Dosado, Michael Vincent BS CS IV 201904469 CMSC197 Machine Learning 3rd Homework

Problem Set: Implementing Linear Regression with Gradient Descent as Optimization Function

1. What are the optimal weights found by your implemented gradient descent? Plug it into the linear model:

$$h_\theta(x) = \theta_0 + \theta_1 TV + \theta_2 Radio + \theta_3 Newspaper$$

What are your interpretations regarding the formed linear model?

Optimal Weights:
Train Set : [-0.0028688 0.74683357 0.53207527 -0.00539423]
Test Set: [0.00753891 0.77605269 0.54897292 0.00456605]
Both/Entire Set: [-0.0028688 0.74683357 0.53207527 -0.00539423]

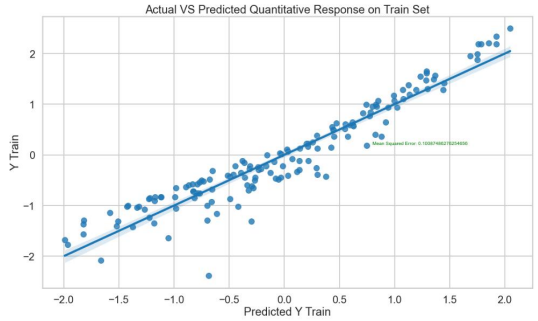
Plugging it into the Linear Model:

hΘ(x) = -0.002868804851867801 + 0.7468335737103866*TV + 0.5320752735031187*Radio + -0.005394228342171964*Newspaper

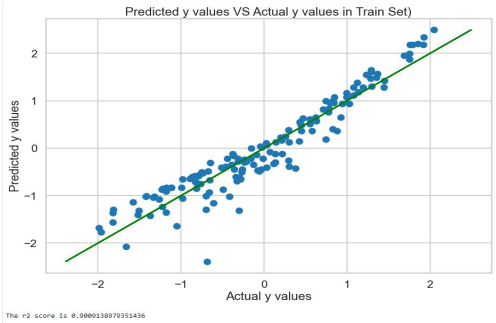
Observing the Linear Model, from the Train Set w/ the least Bias, my interpretation is that the TV Ads has the greatest significance, followed by the Radio Ads, and Lastly the Newspaper Ads.

2. Provide a scatter plot of the $(\widehat{y^{(i)}})$ and $y^{(i)}$ for both the train and test set. Is there a trend? Provide an r2 score (also available in sklearn).

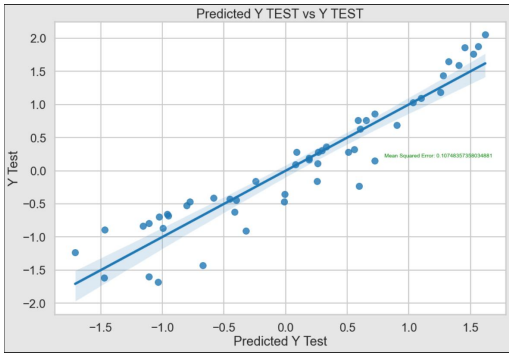
Scatter Plot -- Train Set:
In #6



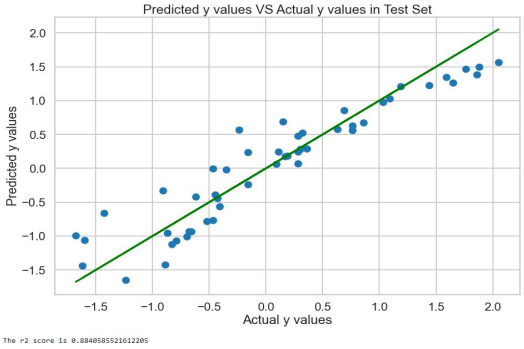
In Scratch



Scatter Plot -- Train Set:
In #7



In Scratch



The R2 Score
Train Set : 0.9009138979351436
Test Set : 0.8840585521612205

3. What happens to the error, r2, and cost as the number of iterations increase? Show your data and proof. You can alternatively plot your result data for visualization and check until 50000 iterations or more (actually)

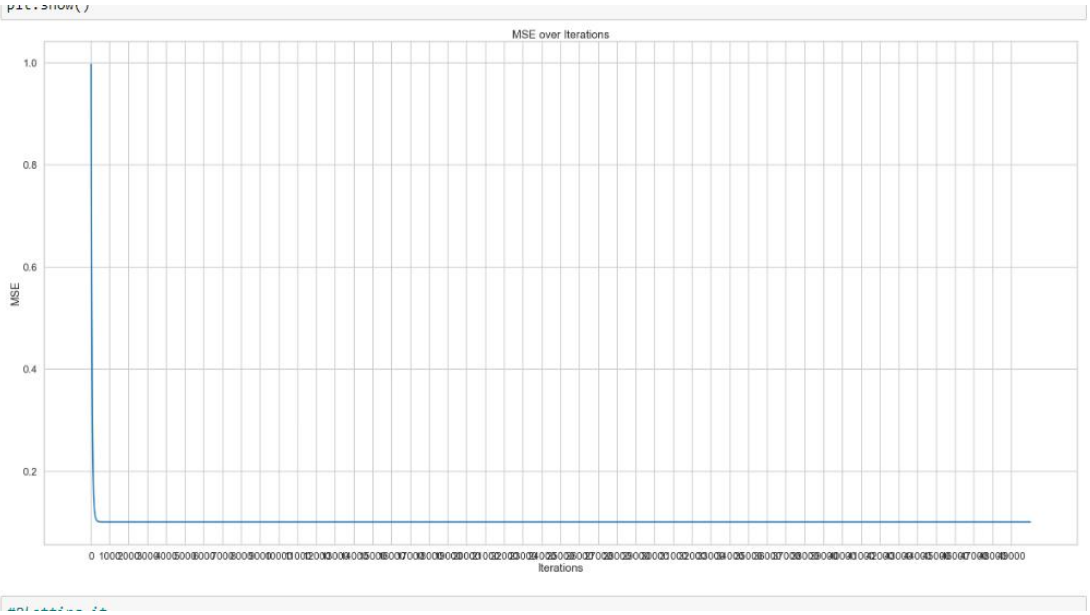
Out[112]:

	Cost	MSE/Error	R2
0	0.330243	0.997352	0.020333
1	0.323932	0.976576	0.040741
2	0.317771	0.956308	0.060649
3	0.311758	0.936536	0.080071
4	0.305888	0.917246	0.099019
...	...	...	...
49995	0.050437	0.100875	0.900914
49996	0.050437	0.100875	0.900914
49997	0.050437	0.100875	0.900914
49998	0.050437	0.100875	0.900914
49999	0.050437	0.100875	0.900914

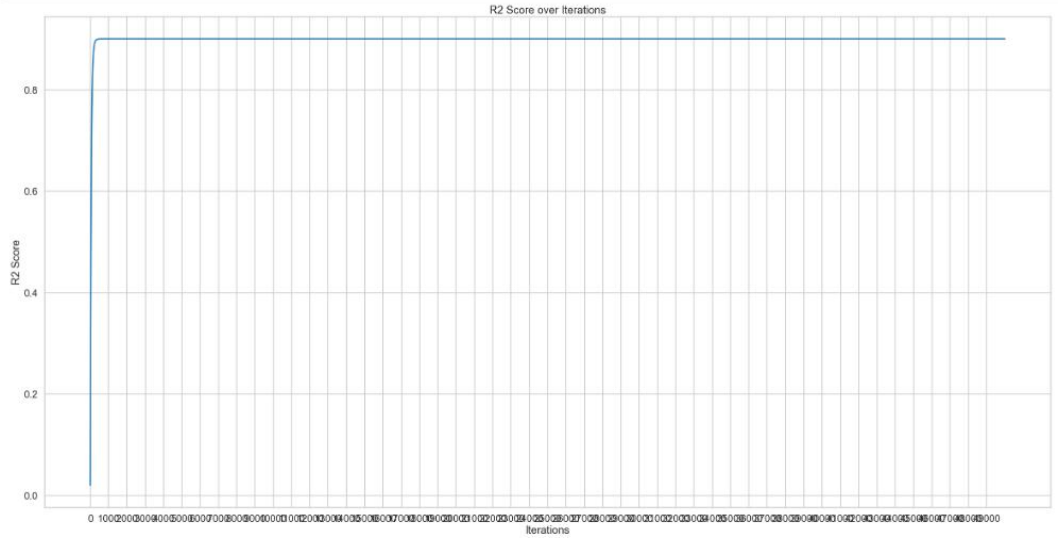
50000 rows × 3 columns

From my observation, the trend for each iteration is as follows:
- Cost has a decreasing trend
- MSE has a decreasing trend
- R2 score increases

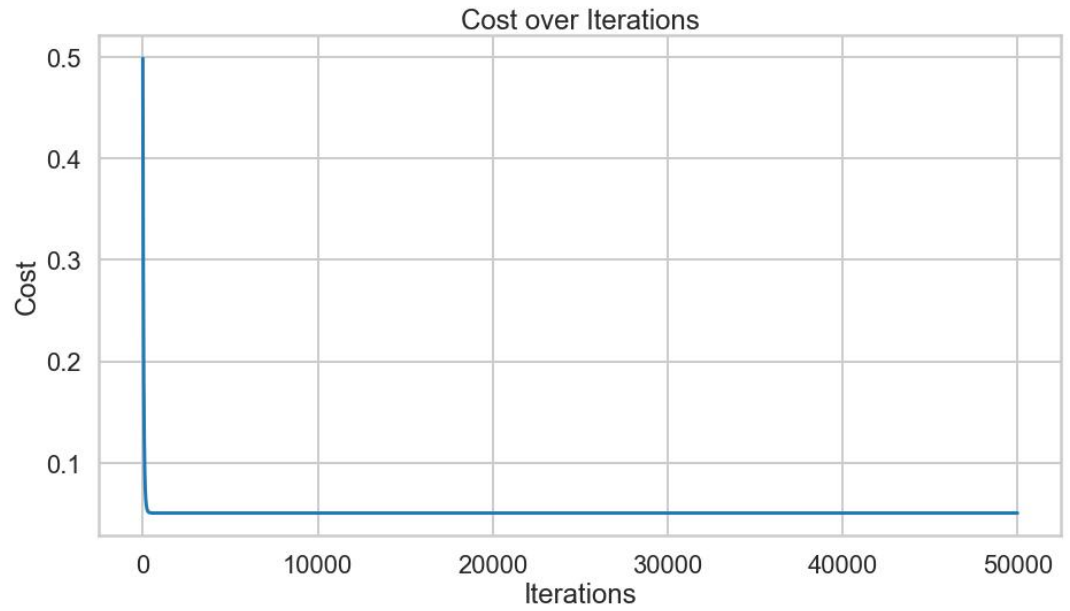
PLOT FOR MSE over Iterations(5000)



PLOT FOR R2 Scores over Iterations



PLOT OF COST over ITERATIONS using the PLOT_COST Function:

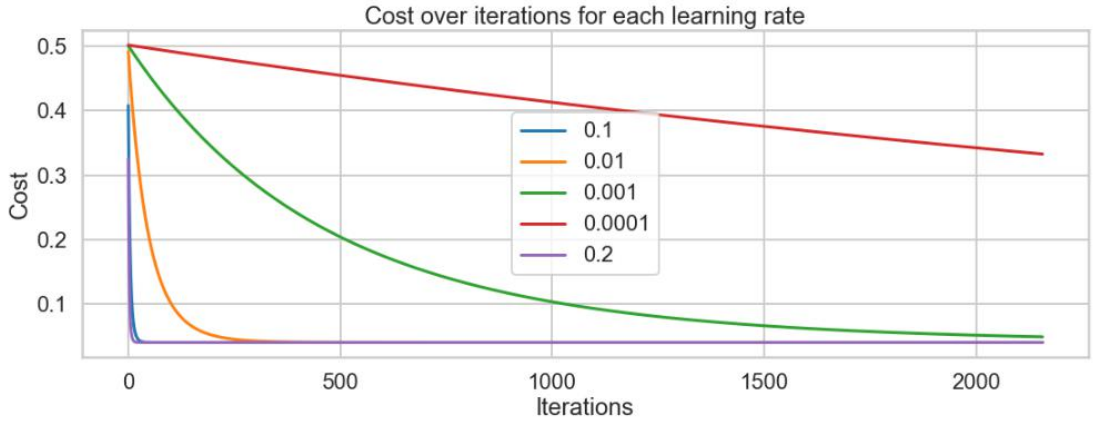


4. Once you determine the optimal number of iterations, check the effect on the cost and error as you change the learning rate. The common learning rates in machine learning include 0.1, 0.01, 0.001, 0.0001, 0.2 but you have the option to include others. Visualize the cost function (vs the optimal number of iterations) of each learning rate in ONLY ONE PLOT. Provide your analysis

```
:  # 4    Once you determine the optimal number of iterations, check the effect on the cost and
   # error as you change the learning rate. The common learning rates in machine learning
   # include 0.1, 0.01, 0.001, 0.0001, 0.2 but you have the option to include others. Visualize
   # the cost function (vs the optimal number of iterations) of each learning rate in ONLY ONE
   # PLOT. Provide your analysis.


   #Determining the optimal number of iterations
   def optimal_iterations(max_r2, r2_arr):
       for j in range(len(r2_arr)):
           if r2_arr[j] == max_r2:
               return j

   optimal_num = optimal_iterations(max(r2_list), r2_list)
   print(f"The optimal number of iterations is {optimal_num}")
```

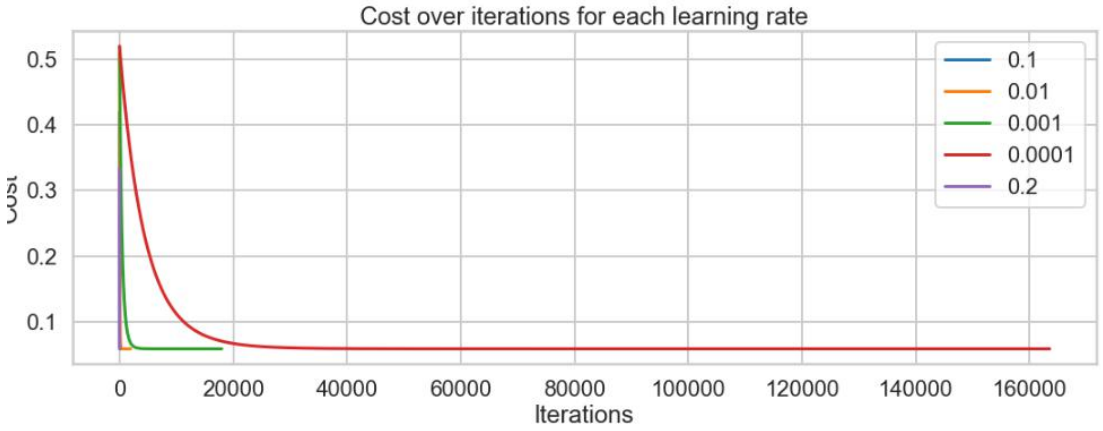
```
The optimal number of iterations is 3058
```

```
The cost at the optimal number of iterations is 0.039768252462286276
The weights at the optimal number of iterations are [-0.00483549  0.76584986  0.58454136 -0.04815294]

The cost at the last iteration is 0.03976825246228625
The weights at the last iteration are [-0.00483549  0.76584986  0.58454137 -0.04815295]
```



For my Analysis, the relationship between the cost and the number of iterations for each learning rate/alpha is inversely proportional such that, as the number of iterations per learning rate increase, the cost decreases.

5. Is there a relationship on the learning rate and the number of iterations?



- From my analysis, it seems that the number of Iterations is limited in relation to the value of the learning rate such that the smaller the learning rate, the more iterations it has. The cost is also lowered/decreased much faster when it has a bigger learning rate value. As we can see on the plot, the largest learning rate value (0.2, 0.1,…) has a more steeper line compared the the smallest learning rate value (0.0001) that has a more slope/curve line. But they all decrease further as the more iterations happen.

6. Compare the results with the results of ordinary least squares function

-- From item #6 of the code solution, We have taken the OLS Value for the TRAIN and TEST Set

In [33]:
```
# EVALUATION: As seen above, the Mean Squared Error (MSE) is 0.11102077.

# Utilize MSE and R2 to assess performance for the TRAIN set.
# r2 score and mean squared error from Sklearn
MSE_train = mean_squared_error(y_training, y_hat_train)
R2_train = r2_score(y_training, y_hat_train)

print("MSE for TRAIN set:", MSE_train)
print("R2 for TRAIN set:", R2_train)

# verify R2 using ols from statsmodels
ols = LinearRegression()
ols.fit(X_training, y_training)
print("OLS for TRAIN set:", ols.score(X_training, y_training))


# The model is more accurate the closer the MSE value is to zero.
# As a result, we may say that our model is already effective.
```

```
MSE for TRAIN set: 0.11755860451522997
R2 for TRAIN set: 0.8870107435670415
OLS for TRAIN set: 0.8870107435670415
```

In [66]:
```
# EVALUATION: The Mean Squared Error is 0.118558126963
# It means that our model, which was developed using the training set, was able to accurately predict the test set.
# However, the MSE in the test dataset is higher than the MSE in our train dataset, indicating that it isn't any better.

# Utilize MSE and R2 to assess performance for the TEST set.
# r2 score and mean squared error from Sklearn
MSE_test = mean_squared_error(y_testing, y_hat_test)
R2_test = r2_score(y_testing, y_hat_test)

print("MSE for TEST set:", MSE_test)
print("R2 for TEST set:", R2_test)

# verify R2 using ols from statsmodels
ols = LinearRegression()
ols.fit(X_testing, y_testing)
print("OLS for TEST set:", ols.score(X_testing, y_testing))
```

```
MSE for TEST set: 0.05354117153776757
R2 for TEST set: 0.9390661407164399
OLS for TEST set: 0.9390661407164399
```

R2Score for TRAIN SET: 0.887010743

OLS for TRAIN set: 0.8870107435670415

R2 Score for TRAIN SET: 0.887010743

OLS for TEST set: 0.9390661407164399

From my analysis, we can see that the values of the OLS and the R2 score are closely similar.