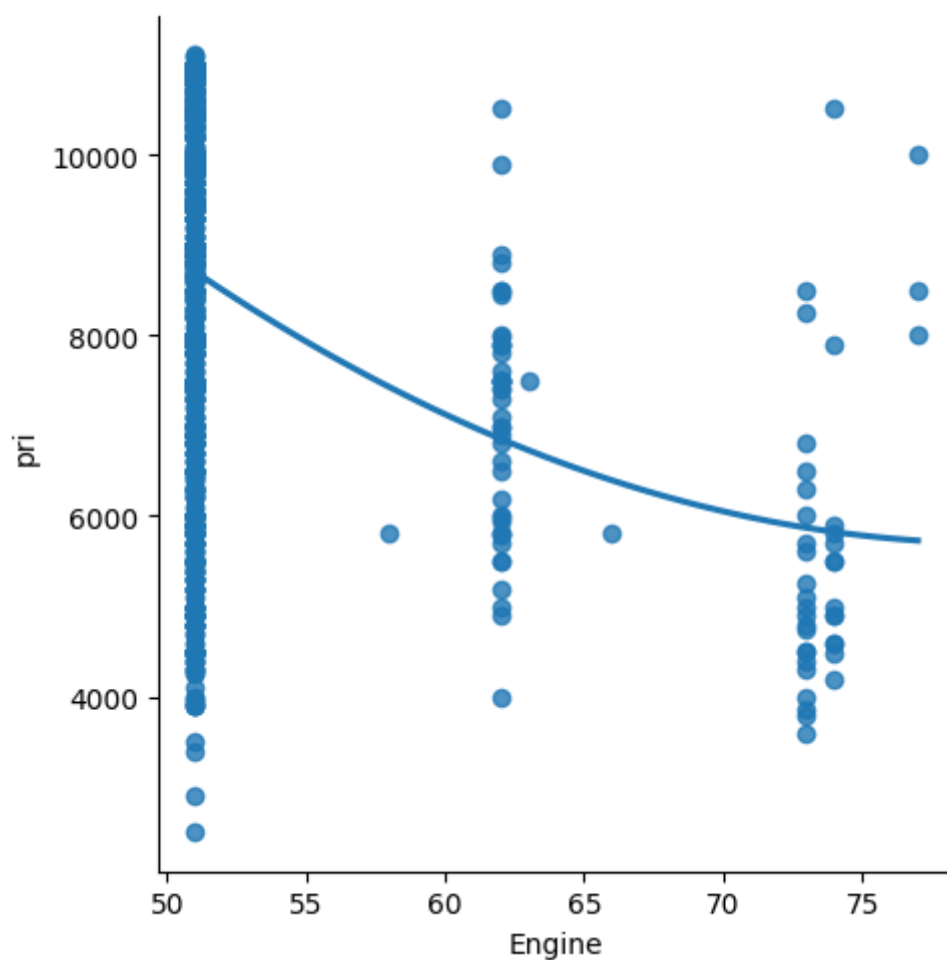
```
df=df[["engine_power","price"]]  
df.columns=["Engine","pri"]
```

In [56]:

```
sns.lmplot(x='Engine',y='pri',data=df,order=2,ci=None)
```

Out[56]:

<seaborn.axisgrid.FacetGrid at 0x276059543d0>



In [57]:

```
df.describe()
```

Out[57]:

	Engine	pri
count	1538.000000	1538.000000
mean	51.904421	8576.003901
std	3.988023	1939.958641
min	51.000000	2500.000000
25%	51.000000	7122.500000
50%	51.000000	9000.000000
75%	51.000000	10000.000000
max	77.000000	11100.000000

In [58]:

```
df.fillna(method="ffill")
```

Out[58]:

	Engine	pri
0	51	8900
1	51	8800
2	74	4200
3	51	6000
4	73	5700
...
1533	51	5200
1534	74	4600
1535	51	7500
1536	51	5990
1537	51	7900

1538 rows × 2 columns

In [59]:

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

In [60]:

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

C:\Users\lenovo\AppData\Local\Temp\ipykernel_8348\1379821321.py:1: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

In [61]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

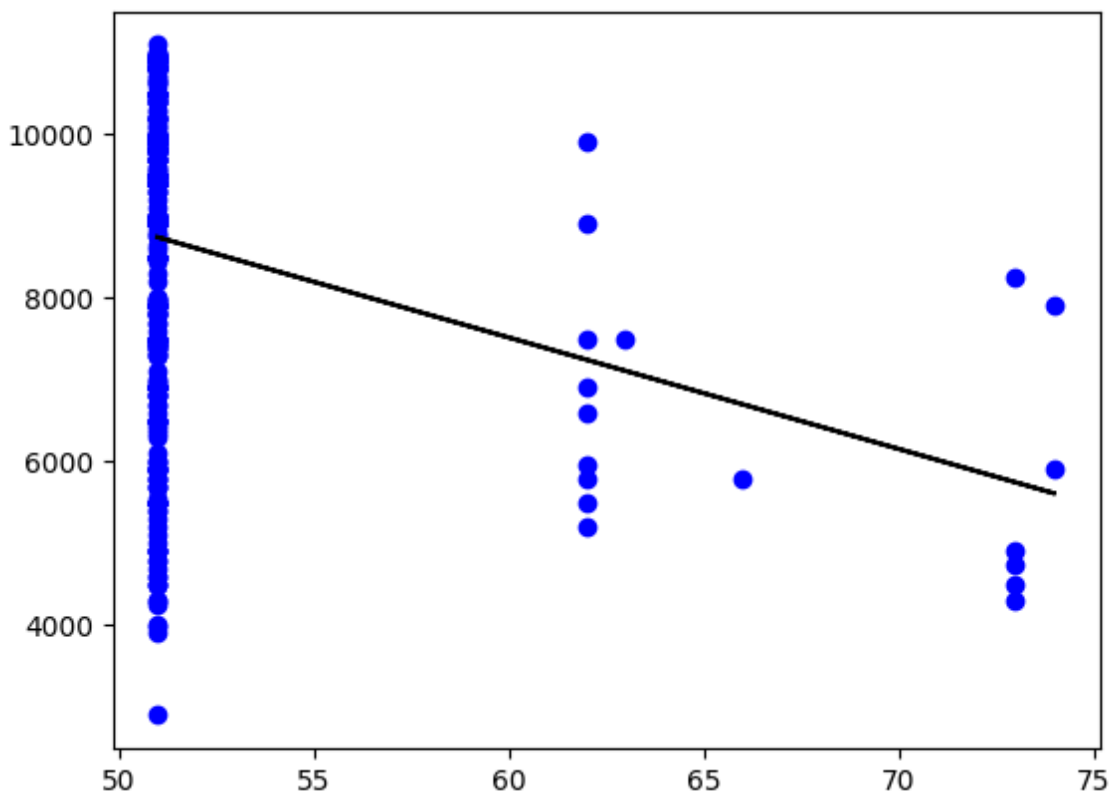
In [62]:

```
regr=LinearRegression()  
regr.fit(x_train,y_train)  
print(regr.score(x_test,y_test))
```

```
0.05091069486195754
```

In [63]:

```
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 [68]:

```
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: 0.05091069486195754