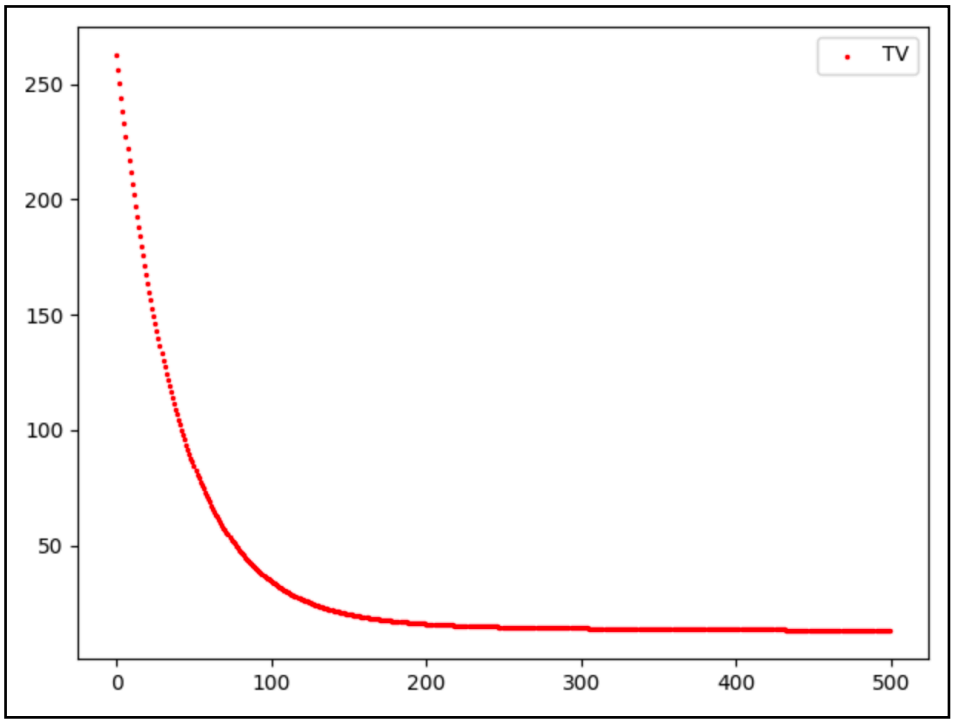
Q1. You have to report the $\theta$ parameters in step 3 when you are using TV feature.

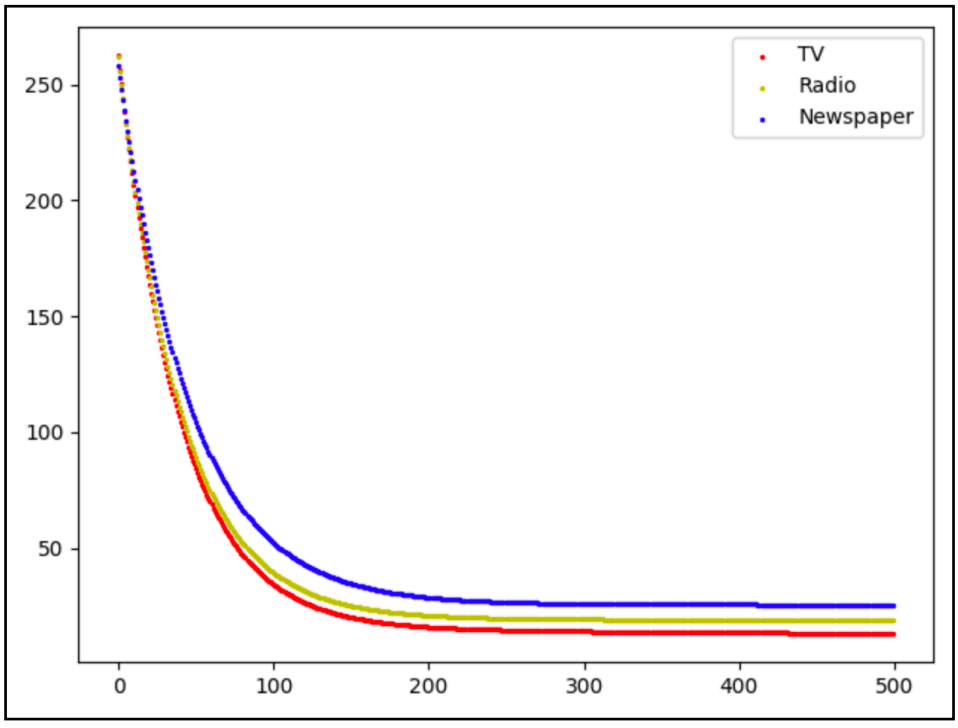
$\theta_0$ = 10.11283406777275

$\theta_1$ = 8.271831294479673

Q2. A plot, which visualises the change in cost function J($\theta$) at each iteration.

As the graph shows below: only use TV feature, and use all features in one graph



Q3. RMSE for your training set when you use TV feature.

RMSE = 3.6403454893687783

Q4. RMSE for test set, when you use TV feature.

RMSE = 3.9085603448797355

Q5. RMSE for test set, when you use Radio feature.

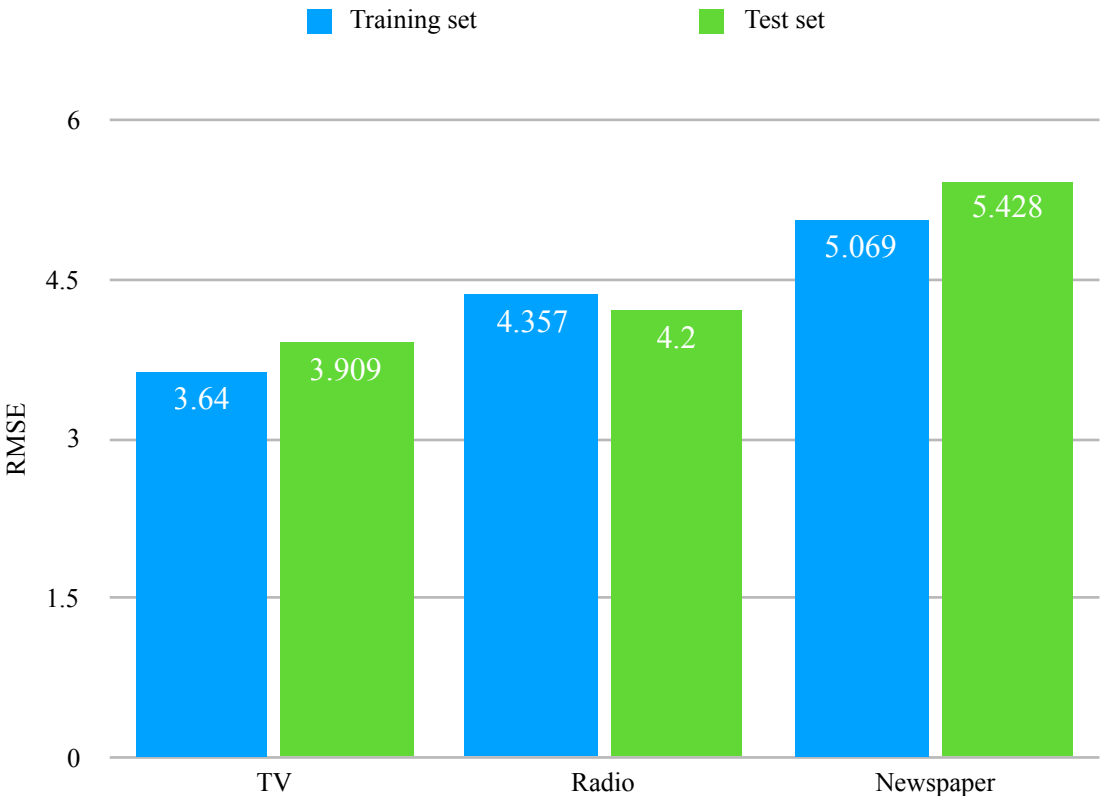
RMSE = 4.20042579511254

Q6. RMSE for test set, when you use newspaper feature.

RMSE = 5.427909854899054

Q7. Compare the performance of your three models and rank them accordingly.

As the graph shows below, which stands for the RMSE for training set and test set by using TV, Radio and Newspaper features respectively. As we all know, RMSE is the best value and criteria if we only care about prediction accuracy. As we can see, the RMSE by using TV feature for both training and test set is the smallest in all of the three models.




The picture below also shows the original value and the predict value by using TV, Radio and Newspaper feature respectively, the TV feature shows the smallest gap which are very close to original values. So using TV feature rank the first, followed by using Radio feature and the last one is using Newspaper feature.

```
original value
 [10.8  9.9  5.9 19.6 17.3  7.6  9.7 12.8 25.5 13.4]
use tv feature
 [11.1982147  12.2052689  10.57440058 14.75927025 14.2809195  11.16184886
 12.72837761 15.04460227 18.02660165 16.58595467]
use radio feature
 [16.63346409 12.34919035 11.40184269 16.76071975 15.85579064 11.34528462
 11.51495883 12.13709759 16.76071975 12.03812097]
use newspaper feature
 [13.15228619 13.15903626 14.02304508 13.07803544 13.15903626 13.42228895
 13.22991199 13.1725364  15.19080699 13.25016219]
```

## Code :

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
def pre_processing(array):
    trnsp = np.array(array).transpose()
    for sublist in trnsp:
        min_x = min(sublist)
        max_x = max(sublist)
        for i in range(len(sublist)):
            sublist[i] = (sublist[i] - min_x) / (max_x - min_x)
    normalisation = trnsp.transpose()
    return normalisation
def batch_gradient_descent(max_iteration, j, x, y):
    theta_0 = -1
    theta_j = -0.5
    alpha = 0.01
    x_T = x.transpose()
    j_funct = []
    for _ in range(max_iteration):
        hypothesis = theta_0 + theta_j * x_T[j]
        loss = y - hypothesis
        gradient_0 = sum(loss) / m
        gradient_1 = np.dot(loss, x_T[j]) / m
        theta_0 = theta_0 + alpha * gradient_0
        theta_j = theta_j + alpha * gradient_1
        j_funct.append(sum(np.square(loss)) / m)
    return theta_0, theta_j, j_funct
```

```
def predict(theta_0, theta_j, x, j):
    x_T = x.transpose()[j]
    res = theta_0 + theta_j * x_T
    return res
def evaluate_rmse(t0, tj, x, y, j):
    loss = (t0 + tj * x.transpose()[j]) - y
    length = len(loss)
    rmse = math.sqrt(sum(np.square(loss)) / length)
    return rmse
def plot(a, b, c):
    axis_x = [i for i in range(500)]
    plt.scatter(axis_x, a, color="r", s=2, label='TV')
    plt.scatter(axis_x, b, color="y", s=2, label='Radio')
    plt.scatter(axis_x, c, color="b", s=2, label='Newspaper')
    plt.legend()
    plt.show()


m = 190
data_frame = pd.read_csv("Advertising.csv")
data = data_frame.values[:]
y_train = data[:m, -1]
y_test = data[m:, -1]
x = data[:, 1: -1]
normal_x = pre_processing(x)
normal_x_train = normal_x[:m, :]
normal_x_test = normal_x[m:, :]
tv_0, tv_j, tv_j_funct = batch_gradient_descent(500, 0, normal_x_train, y_train)
r_0, r_j, r_j_funct = batch_gradient_descent(500, 1, normal_x_train, y_train)
n_0, n_j, n_j_funct = batch_gradient_descent(500, 2, normal_x_train, y_train)
print("Q1: ",tv_0, tv_j)
```

```
plot(tv_j_funct, r_j_funct, n_j_funct)

print("Q3: ", evaluate_rmse(tv_0, tv_j, normal_x_train, y_train, 0))

print("Q4: ", evaluate_rmse(tv_0, tv_j, normal_x_test, y_test, 0))

print("Q5: ", evaluate_rmse(r_0, r_j, normal_x_test, y_test, 1))

print("Q6: ", evaluate_rmse(n_0, n_j, normal_x_test, y_test, 2))
```