CSE 514A Data Mining: Assignment 1

Danni Beaulieu (ID 497567)

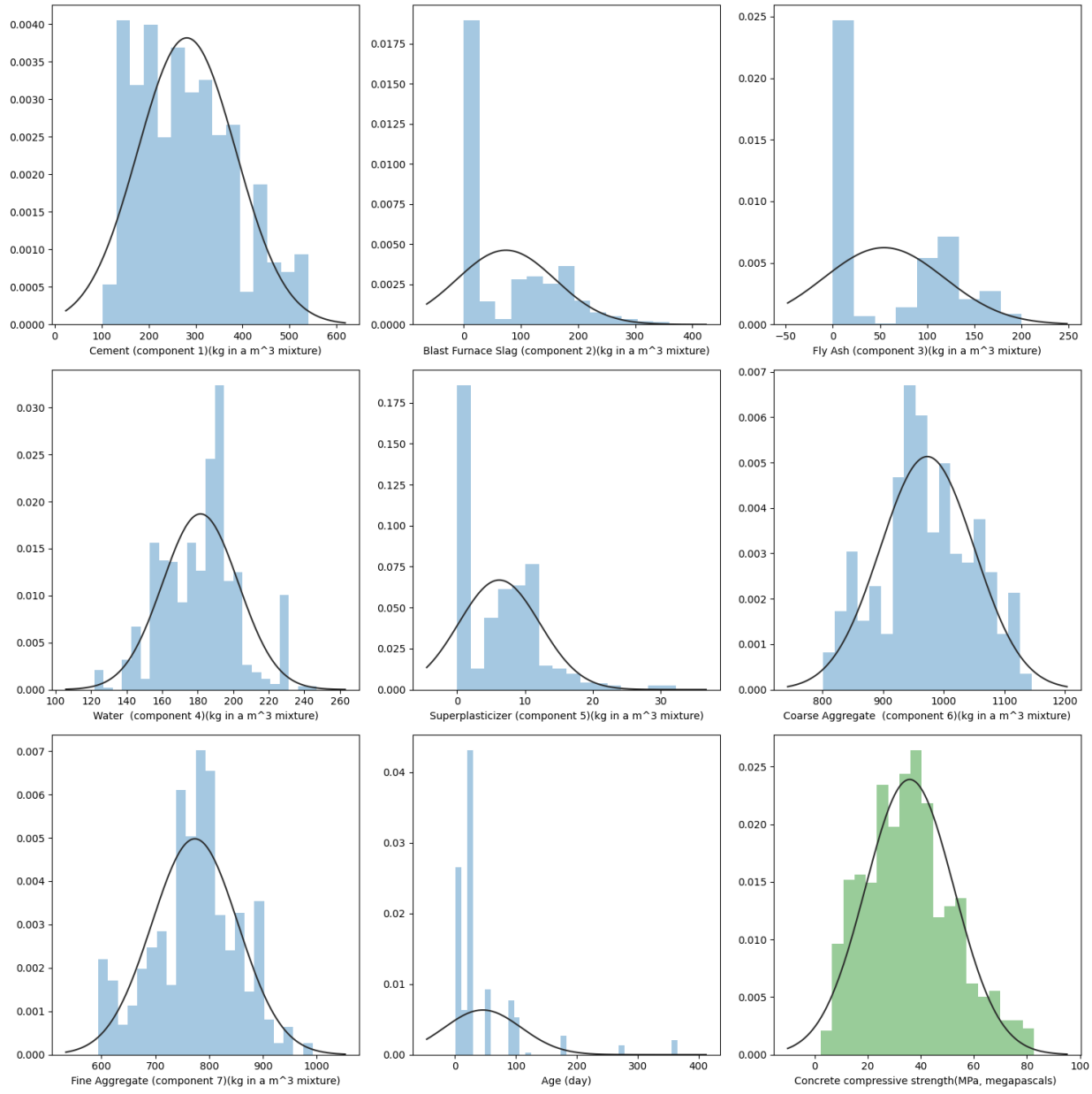
Introduction

In the UCI “Concrete Compressive Strength” data set, there are 8 input features which presumably have some bearing on the final observed compressive strength of concrete. This project seeks to identify which of these predictor variables have more or less influence on this response. This will be discovered through usage of a gradient descent algorithm after preprocessing the data via normalization. The real world application of this analysis with the most obvious significance is using it to determine how to make concrete with the most compressive strength possible. This information could be utilized in any number of construction designs, from buildings to bridges and support columns. Essentially, this would assist any venture in which concrete is used as a material and the result must withstand high compression.

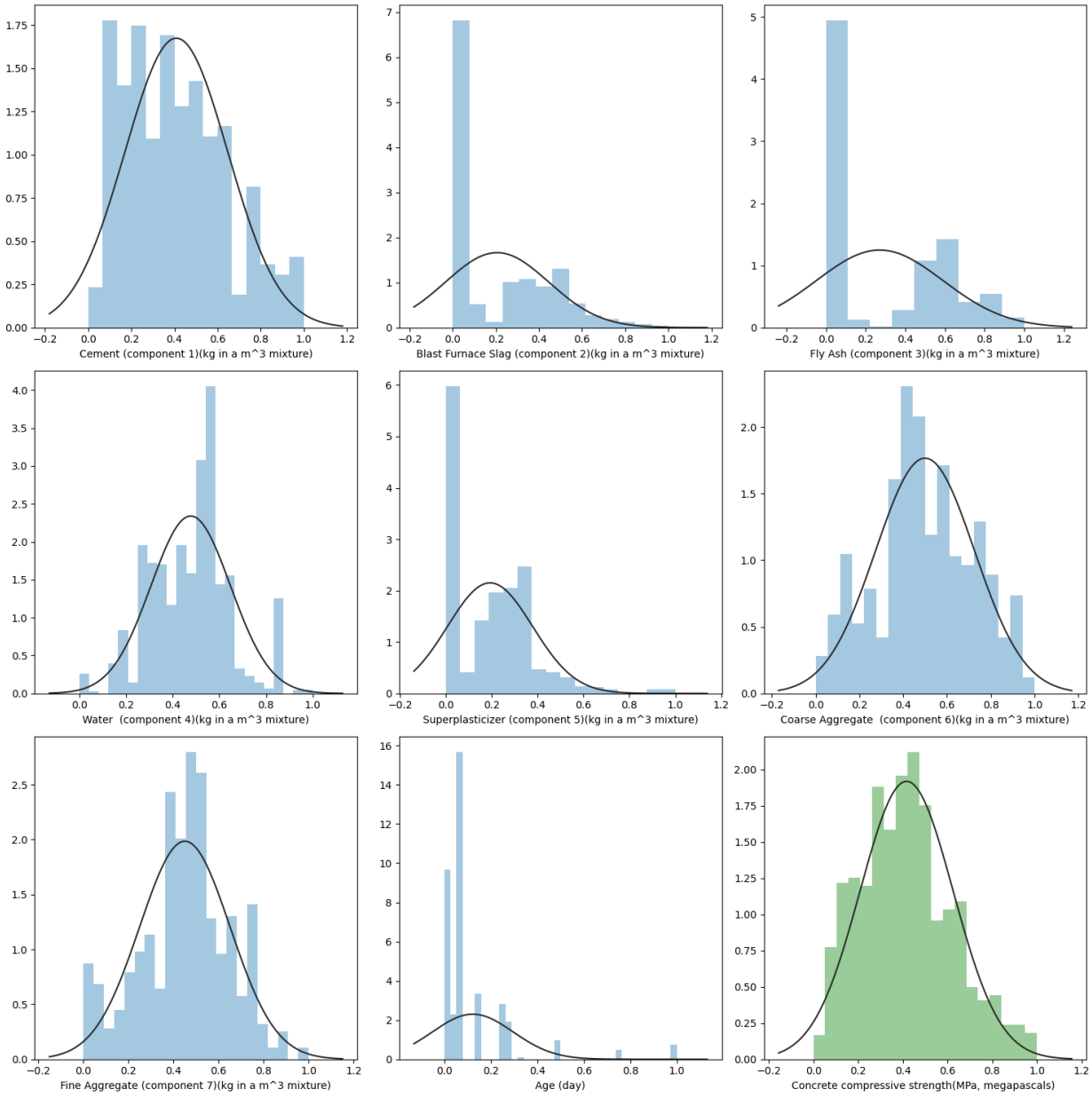
Before preprocessing the data, I looked at the distributions of observed values for each feature and the response values. When deciding between standardization or normalization as a preprocessing step, it is critical to identify whether the data set likely adheres to a Gaussian, or normal, distribution. Since standardization rescales data to align with a standard normal distribution, that is, mean of zero and standard deviation of one, this really only makes sense for data already somewhat of this type. While the label, “compressive strength” does seem close to Gaussian, the underlying data for each feature does not necessarily follow this trend. To be conservative, I then chose to normalize instead of standardize the data. The drawback to this is that normalization may be more affected by outliers than standardization.

This project uses full gradient descent rather than stochastic gradient descent. Since the dataset is relatively small, using only 900 observations for training, the lower variance of full gradient descent as compared with stochastic gradient descent are worth the computational expense of processing the full dataset on each iteration. If the algorithm appears to get stuck in a local minima or shows poor generalization, switching to stochastic gradient descent may be beneficial. Initially, experiments were run employing an adaptive learning rate in which the input step size was multiplied by 1.01 if loss is less than the previous loss, which indicates we are getting closer, otherwise it multiplied the input step size by 0.5 when we are still far from our target. A few other different learning rates were run to see which is more suitable to the problem space. Due to large disparities in magnitudes between the preprocessed and non-preprocessed data, a simple constant learning rate system was chosen in tandem with cross validation on training data to choose which learning rate was appropriate for each feature space and model. Similarly for the stopping criterion(s), after some exploration, the max number of iterations was set to 10,000 and an exit condition tolerance of 1e-02 was set against which the method tests the infinity norm of the gradient.

Raw Distribution Plots Against Gaussian



Normalized Distribution Plots Against Gaussian



Pseudocode

```
# w: d length weight vector
# xTr:dxn matrix (each column is an input vector)
# yTr:1xn matrix (each entry is a label)
# d: dimensions (8 features and 1 for bias)
# n: number of samples
```

Gradient Descent

```
def grdescent (f, w0, step, max=10000, tolerance=1e-2):
      w = w0
      for t in max:
            gradient = f(w)
            w = w - step * gradient
            if infinity norm < tolerance:
                  break
      return w
```

MSE Regression

```
def msereg (w, xTr, yTr):
      XT = xTr.transpose()
      XTw = XT.dot(w)
      XTwMinusy = XTw - yTr.transpose()
      XTwMinusyT = XTwMinusy.transpose()
      gradient = 2 * xTr.dot(XTwMinusy) / yTr.shape[1]
      return gradient
```

MAE Regression

```
def maereg (w,xTr,yTr):
    XT = xTr.transpose()
    XTw = XT.dot(w)
    YMinusXTw = yTr.transpose() - XTw
    YMinusXTwSign = np.sign(YMinusXTw)
    gradient = -1 * xTr.dot(YMinusXTwSign) / yTr.shape[1]
    return gradient
```

Ridge Regression

```
def ridge(w,xTr,yTr,lambdaa):
    XT = xTr.transpose()
    XTw = XT.dot(w)
    XTwMinusy = XTw - yTr.transpose()
    g_loss = 2 * xTr.dot(XTwMinusy) / yTr.shape[1]
    g_reg = 2 * lambdaa * w / yTr.shape[1]
    gradient = g_loss + g_reg
    return gradient
```

Implementation Details

The pseudocode for gradient descent above (`grdescent`) allows flexibility in the choice of function I use so long as it returns the gradient. For linear regression, we have our MSE loss represented by: $(1/n)(X^Tw-y)^T(X^Tw-y)$, where X is a matrix, w is our weights, y is our labels, and n is our number of samples. Here, we can add a regularizer, say the L2 Ridge, to get a new loss equation: $(1/n)(X^Tw-y)^T(X^Tw-y) + \lambda w^Tw$. Without the regularizer, this yields a gradient of: $(2/n)X(X^Tw-y)$. With the regularizer we have a gradient of: $(2/n)X(X^Tw-y) + 2\lambda w$. For convenience, to account for the bias term, we augment the input matrix or vector with an additional feature consisting of all values equal to one. Likewise we create an additional dimension in the weights vector; since this "weight" is simply multiplied by one it actually gets added in correctly as our bias term. For MAE, our loss is represented by $(1/n)\Sigma|y-X^Tw|$ and the corresponding gradient is $(1/n)X[(y-Xw)/|y-Xw|]$, so that if $Xw > y$, we have $-(1/n)X$ and if $Xw < y$ we have $(1/n)X$. Since the gradient is not defined where Xw is equal to y, we must choose how the algorithm proceeds. Since $[a-b/|a-b|] = sign(a-b)$, I have elected to use the sign function and to return gradient as zero in these cases.

Experiment Details

For these experiments, I create a preprocessed and non-preprocessed copy of the data. I sample the data without replacement to create splits of 900 training samples and 130 test samples. I then do a K-fold cross-validation to find the best learning rate for the MSE and MAE models and the best lambda for the Ridge regularizer. I train the models with these hyperparameters and then see how they perform with the test data, recording the loss and R-squared values for both train and test. Finally, I save plots of the model's performance against the training data. Different possible learning rates were used for preprocessed versus non-preprocessed data to avoid scaling issues which resulted in an exploding gradient for the non-preprocessed data. This makes sense because normalization rescales values to be between 0 and 1 whereas the original data had a maximum value about 5 orders of magnitude higher. The possible values used for the non-preprocessed data were [1e-07, 1e-08, 1e-09, 1e-10]. The possible values used for the preprocessed data were [1e-02, 1e-03, 1e-04, 1e-05].

As a note, I ran the full project several times, and I have maintained a project directory with the results for each trial. For our purposes, however, we wish to analyze only one full run of the experiment, so I have chosen a representative trial to analyze. Also note, table values have mostly been truncated to 3 decimals, however more precise data is available in the raw output files. In the raw output files, multivariate models are labeled as "feature: -1" while the univariate models are labeled numerically beginning with "feature: 0" as the first feature (Age).

The full project can be found here: https://github.com/danni-beaulieu/data-mining-hw-1
The trial is located here: https://github.com/danni-beaulieu/data-mining-hw-1/tree/main/trial-2

Results

Variance Explained (R-Squared)

Train: Not Preprocessed

Feature	All	1	2	3	4	5	6	7	8
MSE	0.565	0.165	-2.275	-2.749	-0.201	-2.747	-0.081	-0.124	-2.175
Ridge	-0.007	0.165	-2.275	-2.749	-0.201	-2.747	-0.081	-0.124	-2.175
MAE	-21719.341	-29.959	-10.598	-6.738	-13.645	-3.264	-0.081	-0.125	-5.250

Test: Not Preprocessed

Feature	All	1	2	3	4	5	6	7	8
MSE	0.540	0.260	-2.122	-3.049	-0.211	-2.725	-0.108	-0.087	-2.152
Ridge	-0.050	0.260	-2.122	-3.049	-0.211	-2.725	-0.108	-0.087	-2.152
MAE	-23144.652	-32.402	-10.404	-7.396	-16.205	-3.223	-0.105	-0.084	-5.276

Train: Preprocessed

Feature	All	1	2	3	4	5	6	7	8
MSE	0.512	0.244	0.005	0.003	0.051	0.108	0.014	0.003	0.083
Ridge	0.509	0.244	0.005	0.003	0.051	0.108	0.014	0.003	0.083
MAE	0.591	0.243	0.009	0.002	0.047	0.113	0.025	0.018	0.067

Test: Preprocessed

Feature	All	1	2	3	4	5	6	7	8
MSE	0.581	0.237	0.026	0.001	0.056	0.184	-0.0004	0.001	0.170
Ridge	0.591	0.237	0.026	0.001	0.056	0.184	-0.0004	0.001	0.170
MAE	0.646	0.237	0.009	-0.005	0.044	0.176	-0.018	0.027	0.149

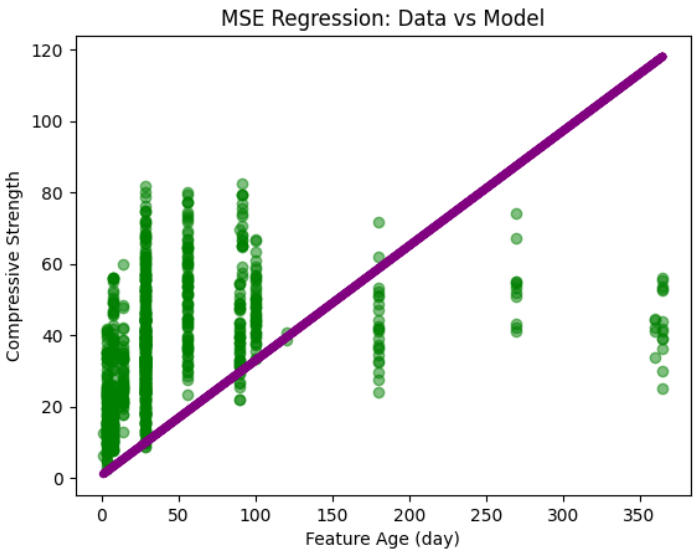
Hyperparameters (Not Preprocessed): Learning Rate and Lambda

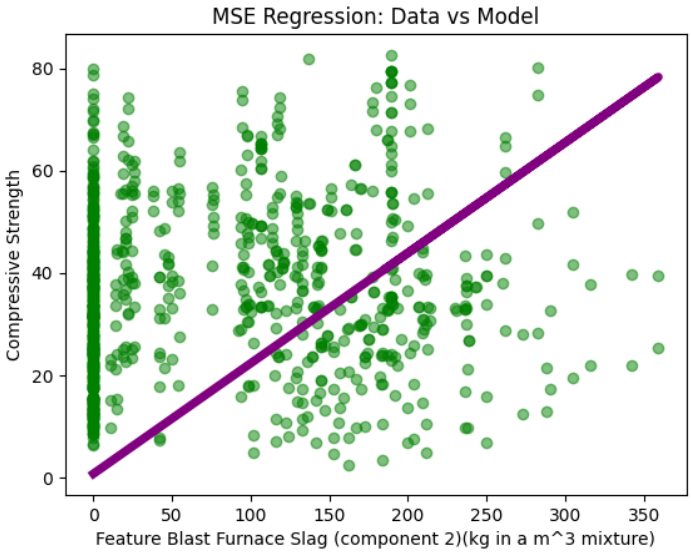
Feature	All	1	2	3	4	5	6	7	8
MSE (Eta)	1e-07	1e-07	1e-07	1e-07	1e-08	1e-07	1e-07	1e-07	1e-07
Ridge (Lambda)	1e-07	1e-09	1e-09	1e-10	1e-07	1e-10	1e-08	1e-09	1e-10
MAE (Eta)	1e-07	1e-07	1e-07	1e-07	1e-07	1e-07	1e-07	1e-07	1e-07

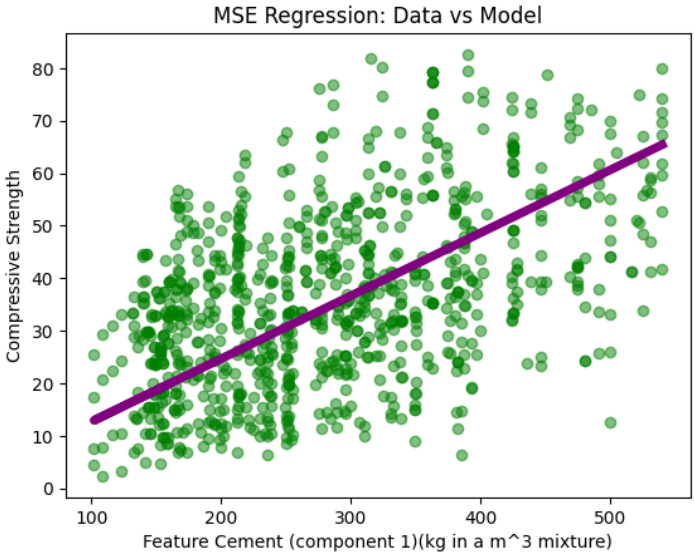
Hyperparameters (Preprocessed): Learning Rate and Lambda

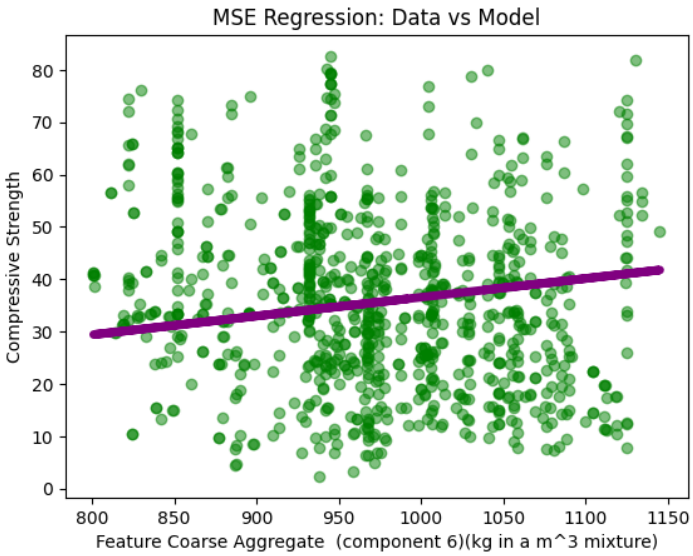
Feature	All	1	2	3	4	5	6	7	8
MSE (Eta)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.001
Ridge (Lambda)	1e-05	0.01	0.01	0.01	1e-05	0.01	1e-05	1e-05	0.01
MAE (Eta)	0.01	0.0001	0.01	0.01	0.01	0.01	0.01	0.01	0.01

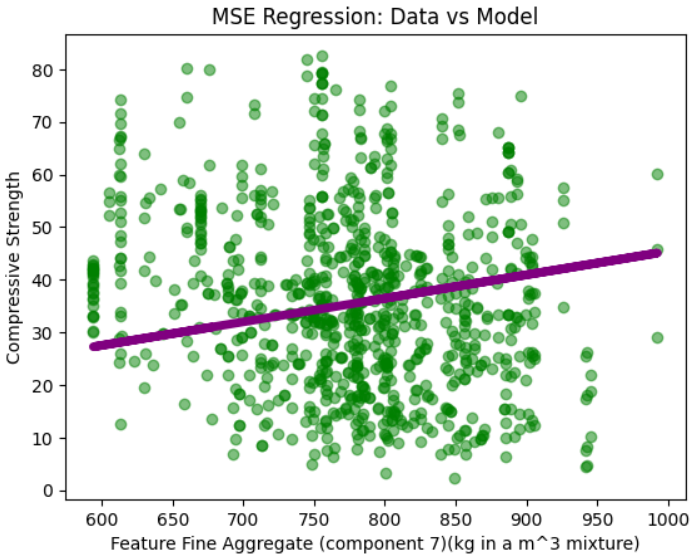
MSE: Non-Preprocessed Training Data and Model Predictions

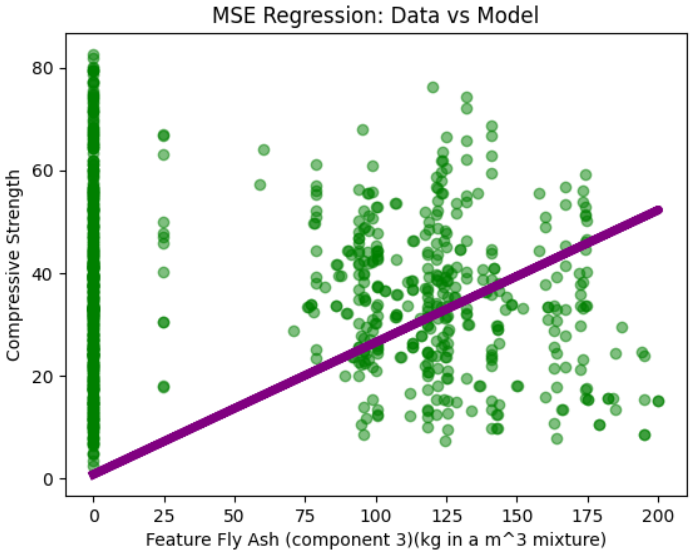


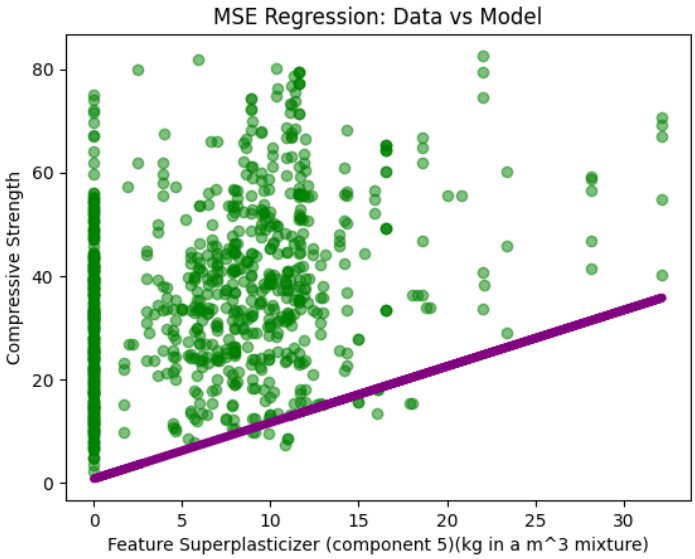


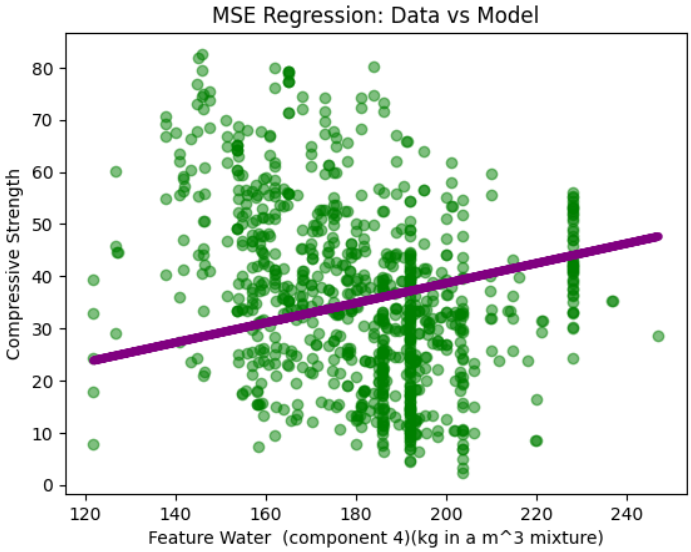




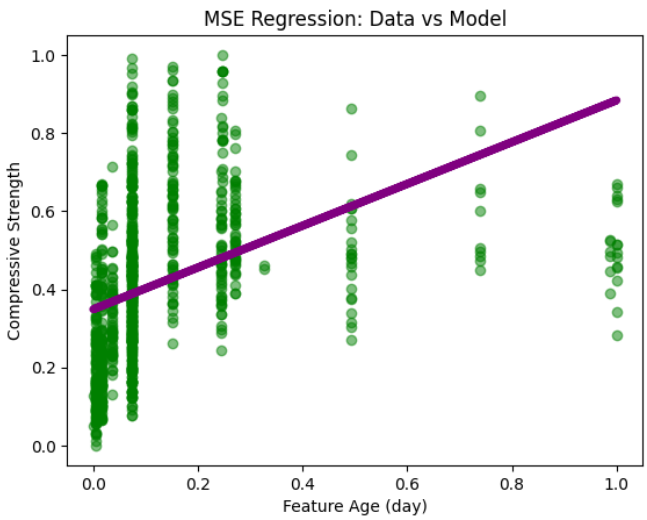

MSE Regression: Data vs Model
Compressive Strength
Feature Fine Aggregate (component 7)(kg in a m^3 mixture)

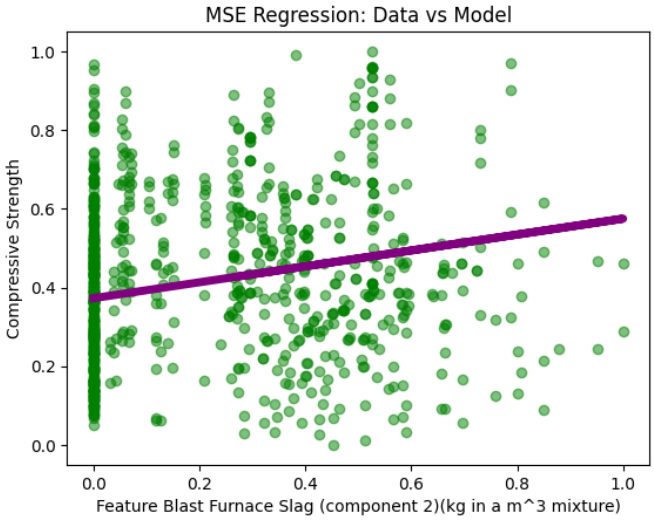

MSE Regression: Data vs Model
Compressive Strength
Feature Fly Ash (component 3)(kg in a m^3 mixture)

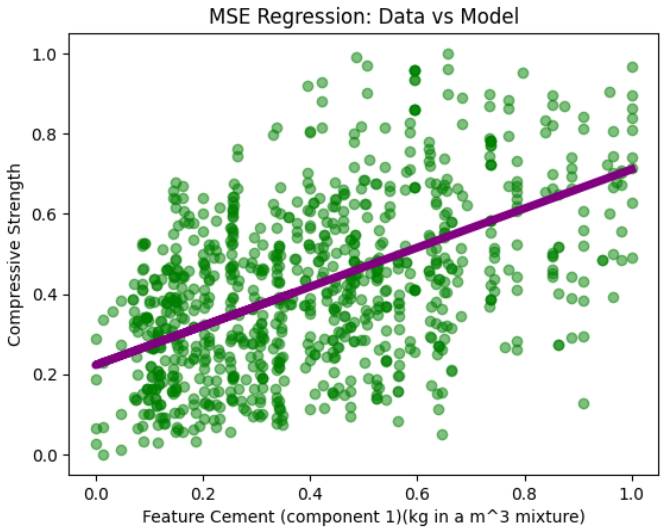

MSE Regression: Data vs Model
Compressive Strength
Feature Superplasticizer (component 5)(kg in a m^3 mixture)

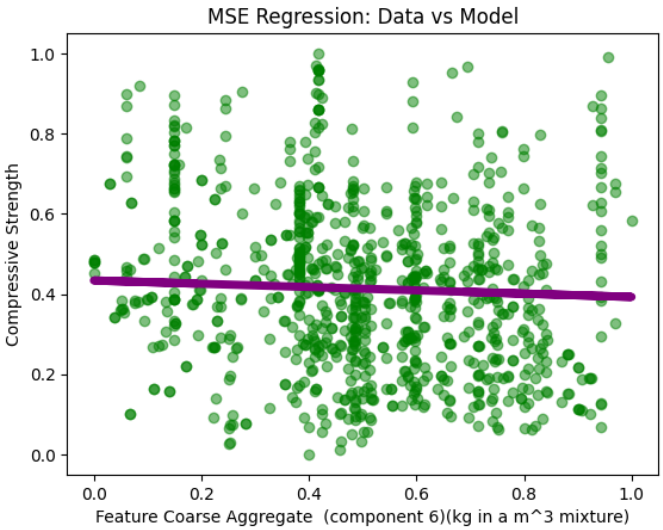

MSE Regression: Data vs Model
Compressive Strength
Feature Water (component 4)(kg in a m^3 mixture)

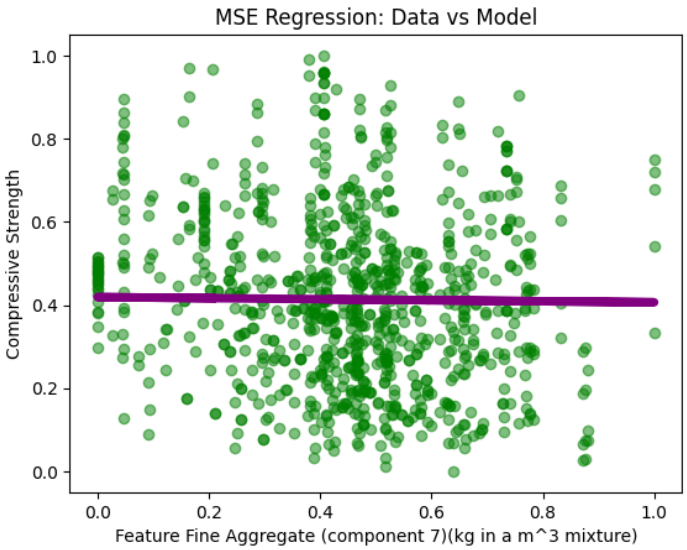
MSE: Preprocessed Training Data and Model Predictions

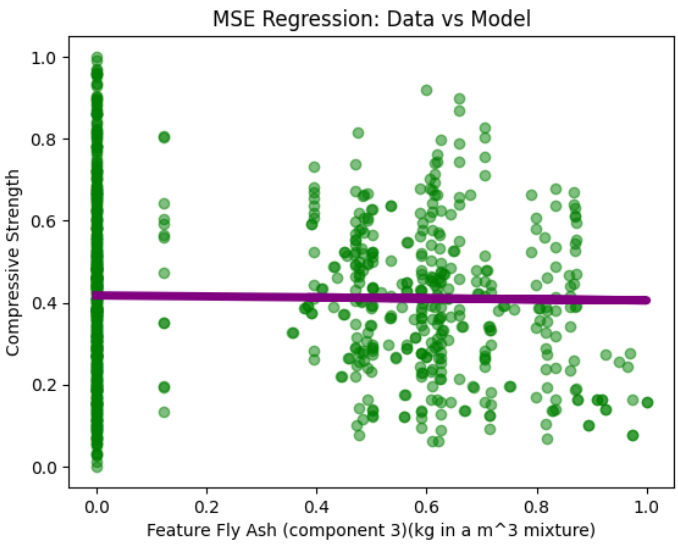


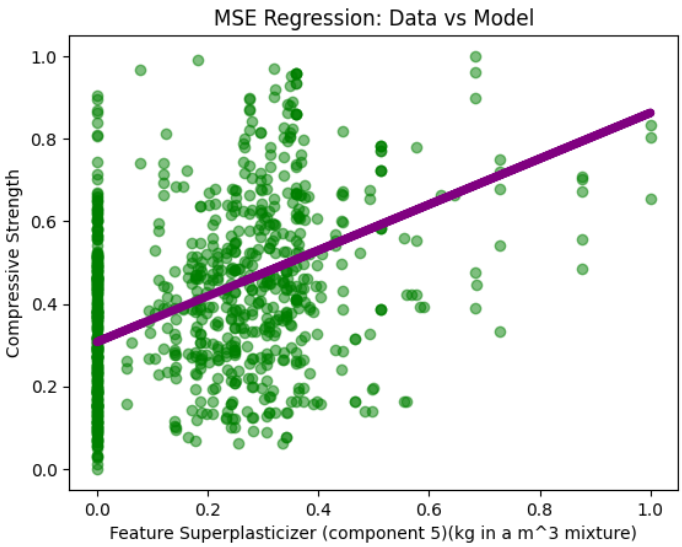


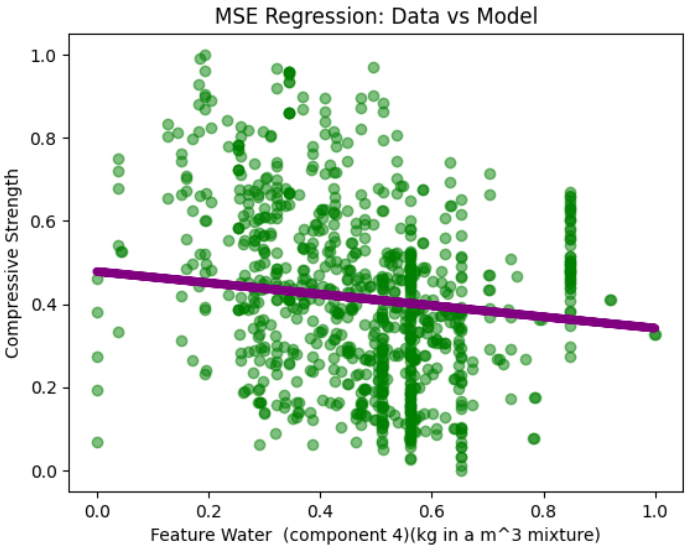




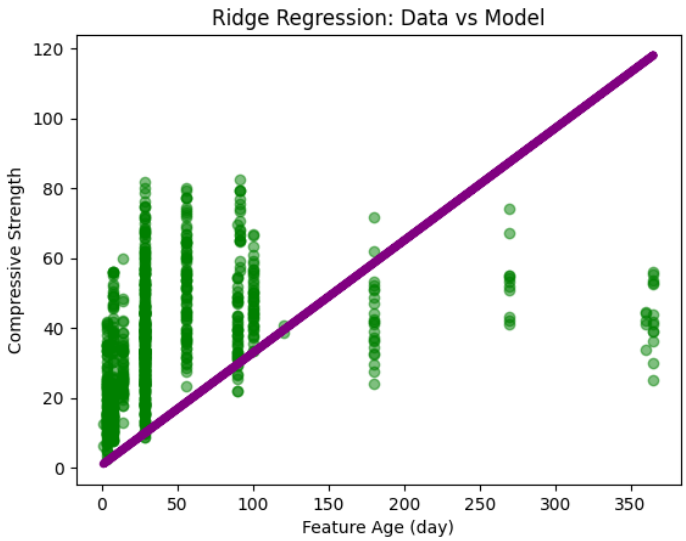

MSE Regression: Data vs Model
Compressive Strength
Feature Fine Aggregate (component 7)(kg in a m^3 mixture)

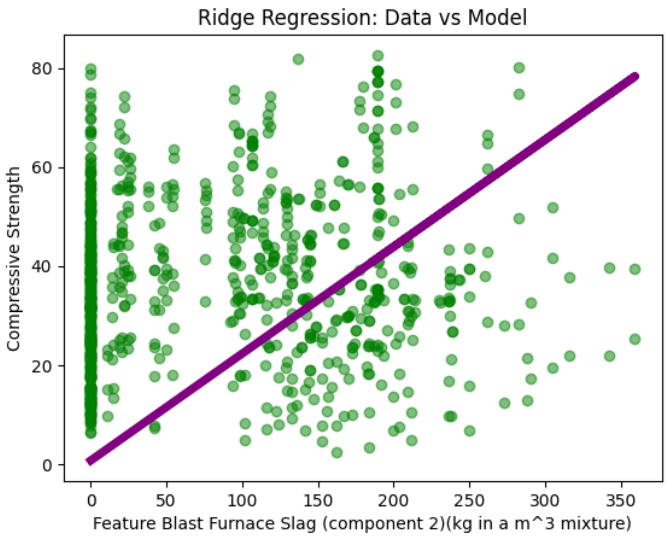

MSE Regression: Data vs Model
Compressive Strength
Feature Fly Ash (component 3)(kg in a m^3 mixture)

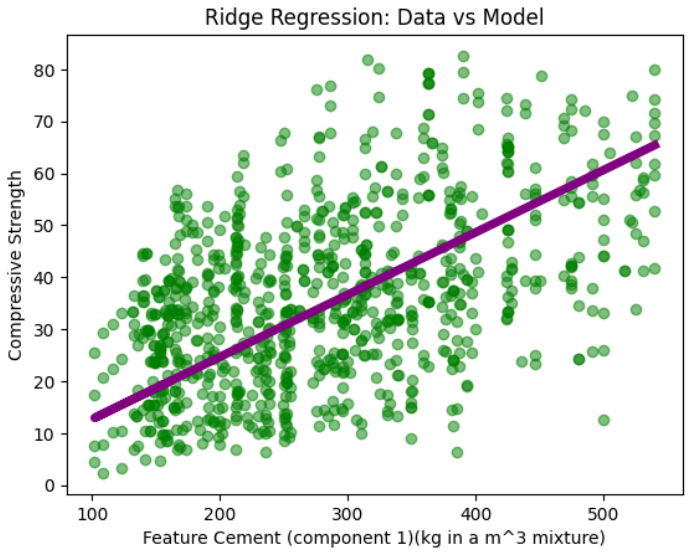

MSE Regression: Data vs Model
Compressive Strength
Feature Superplasticizer (component 5)(kg in a m^3 mixture)

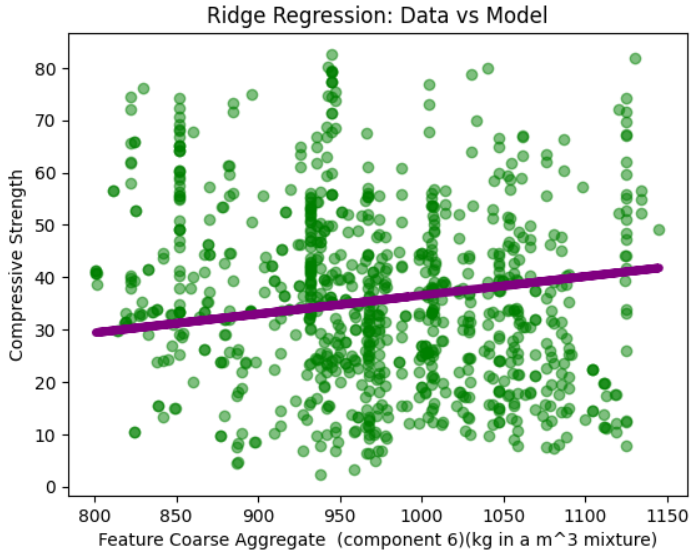

MSE Regression: Data vs Model
Compressive Strength
Feature Water (component 4)(kg in a m^3 mixture)

Ridge: Non-Preprocessed Training Data and Model Predictions

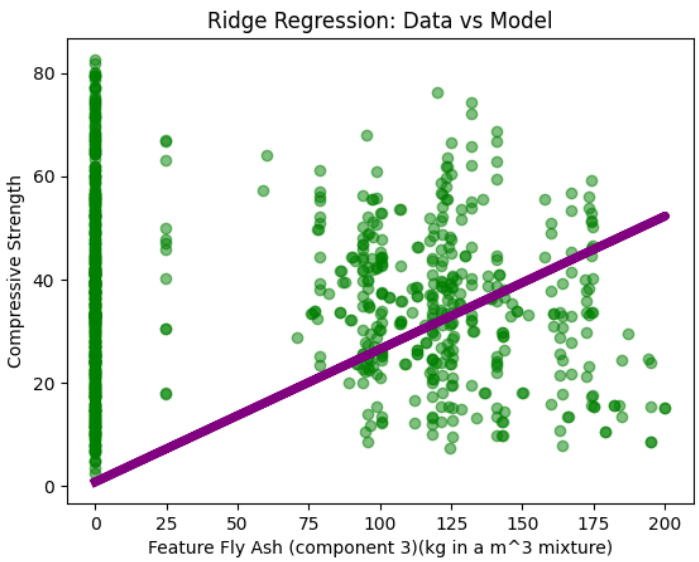


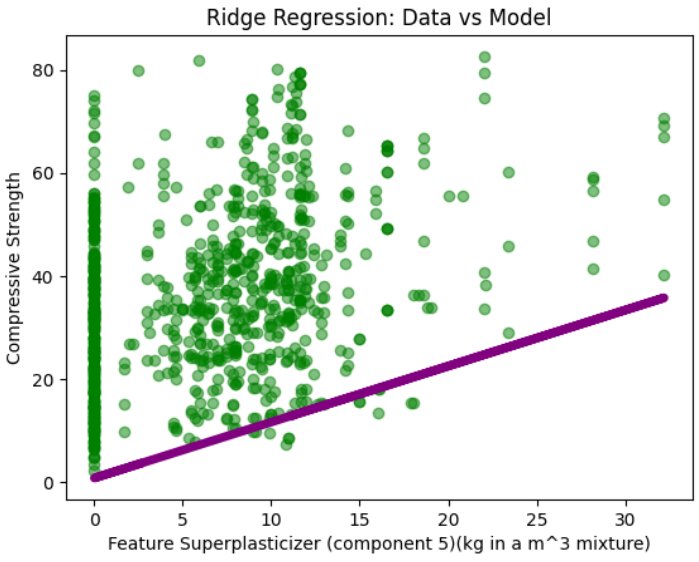


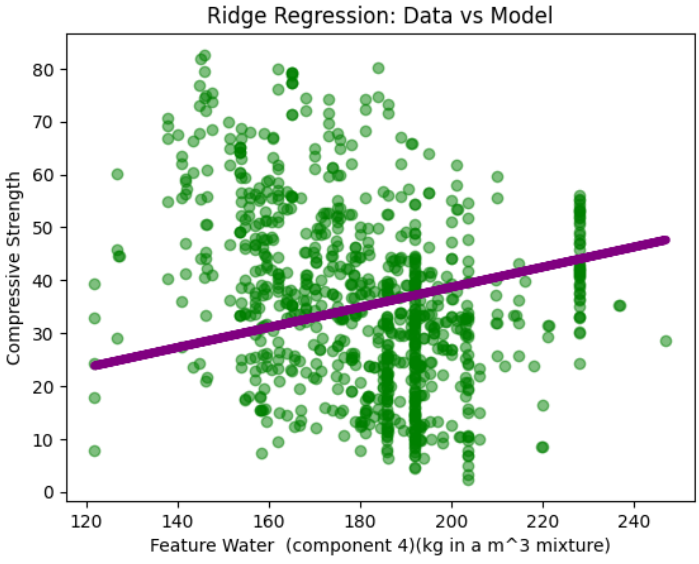




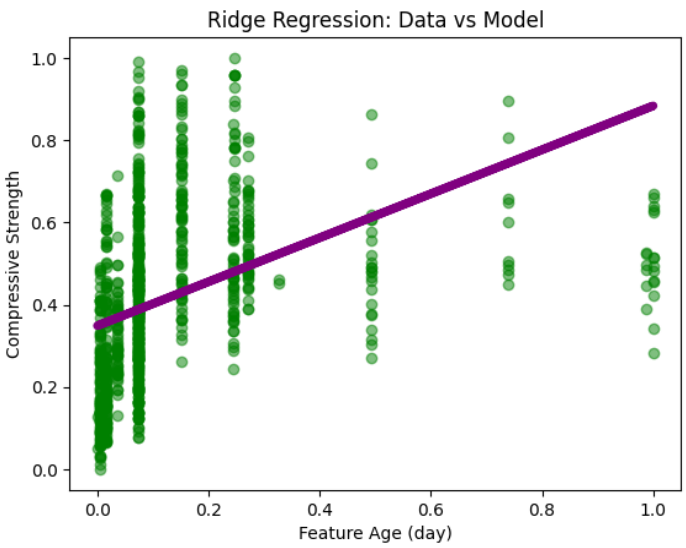

Ridge Regression: Data vs Model
Compressive Strength
Feature Fine Aggregate (component 7)(kg in a m^3 mixture)

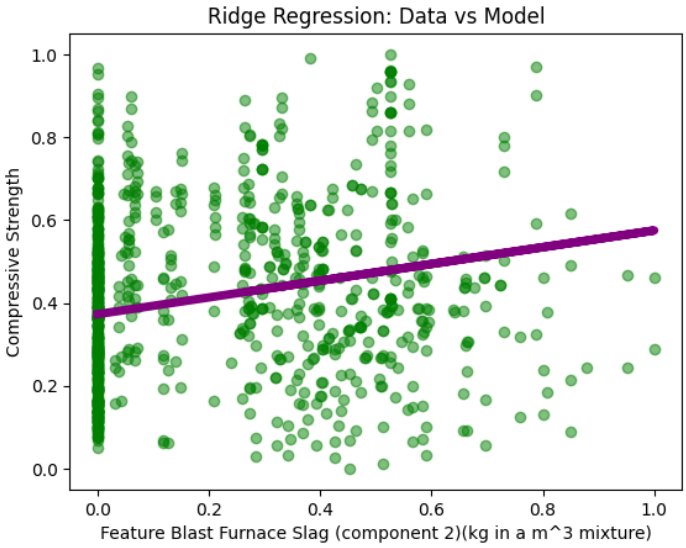

Ridge Regression: Data vs Model
Compressive Strength
Feature Fly Ash (component 3)(kg in a m^3 mixture)

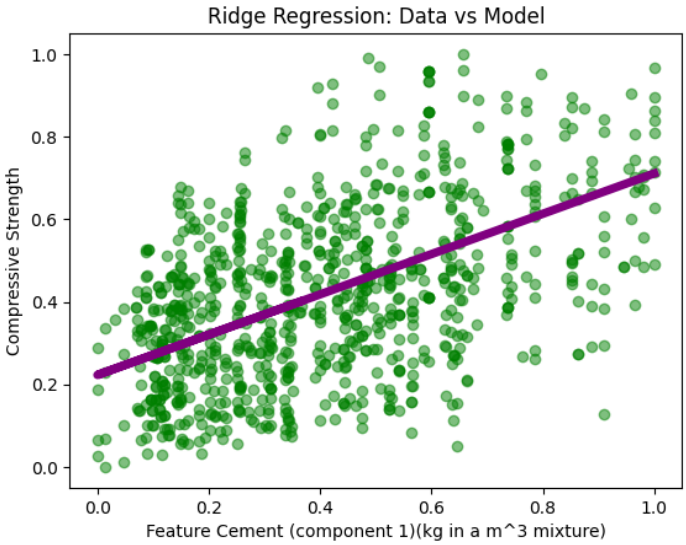

Ridge Regression: Data vs Model
Compressive Strength
Feature Superplasticizer (component 5)(kg in a m^3 mixture)

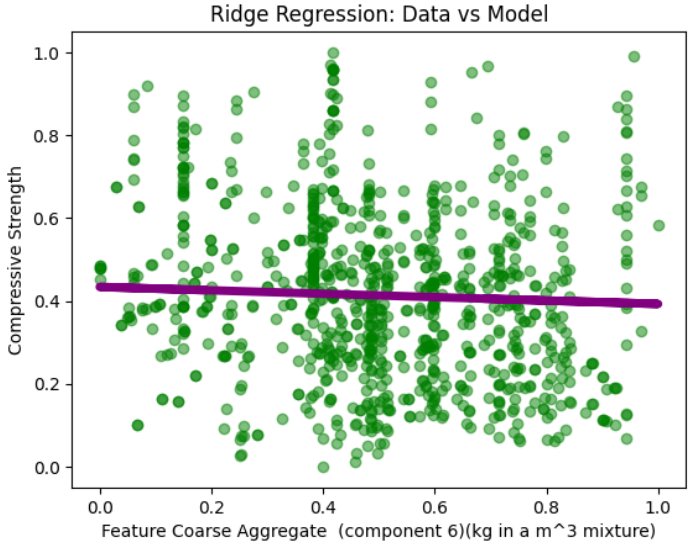

Ridge Regression: Data vs Model
Compressive Strength
Feature Water (component 4)(kg in a m^3 mixture)

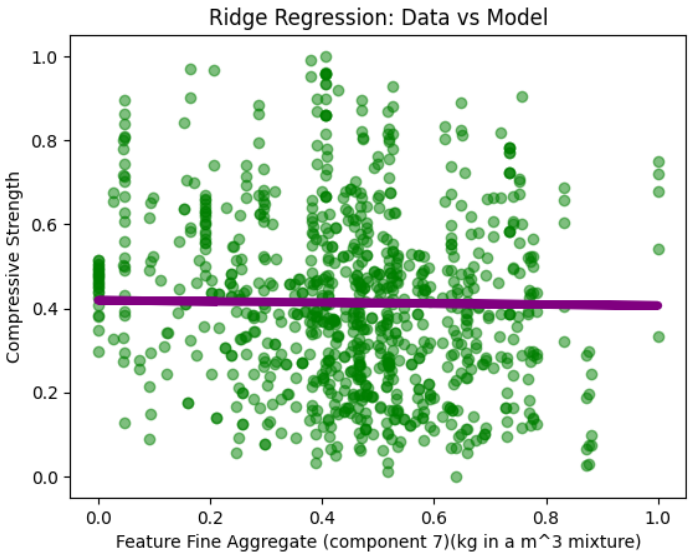
Ridge: Preprocessed Training Data and Model Predictions

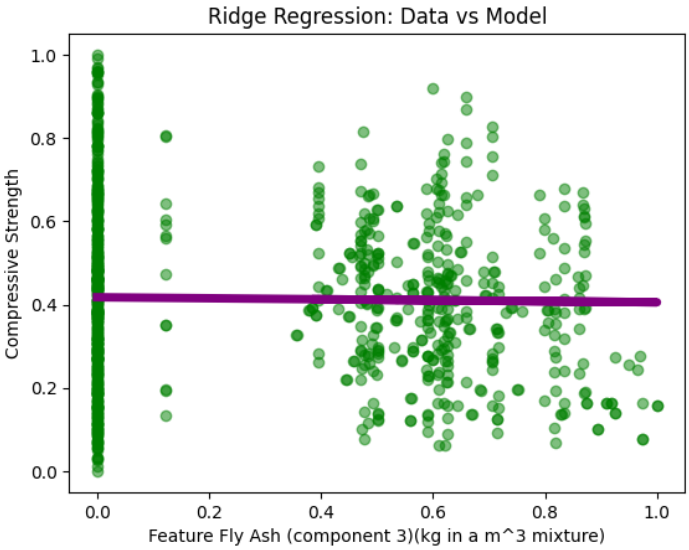


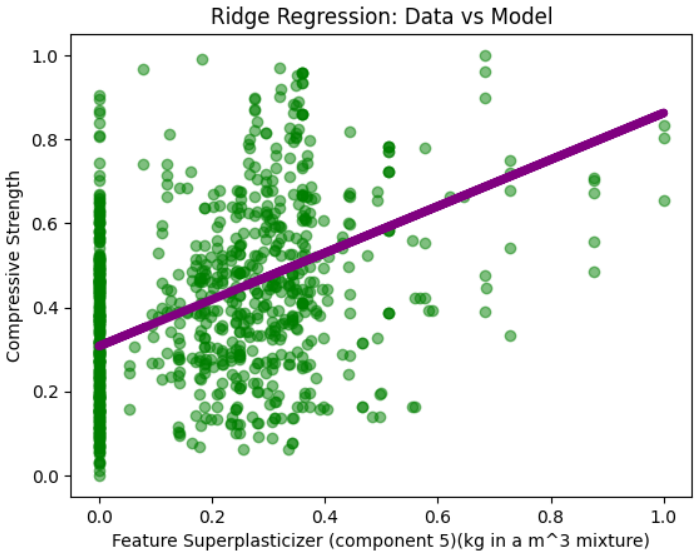


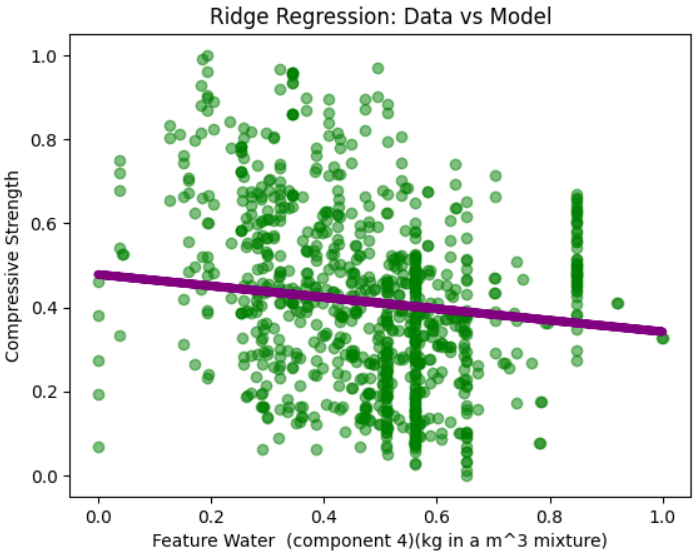




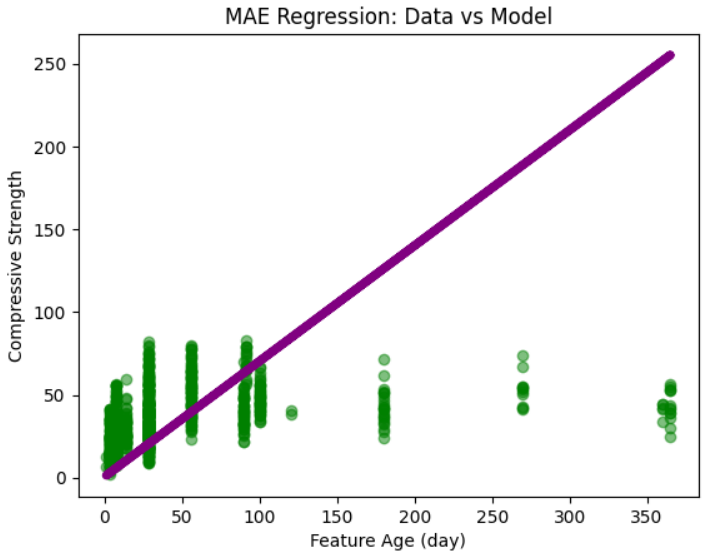

Ridge Regression: Data vs Model
Compressive Strength
Feature Fine Aggregate (component 7)(kg in a m^3 mixture)

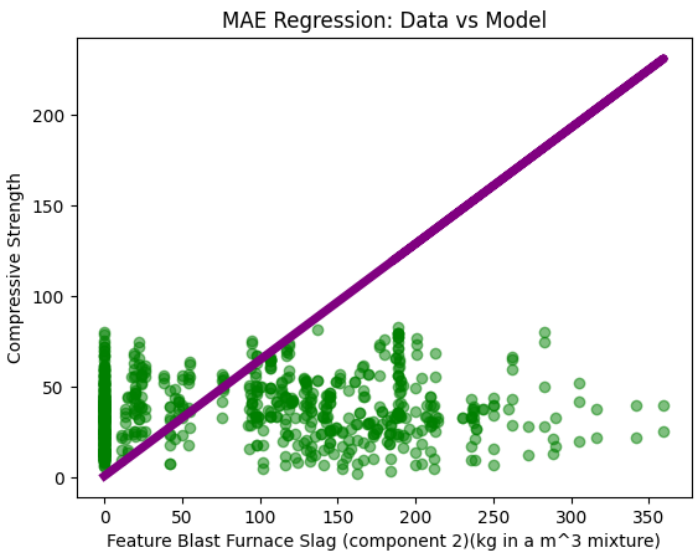

Ridge Regression: Data vs Model
Compressive Strength
Feature Fly Ash (component 3)(kg in a m^3 mixture)

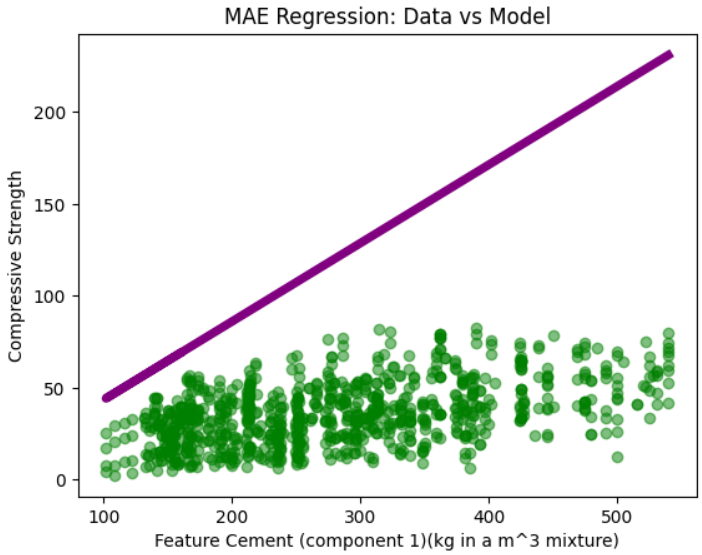

Ridge Regression: Data vs Model
Compressive Strength
Feature Superplasticizer (component 5)(kg in a m^3 mixture)

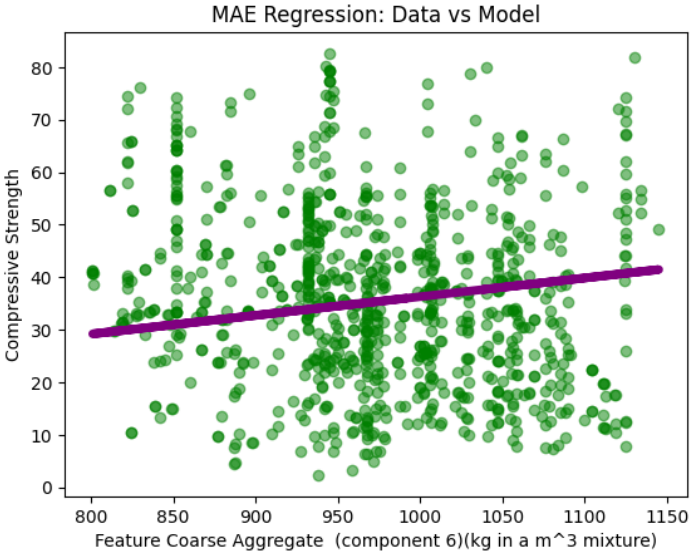

Ridge Regression: Data vs Model
Compressive Strength
Feature Water (component 4)(kg in a m^3 mixture)

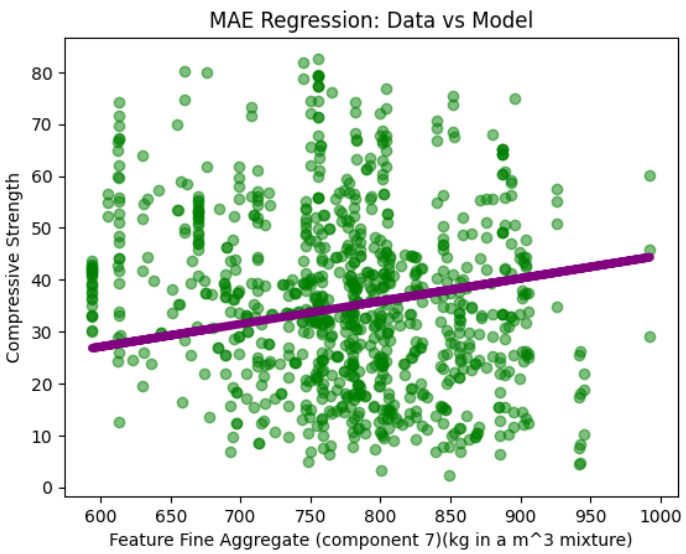
MAE: Non-Preprocessed Training Data and Model Predictions

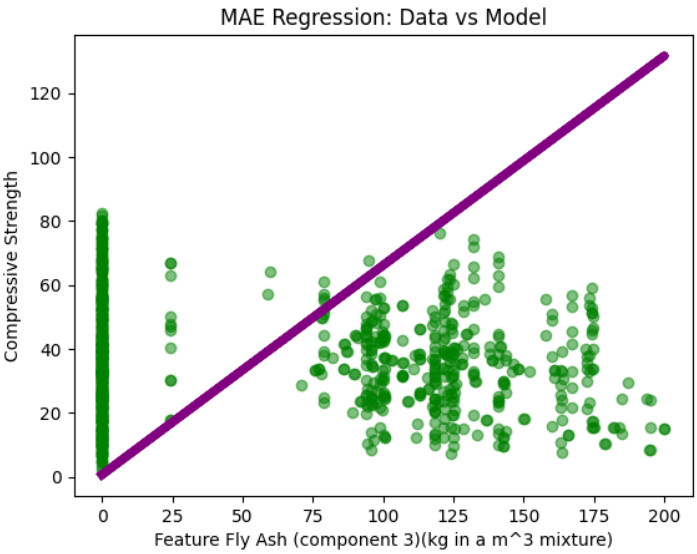


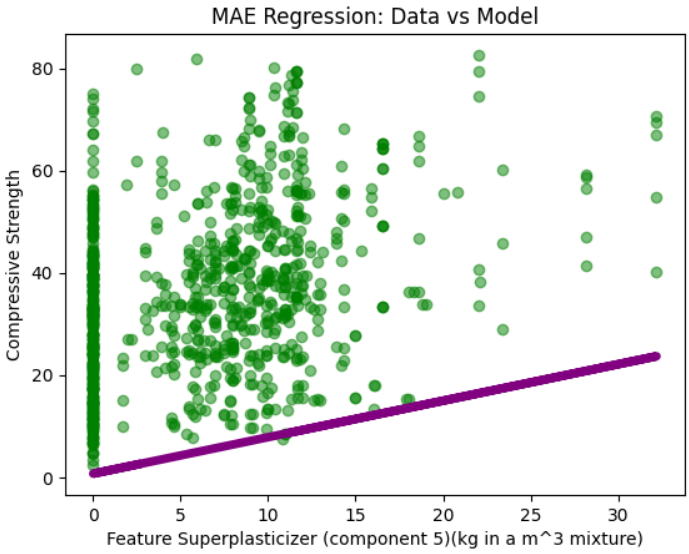


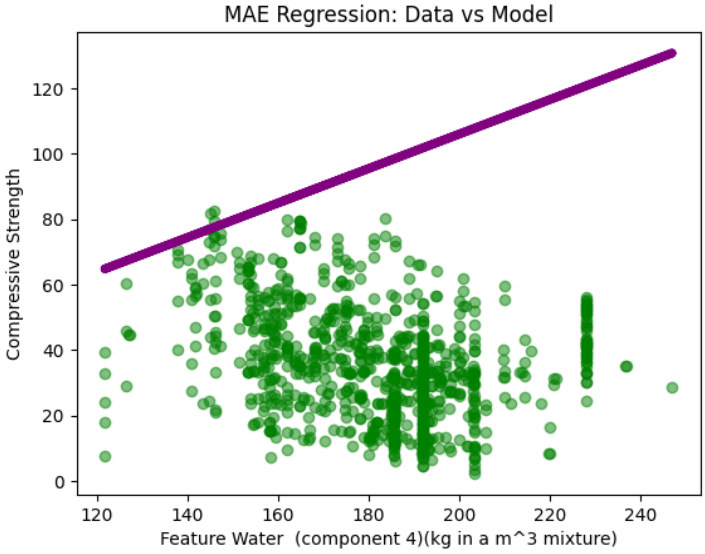




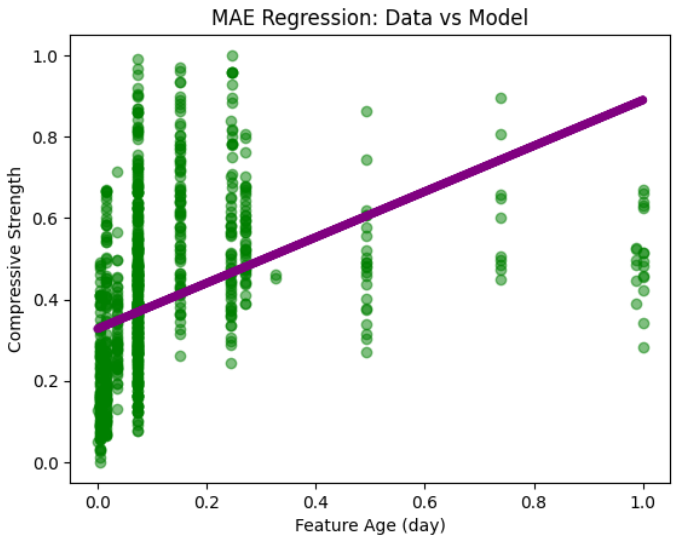

MAE Regression: Data vs Model
Compressive Strength
Feature Fine Aggregate (component 7)(kg in a m^3 mixture)

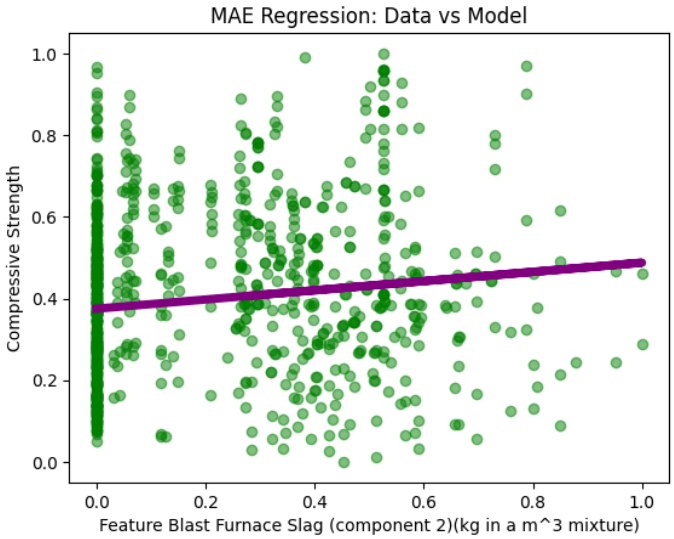

MAE Regression: Data vs Model
Compressive Strength
Feature Fly Ash (component 3)(kg in a m^3 mixture)

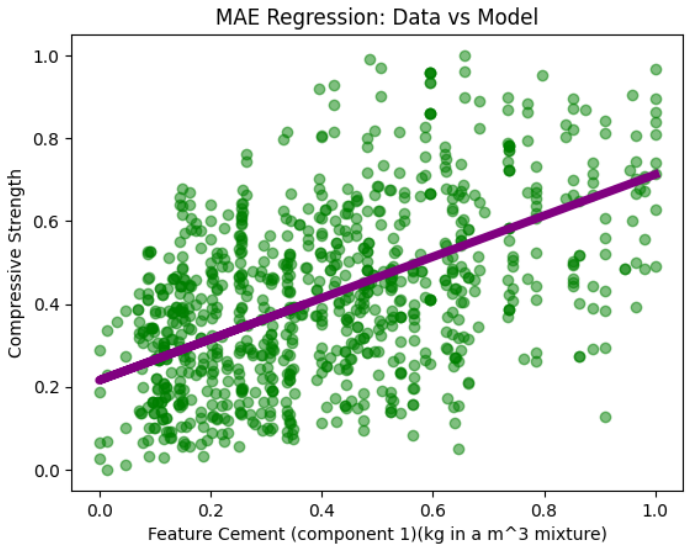

MAE Regression: Data vs Model
Compressive Strength
Feature Superplasticizer (component 5)(kg in a m^3 mixture)

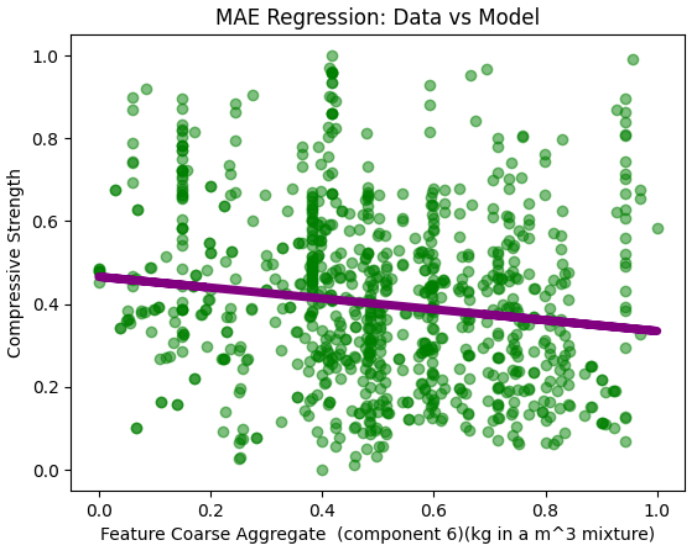

MAE Regression: Data vs Model
Compressive Strength
Feature Water (component 4)(kg in a m^3 mixture)

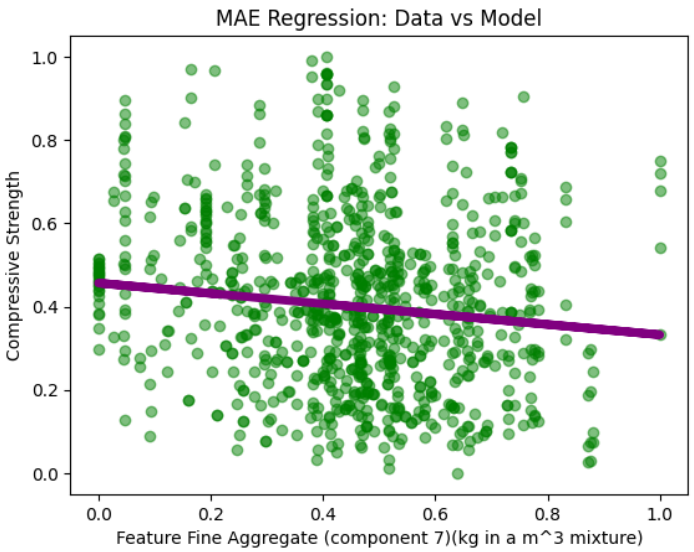
MAE: Preprocessed Training Data and Model Predictions

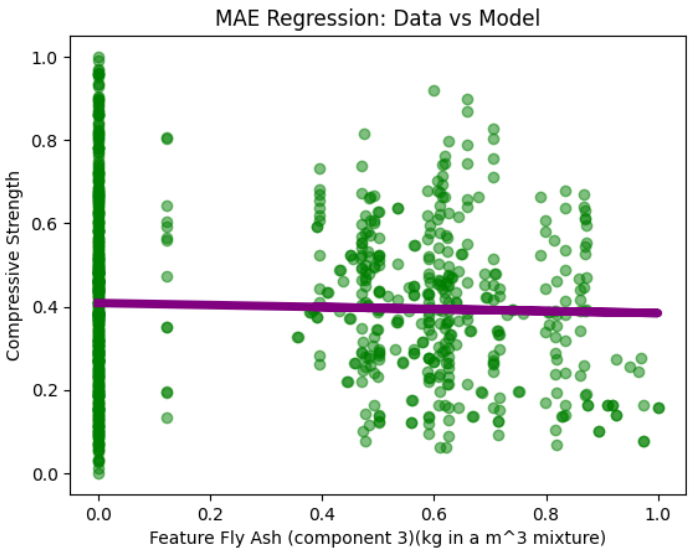


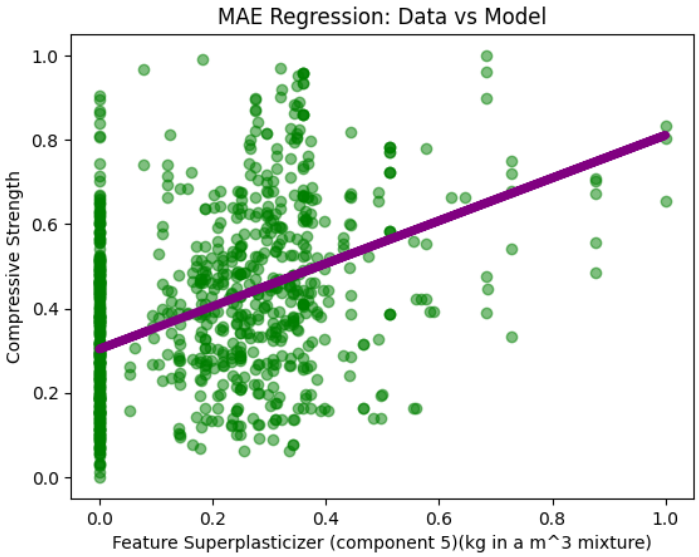


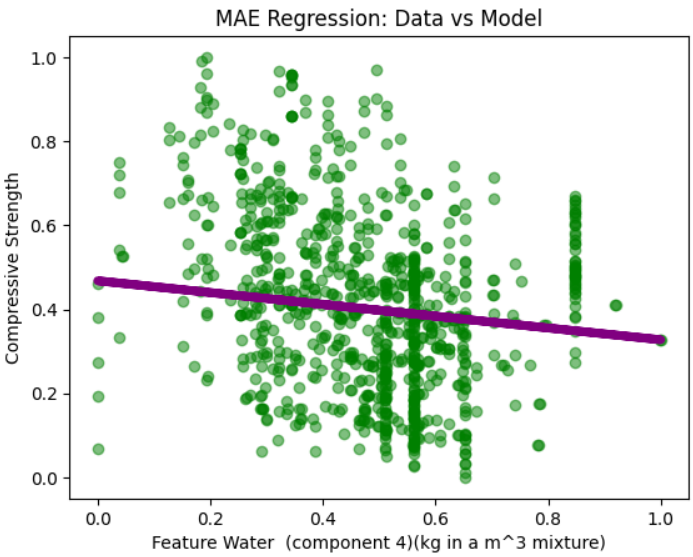





MAE Regression: Data vs Model
Compressive Strength
Feature Fine Aggregate (component 7)(kg in a m^3 mixture)


MAE Regression: Data vs Model
Compressive Strength
Feature Fly Ash (component 3)(kg in a m^3 mixture)


MAE Regression: Data vs Model
Compressive Strength
Feature Superplasticizer (component 5)(kg in a m^3 mixture)


MAE Regression: Data vs Model
Compressive Strength
Feature Water (component 4)(kg in a m^3 mixture)

Discussion

Overall, the preprocessed multivariate models performed the best, with R-squared values consistently between 0.5 and 0.7 for all MSE, Ridge, and MAE loss functions. Interestingly, for the non-preprocessed data, the only models able to achieve a positive R-squared value were MSE using multivariate regression and both MSE and Ridge using univariate regression of Feature 1 (Age); these were positive for both training and testing. However, for the preprocessed data, nearly all of the R-squared values were positive for both training and testing, with the exception of Feature 6 (Fly Ash) for testing. This indicates preprocessing the data with normalization added to the effectiveness of the models. Additionally, this is a signal that Feature 1 (Age) may be the feature most correlated with compressive strength outcomes, while Feature 6 (Fly Ash) may be the feature least correlated with compressive strength outcomes. Indeed, when looking further into the preprocessed results, it can be seen that Feature 1 (Age) shows R-squared values between 0.2 and 0.3 for MSE, Ridge, and MAE loss functions, outperforming the univariate regressions for other features by a strong margin; the closest runners up are Feature 5 (Fine Aggregate) with values between 0.1-0.2 and Feature 8 (Water) with values between 0.06-0.18.

This table also indicates that the choice of models may have resulted in good generalization since the testing values for all the multivariate models and for the features 2, 4, 5, and 8 are equal to or higher than the training values; unfortunately this did not fully hold for features 1, 3, 6, or 7, although 1, 3, and 7 are still quite close. With the preprocessed data, MAE outperformed MSE and Ridge for the multivariate models and the univariate Feature 7 (Superplasticizer) models in both test and train as well as for Features 2, 5, and 6 in train. MSE and Ridge were equal to or outperformed MAE for the remaining preprocessed models. This makes sense because MAE is more robust to outliers and we have a lot of outliers around zero for Feature 7. Comparing MSE and Ridge with the preprocessed data, the models were mostly similar, with slightly higher variance explained for MSE in the multivariate model using training data and slightly higher variance explained for Ridge in the multivariate model using testing data. This, too, makes sense as it confirms the use of Ridge regression for increasing generalization of a model.

Preprocessing the data helped considerably with consistency, reliability, and explainability, especially with Feature 7 (Superplasticizer) and the MAE models (which use absolute error). As mentioned in Experiment Details, in order to avoid the exploding gradient problem, different possible learning rate values were used for the non-preprocessed data ([1e-07, 1e-08, 1e-09, 1e-10]) versus the preprocessed data ([1e-02, 1e-03, 1e-04, 1e-05]). Values across those spectrums were chosen during the K-fold step. The most common learning rate for the preprocessed data was 0.01, and most of the Ridge models chose either 0.01 or 1e-05 for the lambda constant. The Ridge regression models all used the same learning rate as their corresponding MSE model. Largely, the models for both the non-preprocessed and preprocessed data were fairly quick to train.

In conclusion, I would recommend prioritizing the Age feature for maximizing compressive strength, followed by the Fine Aggregate feature and the Water feature. As an estimate, perhaps keep the Age around 30-100 days, the Fine Aggregate between 745-765 kg, and the Water 145-185 kg.