Duvall Pinkney week 5 homework

This homework assignment aims to build a simple linear regression model using the `radio` feature.
(Source of data: https://www.statlearning.com/s/Advertising.csv)

1. Apply the normal equation to calculate parameter values for the best fit.

2. Display the regression line with the training data points.

3. Use `sklearn` to build the same model. Verify that the parameters values are the same as those from the normal equation.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
url = "https://www.statlearning.com/s/Advertising.csv"
dataframe = pd.read_csv(url)
dataframe.head()
```

Out[2]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [3]:
```
dataframe.describe()
```

Out[3]:

	Unnamed: 0	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

In [4]:
```
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
Unnamed: 0    200 non-null int64
TV            200 non-null float64
radio         200 non-null float64
newspaper     200 non-null float64
sales         200 non-null float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

In [5]:
```
dataframe.corr()
```

Out[5]:

	Unnamed: 0	TV	radio	newspaper	sales
Unnamed: 0	1.000000	0.017715	-0.110680	-0.154944	-0.051616
TV	0.017715	1.000000	0.054809	0.056648	0.782224
radio	-0.110680	0.054809	1.000000	0.354104	0.576223
newspaper	-0.154944	0.056648	0.354104	1.000000	0.228299
sales	-0.051616	0.782224	0.576223	0.228299	1.000000

In [7]:
```
# Train a linear regression model using sklearn
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression()
model_lr.fit(dataframe[['TV']], dataframe[['sales']])
```

Out[7]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

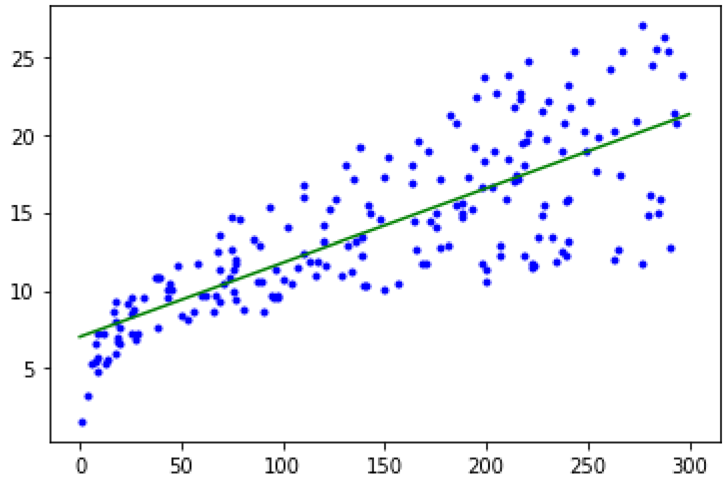
In [8]:
```
# The coef_ and intercept_ attributes contain parameter values
print(model_lr.coef_)
print(model_lr.intercept_)
```

```
[[0.04753664]]
[7.03259355]
```

In [10]:
```
# Plot the data points and the optimal regression line.
m = model_lr.coef_[0, 0]   # slope
b = model_lr.intercept_[0] # y-intercept

plt.plot(dataframe['TV'], dataframe['sales'], 'b.')
x_coordinates = np.array([0,300])
y_coordinates = x_coordinates * m + b
plt.plot(x_coordinates, y_coordinates, 'g-')
```

Out[10]: [<matplotlib.lines.Line2D at 0x12c58379f08>]



In [11]:
```python
# Write a function that produces the squared error given beta0, beta1, data, and i
def get_squared_error(beta0, beta1, dataframe, i):
    """
    This function returns the squared error on Record i.
    """

    x = dataframe.loc[i, 'TV']
    y = dataframe.loc[i, 'sales']

    prediction = beta0 + beta1 * x

    squared_error = (y - prediction) ** 2

    return squared_error
```

In [13]:
```python
# Example:
# Calculate the squared error of the model on the first record.

beta0 = 7.03
beta1 = 0.04

x1 = dataframe.loc[1, 'TV'] # 230.1
y1 = dataframe.loc[1, 'sales'] #22.1
print("x1, y1:", x1, y1)

# Calculate f(x1) = beta0 + beta1 * x1
prediction1 = beta0 + beta1 * x1
print("Prediction on Record 1:", prediction1)

# Calculate the squared error (y1 - f(x1)) ** 2
error1 = (y1 - prediction1) ** 2
print("Squared error on Record 1:", error1)
```

```
x1, y1: 44.5 10.4
Prediction on Record 1: 8.81
Squared error on Record 1: 2.5280999999999993
```

In [14]:
```
# Example:
# Calculate the squared error of the model on an arbitrary record.

beta0 = 7.03
beta1 = 0.04

i = 123  # index of the record
xi = dataframe.loc[i, 'TV']
yi = dataframe.loc[i, 'sales']
print("xi, yi:", xi, yi)

# Calculate f(xi)
predictioni = beta0 + beta1 * xi

# Calculate the squared error (yi - f(xi)) ** 2
errori = (yi - predictioni) ** 2

print("Squared error:", errori)
```

```
xi, yi: 123.1 15.2
Squared error: 10.536515999999992
```

In [16]:
```
get_squared_error(beta0, beta1, dataframe, i)
```

Out[16]:
```
10.536515999999992
```

```
X = np.array([0,300])
```

In [25]:
```
X = np.hstack([np.ones([200, 0]),dataframe[["TV"]].values])
print(X)
```

```
[[230.1]
 [ 44.5]
 [ 17.2]
 [151.5]
 [180.8]
 [  8.7]
 [ 57.5]
 [120.2]
 [  8.6]
 [199.8]
 [ 66.1]
 [214.7]
 [ 23.8]
 [ 97.5]
 [204.1]
 [195.4]
 [ 67.8]
 [281.4]
 [ 69.2]
```

In [26]:
```
y = dataframe[["sales"]].values
print(y)
```

```
 [27. ]
 [20.2]
 [11.7]
 [11.8]
 [12.6]
 [10.5]
 [12.2]
 [ 8.7]
 [26.2]
 [17.6]
 [22.6]
 [10.3]
 [17.3]
 [15.9]
 [ 6.7]
 [10.8]
 [ 9.9]
 [ 5.9]
 [19.6]
```

In [28]:
```
theta = np.linalg.inv(X.T.dot(X)).dot(X.T)
print("Theta: ", theta)
```

```
Theta:  [[3.97332578e-05 7.68418067e-06 2.97006534e-06 2.61607499e-05
  3.12202217e-05 1.50230049e-06 9.92899750e-06 2.07559217e-05
  1.48503267e-06 3.45011078e-05 1.14140302e-05 3.70740133e-05
  4.10974157e-06 1.68361262e-05 3.52436242e-05 3.37413237e-05
  1.17075831e-05 4.85916504e-05 1.19493326e-05 2.54355014e-05
  3.77129227e-05 4.09938088e-05 2.27935247e-06 3.94224370e-05
  1.07578529e-05 4.53971033e-05 2.46757173e-05 4.14600400e-05
  4.29623405e-05 1.21910821e-05 5.05774499e-05 1.94953707e-05
  1.67843227e-05 4.58633345e-05 1.65253054e-05 5.01975578e-05
  4.60878162e-05 1.28990628e-05 7.44243117e-06 3.93706336e-05
  3.49673390e-05 3.05640445e-05 5.06983246e-05 3.57271232e-05
  4.33422326e-06 3.02359559e-05 1.54892361e-05 4.14255043e-05
  3.92324910e-05 1.15521727e-05 3.45011078e-05 1.73368930e-05
  3.73675662e-05 3.15310425e-05 4.53625677e-05 3.43456974e-05
  1.26055099e-06 2.35187732e-05 3.64005682e-05 3.63833004e-05
  9.23828463e-06 4.51208182e-05 4.13218974e-05 1.77340529e-05
  2.26381143e-05 1.19147970e-05 5.43936385e-06 2.40540757e-05
  4.09938088e-05 3.74366375e-05 3.43802331e-05 1.89600683e-05
  4.62777622e-06 2.23445613e-05 3.68495316e-05 2.91826187e-06
  4.74865098e-06 2.08077252e-05 9.32462374e-07 2.00306732e-05
  1.31926158e-05 4.14082365e-05 1.30026698e-05 1.18111901e-05
  3.68667994e-05 3.33614316e-05 1.31753480e-05 1.91154787e-05
  1.52474866e-05 1.89600683e-05 2.31906846e-05 4.93859702e-06
  3.75920479e-05 4.33249647e-05 1.85456405e-05 2.81983529e-05
  3.41212157e-05 3.19282024e-05 5.00248796e-05 2.33460950e-05
  3.84036355e-05 5.11818236e-05 4.83844365e-05 3.24462370e-05
  4.11319514e-05 2.38123262e-05 4.31695543e-06 1.56101108e-05
  2.26208465e-06 4.41020167e-05 3.89907415e-05 4.17363251e-05
  3.03395628e-05 3.61933544e-05 1.35034366e-05 1.29681341e-05
  2.40368079e-05 1.31926158e-05 2.17056519e-05 3.34995742e-06
  2.43994321e-05 3.24635049e-06 3.86799207e-05 2.12566886e-05
  3.96296509e-05 1.50575406e-05 1.34689010e-06 1.38487930e-05
  3.80410113e-05 1.02916218e-05 1.20874752e-07 4.57942632e-05
  1.45049703e-06 3.79546722e-05 6.37182622e-06 8.34035790e-06
  4.42056236e-06 4.72620281e-05 7.42516335e-06 3.19282024e-05
  1.26745812e-05 3.34477707e-05 3.80755469e-05 1.80621415e-05
  1.66116445e-05 2.42267539e-05 4.14600400e-05 4.19953425e-05
  6.56177226e-06 7.71871632e-06 4.84707756e-05 2.08940643e-05
  3.41212157e-05 2.95797786e-05 3.24289692e-05 7.07980691e-07
```

```
1.62144846e-05 2.58671970e-05 2.02033514e-06 2.27417212e-05
2.97869925e-05 1.47985232e-05 3.25325761e-05 2.82328885e-05
2.02378871e-05 4.04930420e-05 3.09094009e-06 3.57098553e-05
3.71948880e-05 4.90924172e-05 8.63391087e-06 2.84055668e-05
3.38449306e-06 2.90790118e-05 3.84036355e-05 4.78145984e-05
4.28932692e-05 2.93898326e-05 4.77800627e-05 2.85955128e-05
2.70414088e-05 3.77301905e-05 9.70451581e-06 4.96622553e-05
4.38257316e-05 3.53990346e-05 2.40886113e-05 3.29988073e-05
4.93859702e-05 3.22908266e-06 6.82078958e-06 1.30372054e-05
2.97006534e-06 2.88027267e-05 2.58499291e-05 6.59630790e-06
1.62662881e-05 3.05640445e-05 4.89715424e-05 4.00786142e-05]]
```

In [ ]: