CS 6375
ASSIGNMENT 1

Names of students in your group:
Terisha Kolencherry Prax

Number of free late days used: None
Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

- Dataset: https://archive.ics.uci.edu/dataset/597/productivity+prediction+of+garment+employees
- Gradient Descent Lab from Class: https://colab.research.google.com/drive/1rmblKcJUf0A18GMk7Jx4u6DXnQLf32V6?usp=sharing#scrollTo=QVz-JbxFJXOW
- Gradient Descent in R: (helpful for orienting logic)
- https://oindrilasen.com/2018/02/compute-gradient-descent-of-a-multivariate-linear-regression-model-in-r/#:~:text=Similar%20to%20the%20Gradient%20Descent%20for%20a%20Univariate,%28i%29%29.%20xj%20%28i%29%20where%20j%20%3D%200%2C1%2C2%E2%80%A6n%20%7D
- Alternate Multiple Linear Regression with Gradient Descent (used generating random weights and using the dot product of A^T * A for squaring the matrix)
- https://www.kaggle.com/code/rakend/multiple-linear-regression-with-gradient-descent
- R-Squared: https://en.wikipedia.org/wiki/Coefficient_of_determination
- Reading in Hosted File: https://stackoverflow.com/questions/32400867/pandas-read-csv-from-url
- Residuals: https://online.stat.psu.edu/stat462/node/120/

Assignment 1 Write Up

Executive Summary

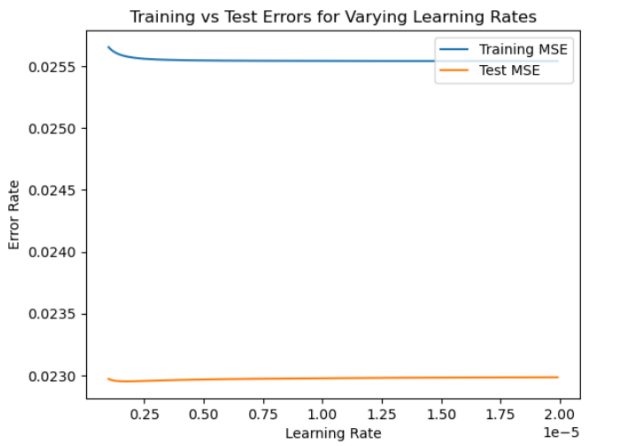
This project took a look at productivity data for garment workers. The response variable was actual productivity score. Initially, five predictors were selected for regression, but the optimal models with and without the library ended up using only one predictor – targeted productivity. The optimal model minimized MSE without seemingly overfitting the data and would be helpful for prediction.

I don't think that either model is the best model possible. A couple of ways I might improve the model given more time would be:

- Focus on what predictors are strongly correlated to targeted productivity. Targeted productivity seems to be a metric that is formulated from other data points so a more robust model might look to the data points that make up that metric.
- Do a log or square root transformation for some of the predictors with larger numbers. Gradient descent can work better when all the data is scaled, but in this case I didn't want to blindly scale all the predictors since it's a mix of qualitative and quantitative data. For example, I assigned numbers to each quarter (1-4). If I had scaled the data, I could have ended up with a quarter value of 1.3 but that number has no significance on its own in this context because the scaling isn't done in relation to what month is assigned to the data.
- It's possible that this data isn't best represented by a linear model.

The final non-library and library gradient descent models are shown below:

```
****MODEL PARAMETERS****

Weights
w_ 0  :  0.3745401188473625 -------> 0.15751497340626586
w_ 1  :  0.9507143064099162 -------> 0.7902068560712477
Iterations :  100000
 Threshold :  1e-05
 Number of Predictors :  1

Minimum Test MSE 0.02295117327268457  for Learning Rate =  1.7000000000000007e-06
```




```
WEIGHT ESTIMATION
w_ 0  :  0.2571907907828791
w_ 1  :  0.19284891025734205

Minimum Test MSE 0.0326880814009939  for Learning Rate =  1.4600000000000014e-05
```

Part Zero: Pre-Processing

When looking at the data the only column that was missing values was *wip*, which stands for works in progress (see Table 0.1).

Name	Description	Name	Description	Name	Description
date	Date in MM-DD-YYYY	no_of_workers	# of workers on team	over_time	Amount of overtime in minutes
day	Day of the Week	no_of_style_change	# of changes in the style of a particular product	incentive	Amount of financial incentive offered (BDT)
quarter	Quarter of the Year	targeted_productivity	Set by the Authority for the team daily	idle_time	Amount of time production was interrupted
department	Associated Department	smv	Standard minute value – allocated time for each task	idle_men	Number of workers idle due to production interruption
team_no	Associated Team #	wip	Number of unfinished items for products	actual_productivity	Actual producitivity % - ranges from 0 to 1

Table 0.1

In order to assign an appropriate value to the empty cells, I looked to see if there were any rows where there were zero works in progress. Since there were not, I filled all NaN cells with zero.

Upon checking the unique values for the *quarter* column, it was noted that there were 5 quarters in the year. In order to understand what the error was, I looked up the rows that had “Quarter 5” as a value and noted that all of them were supposed to be included in Q1, and that replacement was made.

Next, the *department* column was cleaned up. There were two “finishing” values and one “sewing” value. These two different department names were given a numerical value to help view the correlation between *department* and *actual_productivity*. A similar process was used to transform the days of the week to numerical values.

Once all qualitative predictors had been transformed, the correlation matrix was generated (Figure 0.1).

actual_productivity	1.000000
targeted_productivity	0.421594
department	0.087624
incentive	0.076538
wip	0.047389
day	-0.005104
over_time	-0.054206
no_of_workers	-0.057991
idle_time	-0.080851
smv	-0.122089
quarter	-0.123779
team	-0.148753
idle_men	-0.181734
no_of_style_change	-0.207366

Figure 0.1

Of interest here is the correlation between the response variable, actual productivity, and the predictors. The strongest correlations were with targeted productivity, number of style changes, idle men, quarter, and team. These predictors and the response variable were put into a slimmed down dataframe that would be used for the majority of the regression evaluation (Figure 0.2)

	quarter	targeted_productivity	no_of_style_change	idle_men	team	actual_productivity
count	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000
mean	2.252297	0.729632	0.150376	0.369256	6.426901	0.735091
std	1.130974	0.097891	0.427848	3.268987	3.463963	0.174488
min	1.000000	0.070000	0.000000	0.000000	1.000000	0.233705
25%	1.000000	0.700000	0.000000	0.000000	3.000000	0.650307
50%	2.000000	0.750000	0.000000	0.000000	6.000000	0.773333
75%	3.000000	0.800000	0.000000	0.000000	9.000000	0.850253
max	4.000000	0.800000	2.000000	45.000000	12.000000	1.120437

Figure 0.2

Finally, before beginning the regression analysis, a scatterplot of all the selected predictors was generated (Figure 0.3). The first takeaway from this plot is the difference in scales for the various predictors, which was further illustrated by Figure 0.2. Later in this report, we will look at scaling options. The second takeaway is that the trend of the data doesn't seem to be strongly linear. The means of the various predictors at different values don't strongly increase of decrease. This conclusion is supported by Figure 0.1.

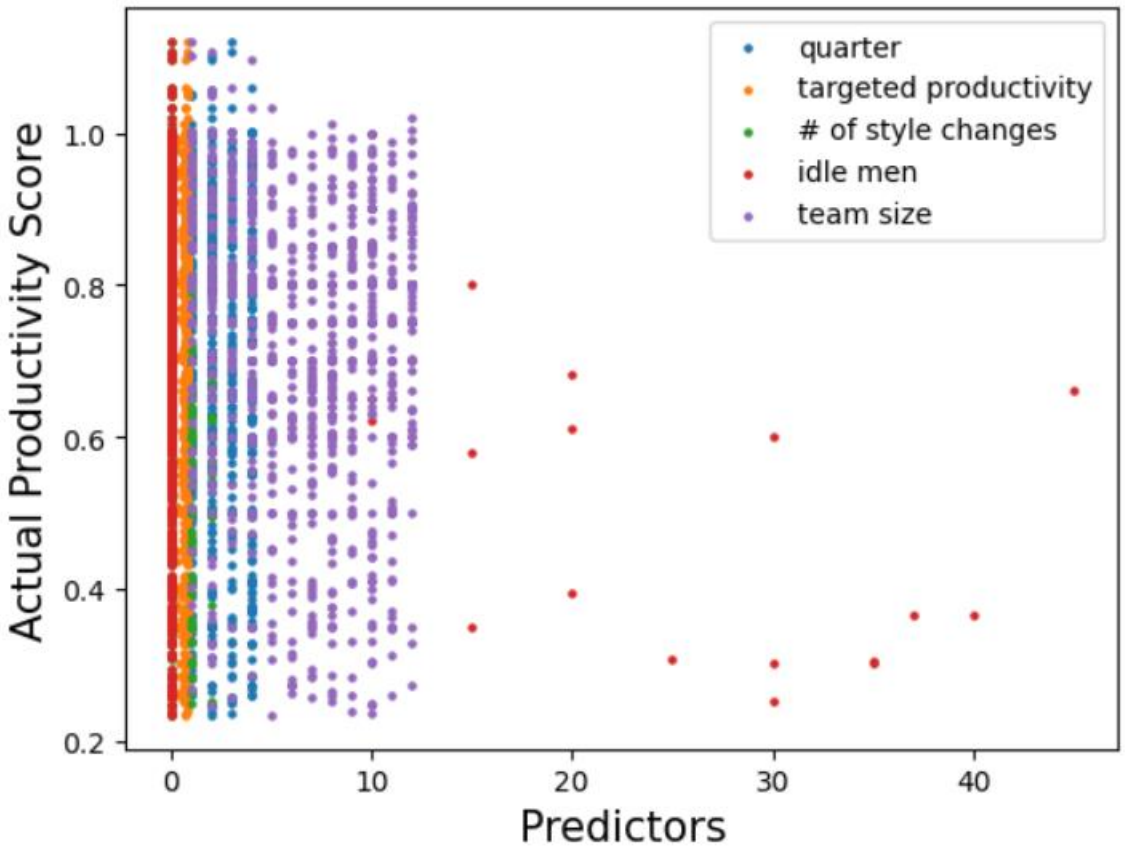


Figure 0.3

Part One: DIY Gradient Descent

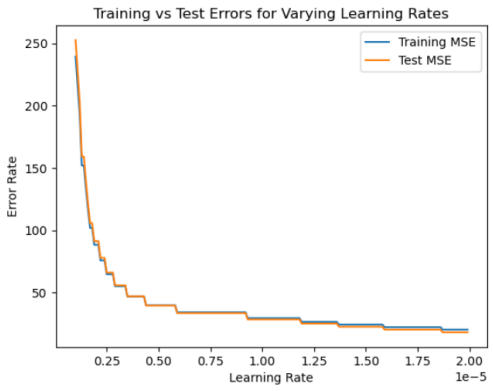
Experimenting with Varying Learning Rates, Iterations, and Thresholds

For each round, an object was created from a dataframe that specified the response variable column name and the number of predictors. Initially, each regression started with 5 predictors. The different rounds of training used different starting weights, looped through various learning rates, and had different iteration values. In the beginning, the models generated moderate mean squared errors, but there was concern about overfitting because the training error rate was lower than the testing error rate. These were the key takeaways:

- While starting weights were randomly generated, starting weights pulled from a (0,1) distribution had better errors versus starting weights pulled from distributions with higher upper bounds (Figures 1.1 and 1.2)
- For this particular set of data, a higher number of iterations led to smaller errors
- For this particular set of data, a smaller threshold led to smaller errors
- For this model, learning rates seemed to plateau towards the end

```
****MODEL PARAMETERS****

Weights
w_ 0  :  6.0 -------> 4.775748942794789
w_ 1  :  3.0 -------> 0.07092332039438708
w_ 2  :  4.0 -------> 3.1154425326298245
w_ 3  :  6.0 -------> 5.643560463394214
w_ 4  :  2.0 -------> 0.3235840897557365
w_ 5  :  7.0 -------> -0.9381289305425192
Iterations :  100
 Threshold :  0.1
 Number of Predictors :  5

Minimum Test MSE 18.30359047398677  for Learning Rate =  1.8700000000000017e-05
```

```
****MODEL PARAMETERS****

Weights
w_ 0  :  0.6722785586307918 -------> 0.36874433840601695
w_ 1  :  0.4880783992405837 -------> -0.010558652720481124
w_ 2  :  0.8254951740358963 -------> 0.6219027836157457
w_ 3  :  0.031446387626298145 -------> -0.0393625287535018 66
w_ 4  :  0.8080499633648477 -------> -0.009114088967741867
w_ 5  :  0.5656174196105306 -------> -0.008742295191248606
Iterations :  100000
 Threshold :  1e-05
 Number of Predictors :  5

Minimum Test MSE 0.022506584802716886  for Learning Rate =  1.990000000000002e-05
```



Figures 1.1 and 1.2

Once there was what was determined to be a "good enough" model, I looked at the residual plots for the training data (Figure 1.3). One can see the data points have a trumpet shape, which can indicate that there is a need for scaling.

Figure 1.3

Scaling Predictors

When looking at the boxplots of the predictors, the first thing that stood out was the number of outliers for the *idle_men* predictor (Figure 1.4).

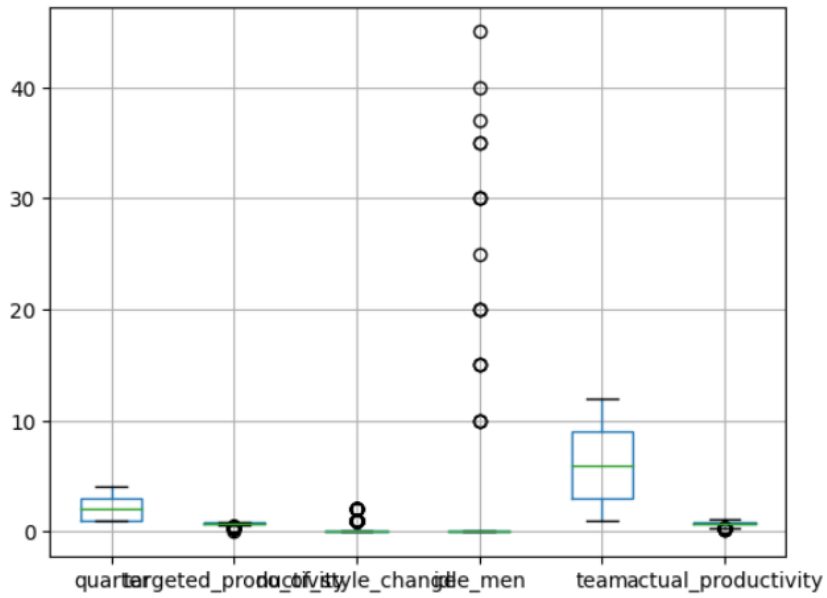


Figure 1.4

I looked at what the data would look like if I dropped those outliers and settled on a cutoff of 20. That cutoff would still include about 90% of the data. However, looking at the correlation coefficients before vs after there wasn't a significant difference so I decided to look at taking out weak predictors (Figure 1.5).

```
Correlations Before Removing Outliers in idle_men Predictor
 quarter                -0.123779
targeted_productivity    0.421594
no_of_style_change      -0.207366
idle_men                -0.181734
team                    -0.148753
actual_productivity      1.000000
Name: actual_productivity, dtype: float64
```

```
Correlations After Removing Outliers in idle_men Predictor
 quarter                -0.125209
targeted_productivity    0.421826
no_of_style_change      -0.191510
idle_men                -0.073213
team                    -0.148362
actual_productivity      1.000000
Name: actual_productivity, dtype: float64
```

Figure 1.5

Taking out Weak Predictors

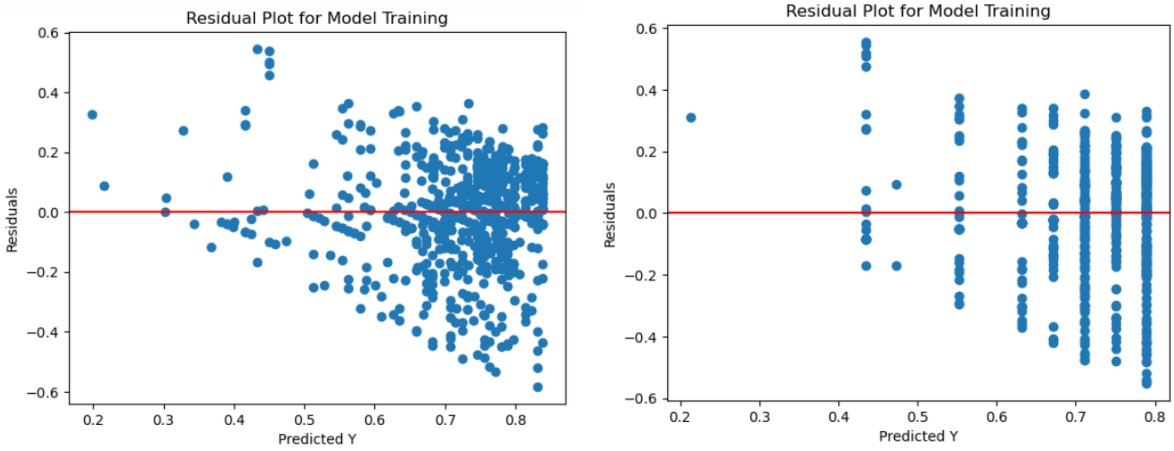
I looked at how strongly the predictors correlated with each other to see if there were any strong correlations that might lead to double-counting a predictor's influence. There was a decent positive correlation between the number of style changes, quarter, and targeted productivity (Figure 1.6). Since targeted productivity had a larger correlation, I dropped the other two predictors and re-ran my optimal model for five predictors.

	quarter	targeted_productivity	no_of_style_change	idle_men	team	actual_productivity
quarter	1.000000	-0.105497	0.249836	-0.010972	0.022426	-0.123779
targeted_productivity	-0.105497	1.000000	-0.209294	-0.053818	0.030274	0.421594
no_of_style_change	0.249836	-0.209294	1.000000	0.133632	-0.011194	-0.207366
idle_men	-0.010972	-0.053818	0.133632	1.000000	0.026974	-0.181734
team	0.022426	0.030274	-0.011194	0.026974	1.000000	-0.148753

Figure 1.6

There wasn't a significant decrease in minimum MSE, but the residual plot did look more random, which is a good sign (Figure 1.7). I went ahead and dropped all predictors except for targeted productivity and re-ran the regression. The was a small .01 increase in MSE, but the residual plot

looks less trumpet-like (Figure 1.8). Since the residual plot looks better and the MSE isn't very different from the five-predictor model, I went with the simpler model.

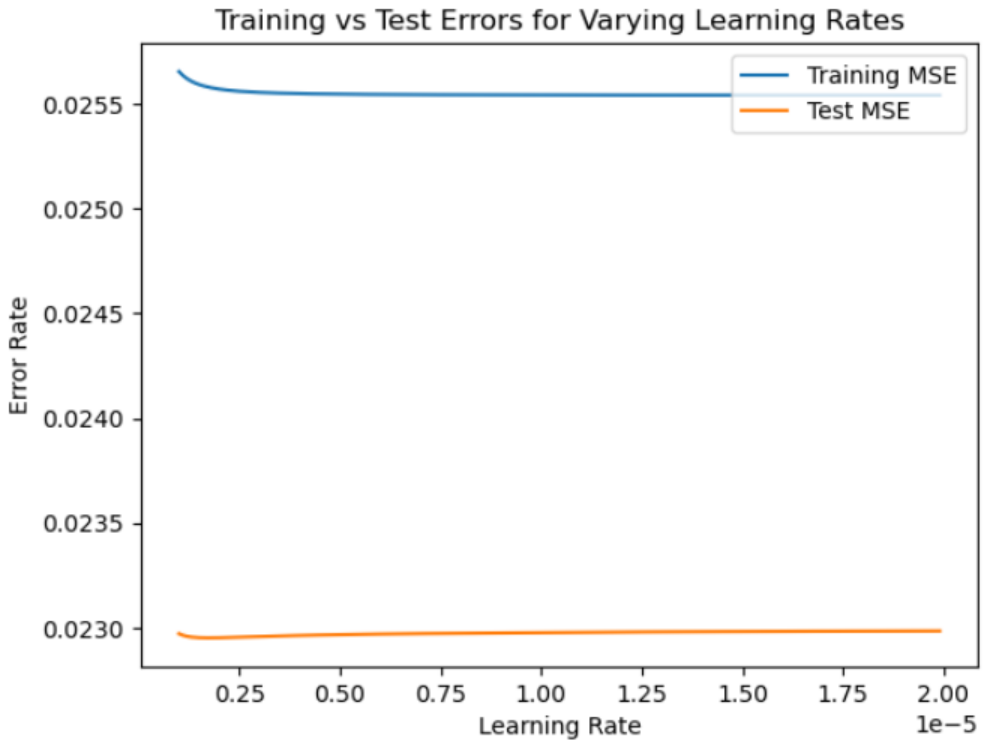


Figures 1.7 (left) and 1.8 (right)

Final Model

$$\hat{y} = 0.158 + 0.790 * x_1$$

```
****MODEL PARAMETERS****

Weights
w_ 0  :  0.3745401188473625 -------> 0.15751497340626586
w_ 1  :  0.9507143064099162 -------> 0.7902068560712477
Iterations :  100000
 Threshold :  1e-05
 Number of Predictors :  1

 Minimum Test MSE 0.02295117327268457  for Learning Rate =  1.7000000000000007e-06
```



Part Two:

Varying Learning Rates, Iterations, Threshold Values

Similar to Part One, I varied the learning rates, the number of iterations, and threshold values. The takeaways are similar here – higher number of iterations, and lower threshold values help decrease MSE (Figures 2.1 and 2.2)

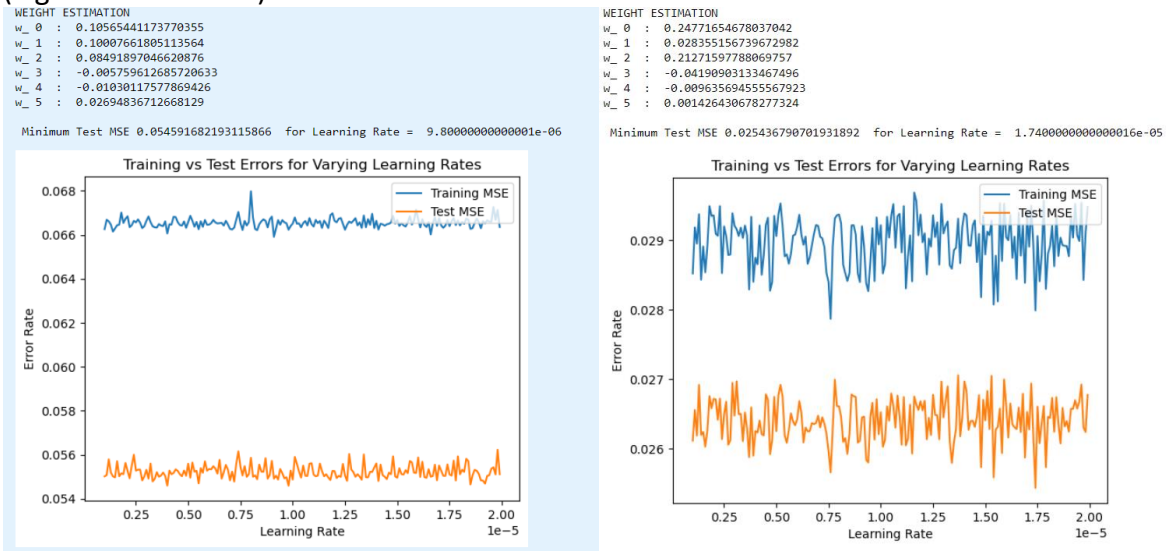


Figure 2.1 (left) and 2.2 (right)

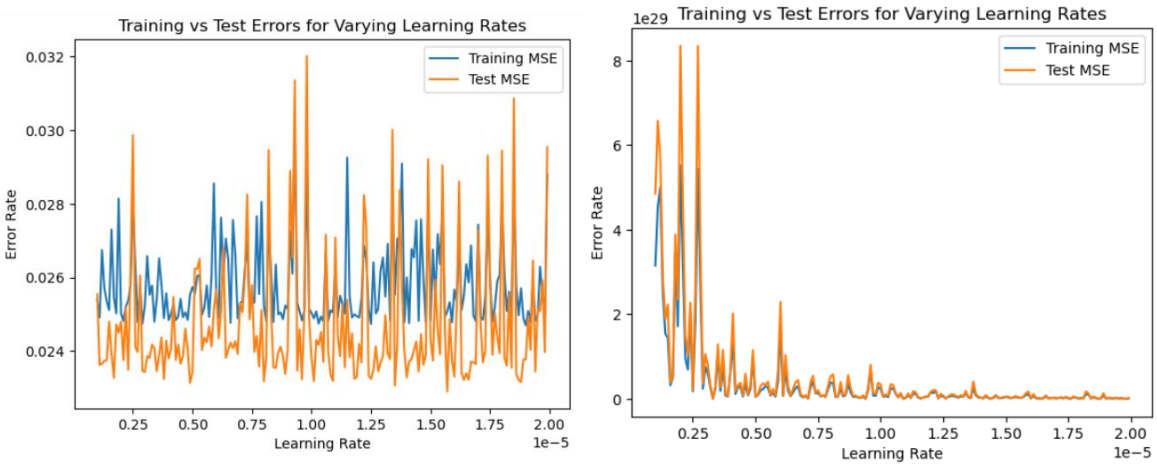
Changing Learning Rate Schedule

The SGDRegressor function has an option for the learning rate schedule (Table 2.1) and I tried all four.

Schedule	**Formula**
constant	eta = eta0
optimal	eta = 1.0 / (alpha * (t+t0))
invscaling	eta = eta0 / pow(t, power_t)
adaptive	eta = eta0 as long as the training keeps decreasing. If consecutive epochs fail to decrease the training loss by the tolerance, the current learning rate is divided by 5

Table 2.1

Both the constant and optimal settings seemed to lead to overfitting with test MSEs above training MSEs (Figures 2.3 and 2.4), but the invscaling schedule resulted in the lowest MSE without overfitting.



Figures 2.3 (left) and 2.4 (right

Regression with Fewer Predictors

Similar to Part One, I re-ran my optimal 5-predictor regression parameters but with fewer predictors. Test MSEs were similar, but for one predictor the test seemed to be better with a much higher threshold.

Final Model Equation

$$\hat{y} = 0.257 + 0.193 * x_1$$

```
WEIGHT ESTIMATION
w_ 0  :  0.25724519382555755
w_ 1  :  0.1926233020022272

Minimum Test MSE 0.03270972545195711  for Learning Rate =  6.800000000000005e-06
```