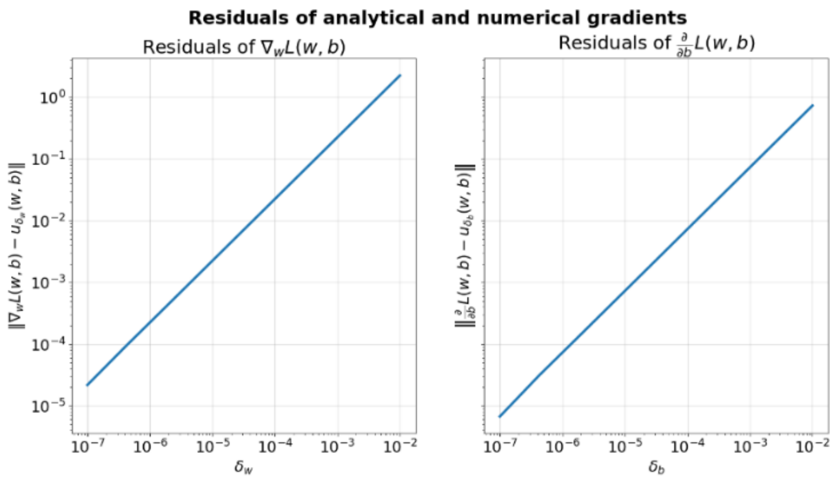
Section 1

(A1)

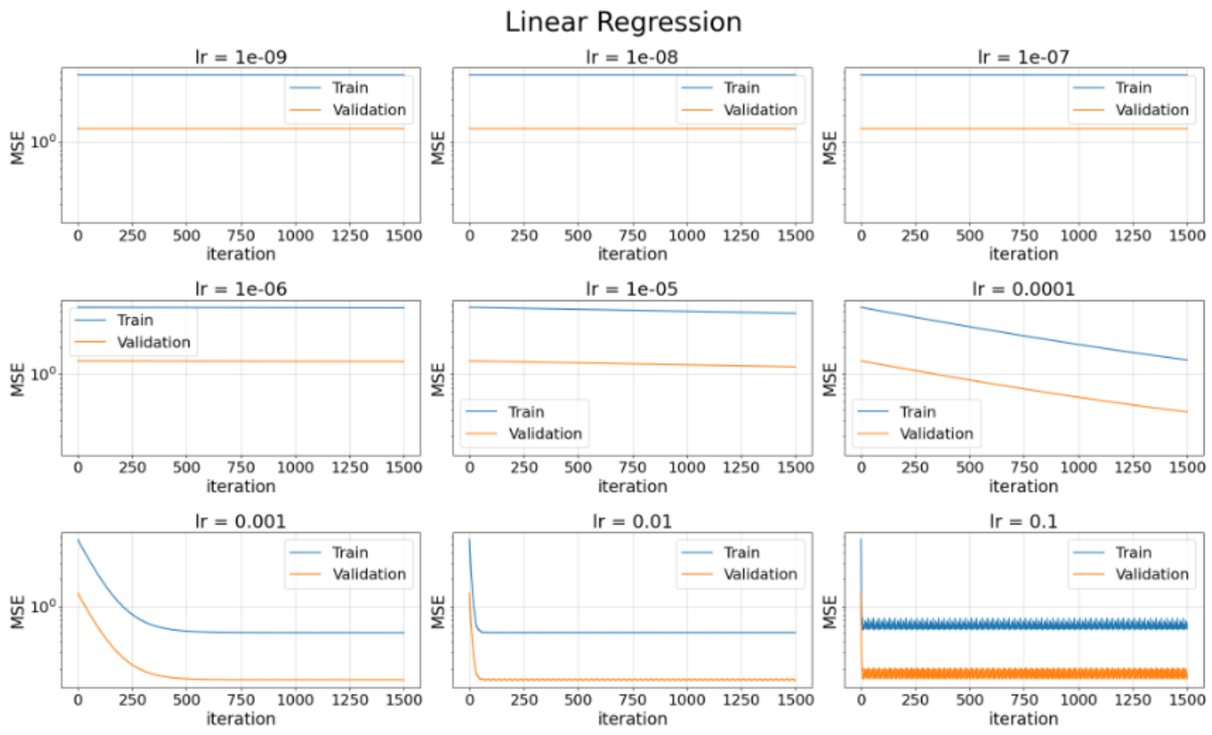
$$L(w,b) = \frac{1}{m}\,\Sigma_{i=1}^{m}\left(x^i w + b - y^i\right)^2$$

$$\frac{\boldsymbol{\partial L(w,b)}}{\boldsymbol{\partial b}} = \frac{\mathbf{2}}{\boldsymbol{m}}\Sigma_{i=1}^{m}\left(x^i w + b - y^i\right)$$

(A2)



(A3)



lr size = 0.01, Best train loss = 0.508229717973801, Best validation loss = 0.14791003436298927

lr = 0.01 is the optimal learning rate since it has the best validation loss out of all other learning rates, it's not jittery as lr = 0.1(step size is too big to find settle for a minimum) and out of all non-jittery graphs – it takes the least steps to get to the minimum.

It doesn't make sense for it to increase the number of steps above 1500 since it's way more then enough for it to get to it to descent to the minimum

Section 2

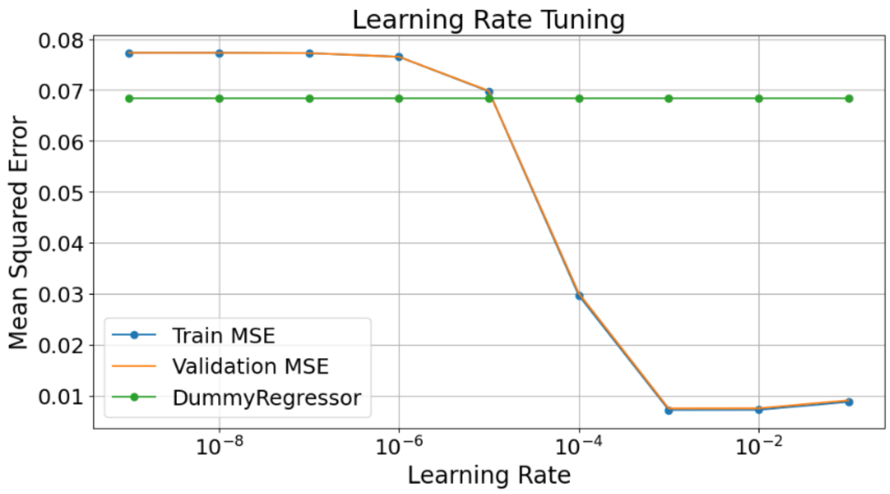
(A4)

Model	***Section***	***Train MSE***	***Valid MSE***
Dummy	***2***	***0.0684***	***0.0685***

(A5)

Model	***Section***	***Train MSE***	***Valid MSE***
		Cross validated	
Dummy	***2***	***0.0684***	***0.0685***
Linear	***2***	***0.0072***	***0.0075***

- ***Optimal Learning Rate: 0.001; Validation MSE: 0.0075***



Had we chosen not to normalize features beforehand, would the training performance of these two models have changed (assume there are no numerical errors)? Explain.

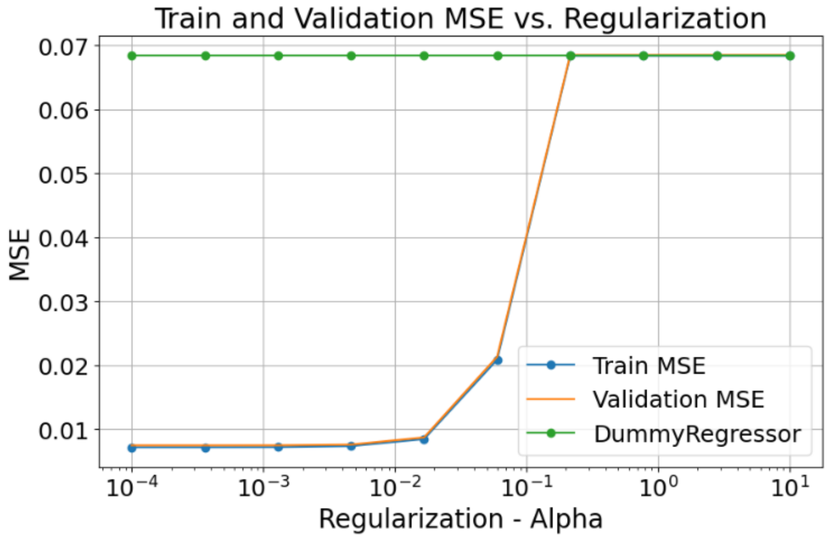
(A6)

Linear Regressor: *yes, it could lead to the GD algorithm to converge slower or even fail to converge; because the learning rate might become too small for some features and too large for others, leading to inefficient training*

Dummy Regressor: *no, because it always predicts the mean of the target values and does not use the features for predictions.*

Section 3

(A7)



- ***Optimal Regularization (Alpha): 0.000359; Validation MSE: 0.00748***

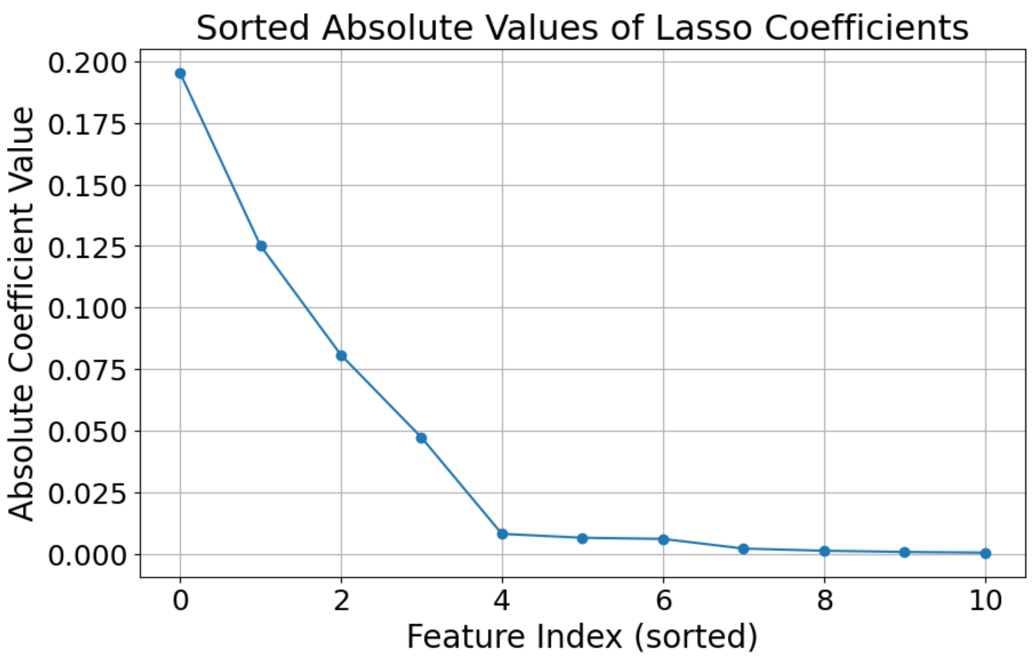
(A8)

Model	***Section***	***Train MSE***	***Valid MSE***
		Cross validated	
Dummy	***2***	***0.0684***	***0.0685***
Linear	***2***	***0.0072***	***0.0075***
Lasso	***3***	***0.00719***	***0.00748***

(A9) *Top 5 Features by Coefficient (in absolute value):*

1. *PCR_04: 0.19544417844798376*
2. *sugar_levels: 0.12519906916152487*
3. *PCR_02: 0.08079651821350847*
4. *PCR_06: 0.04728114417158705*
5. *PCR_03: 0.008124744282097194*

(A10)



(A11) *as was taught in the Tutorial, the l^1 regularization in the Lasso Regression causes "variable selection" by shrinking some coefficients to zero; meaning that larger magnitudes suggest that the feature has a stronger effect on the target variable.*

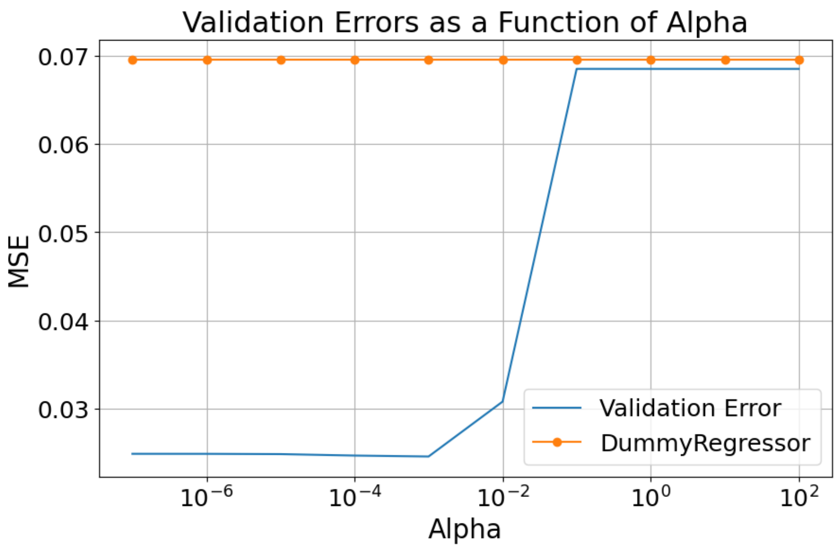
(A12) *Features with larger scales would dominate the regularization term, leading to biased results, which might result in the learned coefficients of the Lasso model to be significantly affected.*

(A13) *If we had used Ridge regression instead, there would be less coefficients that are close to zero, and all features would generally be included in the model with reduced magnitudes*

Section 4

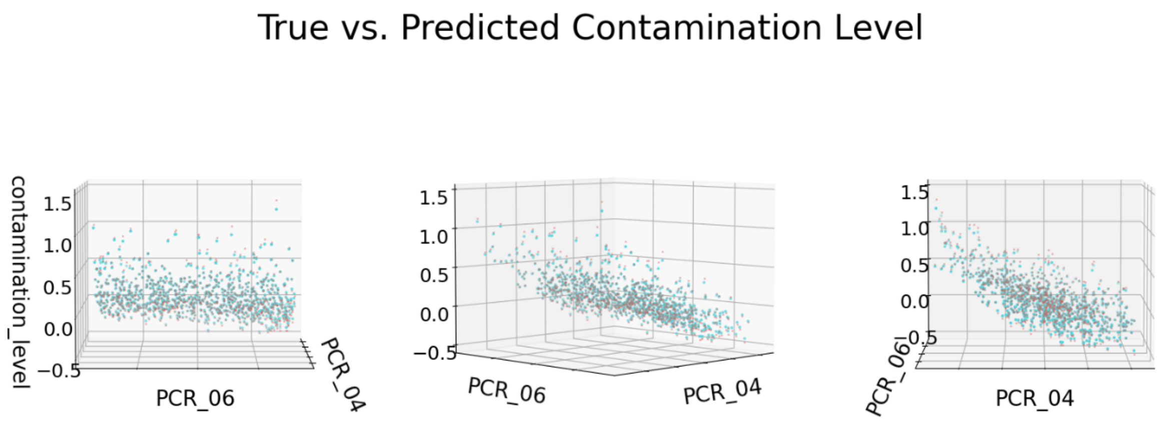
(A14) *after the poly-mapping, the features will have different scales because of their different degrees (x compared to x^3 for example), we will see a big difference if their original values are large; without normalization they will contribute unevenly to the model's prediction. Therefore, we should normalize to ensure that the features are on a similar scale - which improves the performance of gradient-based optimization algorithms.*

(A15)



- ***Optimal Regularization (Alpha): 0.001; Validation MSE: 0.025***

(A16)

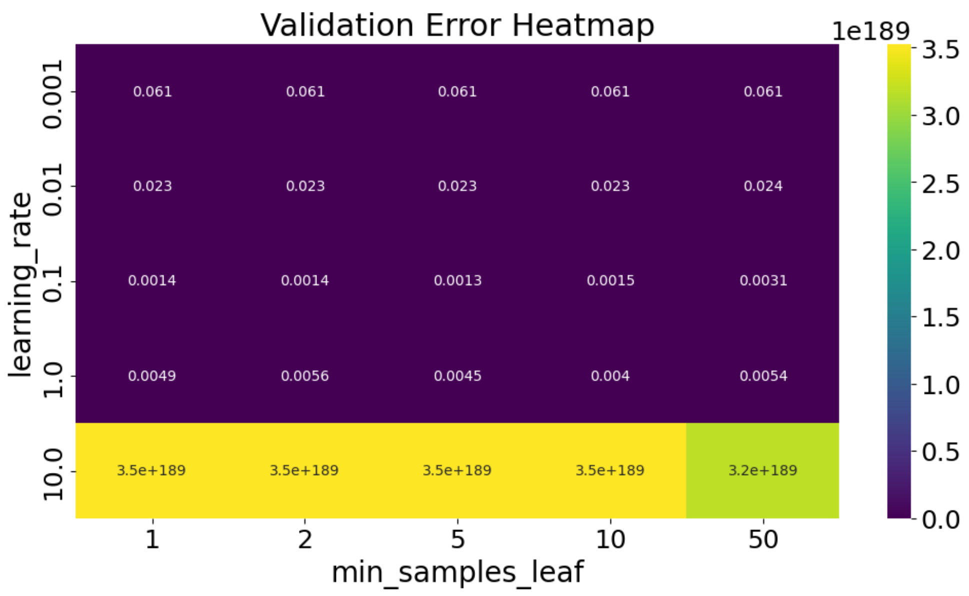


(A17)

Model	***Section***	***Train MSE***	***Valid MSE***
		Cross validated	
Dummy	***2***	***0.0684***	***0.0685***
Linear	***2***	***0.0072***	***0.0075***
Lasso	***3***	***0.00719***	***0.00748***
Polynomial Lasso	***4***	***0.0003***	***0.0246***

Section 5

(A18)



Best Parameters: *gbm__learning_rate: 0.1, gbm__min_samples_leaf: 5*

Best Training Error (MSE): *0.00038*

Best Validation Error (MSE): *0.00131*

(A19)

Model	***Section***	***Train MSE***	***Valid MSE***
		Cross validated	
Dummy	***2***	***0.0684***	***0.0685***
Linear	***2***	***0.0072***	***0.0075***
Lasso	***3***	***0.00719***	***0.00748***
Polynomial Lasso	***4***	***0.0242***	***0.0246***
GBM Regressor	***5***	***0.00038***	***0.00131***

Section 6

(A20)

Model	*Section*	*Train MSE*	*Valid MSE*	*Test MSE*
		Cross validated		*Retrained*
Dummy	*2*	*0.0684*	*0.0685*	*0.06790*
Linear	*2*	*0.0072*	*0.0075*	*774.87522*
Lasso	*3*	*0.00719*	*0.00748*	*778.03870*
Polynomial Lasso	*4*	*0.0003*	*0.0246*	*1097182.1595*
GBM Regressor	*5*	*0.00038*	*0.00131*	*0.47130*

Although all of our models underperformed compared to our baseline, the intelligent dummy, the GBM Regressor performed best on the test set as it had the best outcome with the unseen data. We can see that the Linear Regressor and the Lasso have somewhat failed to generalize to unseen data, furthermore as the results are very similar, we can also see that our linear regressor is on par with the lasso regressor. At the bottom of it all the Poly' Lasso had extremely overfitted the training data (as expected because of the small alpha -> high complexity of the model we chose).

In conclusion, GMB (which is more robust and less prone to overfitting) performed best – as expected when comparing an ensemble model to regular & simpler models.