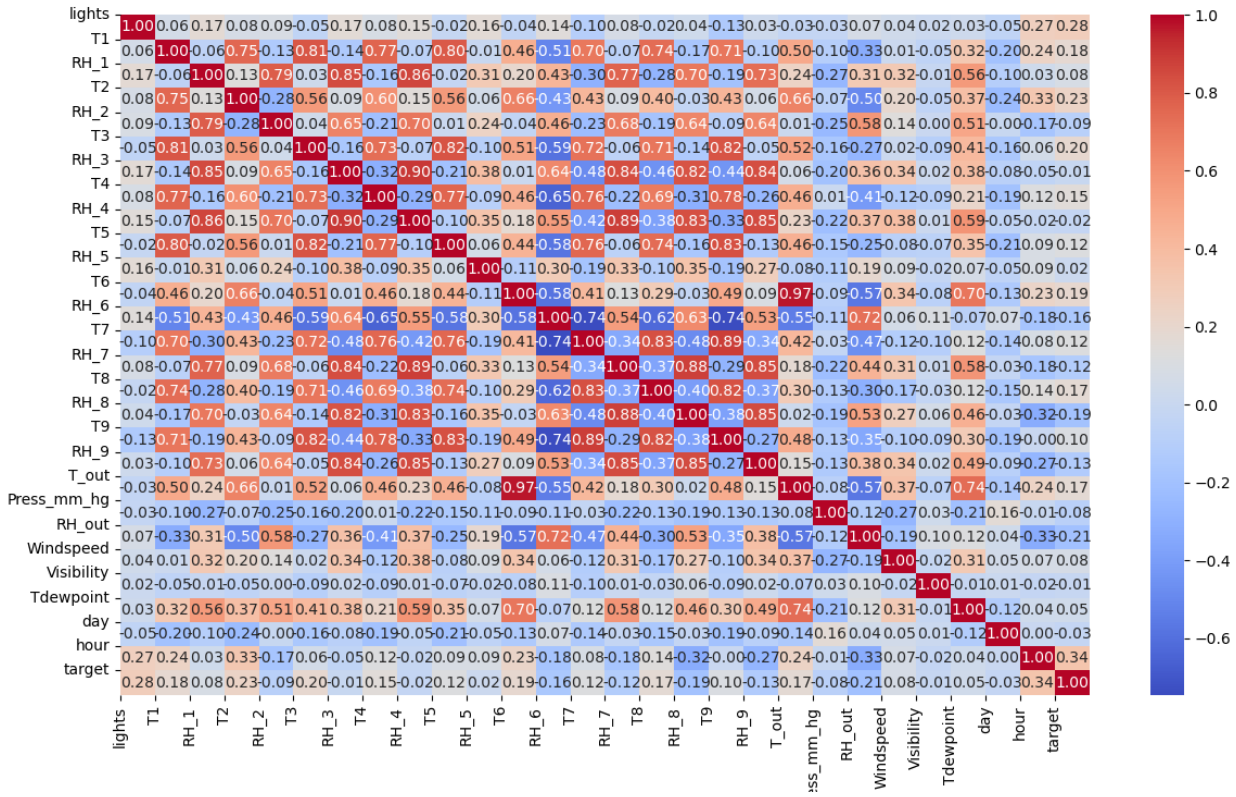
CS 334 Machine Learning HW #2

1(a.) The features that I chose to extract from the 'date' column is the hour of the day and the day of the month. The selection was made based on how the training data and test data was selected. The month of the year would have been an interesting variable to look at too, but since our training and test data are chosen based on that variable, it would be irrelevant. The same can be said for the year, although looking at yearly energy consumption rates would be interesting, both our training and test data does not reflect a large enough variety of different years.

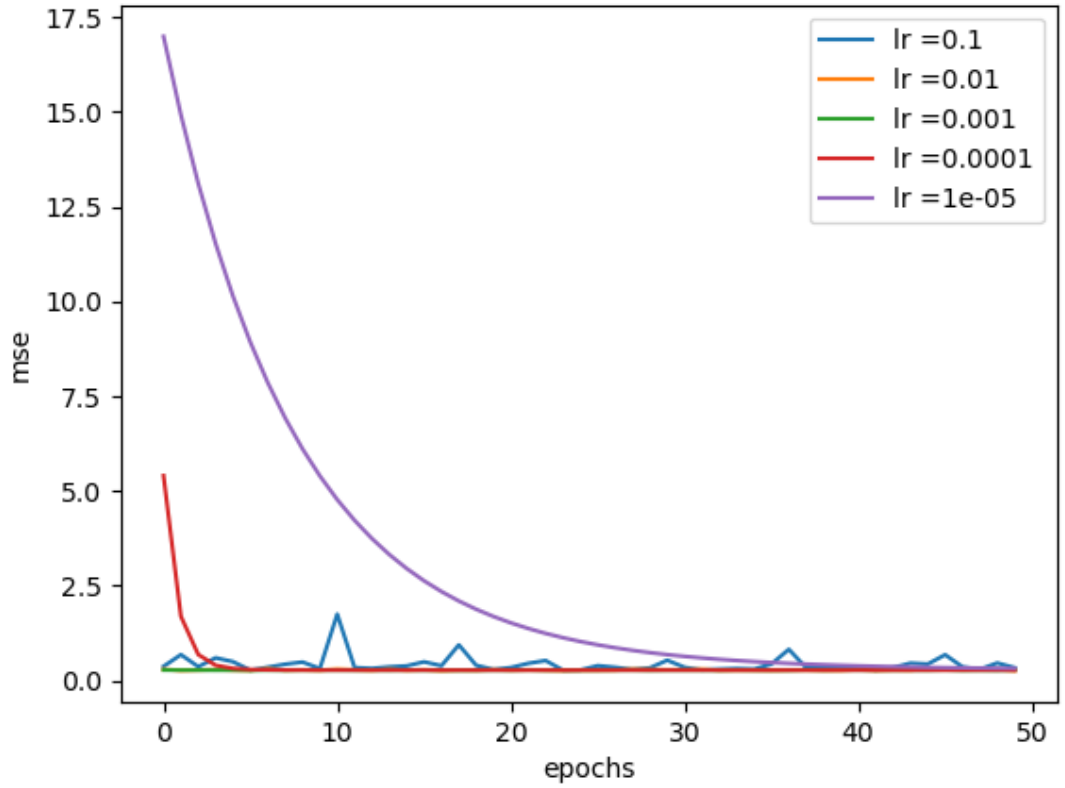
1(d.) Correlation Matrix as a HeatMap



1(e.) Looking at the heatmap above, I think the most relevant features to select are hour, T2, and lights. Although none of them are higher than 0.5, these variables have the highest correlation with the target. In addition to having a high correlation with the target, they also have relatively lower correlations to each other (with the highest one being around 0.3).

1(g.) Since we will be training a linear regression model, at the very minimum, we should normalize the data to scale. For this purpose, I have selected to use a gaussian distribution. Moreover, we can also decide to add a columns of 1's as a preprocessing step for the bias coefficients if the we know the data will only be used for linear regression, but I choose to do this in the linear regression model code instead so that data that isn't preprocessed in this regard can still be passed into the train_predict functions without issue.

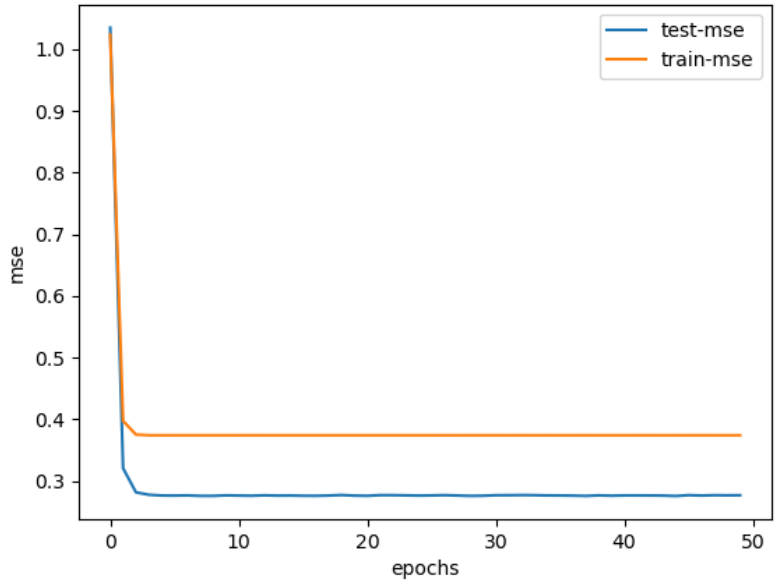
3(c.)



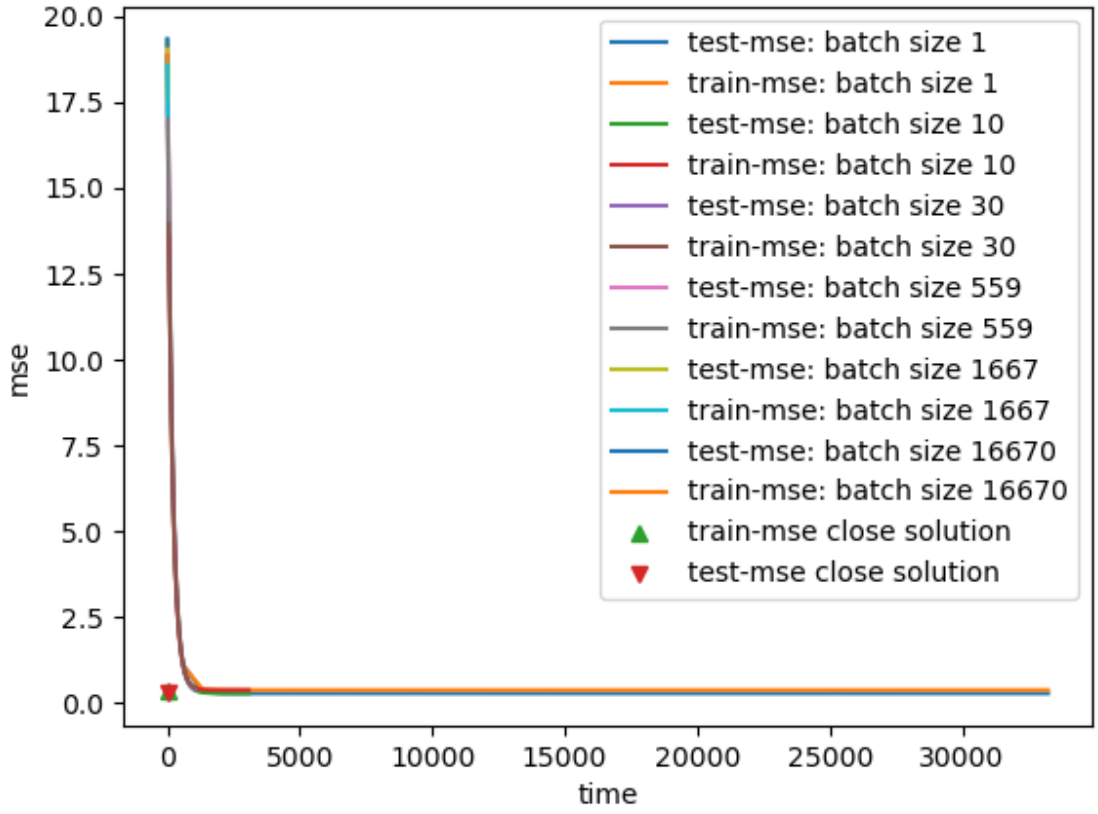
*It is hard to see on this graph, but the best learning rate is 0.0001. Objectively, this learning rate had the lowest mse out of all the learning rates yet maintaining a consistent learning rate. The 0.000001 learning rate was too slow for the number of epochs tested and the 0.001 learning rate had a high learning rate but failed to have a competitive mse value. It is interesting to note that the learning rate of 0.001 had a mse relatively close to the 0.0001 curve, so 0.001 could also be a valid candidate if the user wants to sacrifice a slight amount of accuracy for computational speed. The learning rate of 1 was removed from this graph because it diverges to infinity.

3(d.)

Here is the plot for test-mse and train-mse generated from a learning rate of 0.0001.



4(a.)



Here are the raw mse curves generated from all the conditions with respect to time with the ideal learning rate found in the previous question.

The graph is hard to see so I have included these following:

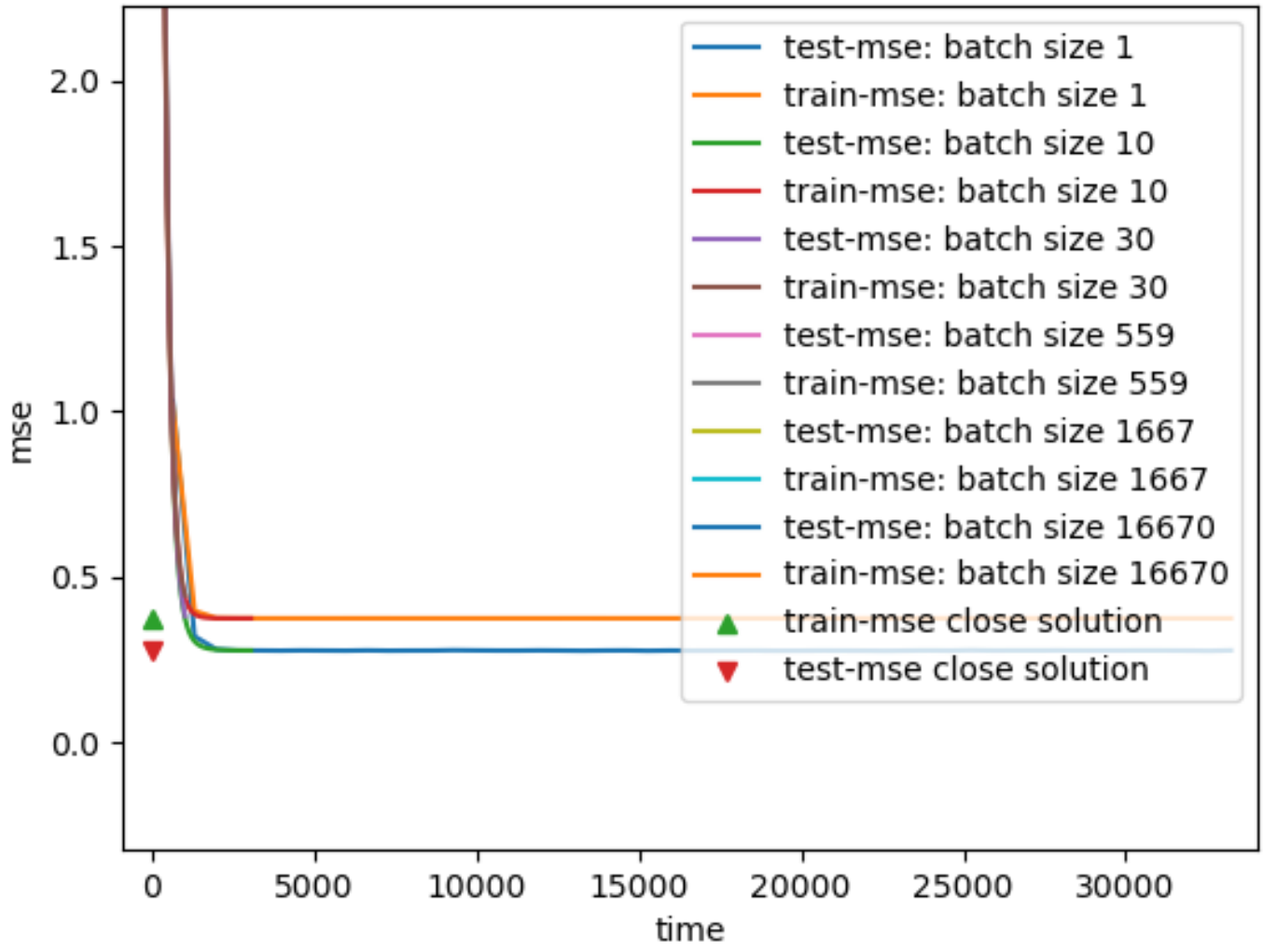


Figure 1: Zoomed onto final mse for clearer view.

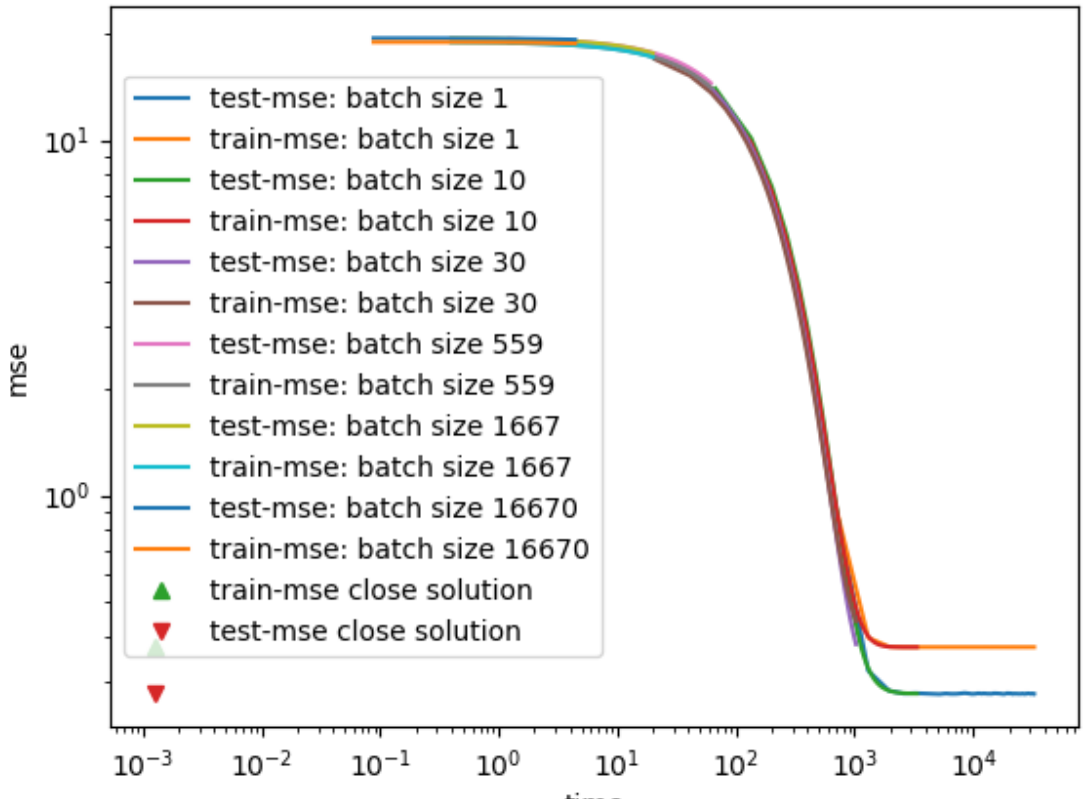


Figure 2: Scaled x (time) and y.

Observations:

*Train mse is always higher than Test mse (under same conditions)

Higher batch size ⇒ less total time.

Lower batch size ⇒ lower mse.

4(b.)

From my observations and data (tested at max 50 epochs), the close solution is preferred over the mini-batch and SGD methods. They generated relatively low loss (mse) with the shortest computational time. In terms of SGD and the mini-batches, SGD seems to have the best mse but take the longest computational time. As the batch sizes increase, there seems to be a trade off of more mse for a faster computational time.