In [10]:

```
#step 1-importing all the requried libraries
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 [11]:

```
#Step-2: Reading the Dataset
df=pd.read_csv(r"C:\Users\mural\Downloads\bottle.csv.zip")
df
```

```
C:\Users\mural\AppData\Local\Temp\ipykernel_15008\447131393.py:2: DtypeWarning: Columns
(47,73) have mixed types. Specify dtype option on import or set low_memory=False.
  df=pd.read_csv(r"C:\Users\mural\Downloads\bottle.csv.zip")
```

Out[11]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHA
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900	NaN	...	N
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	NaN	...	N
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	NaN	...	N
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	NaN	...	N
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	NaN	...	N
...	...	...	...	...	...	...	...	...	...	...	...	
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055	108.74	...	C
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072	108.74	...	C
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911	108.46	...	C
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426	107.74	...	C
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297	105.66	...	C

In [12]:

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

In [13]:

```
df.head(10)
```

Out[13]:

	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

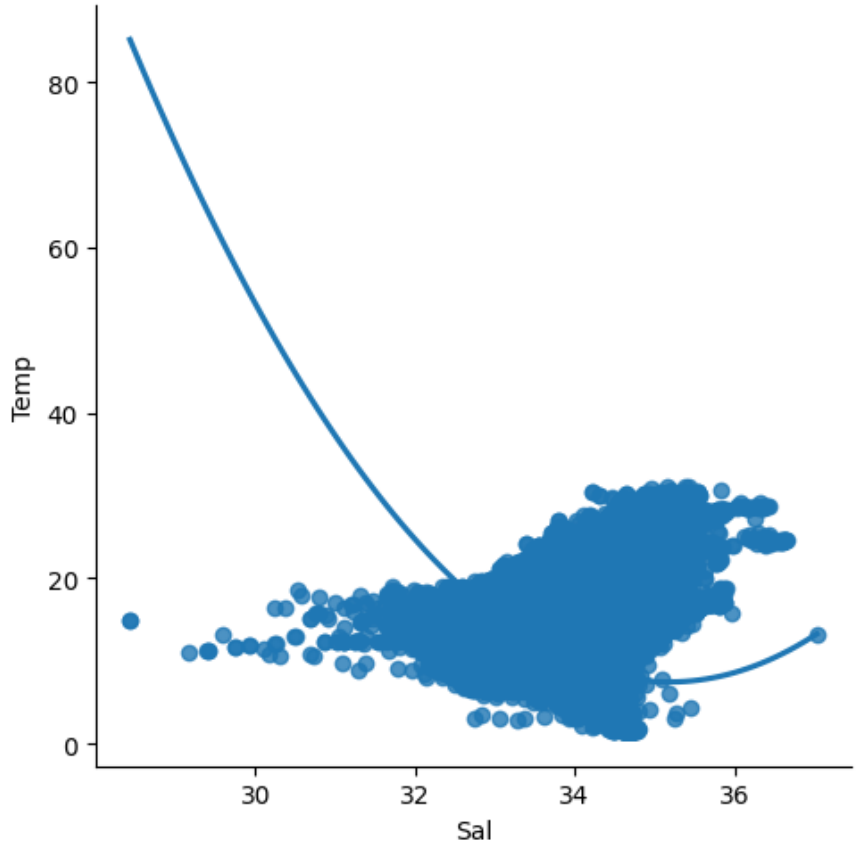
864863 rows × 74 columns

In [14]:

```
#Step-3: Exploring the Data Scatter - plotting the data scatter
sns.lmplot(x="Sal",y="Temp", data = df, order = 2, ci = None)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x2cd16e5ab90>
```



In [15]:

```
df.describe()
```

Out[15]:

	Sal	Temp
count	817509.000000	853900.000000
mean	33.840350	10.799677
std	0.461843	4.243825
min	28.431000	1.440000
25%	33.488000	7.680000
50%	33.863000	10.060000
75%	34.196900	13.880000
max	37.034000	31.140000

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Sal     817509 non-null  float64
 1   Temp    853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [17]:

```
#Step-4: Data cleaning - Eliminating NaN OR missing input numbers

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

```
C:\Users\mural\AppData\Local\Temp\ipykernel_15008\3532286049.py:3: SettingWithCopyWarnin
g:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method ='ffill', inplace = True)
```

In [18]:

```
# Step-5: Training Our Model
X = np.array(df['Sal']).reshape(-1, 1)
y = np.array(df['Temp']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

In [19]:

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

```
C:\Users\mural\AppData\Local\Temp\ipykernel_15008\1791587065.py:1: SettingWithCopyWarnin
g:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace = True)
```

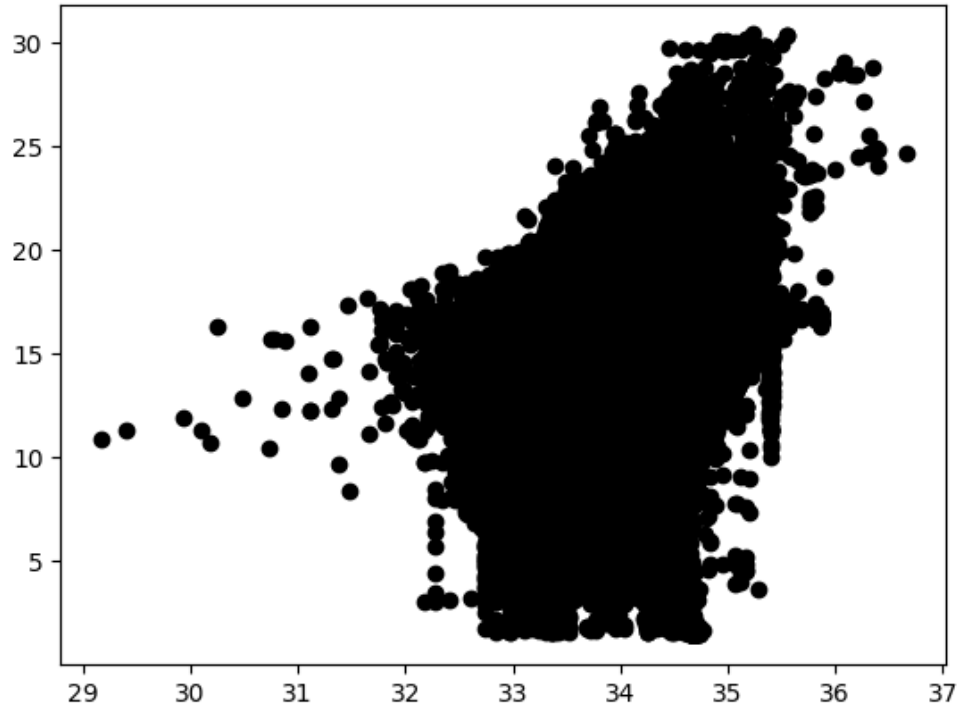
In [20]:

```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.25)
# Splitting the data into training data and test data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.20793879838956597
```

In [21]:

```
#step-6: Exploring Our Results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'k')
plt.show()
# Data scatter of predicted values
```

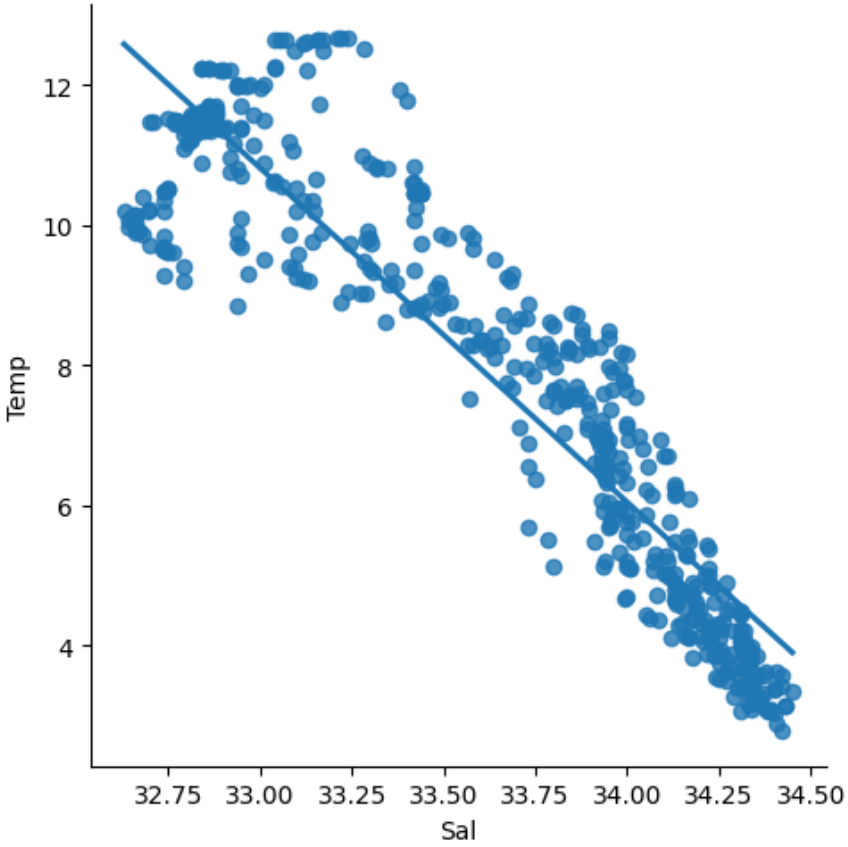


In [22]:

```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="Sal", y ="Temp", data = df500, order = 1, ci = None)
```

Out[22]:

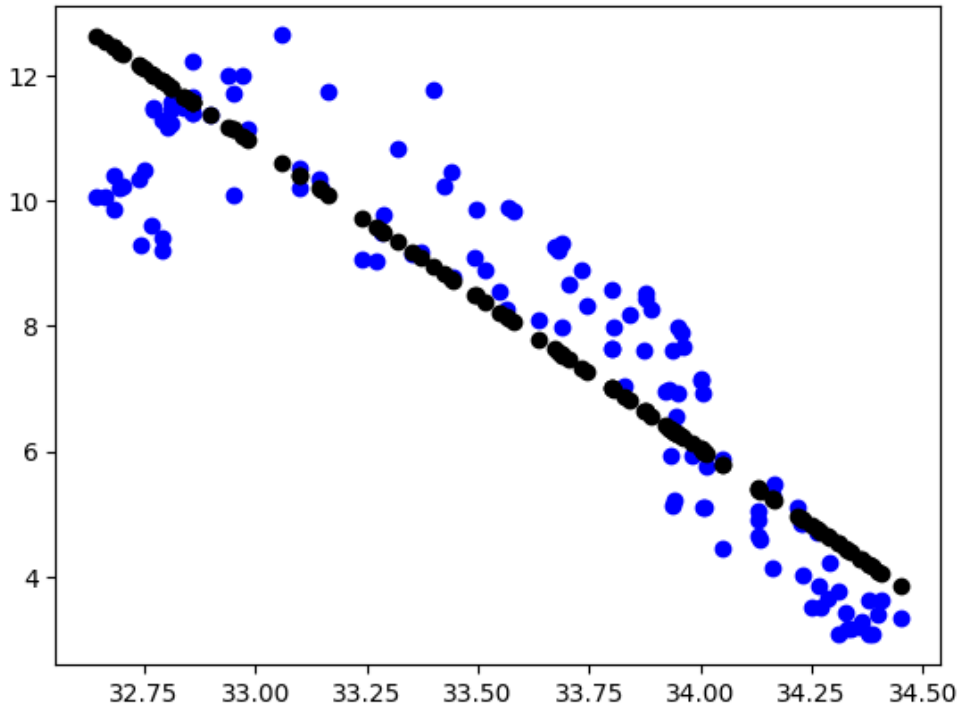
```
<seaborn.axisgrid.FacetGrid at 0x2cd17792e50>
```



In [23]:

```
df500.fillna(method = 'ffill', inplace = True)
X = np.array(df500['Sal']).reshape(-1, 1)
y = np.array(df500['Temp']).reshape(-1, 1)
df500.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.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: 0.8221495373138488
```



In [24]:

```
#Step-8: Evaluation of model

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

#Train the model

model = LinearRegression()

model.fit(X_train, y_train)

#Evaluating the model on the test set

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)

print("R2 score:",r2)
```

```
R2 score: 0.8221495373138488
```

In [46]:

```
#step 9-conclusion:
#Data set we have taken is poor for linear model but with the smaller data works well with linear model
```

In [31]:

```
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 [32]:

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

Out[32]:

	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
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [33]:

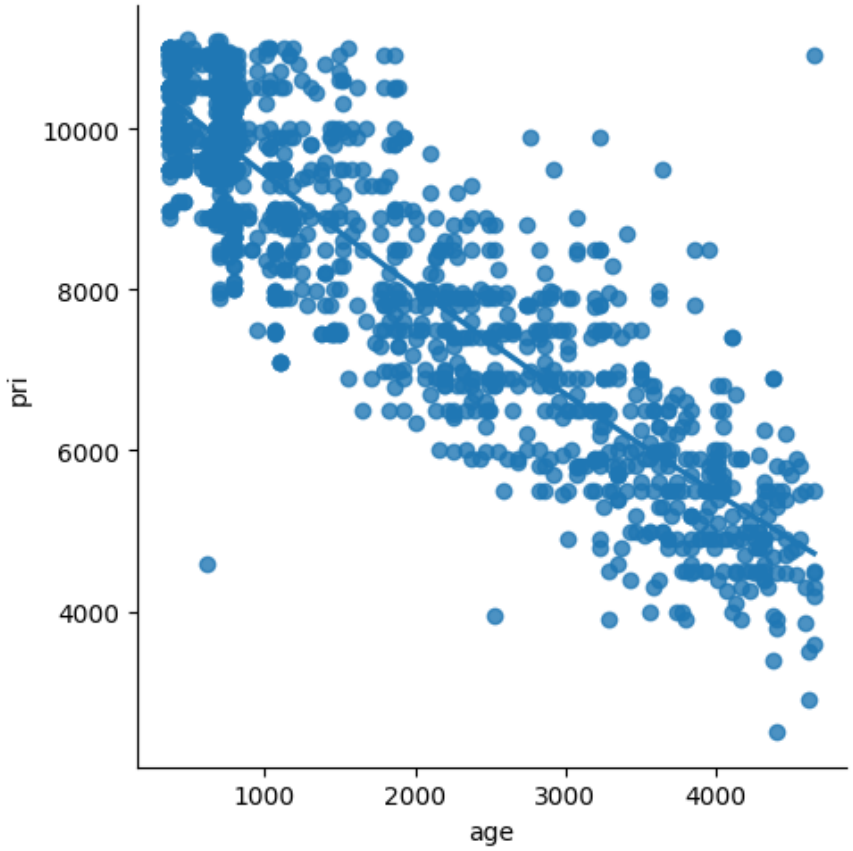
```
df=df[['age_in_days','price']]
df.columns=['age','pri']
```

In [34]:

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

Out[34]:

```
<seaborn.axisgrid.FacetGrid at 0x2cd1a934e50>
```



In [35]:

```
df.head(10)
```

Out[35]:

	age	pri
0	882	8900
1	1186	8800
2	4658	4200
3	2739	6000
4	3074	5700
5	3623	7900
6	731	10750
7	1521	9190
8	4049	5600
9	3653	6000

In [36]:

```
df.describe()
```

Out[36]:

	age	pri
count	1538.000000	1538.000000
mean	1650.980494	8576.003901
std	1289.522278	1939.958641
min	366.000000	2500.000000
25%	670.000000	7122.500000
50%	1035.000000	9000.000000
75%	2616.000000	10000.000000
max	4658.000000	11100.000000

In [37]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     1538 non-null   int64
 1   pri     1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [38]:

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

```
C:\Users\mural\AppData\Local\Temp\ipykernel_15008\48824337.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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method ='ffill', inplace = True)
```

In [39]:

```
# Step-5: Training Our Model
X = np.array(df['age']).reshape(-1, 1)
y = np.array(df['pri']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

In [40]:

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

```
C:\Users\mural\AppData\Local\Temp\ipykernel_15008\1791587065.py:1: SettingWithCopyWarnin
g:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace = True)
```

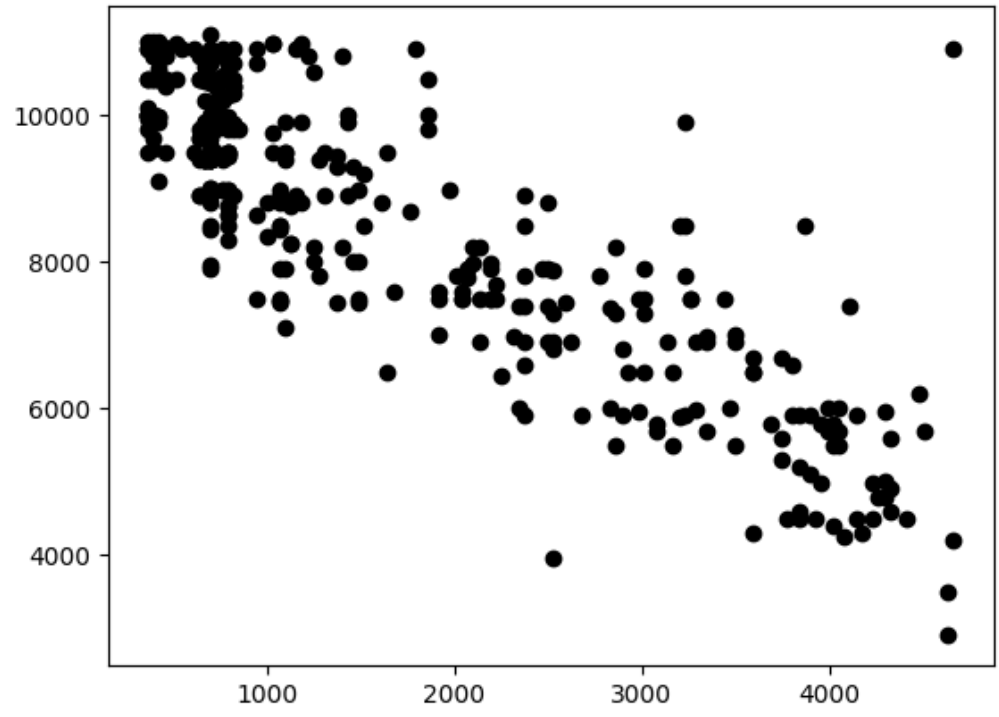
In [41]:

```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.25)
# Splitting the data into training data and test data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.7583527651957717
```

In [42]:

```
#step-6: Exploring Our Results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'k')
plt.show()
# Data scatter of predicted values
```

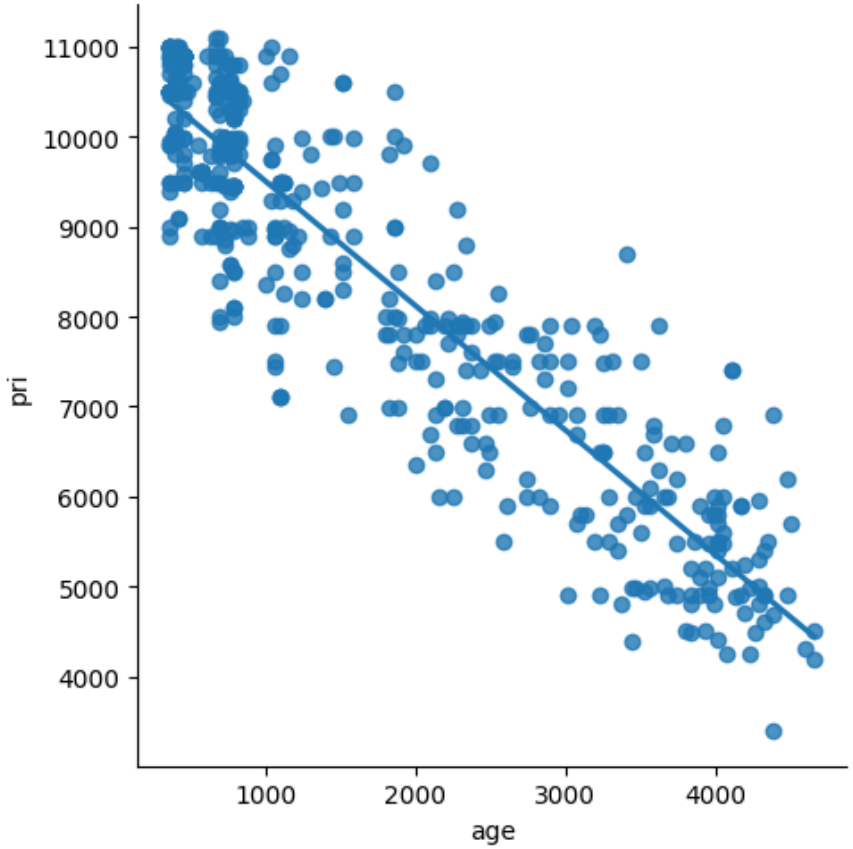


In [43]:

```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="age", y ="pri", data = df500, order = 1, ci = None)
```

Out[43]:

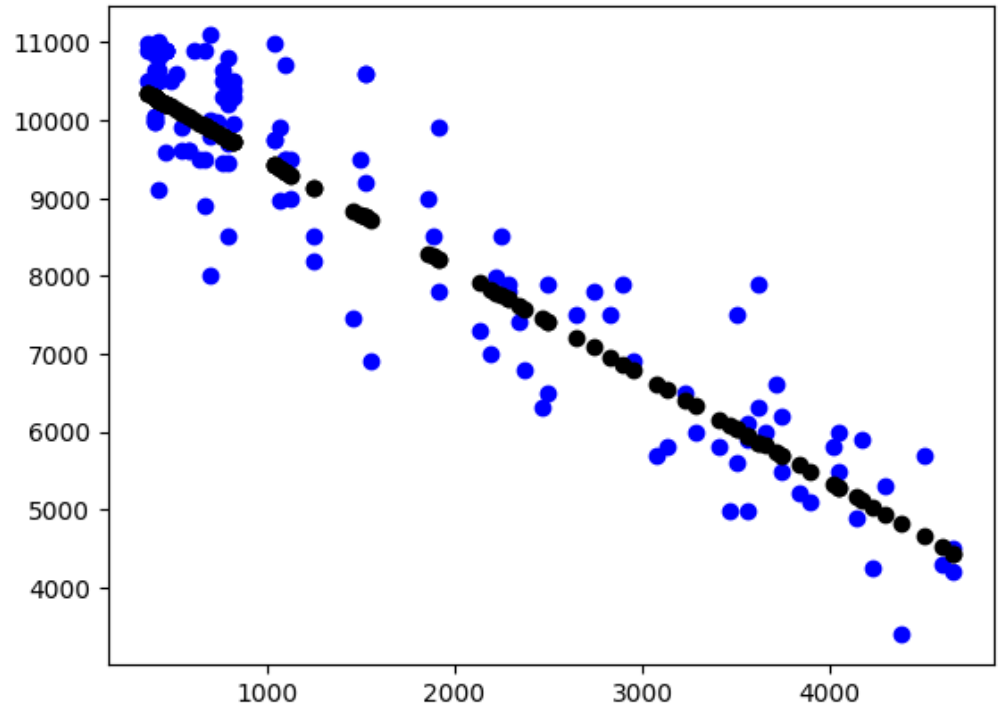
```
<seaborn.axisgrid.FacetGrid at 0x2cd5ea46550>
```



In [44]:

```
df500.fillna(method = 'ffill', inplace = True)
X = np.array(df500['age']).reshape(-1, 1)
y = np.array(df500['pri']).reshape(-1, 1)
df500.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.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: 0.8696812646983724
```



In [48]:

```
#Step-8: Evaluation of model

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

#Train the model

model = LinearRegression()

model.fit(X_train, y_train)

#Evaluating the model on the test set

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)

print("R2 score:",r2)
```

```
R2 score: 0.8696812646983724
```

In [47]:

```
#step 9-conclusion:
#Data set we have taken is poor for linear model but with the smaller data works well with linear model
```

In [49]:

```
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 [51]:

```
df=pd.read_csv(r"C:\Users\mural\Downloads\Housing.csv")
df#step 9-conclusion:
#Data set we have taken is poor for linear model but with the smaller data works well with linear model
```

Out[51]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	aircon
0	13300000	7420	4	2	3	yes	no	no	no	
1	12250000	8960	4	4	4	yes	no	no	no	
2	12250000	9960	3	2	2	yes	no	yes	no	
3	12215000	7500	4	2	2	yes	no	yes	no	
4	11410000	7420	4	1	2	yes	yes	yes	no	
...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no	yes	no	
541	1767150	2400	3	1	1	no	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	no	
543	1750000	2910	3	1	1	no	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	no	

545 rows × 13 columns

In [52]:

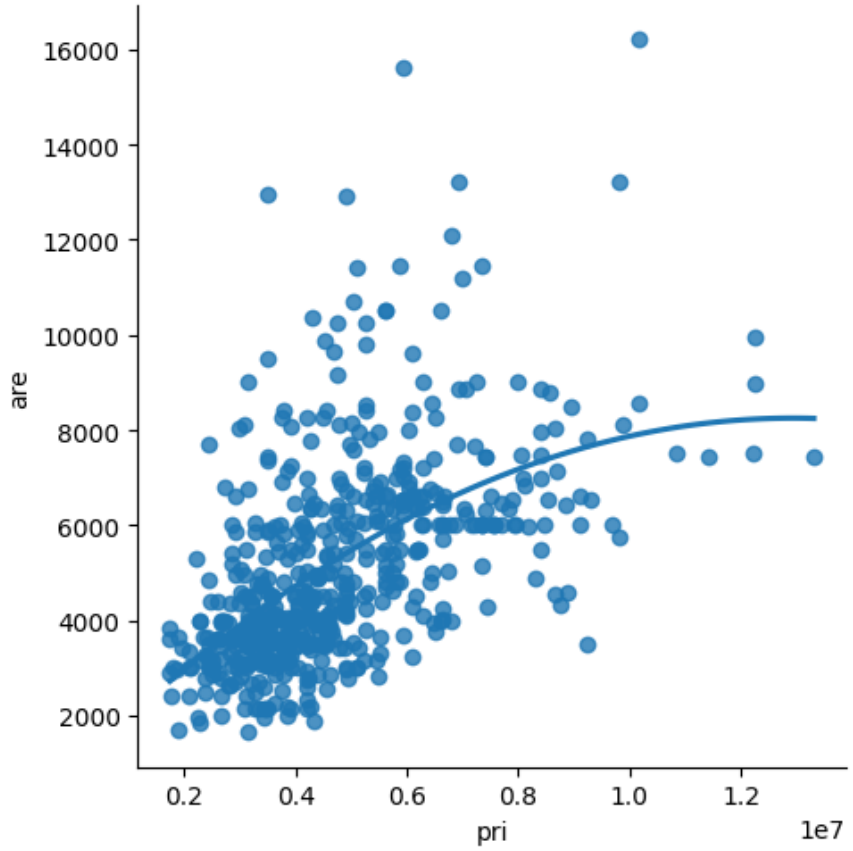
```
df=df[['price','area']]
df.columns=['pri','are']
```

In [53]:

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

Out[53]:

```
<seaborn.axisgrid.FacetGrid at 0x2cd41aea990>
```



In [ ]:

```
df.head(10)
```

In [ ]:

```
df.describe()
```

In [ ]:

```
df.info()
```

In [ ]:

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

In [54]:

```
# Step-5: Training Our Model
X = np.array(df['pri']).reshape(-1, 1)
y = np.array(df['are']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

In [55]:

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

```
C:\Users\mural\AppData\Local\Temp\ipykernel_15008\1791587065.py:1: SettingWithCopyWarnin
g:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace = True)
```

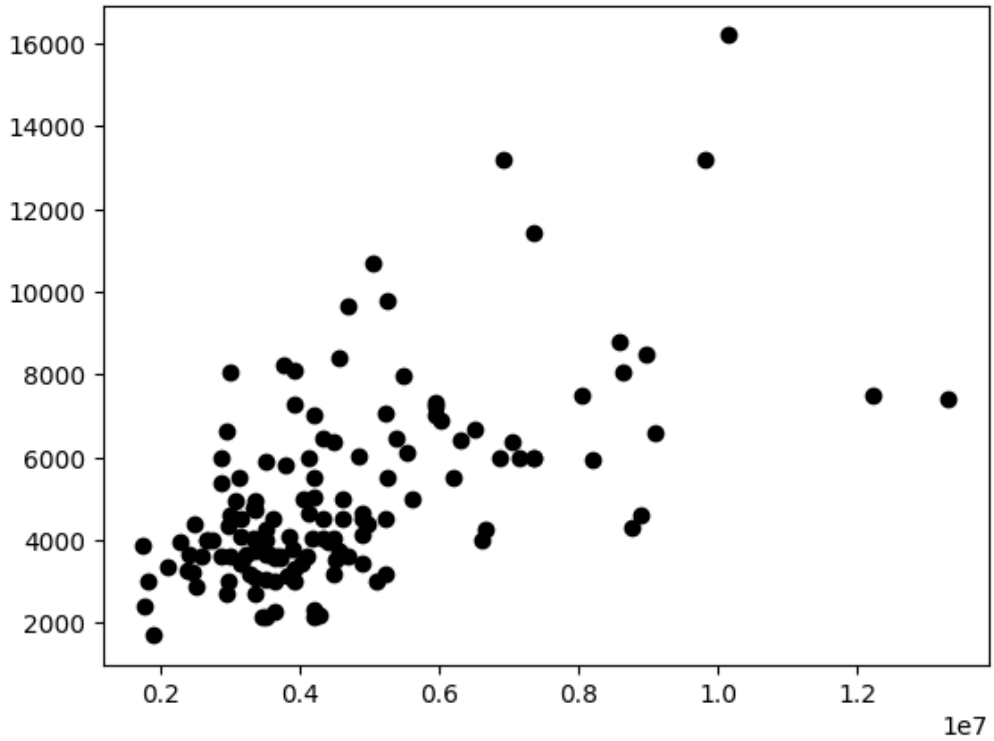
In [56]:

```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.25)
# Splitting the data into training data and test data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.34759413618913637
```

In [57]:

```
#step-6: Exploring Our Results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'k')
plt.show()
# Data scatter of predicted values
```

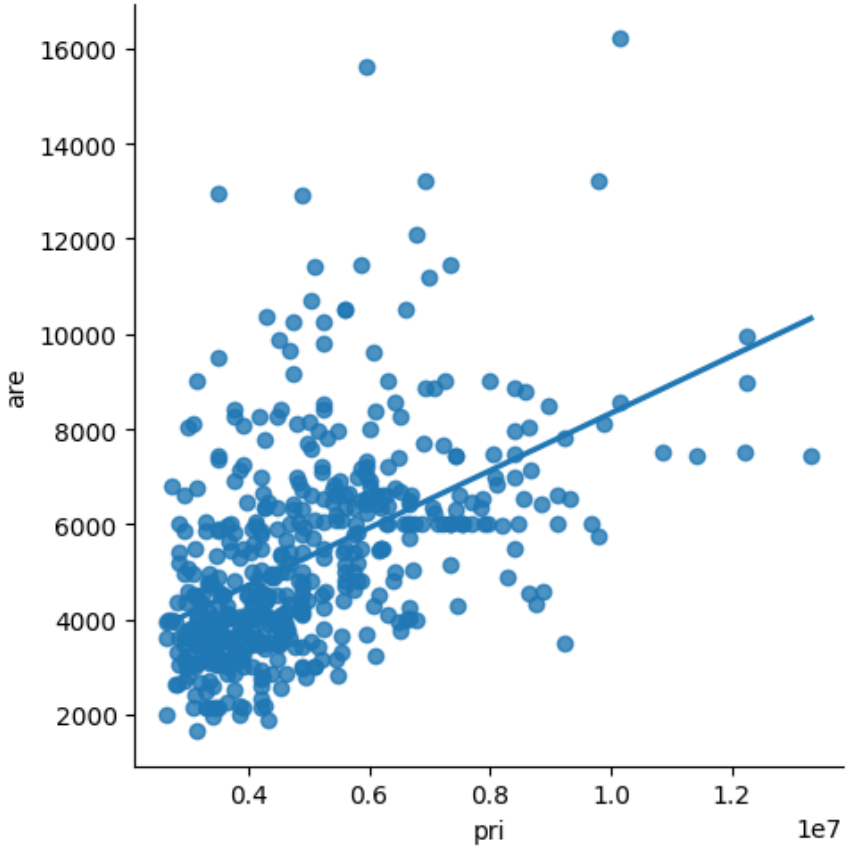


In [59]:

```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="pri", y ="are", data = df500, order = 1, ci = None)
```

Out[59]:

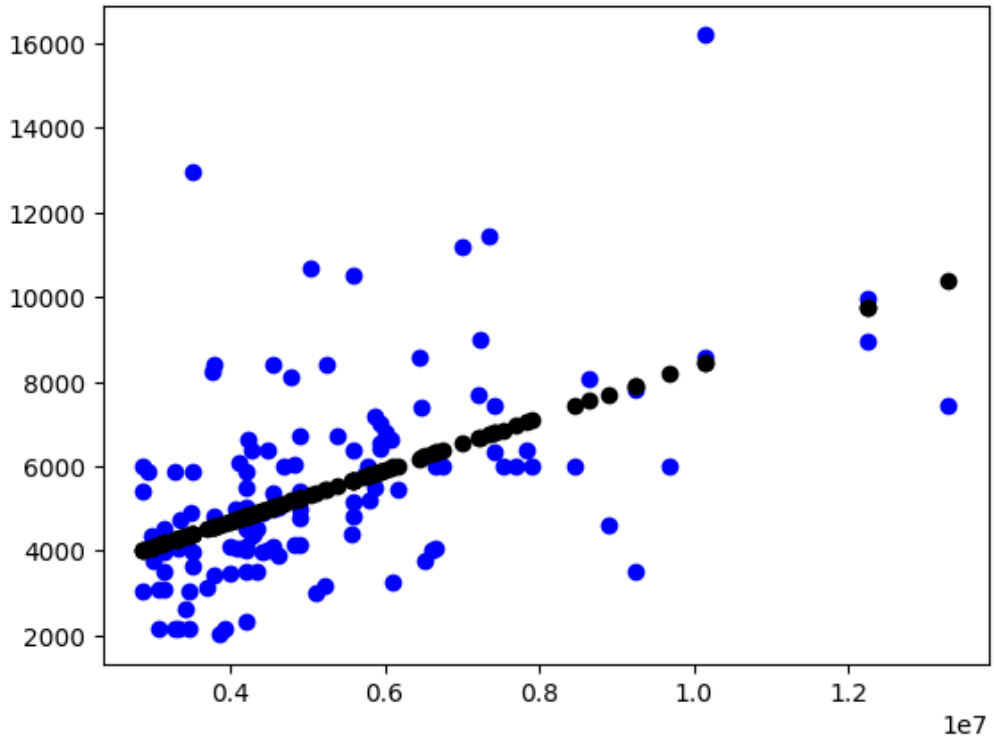
```
<seaborn.axisgrid.FacetGrid at 0x2cd41afac50>
```



In [60]:

```
df500.fillna(method = 'ffill', inplace = True)
X = np.array(df500['pri']).reshape(-1, 1)
y = np.array(df500['are']).reshape(-1, 1)
df500.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.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: 0.25833575748396476
```



In [61]:

```
#Step-8: Evaluation of model

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

#Train the model

model = LinearRegression()

model.fit(X_train, y_train)

#Evaluating the model on the test set

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)

print("R2 score:",r2)
```

```
R2 score: 0.25833575748396476
```

In [62]:

```
#step 9-conclusion:
#Data set we have taken is poor for linear model but with the smaller data works well with linear model
```

In [ ]: