In [1]:

```
import pandas as pd
import numpy as np
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 [2]:

```
df=pd.read_csv(r"C:\Users\Arshad Shaik\Downloads\Advertising.csv")
df
```

Out[2]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [3]:

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

Out[3]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6

In [4]:

```
df.describe()
```

Out[4]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [6]:

```
df=df[['TV','Radio']]
df.columns=['TV','Radio']
df.head(10)
```

Out[6]:

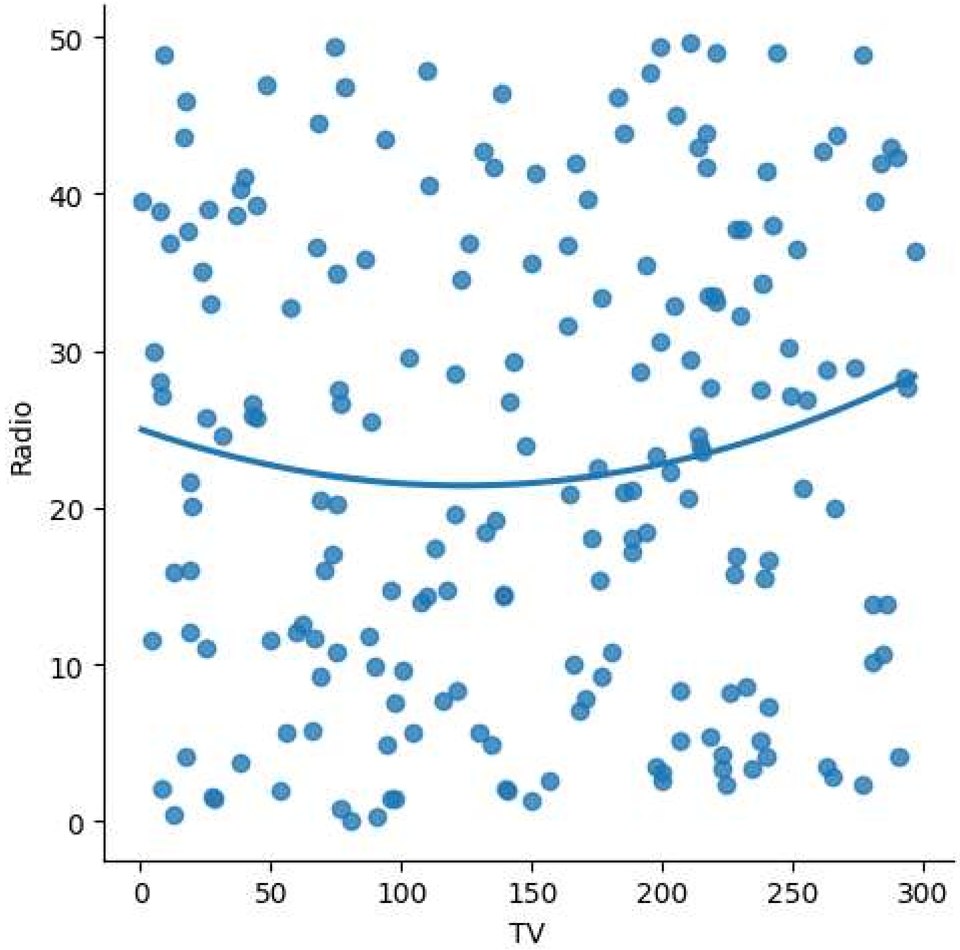
	TV	Radio
0	230.1	37.8
1	44.5	39.3
2	17.2	45.9
3	151.5	41.3
4	180.8	10.8
5	8.7	48.9
6	57.5	32.8
7	120.2	19.6
8	8.6	2.1
9	199.8	2.6

In [7]:

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

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x2178295c610>
```



In [8]:

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

```
C:\Users\Arshad Shaik\AppData\Local\Temp\ipykernel_2652\1844562654.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.fillna(method="ffill",inplace=True)
```

In [9]:

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

In [10]:

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

```
C:\Users\Arshad Shaik\AppData\Local\Temp\ipykernel_2652\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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

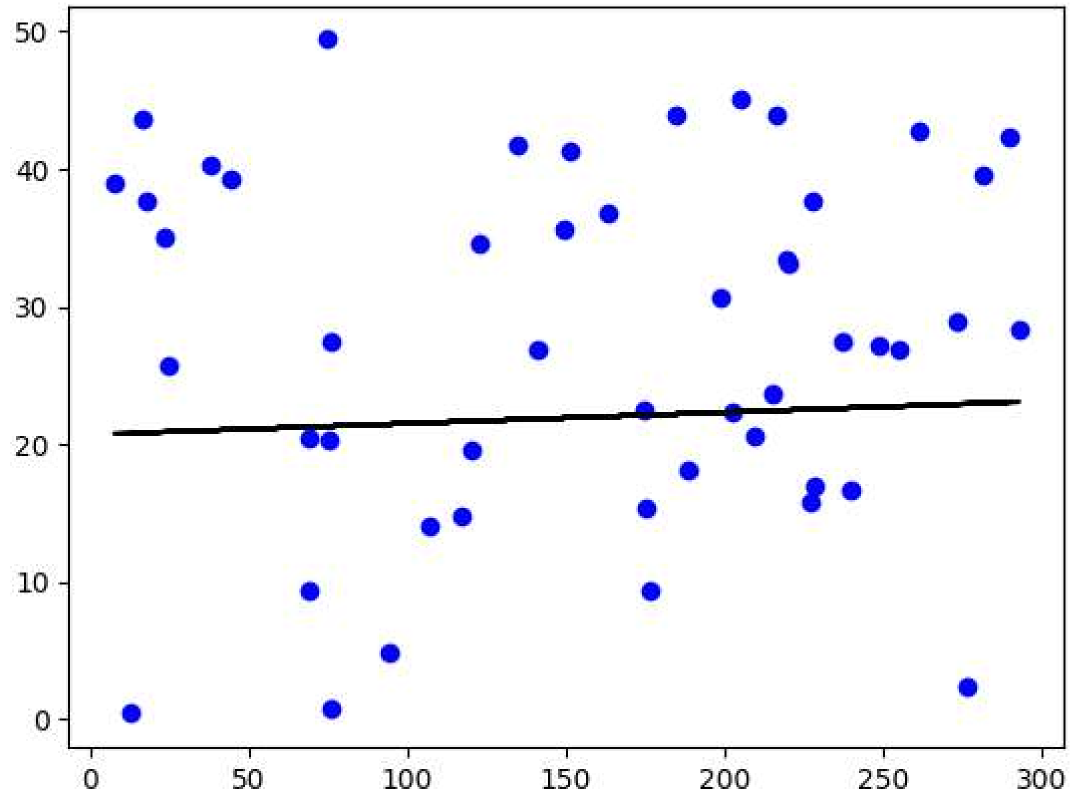
In [11]:

```
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(regr.score(x_test,y_test))
```

```
-0.18653611310505425
```

In [12]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

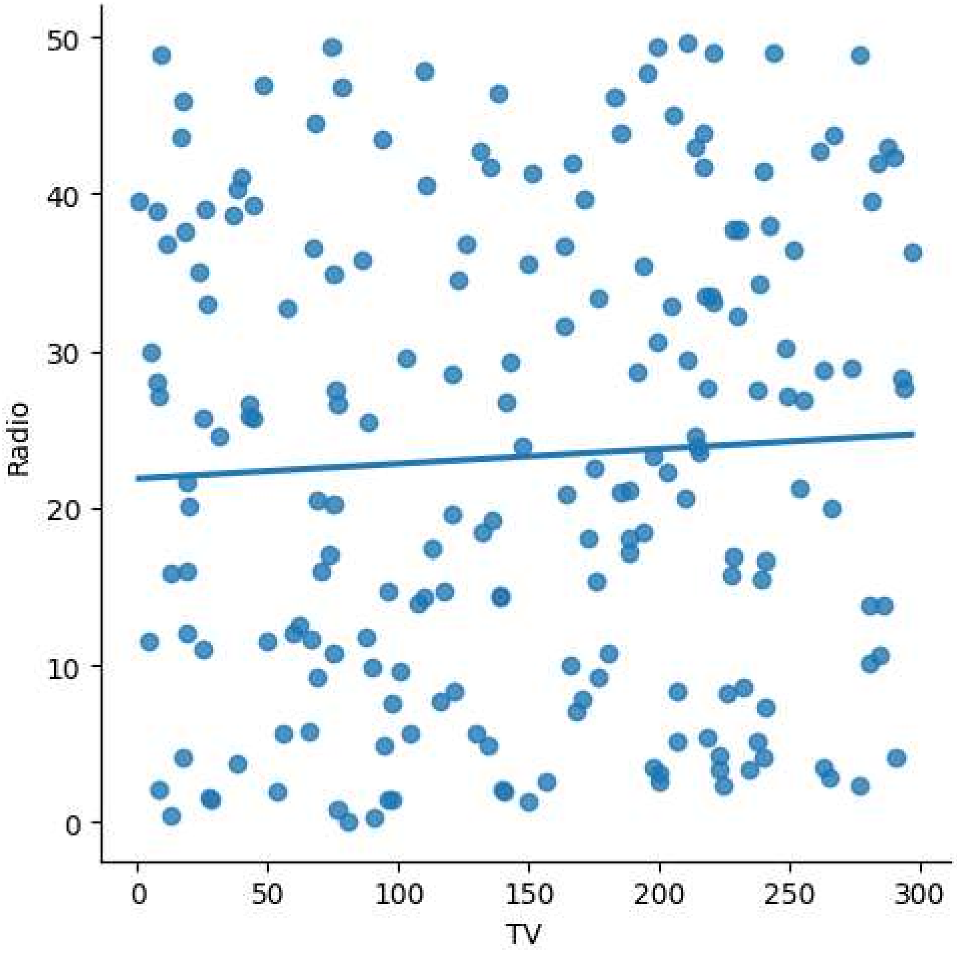


In [13]:

```
df500=df[:][:500]
sns.lmplot(x="TV",y="Radio",data=df500,order=1,ci=None)
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x21782a93d60>
```



In [14]:

```
df500.fillna(method='ffill',inplace=True)
```

In [15]:

```
x=np.array(df500['TV']).reshape(-1,1)
y=np.array(df500['Radio']).reshape(-1,1)
```

In [16]:

```
df500.dropna(inplace=True)
```

In [17]:

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

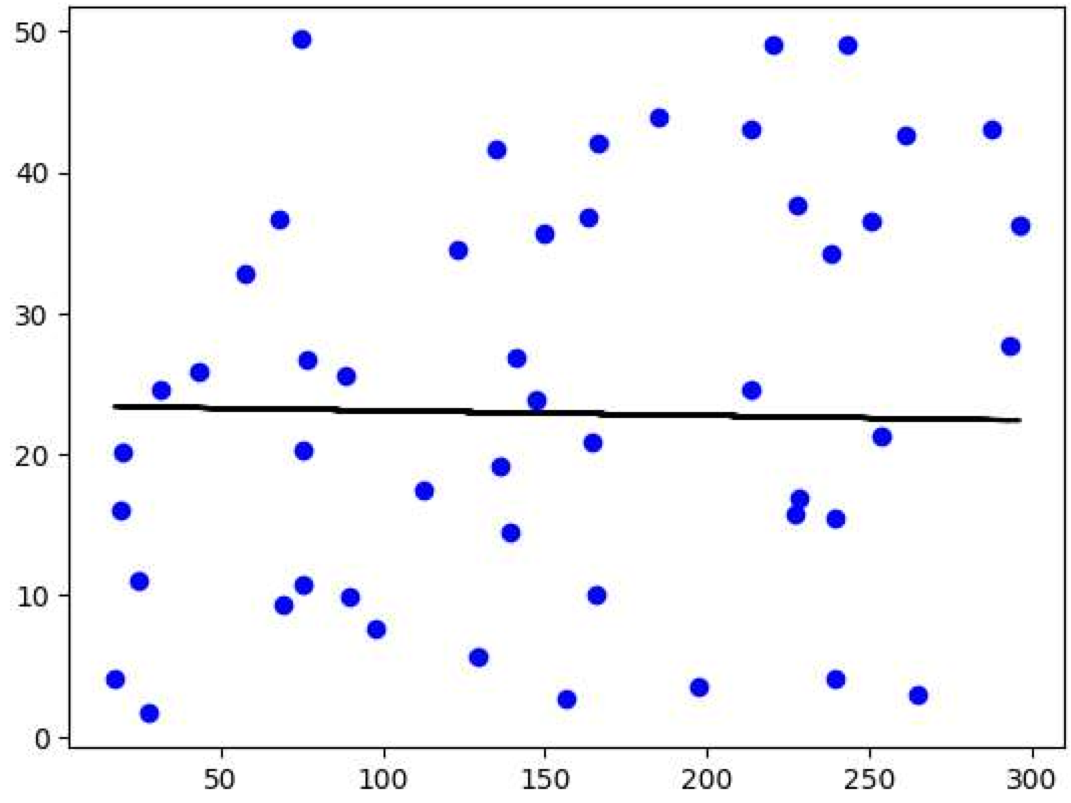
In [18]:

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

```
Regression: -0.021405231321753204
```

In [19]:

```
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 [20]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[20]:

```
▾ LinearRegression
LinearRegression()
```

In [21]:

```
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: -0.021405231321753204
```

In [22]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.00940927]
[21.88043754]
```

In [23]:

```
y_pred_elastic=regr.predict(x_train)
```

In [24]:

```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean squared Error on test set",mean_squared_error)
```

```
Mean squared Error on test set 226.69111670084166
```

In [ ]: