CS 4375

ASSIGNMENT 1

Names of students in your group: Manya Bondada, David Song

Number of free late days used: 0

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.

1. Frost, Jim. "Interquartile Range (IQR): How to Find and Use It." Statistics By Jim, statisticsbyjim.com/basics/interquartile-range/#:~:text=For%20normal%20distributions%2C%20you%20can%20use%20the%20standard,distributions%2C%20and%20the%20IQR%20is%20an%20excellent%20alternative.
2. "Multivariable Gradient Descent for Linear Regression Implemented in Python | Generic Gradient Descent." *YouTube*, uploaded by Bumstriker112, 9 Mar. 2021, [www.youtube.com/watch?v=tHxTyPEgQvg&t=7s](http://www.youtube.com/watch?v=tHxTyPEgQvg&t=7s).
3. Stojiljkovic, Mirko. "Stochastic Gradient Descent Algorithm With Python and NumPy." *https://Realpython.Com/Gradient-descent-algorithm-python/*, realpython.com/gradient-descent-algorithm-python/.
4. Wang, Chi-Feng. "Calculating Gradient Descent Manually." *Towards Data Science*, 24 Oct. 2018, Calculating Gradient Descent Manually | by Chi-Feng Wang | Towards Data Science.
5. Yeh,I-Cheng. (2007). Concrete Compressive Strength. UCI Machine Learning Repository. https://doi.org/10.24432/C5PK67.

Report File

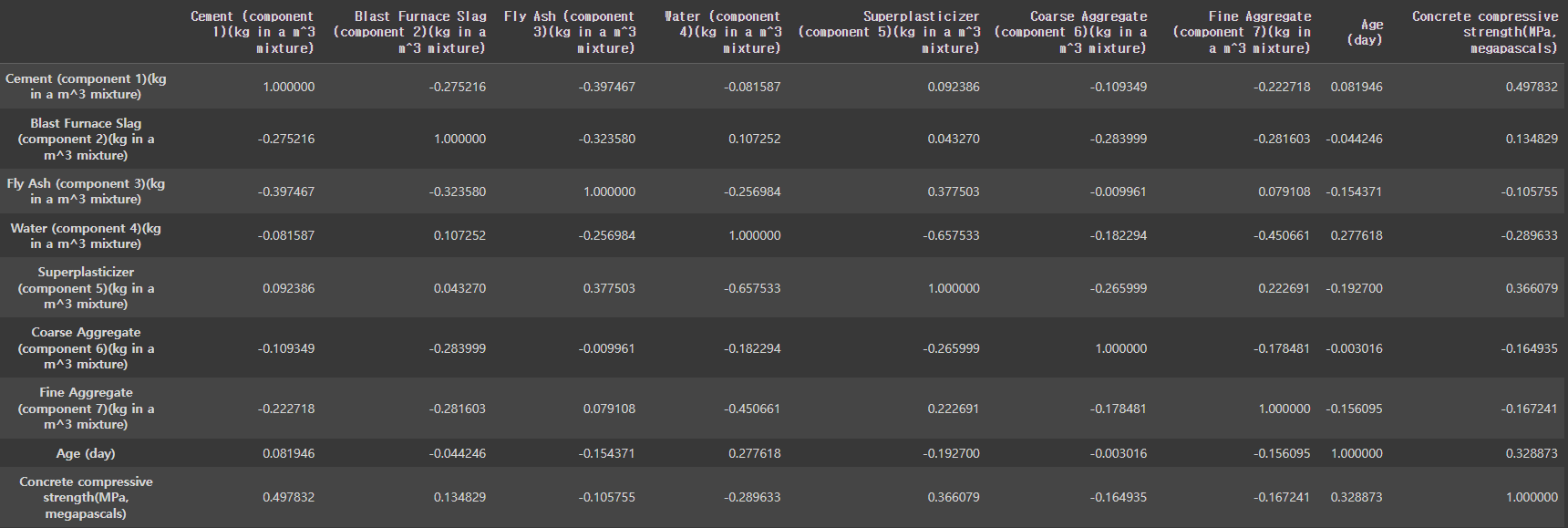
Publically hosted data: [CS4375.001\_Assignment1/Concrete\_Data.csv at main · ManyaBondada/CS4375.001\_Assignment1 (github.com)](https://github.com/ManyaBondada/CS4375.001_Assignment1/blob/main/Concrete_Data.csv)

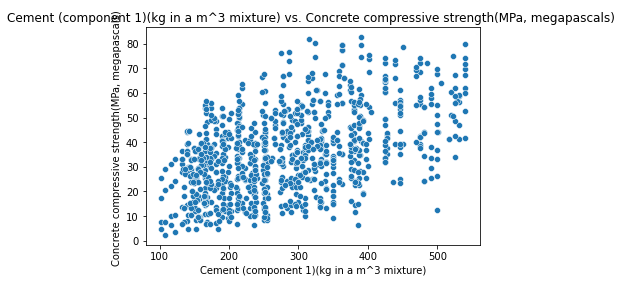
Link to Assignment1\_Part1.ipynb: <https://colab.research.google.com/drive/1NoHUB8Wh6fcLbU8hcCpsHd3V_JmGc4ny?usp=sharing>

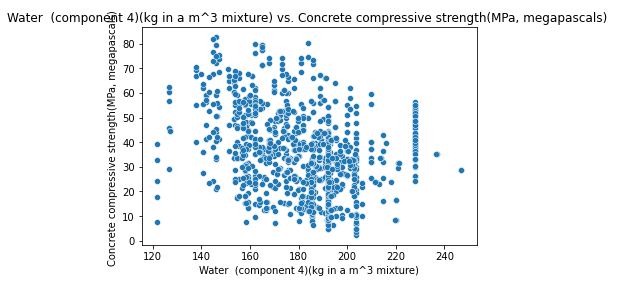
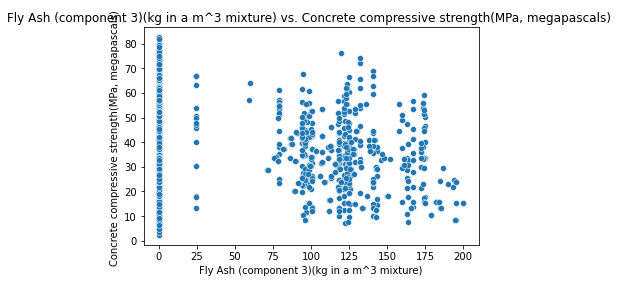
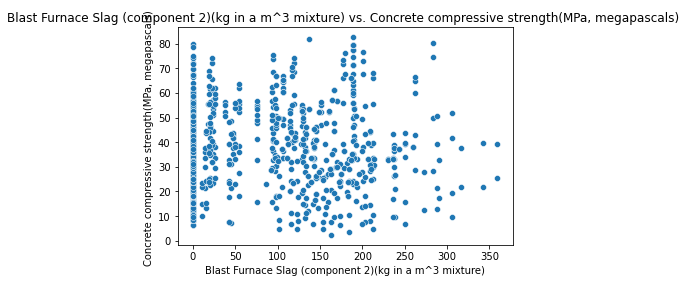
Link to Assignment1\_Part2.ipynb: <https://colab.research.google.com/drive/1VuAQuRw1B6NoYoB-Kh11hGSdgx13nX2p?usp=sharing>

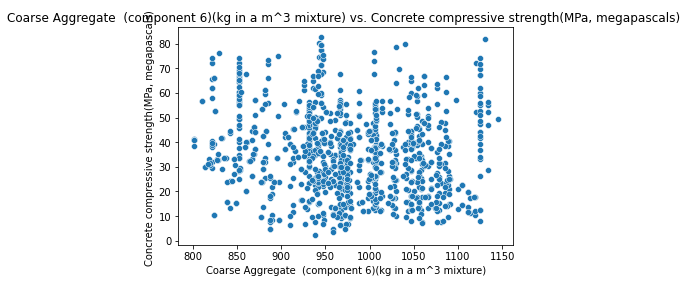
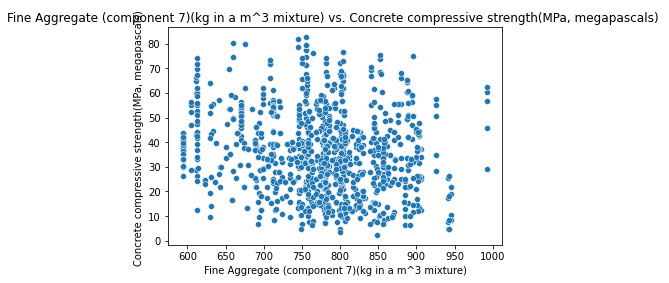
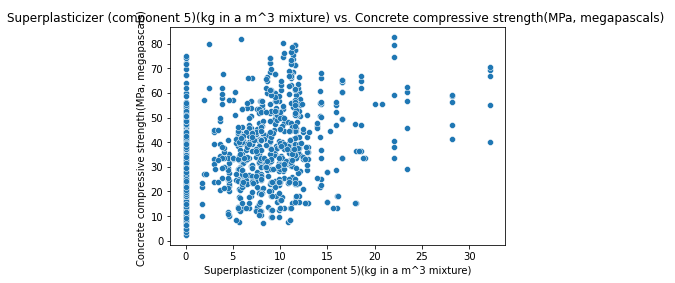
Plots:

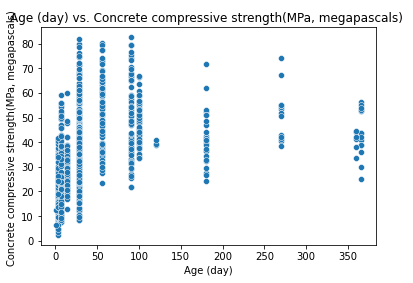
Correlation between attributes and target value:



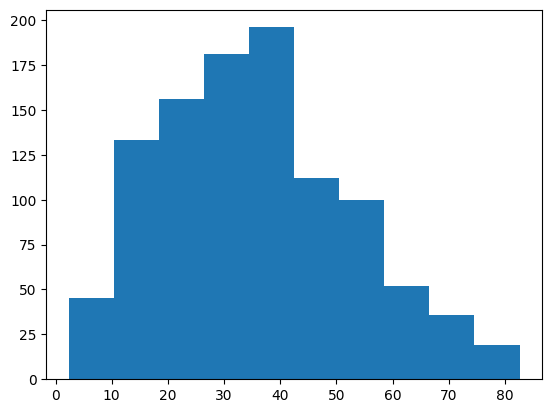
Attributes plotted against target value:



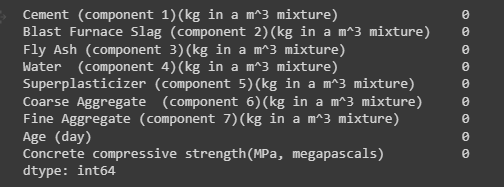




Distribution of Y variable:



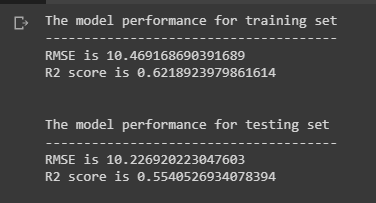
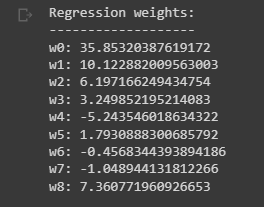
Data contains no null values



Part 1

Trial 1:

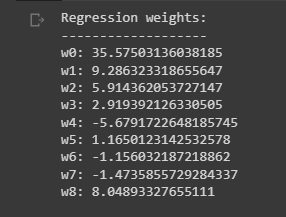
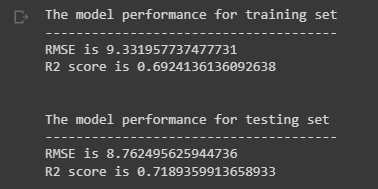
* Data Split: 80% training 20% testing
* Iteration #: 100, Learning rate: 0.1, Starting weights: Array size 9 of 0s
* Results



* The model did not perform as well on the testing set as it did on the training set. There are several factors that might have affected this, such as outliers in the dataset, number of iterations, etc. We plotted the distribution of the Y variable and decided to calculate the IQR of each feature to remove rows with outliers.

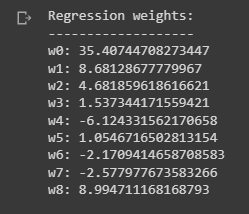
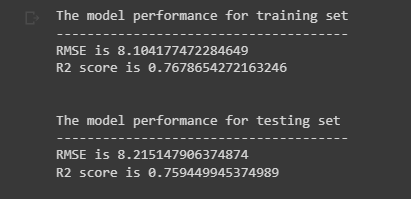
Trial 2:

* Data Split: 80% training 20% testing
* Iteration #: 100, Learning rate: 0.1, Starting weights: Array size 9 of 0s
* Outlier threshold for preprocessed data: 3.0
* Results

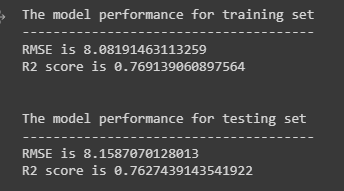
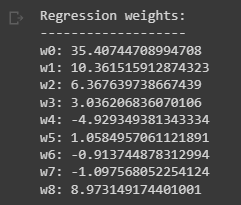


* Adding in the outlier removal function greatly increased the performance of the model on the testing data. We decided to continue hypertuning the parameters to get the RMSE as close to 0 as possible.

Trial 3:

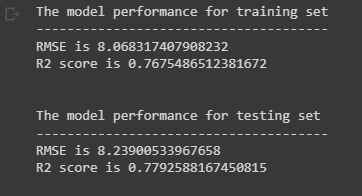
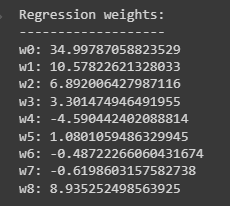
* Data Split: 80% training 20% testing
* Iteration #: 100, Learning rate: 0.1, Starting weights: Array size 9 of 0s
* Outlier threshold for preprocessed data: 1.5
* Results:
* Decreasing the outlier threshold improved the performance of the model. We continued hypertuning parameters to decrease the RMSE.

Trial 4:

* Data Split: 80% training 20% testing
* Iteration #: 10000, Learning rate: 0.1, Starting weights: Array size 9 of 0s
* Outlier threshold for preprocessed data: 1.5
* Results

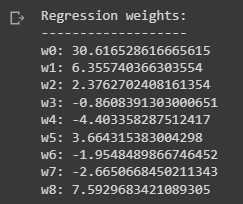
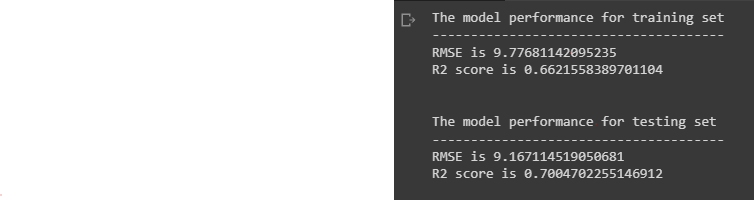
* Increasing the number of iterations increased the performance of the model on the training set

Trial 5:

* Data Split: 90% training 10% testing
* Iteration #: 10000, Learning rate: 0.1, Starting weights: Array size 9 of 0s
* Outlier threshold for preprocessed data: 1.5
* Results
* Changing the training:testing ratio improved the R^2 metric of the test data, but not by a significant amount. The RMSE increased slightly, so we will keep the data split 80% 20%

Trial 6:

* Data Split: 80% training 20% testing
* Iteration #: 10000, Learning rate: 0.0001, Starting weights: Array size 9 of 0s
* Outlier threshold for preprocessed data: 1.5
* Results



* Changing the learning rate decreased the performance of the model, so we decided to leave it at 0.1

Best parameters:

From the trials we conducted, we concluded that our best set of parameters are:

* Data Split: 80% training 20% testing
* Iteration #: 10000, Learning rate: 0.1, Starting weights: Array size 9 of 0s
* Outlier threshold for preprocessed data: 1.5

Trial 4 seemed to produce the most consistent R^2 values and the lowest RMSE errors. Overall, the biggest improvement we made was not in our model, but in the way we preprocessed the entire dataset by removing outliers.

Are you satisfied that you have found the best solution?

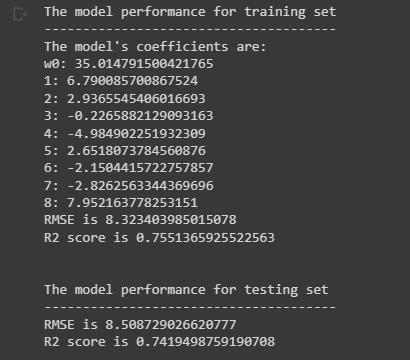
We most likely did not find the best solution solely because we chose the best parameters after conducting 6 trials of training and testing. In reality, there are likely thousands of combinations of parameters and better methods for cleaning the data, so the probability of finding a better performing model is extremely high.

Part 2

We preprocessed our data the same way as part 1, so in this part we focused on tuning the parameters to achieve the best model performance.

Trial 1:

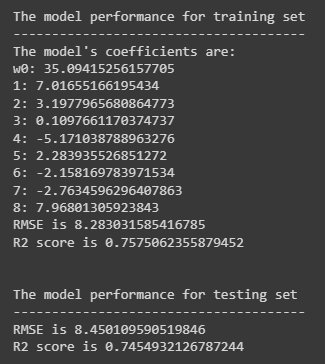
* Data Split: 80% training 20% testing
* Iteration #: 100, Learning rate: 0.1, Initial learning rate: 0.001, tolerance : 0.000001
* Outlier threshold for preprocessed data: 1.5
* Results



* Because we already preprocessed data like part 1, our evaluation metrics ended up being consistent on the first try. We continued to hypertune the parameters to decrease RMSE.

Trial 2:

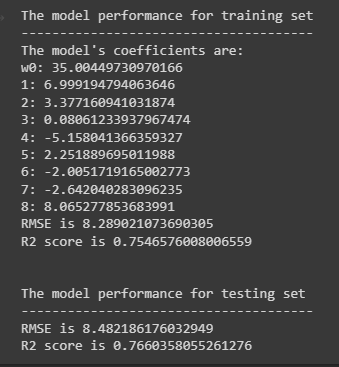
* Data Split: 80% training 20% testing
* Iteration #: 10000000, Learning rate: 0.1, Initial learning rate: 0.001, tolerance : 0.000001
* Outlier threshold for preprocessed data: 1.5
* Results



* RMSE for both training and testing data has slightly decreased.

Trial 3:

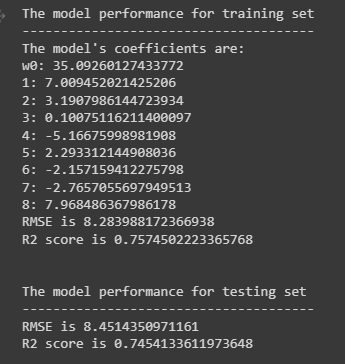
* Data Split: 90% training 10% testing
* Iteration #: 10000000, Learning rate: 0.1, Initial learning rate: 0.001, tolerance : 0.000001
* Outlier threshold for preprocessed data: 1.5
* Results



* Changing the data split did not significantly impact the evaluation metrics, so we continued using a 80% 20% training-testing split in future trials

Trial 4:

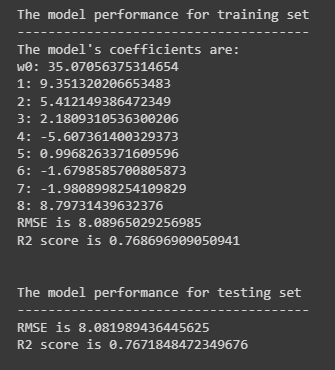
* Data Split: 80% training 20% testing
* Iteration #: 10000000, Learning rate: 0.1, Initial learning rate: 0.001, tolerance : 0.0001
* Outlier threshold for preprocessed data: 1.5
* Results



* Changing the tolerance threshold did not significantly impact the model’s performance either.

Trial 5:

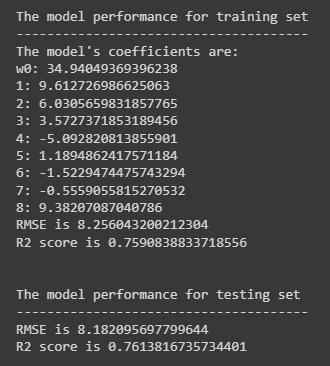
* Data Split: 80% training 20% testing
* Iteration #: 10000000, Learning rate: 0.001, Initial learning rate: 0.001, tolerance : 0.0001
* Outlier threshold for preprocessed data: 1.5
* Results



* Changing the learning rate decreased the RMSE for both training and testing data. It also increased the R2 score

Trial 6:

* Data Split: 80% training 20% testing
* Iteration #: 10000000, Learning rate: 0.001, Initial learning rate: 0.001, tolerance : 0.0001
* Outlier threshold for preprocessed data: 1.5
* Results



* Increasing the model’s initial learning rate decreased the model’s fit to the training data.

Best parameters:

From the trials we conducted, we concluded that our best set of parameters are:

* Data Split: 80% training 20% testing
* Iteration #: 10000000, Learning rate: 0.001, Initial learning rate: 0.001, tolerance : 0.0001
* Outlier threshold for preprocessed data: 1.5

Trial 5 seemed to produce the most consistent R^2 values and the lowest RMSE errors for this part.

Are you satisfied that the package has found the best solution? How can you check?

We think the package found the best weights for the trials that were conducted and the parameters we selected. We checked this by comparing RMSE values and R2 values among trials, ultimately selecting the parameters that produced the lowest RMSE and highest R2. However, in general, since there are so many combinations and parameters and ways to preprocess data, it is very likely that a better solution exists.

Part 1 and 2 best parameter comparisons:

Overall, the RMSE and R2 metrics were very similar for both parts, though the RMSE was slightly lower for the sklearn regression we applied on our testing data in part 2. The weights for both are also similar, only slightly varying by 0.5-1 for each coefficient. However, this variance did not significantly affect the results of our model.