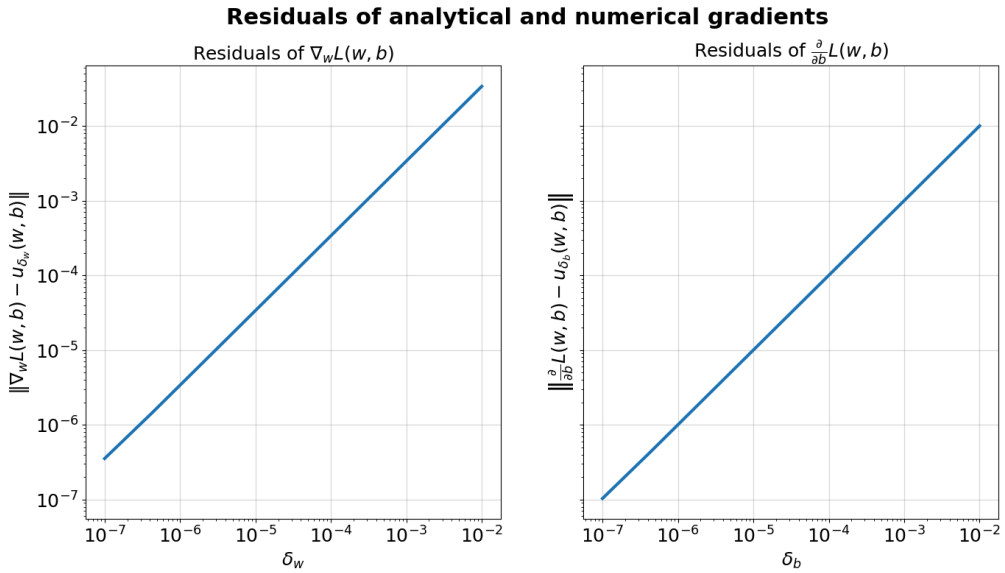
# HW3 - Report

## Q1

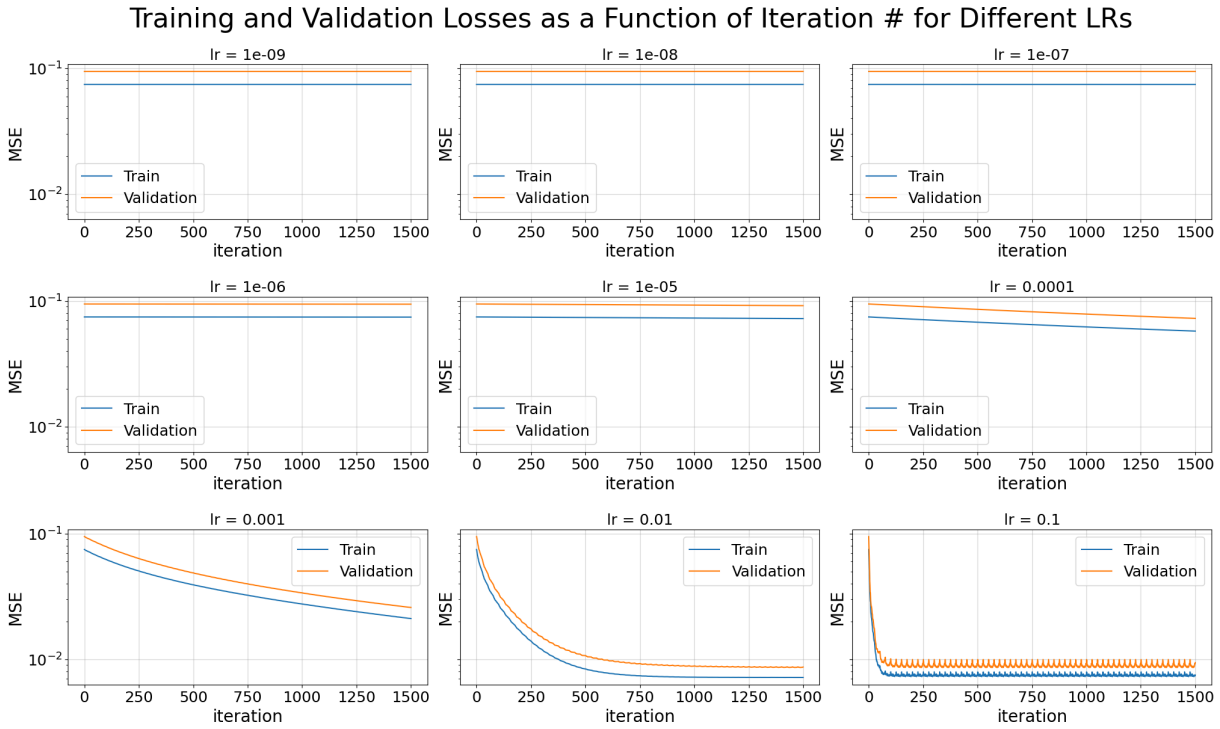
The analytical partial derivative w.r.t $b$:

$$\frac{\partial}{\partial b}L(w,b) = \frac{1}{m}2 \cdot (1_m)^T(Xw + 1_m \cdot b - y)$$

## Q2




## Q3




The optimal learning rate is **<u>0.01</u>** because its training and validation loss decreased the most and steadily compared to other learning rates.

Learning rates 1e-9 to 1e-5 do not demonstrate a decrease in the losses. This is because these learning rates are very small, making the updates to the model parameters insignificant. Thus the model does not learn and improve at all over the 1500 iterations.

For learning rates 0.0001 we see a minimal decrease in the losses, which suggests that the model does learn a little but the learning rate is still relatively too low for significant progress. This is also true for learning rate 0.001 for which the losses decrease even more but are still much worse than the learning rate 0.01.

For the learning rate 0.1, its losses initially decreased significantly but then they ceased to improve, and the loss jumps a lot which indicates that 0.1 is too large as a learning rate, causing the model to make very large updates to its parameters instead of converging to a local minimum.

We see that 0.01 learning rate is enough to reach minimum loss even in less than 1500 steps, and increasing the number of steps will not improve the model.

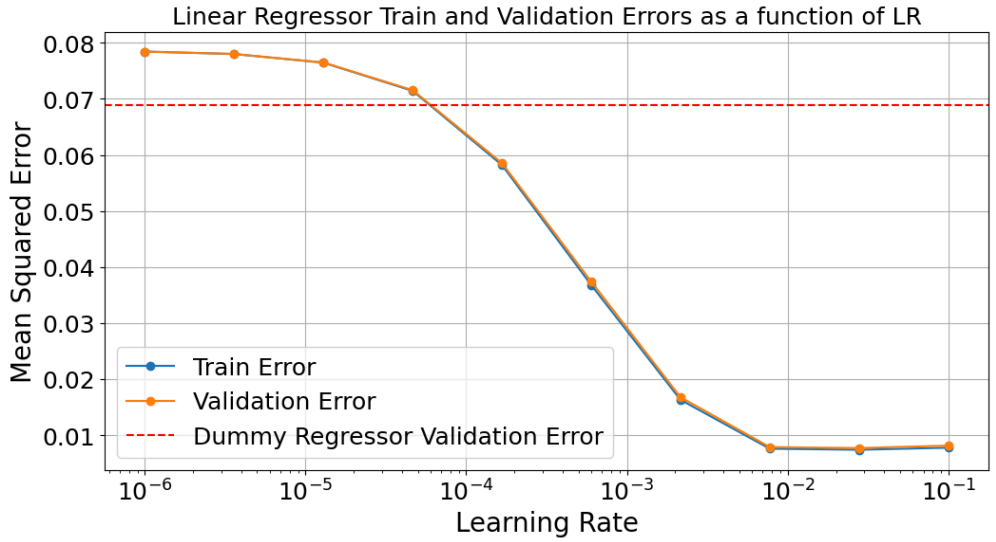
Q4

**Crossed-validated errors of regressors**

| Model | Section | Train MSE | Valid MSE |
|---|---|---|---|
| | | Cross validated | |
| Dummy | 2 | 0.068899 | 0.068933 |

Q5

learning rates tested values: np.logspace(-6, -1, 10)
Number of folds in CV: 5




The optimal learning rate is **0.0278**

**Crossed-validated errors of regressors**

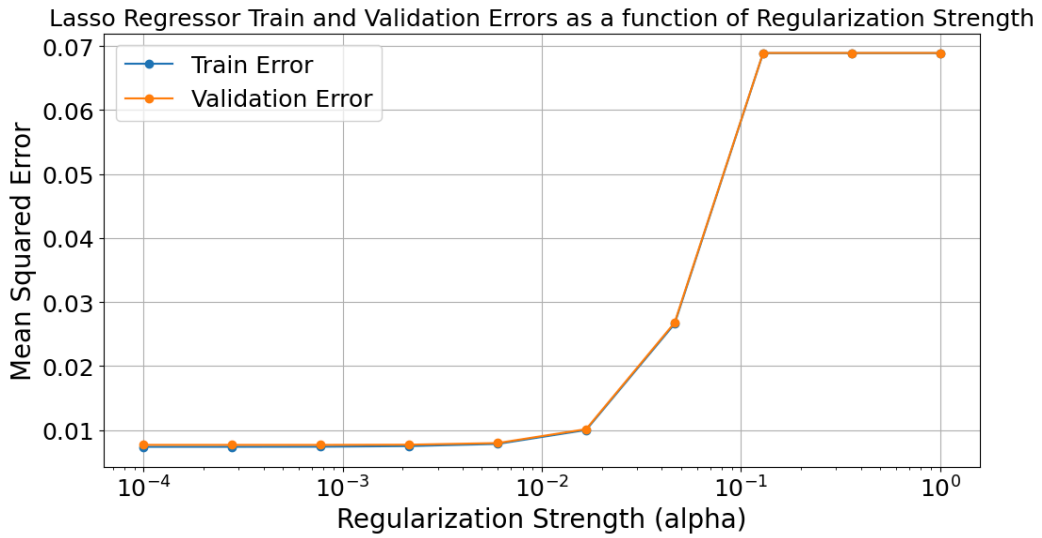
| Model | Section | Train MSE | Valid MSE |
|---|---|---|---|
| | | Cross validated | |
| Dummy | 2 | 0.068899 | 0.068933 |
| Linear | 2 | 0.007412 | 0.007722 |

Q6

The performance of the Dummy Regressor would not change, because it always returns the mean of the target value, which is not affected by feature normalization. However, the linear regressor performance would decrease, because it relies on minimizing the MSE, and if some features are on a way larger scale, they will contribute a lot to the loss, and the regressor would fine-tune to minimize them, rather than taking into consideration all the given features equally.

Q7

Regularization strength values: np.logspace(-4, 0, 10)
Number of folds in CV: 5




The optimal regularization strength: **0.000774**

**Crossed-validated errors of regressors**

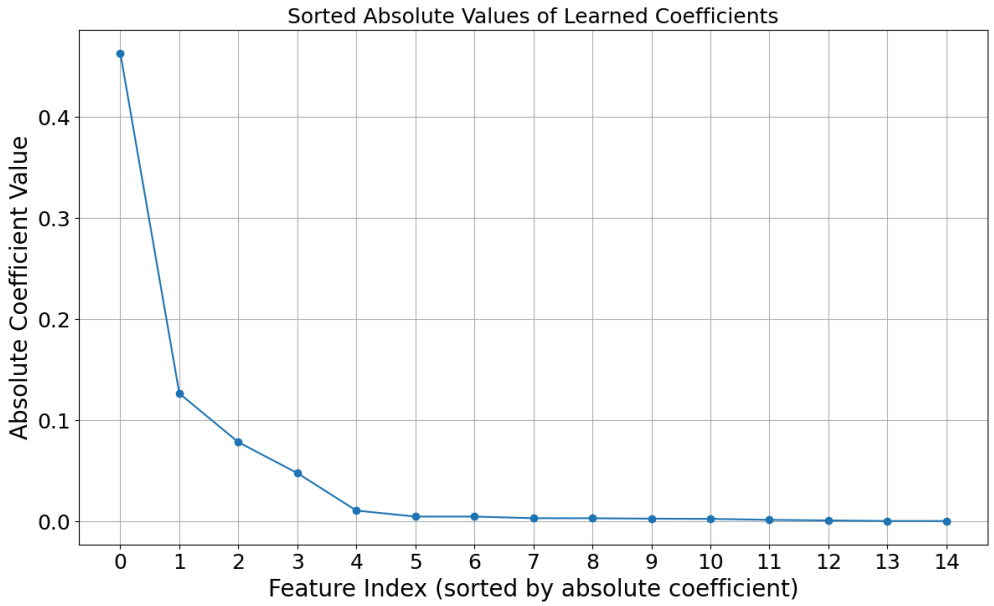
| Model | Section | Train MSE | Valid MSE |
|---|---|---|---|
| | | Cross validated | |
| Dummy | 2 | 0.068899 | 0.068933 |
| Linear | 2 | 0.007412 | 0.007722 |
| Lasso Linear | 3 | 0.0074 | 0.007686 |

Q9

**Top 5 features with the largest coefficients (in absolute value) in the optimal Lasso regressor**

| Feature | Coefficient |
|---|---|
| PCR_04 | 0.462614 |
| sugar_levels | 0.126135 |
| PCR_02 | -0.077948 |
| PCR_06 | -0.047372 |
| PCR_09 | -0.010335 |

Q10



Q11

The magnitude of the coefficients is interesting because it indicates the feature importance[1], i.e. its influence strength on the target variable. Larger magnitudes (in absolute value) mean a greater impact. By comparing feature coefficients we can determine what features contribute to the model more than others, aiding in feature selection. In particular, a zero coefficient indicates that the feature has no contribution to the model and can be considered for removal.
Here we see that the model could still reach high accuracy by just using the top 5 features as stated in Q9, as the rest have very low coefficients.

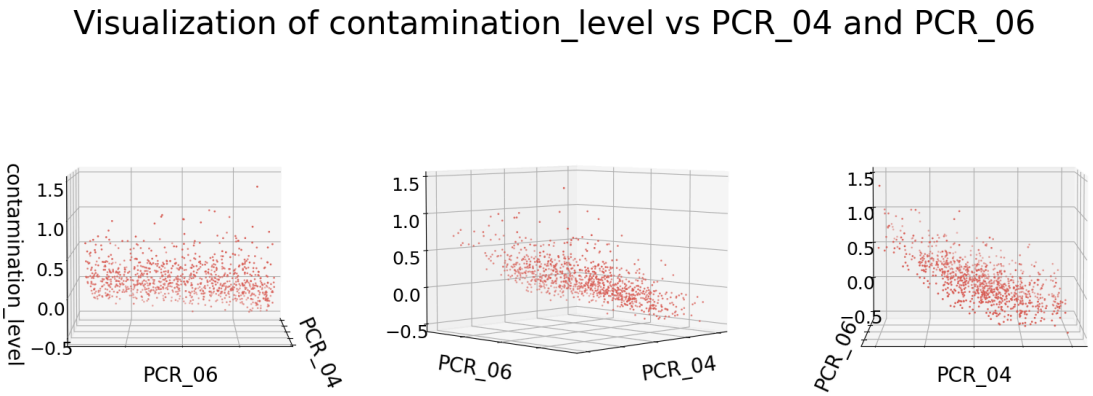
Q12

The training performance of the Lasso model would have changed. Similarly to the Linear Regressor, when the features aren't normalized, differently scaled features can dominate the loss expression, and the model would have to fine-tune them to decrease the loss, resulting in missed patterns that could be found by using all the features.

Q13

I would expect the coefficients of a Ridge regressor to be more uniformly small (but not almost 0). This is because the Lasso regressor uses L1 regularization, which encourages sparse weight vector (and we can see in Q10 that indeed most features have almost 0 coefficient). This can be said to induce "feature selection". On the contrary, the Ridge regressor uses L2 regularization, which penalizes the sum of the squared coefficients. This encourages the overall shrinkage of all coefficients towards zero uniformly (since we normalized the features).

Section 4 Task 2:



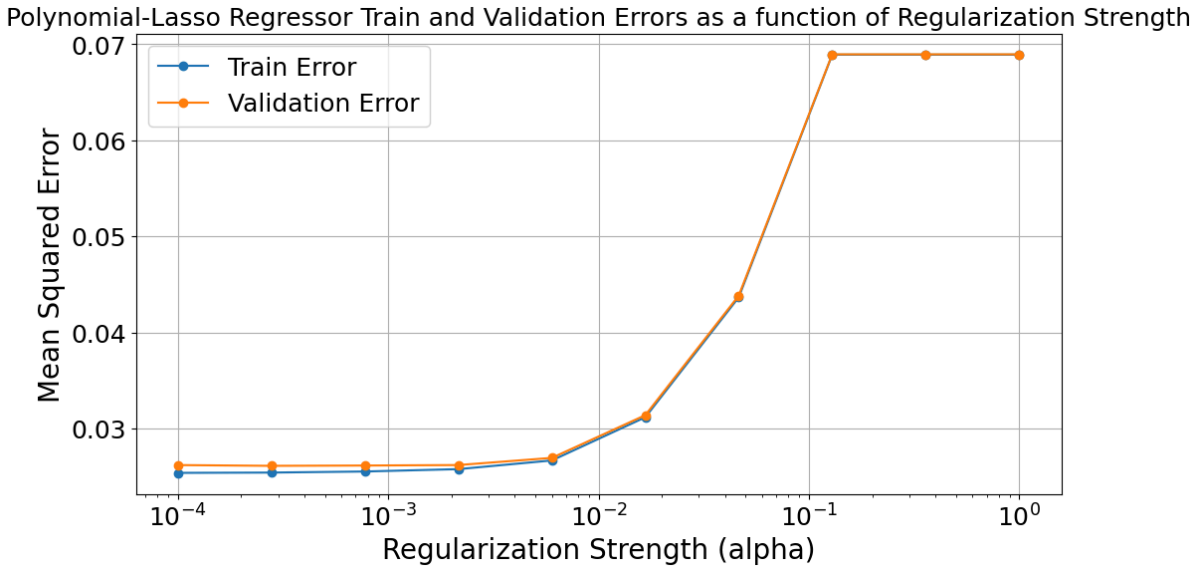

We can see it looks pretty linear (sits on a plane), except when looking at the right-most perspective where we can see some quadratic behavior.

[1] This works because the features are normalized. Had the features not been normalized, the coefficients would be biased because features could be in different scales, as discussed in Q6.

Q14

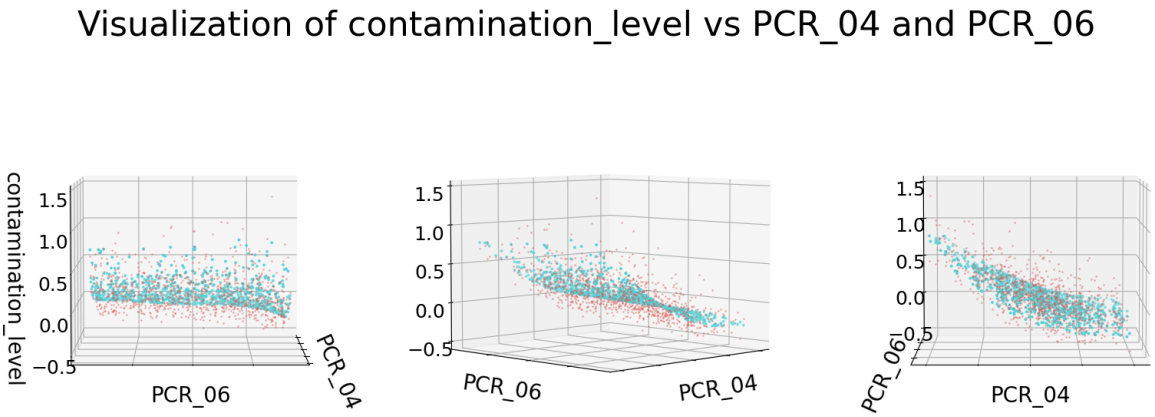
It is important to re-normalize because if we have a monomial x, and we compare it to x^2, for big values of x, x^2 will be a lot bigger. If we don't normalize, the higher-order monomials will have way bigger values than the rest and will dominate the loss expression just like if we don't normalize the features from the input. To use higher-order monomials we normalize them, which lets them keep their behavior (curvature, shape, and so on) while letting all monomials be of equal magnitude, which would be better for the linear regressor to train with.

Q15




Optimal regularization strength: **0.000278**
Optimal validation error: 0.026

Q16




The points in blue are the model's predictions and the points in red are the train dataset. We see the model doesn't fit the data pretty well, which makes sense since the data is sparse and just knowing PCR_04 and PCR_06 isn't a lot of information to

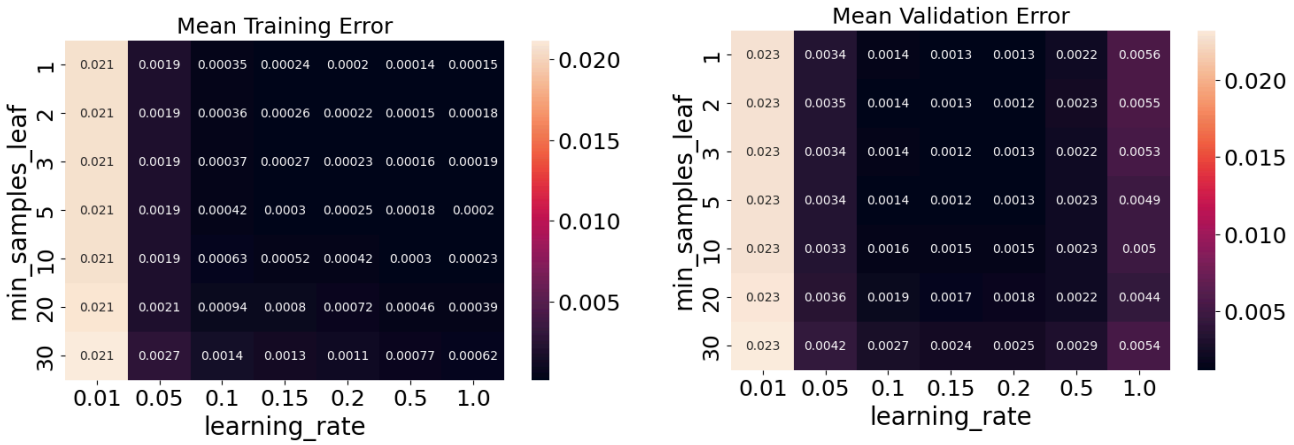
predict the contamination_level with great accuracy. It does seem like it found the closest quadratic curve to the training points.

Q17

**Crossed-validated errors of regressors**

| Model | Section | Train MSE | Valid MSE |
|---|---|---|---|
| | | Cross validated | |
| Dummy | 2 | 0.068899 | 0.068933 |
| Linear | 2 | 0.007412 | 0.007722 |
| Lasso Linear | 3 | 0.0074 | 0.007686 |
| Polynomial Lasso | 4 | 0.025422 | 0.026124 |

Q18




Optimal parameters: **Learning Rate <u>0.15</u>, Min Samples Leaves: <u>3</u>**
Optimal parameters train error: 0.00027477, validation error: 0.001192

Q19

**Crossed-validated errors of regressors**

| Model | Section | Train MSE | Valid MSE |
|---|---|---|---|
| | | Cross validated | |
| Dummy | 2 | 0.068899 | 0.068933 |
| Linear | 2 | 0.007412 | 0.007722 |
| Lasso Linear | 3 | 0.0074 | 0.007686 |

| | | | |
|---|---|---|---|
| Polynomial Lasso | 4 | 0.025422 | 0.026124 |
| GBM Regressor | 5 | 0.000275 | 0.001192 |

Q20

**Crossed-validated and Test errors of all regressors**

| Model | Section | Train MSE | Valid MSE | Test MSE |
|---|---|---|---|---|
| | | Cross validated | | Retrained |
| Dummy | 2 | 0.068899 | 0.068933 | 0.065379 |
| Linear | 2 | 0.007412 | 0.007722 | 0.007742 |
| Lasso Linear | 3 | 0.0074 | 0.007686 | 0.007673 |
| Polynomial Lasso | 4 | 0.025422 | 0.026124 | 0.023838 |
| GBM Regressor | 5 | 0.000275 | 0.001192 | 0.001502 |

The **GBM** model performed best on the test set.

We can see that in terms of overfitting/underfitting:
The GBM model was overfitting (lower train error than test error by an order of 10) but still had the lowest test error of all models. Other models had similar train/valid/test errors. Among them, the Polynomial Lasso can be considered the most underfitting (except the dummy regressor of course) since it has the highest errors, which are also closest to the dummy regressor. This suggests that the polynomial mapping did not contribute to the Lasso model performance. It is also interesting to note that the Linear and the Lasso Linear models achieved very similar MSE in the Train/Valid/Test sets.