In [ ]:

In [ ]:

Linear Regression- USA_Housing

In [24]:
```
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 [25]:
```
df=pd.read_csv(r"C:\Users\jyothi reddy\Downloads\USA_Housing.csv")
df#step 9-conclusion:
#Data set we have taken is poor for linear model but with the smaller data
```

Out[25]:

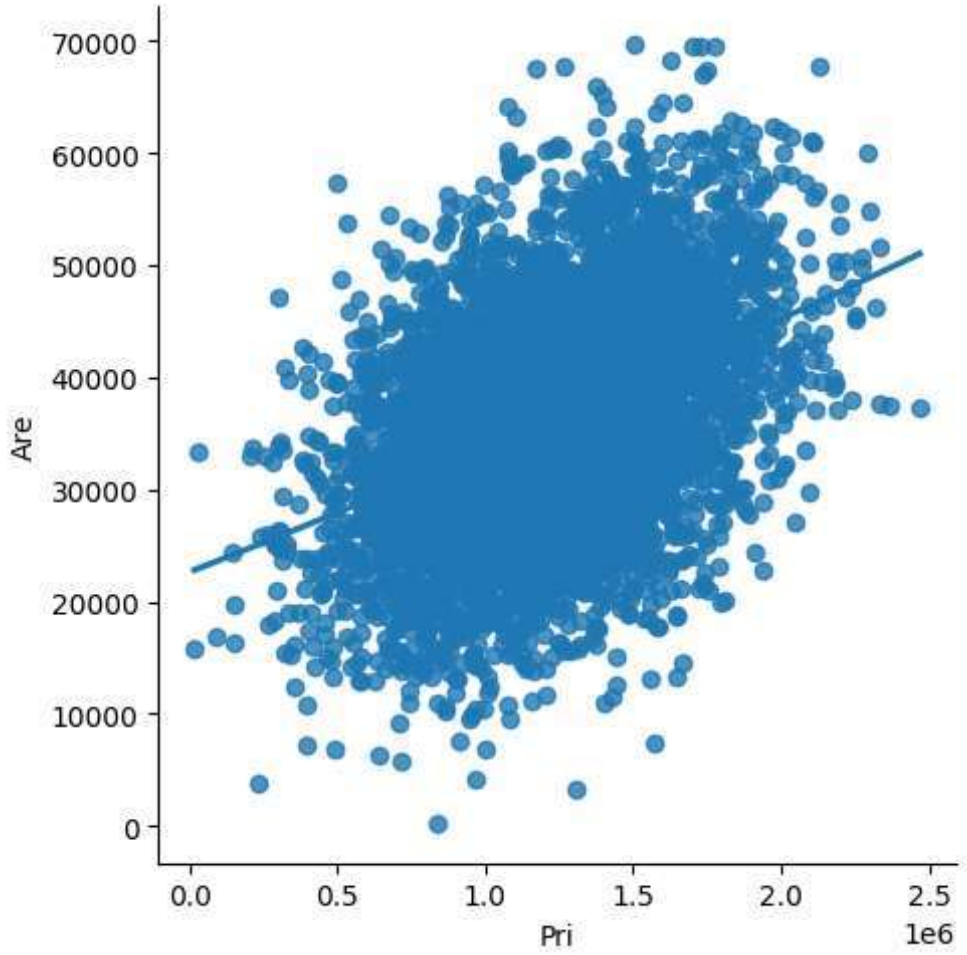
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Micha 674\nL
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Jo Suit Ka
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	91 Stravenue\
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barn
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Ray
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS W AP
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PS 8489\nAP
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 T Suite 076\r
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Walla
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 Ge Apt. 509

5000 rows × 7 columns

In [26]:
```
df=df[['Price','Area Population']]
df.columns=['Pri','Are']
```

In [27]:
```
sns.lmplot(x="Pri",y="Are", data = df, order = 2, ci = None)
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x22c0a0bd8d0>



In [28]:
```
df.head(10)
```

Out[28]:

	Pri	Are
0	1.059034e+06	23086.800503
1	1.505891e+06	40173.072174
2	1.058988e+06	36882.159400
3	1.260617e+06	34310.242831
4	6.309435e+05	26354.109472
5	1.068138e+06	26748.428425
6	1.502056e+06	60828.249085
7	1.573937e+06	36516.358972
8	7.988695e+05	29387.396003
9	1.545155e+06	40149.965749

In [29]:
```
df.describe()
```

Out[29]:

	Pri	Are
count	5.000000e+03	5000.000000
mean	1.232073e+06	36163.516039
std	3.531176e+05	9925.650114
min	1.593866e+04	172.610686
25%	9.975771e+05	29403.928702
50%	1.232669e+06	36199.406689
75%	1.471210e+06	42861.290769
max	2.469066e+06	69621.713378

In [30]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Pri     5000 non-null   float64
 1   Are     5000 non-null   float64
dtypes: float64(2)
memory usage: 78.3 KB
```

In [31]:
```
df.fillna(method ='ffill', inplace = True)
```

```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_11344\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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df.fillna(method ='ffill', inplace = True)
```

In [32]:
```
# 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 [33]:
```
df.dropna(inplace = True)
```

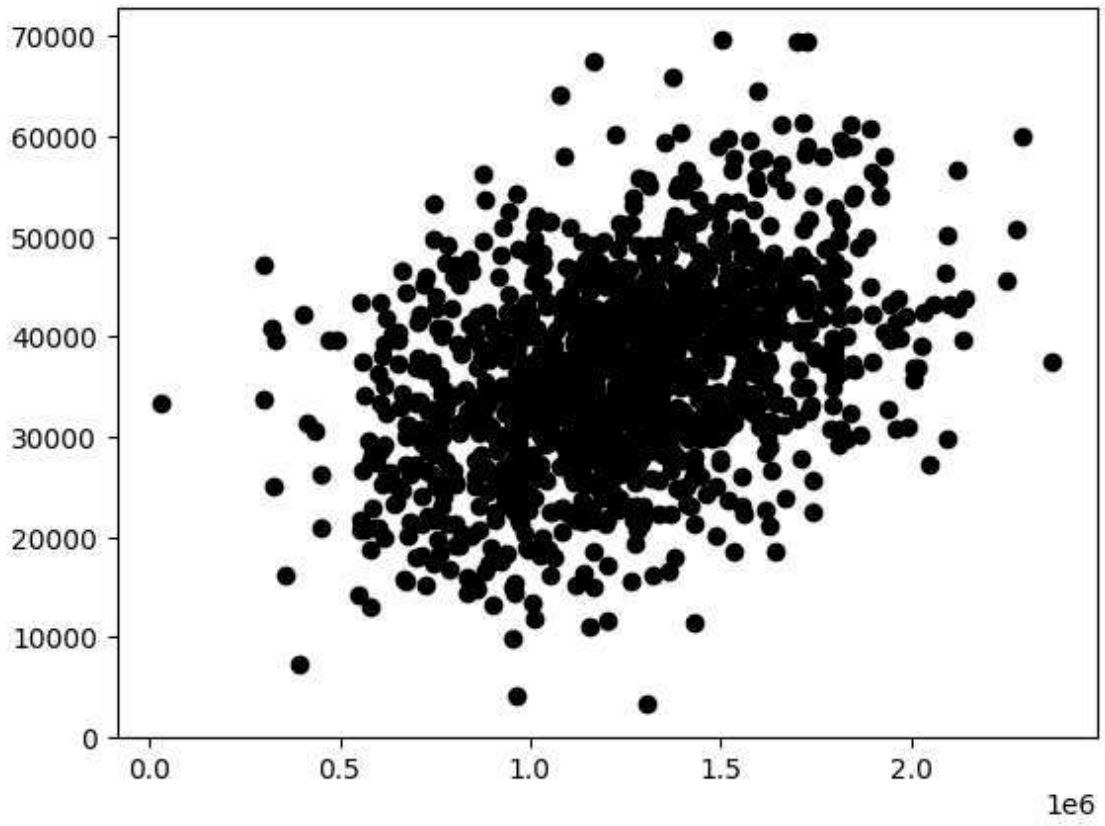
```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_11344\1791587065.py:
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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

In [34]:
```
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))
```

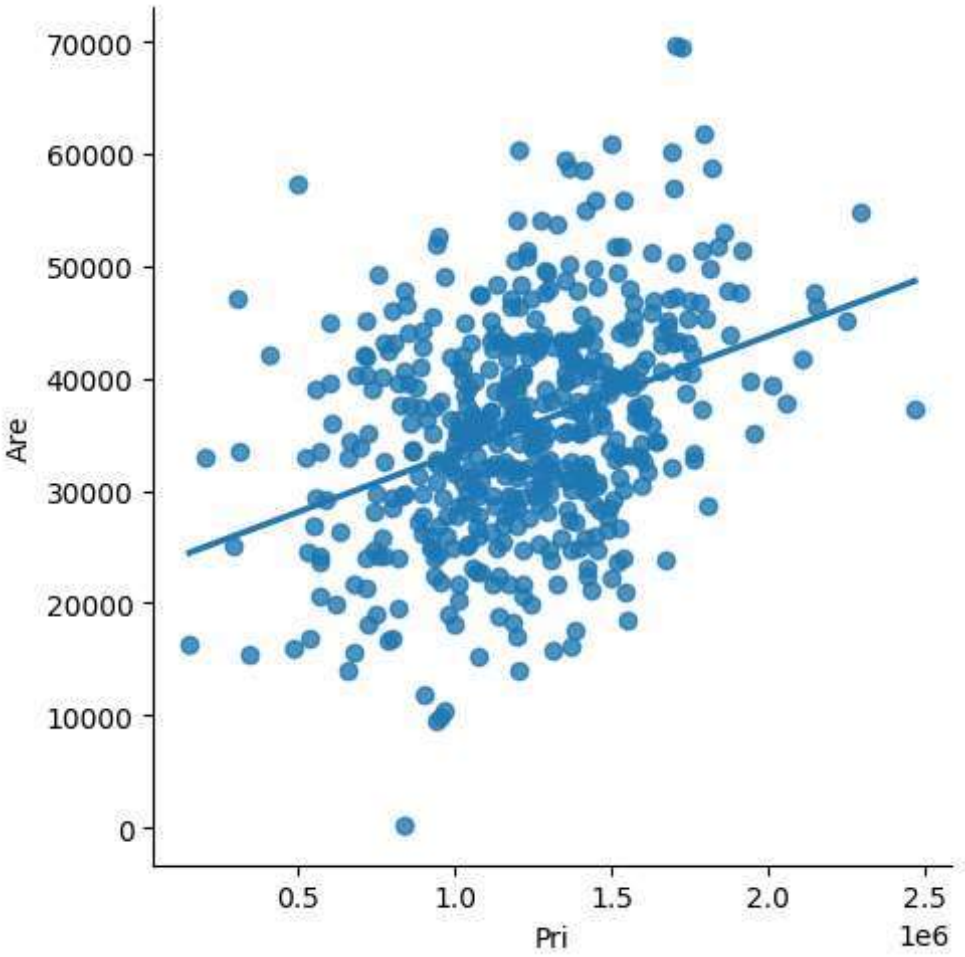
```
0.15220040967892345
```

In [35]:
```
#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
```



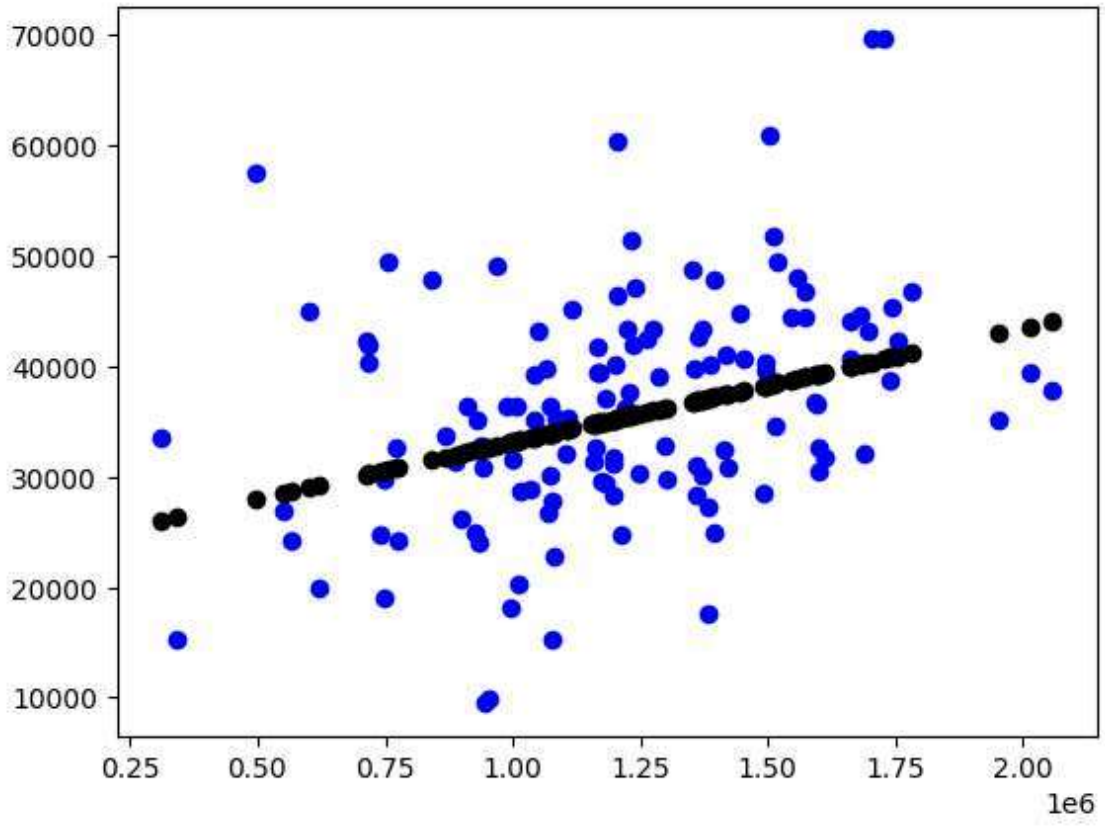
In [36]:
```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="Pri", y ="Are", data = df500, order = 1, ci = None)
```

Out[36]: <seaborn.axisgrid.FacetGrid at 0x22c09ee9190>



In [37]:
```
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: 0.12069840400358822
```



In [38]:
```
#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: 0.12069840400358822
```

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

In [ ]: