hw2

September 7, 2023

1 Layth Aljorani

1.1 Setup

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn.linear_model import LinearRegression
```

[4]:
```
df = pd.read_csv("Lab1/Advertising.csv")
```

[5]:
```
X = df[['TV', 'radio', 'newspaper']] # extract relevant features
print(X) # print dataframe
X = np.c_[X, np.ones(len(X.index))] # convert to numpy matrix + the last␣
↪columns being 1s to account for 'intercept'
```

```
        TV  radio  newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     …     …        …
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7

[200 rows x 3 columns]
```

[6]:
```
Y = df[['sales']] # extract sales
print(Y) # print dataframe
Y = np.c_[Y] # convert to numpy matrix
```

```
     sales
0     22.1
```

```
1      10.4
2       9.3
3      18.5
4      12.9
..      …
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4

[200 rows x 1 columns]
```

1.2 Problem 1

1. Solve the Normal Equation to find the model parameters theta_i (for theta <= i <= N) sale = theta_0 + theta_1(TV) + theta_2(radio) + theta_3(newspaper)

I will first rewrite the equation: Y_i = (X_i)^T * B + e_i

Where: - B, "beta" is a matrix of the coefficents - X_i, is a feature vector of the ith observation - Y_i, the results vector

Normal equation (we want to solve for B):

> (X^T * X) * B = X^T * Y
>
> B = (X^T * X)^-1 * (X^T * Y)

[12]:
```python
# np.linlag.inv Computes the (multiplicative) inverse of a matrix
# The @ operator can be used as a shorthand for np.matmul on ndarrays
# .T is the transpose of a matrix

B = np.linalg.inv(X.T @ X) @ (X.T @ Y)
B
```

[12]:
```
array([[ 4.57646455e-02],
       [ 1.88530017e-01],
       [-1.03749304e-03],
       [ 2.93888937e+00]])
```

[13]:
```python
# pretty printing
print(f" theta_1= {B[0][0]:.8f},\n theta_2= {B[1][0]:.8f},\n theta_3= {B[2][0]:.
 ↪8f},\n theta_0= {B[3][0]:.8f}")
```

```
 theta_1= 0.04576465,
 theta_2= 0.18853002,
 theta_3= -0.00103749,
 theta_0= 2.93888937
```

2. Use Linear Regression to solve this problem

[15]:
```
X = df[['TV', 'radio', 'newspaper']] # extract relevant features
Y = df[['sales']] # extract sales
model = LinearRegression().fit(X, Y)
```

[16]:
```
# pretty printing
print(f" theta_1= {model.coef_[0][0]:.8f},\n theta_2= {model.coef_[0][1]:.
↪8f},\n theta_3= {model.coef_[0][2]:.8f},\n theta_0= {model.intercept_[0]:.
↪8f}")
```

```
 theta_1= 0.04576465,
 theta_2= 0.18853002,
 theta_3= -0.00103749,
 theta_0= 2.93888937
```

3. Compare Perfect match! Both approaches lead to the same results !!

1.3 Problem 2. Perform the following tasks with Scikit-Learn

(1) Scale the features using sklearn.preprocessing.MinMaxScale https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[22]:
```
data = np.c_[X, Y]
```

[23]:
```
scaler = sk.preprocessing.MinMaxScaler()
print(scaler.fit(data))
```

```
MinMaxScaler()
```

[24]:
```
# .data_max_:   Per feature maximum seen in the data
# .data_min_:   Per feature minimum seen in the data
# .data_range_: Per feature range (data_max_ - data_min_) seen in the data

print("max: ", scaler.data_max_)
print("min: ", scaler.data_min_)
print("range: ", scaler.data_range_)
```

```
max:  [296.4  49.6 114.   27. ]
min:  [0.7 0.  0.3 1.6]
range:  [295.7  49.6 113.7  25.4]
```

[25]:
```
scaled_data = scaler.transform(data)
scaled_data
```

[25]:
```
array([[0.77578627, 0.76209677, 0.60598065, 0.80708661],
       [0.1481231 , 0.79233871, 0.39401935, 0.34645669],
       [0.0557998 , 0.92540323, 0.60686016, 0.30314961],
       [0.50997633, 0.83266129, 0.51187335, 0.66535433],
```

```
[0.60906324, 0.21774194, 0.51099384, 0.44488189],
[0.02705445, 0.9858871 , 0.65699208, 0.22047244],
[0.19208657, 0.66129032, 0.20404573, 0.4015748 ],
[0.4041258 , 0.39516129, 0.09938434, 0.45669291],
[0.02671627, 0.04233871, 0.00615655, 0.12598425],
[0.67331755, 0.05241935, 0.18381706, 0.35433071],
[0.2211701 , 0.11693548, 0.21020229, 0.27559055],
[0.72370646, 0.48387097, 0.03254178, 0.62204724],
[0.07811972, 0.70766129, 0.5769569 , 0.2992126 ],
[0.32735881, 0.15322581, 0.06068602, 0.31889764],
[0.68785932, 0.66330645, 0.40193492, 0.68503937],
[0.65843761, 0.96169355, 0.46262093, 0.81889764],
[0.22691917, 0.73790323, 1.        , 0.42913386],
[0.94927291, 0.7983871 , 0.48812665, 0.8976378 ],
[0.2316537 , 0.41330645, 0.15831135, 0.38188976],
[0.49577274, 0.48185484, 0.16534741, 0.51181102],
[0.73621914, 0.55846774, 0.46701847, 0.64566929],
[0.80047345, 0.10282258, 0.20404573, 0.42913386],
[0.04227257, 0.32056452, 0.43359719, 0.15748031],
[0.76969902, 0.34072581, 0.22779244, 0.54724409],
[0.20831924, 0.25403226, 0.15831135, 0.31889764],
[0.8867095 , 0.07056452, 0.16886544, 0.40944882],
[0.4808928 , 0.59072581, 0.10817942, 0.52755906],
[0.80960433, 0.33669355, 0.19876869, 0.56299213],
[0.83902604, 0.54637097, 0.19876869, 0.68110236],
[0.23638823, 0.32258065, 0.35620053, 0.3503937 ],
[0.98816368, 0.57056452, 0.37730871, 0.77952756],
[0.37943862, 0.35080645, 0.33685136, 0.40551181],
[0.32634427, 0.03024194, 0.26121372, 0.31496063],
[0.89584038, 0.40322581, 0.        , 0.62204724],
[0.32127156, 0.02822581, 0.06244503, 0.31102362],
[0.98072371, 0.08266129, 0.07211961, 0.44094488],
[0.90023673, 0.88306452, 0.04133685, 0.93700787],
[0.25025364, 0.99596774, 0.39929639, 0.51574803],
[0.14338857, 0.53830645, 0.3060686 , 0.33464567],
[0.76868448, 0.76008065, 0.27880387, 0.78346457],
[0.68244843, 0.44959677, 0.27528584, 0.59055118],
[0.59621238, 0.6733871 , 0.33773087, 0.61023622],
[0.99053094, 0.55846774, 0.01319261, 0.7519685 ],
[0.69732837, 0.16935484, 0.22955145, 0.44488189],
[0.08251606, 0.51814516, 0.37818821, 0.27165354],
[0.58978695, 0.45362903, 0.27440633, 0.52362205],
[0.30098072, 0.19959677, 0.31134565, 0.35433071],
[0.80892797, 0.83669355, 0.16007036, 0.8503937 ],
[0.76597903, 0.31854839, 0.43623571, 0.51968504],
[0.22387555, 0.2358871 , 0.32102023, 0.31889764],
[0.67331755, 0.0625    , 0.30167106, 0.38582677],
```

```
[0.33716605, 0.19354839, 0.02902375, 0.35826772],
[0.72945553, 0.84072581, 0.34564644, 0.82677165],
[0.61515049, 0.93145161, 0.51363237, 0.77165354],
[0.88603314, 0.58064516, 0.13720317, 0.73228346],
[0.67027393, 0.99596774, 0.52506596, 0.87007874],
[0.02231992, 0.56653226, 0.36147757, 0.15354331],
[0.4582347 , 0.38709677, 0.14335972, 0.45669291],
[0.71051742, 1.        , 0.3289358 , 0.87401575],
[0.71017924, 0.59475806, 0.07915567, 0.66141732],
[0.17855935, 0.04032258, 0.18557608, 0.25590551],
[0.88129861, 0.8608871 , 0.47845207, 0.88976378],
[0.80689888, 0.3125    , 0.23746702, 0.55511811],
[0.3449442 , 0.59677419, 0.07124011, 0.48818898],
[0.44098749, 0.86290323, 0.25153914, 0.64566929],
[0.23097734, 0.1875    , 0.00527704, 0.30314961],
[0.10415962, 0.49596774, 0.01671064, 0.31102362],
[0.4687183 , 0.29233871, 0.08707124, 0.46456693],
[0.80047345, 0.55443548, 0.0941073 , 0.68110236],
[0.73080825, 0.88508065, 0.23658751, 0.81496063],
[0.67095029, 0.61693548, 0.33773087, 0.65748031],
[0.36895502, 0.28830645, 0.27616535, 0.42519685],
[0.08826513, 0.66532258, 0.16710642, 0.28346457],
[0.43523842, 0.11491935, 0.27264732, 0.37007874],
[0.71931011, 0.49596774, 0.11257696, 0.60629921],
[0.05478526, 0.88104839, 0.78364116, 0.27952756],
[0.0906324 , 0.03225806, 0.17941953, 0.20866142],
[0.40514034, 0.57459677, 0.12225154, 0.49606299],
[0.01589449, 0.60282258, 0.08003518, 0.14566929],
[0.38992222, 0.15524194, 0.2005277 , 0.37007874],
[0.25600271, 0.53830645, 0.19349164, 0.4015748 ],
[0.80858979, 0.08266129, 0.32189974, 0.42125984],
[0.25228272, 0.40927419, 0.28320141, 0.38188976],
[0.22894826, 0.89717742, 0.31046614, 0.47244094],
[0.71964829, 0.86693548, 0.294635  , 0.79133858],
[0.65099763, 0.37096774, 0.57519789, 0.53543307],
[0.25566452, 0.55443548, 0.13808267, 0.40944882],
[0.37199865, 0.81854839, 0.5532102 , 0.56692913],
[0.2962462 , 0.5141129 , 0.64291996, 0.44488189],
[0.36895502, 0.96370968, 0.44942832, 0.59448819],
[0.45180927, 0.09879032, 0.07915567, 0.37795276],
[0.09435238, 0.03024194, 0.28759894, 0.22440945],
[0.73385188, 0.67540323, 0.51627089, 0.7007874 ],
[0.84612783, 0.7358871 , 0.63324538, 0.81102362],
[0.36083869, 0.28225806, 0.09322779, 0.38976378],
[0.54988164, 0.63709677, 0.46262093, 0.6023622 ],
[0.66587758, 0.07056452, 0.04925242, 0.3976378 ],
[0.62292864, 0.4233871 , 0.19085312, 0.54724409],
```

```
[0.9773419 , 0.85282258, 0.44766931, 0.93700787],
[0.45485289, 0.84072581, 0.40105541, 0.61417323],
[0.74974636, 0.08669355, 0.4353562 , 0.3976378 ],
[1.        , 0.73185484, 0.88478452, 0.87401575],
[0.94521474, 0.20362903, 0.18557608, 0.51968504],
[0.63307406, 0.34677419, 0.15479332, 0.51574803],
[0.8031789 , 0.69153226, 0.04397537, 0.7519685 ],
[0.46398377, 0.93548387, 0.51627089, 0.69291339],
[0.08217788, 0.22177419, 0.2585752 , 0.22047244],
[0.30334799, 0.00604839, 0.20140721, 0.27952756],
[0.04193439, 0.00806452, 0.22251539, 0.14566929],
[0.86134596, 0.54233871, 0.04573439, 0.71653543],
[0.7612445 , 0.16532258, 0.4942832 , 0.46456693],
[0.81501522, 0.76612903, 0.20140721, 0.79527559],
[0.59181603, 0.31048387, 0.01846966, 0.49212598],
[0.70645925, 0.41532258, 0.09146878, 0.56299213],
[0.26208996, 0.94354839, 0.30079156, 0.51181102],
[0.25160636, 0.70564516, 0.46086192, 0.43307087],
[0.46838011, 0.28830645, 0.22251539, 0.41732283],
[0.25600271, 0.01612903, 0.12752858, 0.30708661],
[0.42272574, 0.74395161, 0.6939314 , 0.56299213],
[0.06323977, 0.32258065, 0.19349164, 0.19685039],
[0.47548191, 0.54032258, 0.40369393, 0.54724409],
[0.06121069, 0.4375    , 0.44063325, 0.21259843],
[0.75515725, 0.0483871 , 0.13456464, 0.39370079],
[0.41393304, 0.69758065, 0.1064204 , 0.53543307],
[0.77375719, 0.65120968, 0.64995602, 0.71259843],
[0.29252621, 0.23790323, 0.22515391, 0.35433071],
[0.02401082, 0.78427419, 0.44239226, 0.19685039],
[0.26885357, 0.        , 0.07827617, 0.28346457],
[0.74264457, 0.98790323, 0.02550572, 0.90944882],
[0.19918837, 0.24193548, 0.3764292 , 0.31889764],
[0.        , 0.7983871 , 0.07387863, 0.        ],
[0.89448766, 0.05846774, 0.37554969, 0.43700787],
[0.02603991, 0.5483871 , 0.01583113, 0.16141732],
[0.74095367, 0.67540323, 0.39401935, 0.70866142],
[0.12242137, 0.77822581, 0.57431838, 0.36220472],
[0.16097396, 0.94758065, 0.07211961, 0.39370079],
[0.08420697, 0.78629032, 0.07915567, 0.31102362],
[0.92323301, 0.58266129, 0.52242744, 0.75590551],
[0.14305039, 0.52217742, 0.17766051, 0.31496063],
[0.62292864, 0.88508065, 0.0123131 , 0.7519685 ],
[0.24585729, 0.34274194, 0.11081794, 0.36614173],
[0.65268854, 0.71370968, 0.66226913, 0.69291339],
[0.74332093, 0.66935484, 0.33069481, 0.72834646],
[0.35136963, 0.11491935, 0.29991205, 0.34645669],
[0.32296246, 0.2983871 , 0.33948989, 0.38582677],
```

```
[0.4721001 , 0.03830645, 0.07651715, 0.34251969],
[0.80960433, 0.14717742, 0.07387863, 0.45669291],
[0.82008793, 0.98790323, 0.38698329, 0.93700787],
[0.12614136, 0.8125    , 0.10202287, 0.36614173],
[0.14879946, 0.52016129, 0.17854002, 0.33464567],
[0.94690565, 0.28024194, 0.32277924, 0.57086614],
[0.40683125, 0.16935484, 0.42568162, 0.39370079],
[0.66587758, 0.46975806, 0.12225154, 0.59055118],
[0.57693608, 0.80040323, 0.3289358 , 0.68503937],
[0.63273588, 0.42540323, 0.08091469, 0.5511811 ],
[0.01149814, 0.23387097, 0.0474934 , 0.06299213],
[0.31518431, 0.87701613, 0.44151275, 0.53937008],
[0.50422726, 0.02620968, 0.21108179, 0.33464567],
[0.03719986, 0.74395161, 0.39489886, 0.22440945],
[0.44301657, 0.37096774, 0.30167106, 0.44488189],
[0.58099425, 0.36491935, 0.26737027, 0.50393701],
[0.2874535 , 0.72177419, 0.43095866, 0.46062992],
[0.63476496, 0.36491935, 0.22251539, 0.52362205],
[0.550558  , 0.74193548, 0.06244503, 0.64566929],
[0.39398039, 0.29637097, 0.04485488, 0.40551181],
[0.79066622, 0.06854839, 0.74318382, 0.40551181],
[0.05816706, 0.75806452, 0.18733509, 0.2519685 ],
[0.69699019, 0.10483871, 0.16798593, 0.41732283],
[0.72607372, 0.47580645, 0.50395778, 0.61023622],
[0.95908015, 0.21370968, 0.05364996, 0.52755906],
[0.16672303, 0.23387097, 0.15919085, 0.26771654],
[0.5539398 , 0.42137097, 0.41424802, 0.50787402],
[0.06391613, 0.40524194, 0.14687775, 0.23622047],
[0.56712885, 0.14314516, 0.10993843, 0.3976378 ],
[0.74974636, 0.06854839, 0.11257696, 0.38976378],
[0.93405479, 0.9858871 , 0.3649956 , 1.        ],
[0.83767332, 0.60887097, 0.1759015 , 0.73228346],
[0.5732161 , 0.15725806, 0.30694811, 0.3976378 ],
[0.93337842, 0.04637097, 0.20580475, 0.4015748 ],
[0.55765979, 0.2016129 , 0.15215479, 0.43307087],
[0.52722354, 0.05241935, 0.0703606 , 0.3503937 ],
[0.73655732, 0.10887097, 0.23834653, 0.41732283],
[0.18769023, 0.11491935, 0.2585752 , 0.27952756],
[0.97024011, 0.86693548, 0.62884785, 0.96850394],
[0.85593507, 0.42943548, 0.26121372, 0.62992126],
[0.69090294, 0.90927419, 0.16974494, 0.82677165],
[0.46939466, 0.04233871, 0.23131047, 0.34251969],
[0.64389584, 0.57862903, 0.15743184, 0.61811024],
[0.96482922, 0.28024194, 0.02990325, 0.56299213],
[0.06087251, 0.24395161, 0.20316623, 0.2007874 ],
[0.13121407, 0.82862903, 0.04837291, 0.36220472],
[0.25295908, 0.21774194, 0.05013193, 0.32677165],
```

```
[0.0557998 , 0.08266129, 0.27528584, 0.16929134],
[0.56171796, 0.84677419, 0.02902375, 0.70866142],
[0.50388908, 0.71774194, 0.05013193, 0.61811024],
[0.12681772, 0.07459677, 0.11873351, 0.23622047],
[0.31619885, 0.09879032, 0.06860158, 0.31889764],
[0.59621238, 0.1875    , 0.05364996, 0.44094488],
[0.95671288, 0.84677419, 0.57959543, 0.94094488],
[0.78254988, 0.1733871 , 0.07387863, 0.46456693]])
```

(2) Split the above scaled data into 80% for training and 20% for test using train_test_split(). Use the same test and training data in the following tasks https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[28]:
```
X = scaled_data[:,:3]
```

[29]:
```
y = scaled_data[:,-1]
```

[30]:
```
#  shuffle=True, random_state=44 : to make the results reproducible
X_train, X_test, y_train, y_test = sk.model_selection.train_test_split(X, y,␣
 ↪test_size=0.20, train_size=.80, shuffle=True, random_state=44)
```

(3) use the ordinary linear regression (OLR) to train the model. Compare the R2 scores from training and test data, and discuss your observations

[32]:
```
# sk.linearmodel.LinearRegression uses OLR by default (OLR has an aplha of 0␣
 ↪basically)
model = LinearRegression()
model.fit(X_train, y_train)
```

[32]:
```
LinearRegression()
```

[33]:
```
print(f"training's data R2: {model.score(X_train, y_train):.5f}")
print(f"testing's data R2: {model.score(X_test, y_test):.5f}")
```

```
training's data R2: 0.90659
testing's data R2: 0.83103
```

Meaning that the model explains approximately 90.65% of the training's data but only 83.1% of the testing dataset

[35]:
```
# **** Alternative EXTRA approach to gather metrics ****

y_pred = model.predict(X_test)

mae = sk.metrics.mean_absolute_error(y_test, y_pred)
mse = sk.metrics.mean_squared_error(y_test, y_pred)
r2 = sk.metrics.r2_score(y_test, y_pred)
```

```python
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Absolute Error: 0.05171885100739434
Mean Squared Error: 0.004652908399823869
R-squared: 0.8310290458396921
```

(4) Use the Ridge regression to train the model (you may explore different hyperparameter). Compare the R2 scores from training and test data, and discuss your observation https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

[38]:
```python
model = sk.linear_model.Ridge(alpha=.08)
model.fit(X_train, y_train)

print(f"training's data R2: {model.score(X_train, y_train):.5f}")
print(f"testing's data R2: {model.score(X_test, y_test):.5f}")
```

```
training's data R2: 0.90656
testing's data R2: 0.83128
```

With alpha=.08, it seems that the model can explain 90.65% of the training data but only 83.12% of the testing data

(5) use the Lasso regression to train the model (you may explore different hyperparameter). Compare the R2 scores from training and test data, and discuss your observations. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

[42]:
```python
model = sk.linear_model.Lasso(alpha=.0002)
model.fit(X_train, y_train)

print(f"training's data R2: {model.score(X_train, y_train):.5f}")
print(f"testing's data R2: {model.score(X_test, y_test):.5f}")
```

```
training's data R2: 0.90653
testing's data R2: 0.83132
```

Meaning that the model explains approximately 90.65% of the training data and 83.13% of the testing data.

It also seems to require a really small alpha so that it is less punishing