Introduction to Machine Learning (02360766)

Homework 3

Submitted by: Gal Porat, Eyal Amdur

ID: 318849270, 315116095

Section 1: Linear regression implementation

(Q1) The analytical partial derivative $\frac{\partial}{\partial b} L(w,b) \in \mathbb{R}$ is:

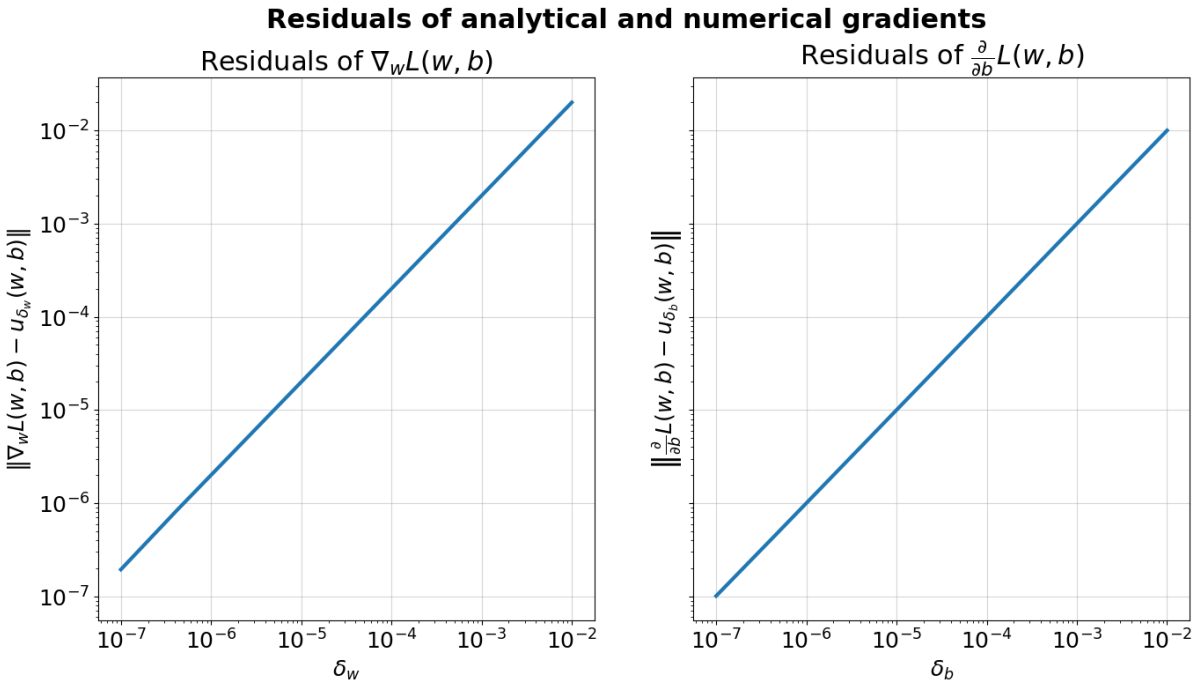
$$\frac{\partial}{\partial b} \boldsymbol{L(w,b)} = \frac{\partial}{\partial b}\left(\frac{1}{m}\sum_{i=1}^{m}(w^T x_i + b - y_i)^2\right) = \frac{\partial}{\partial b}\left(\frac{1}{m}\sum_{i=1}^{m}\left((w^T x_i - y_i) + b\right)^2\right) =$$

$$= \frac{\partial}{\partial b}\left(\frac{1}{m}\sum_{i=1}^{m}((w^T x_i - y_i)^2 + 2(w^T x_i - y_i)b + b^2)\right) =$$

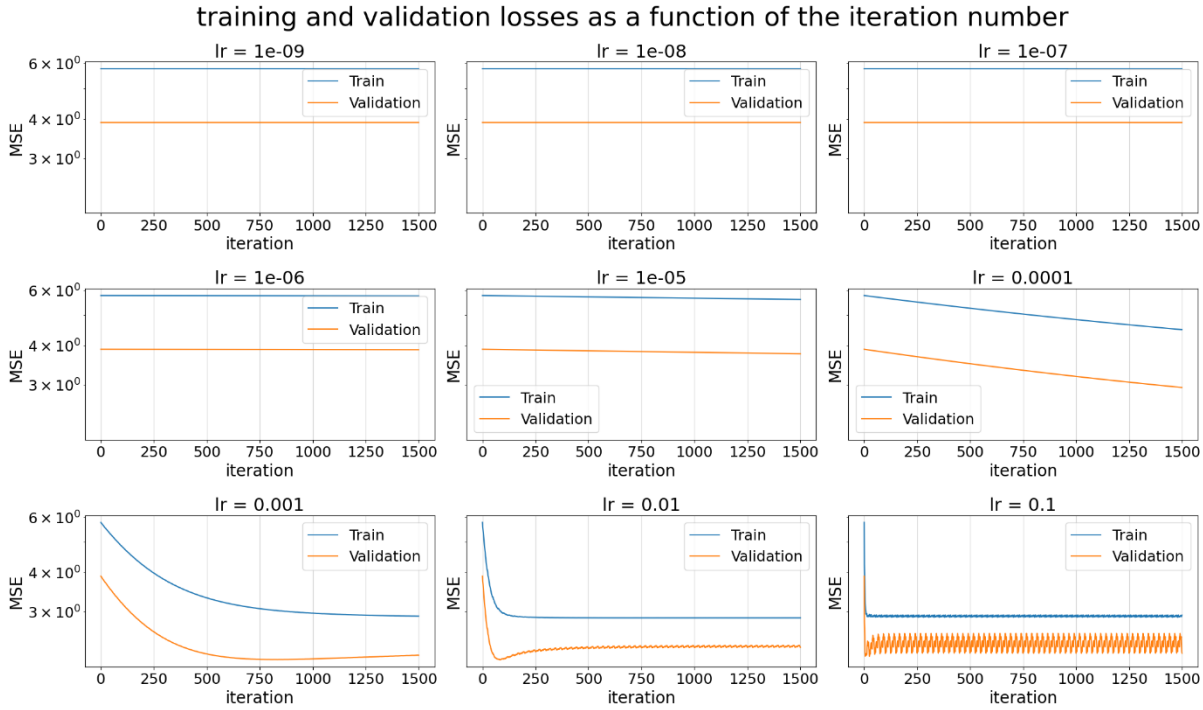
$$= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b}((w^T x_i - y_i)^2 + 2(w^T x_i - y_i)b + b^2) =$$

$$\frac{1}{m}\sum_{i=1}^{m}(2(w^T x_i - y_i) + 2b) = \boldsymbol{\frac{2}{m}\sum_{i=1}^{m}(w^T x_i + b - y_i)}$$

(Q2) Our results:



(Q3) Here are the graphs of the learning rates:



As we can see, as we increase the learning rate, there is an improvement in the rate of error reduction, until we reach to learning rate of `0.001`. If we increase more, the MSE graph becomes less stable. We can explain this by noting that for `1500` iterations, small steps are not "sufficient" to improve the model and reduce the error, whereas larger steps lead to faster convergence (But steps that are too big will cause divergence).
The "optimal" learning rate is `0.001` lr because it achieves stable convergence in about `750` iterations. There's no need to add more iterations, as they will not improve the error.

If we want to achieve the minimum error with the minimum number of iterations, we should choose `0.01` lr. Although it is less stable than `0.001`, it converges faster to a better error.
Note that for `1500` iterations the learning rate of `0.001` is better (lower MSE).

Section 2: Evaluation and Baseline

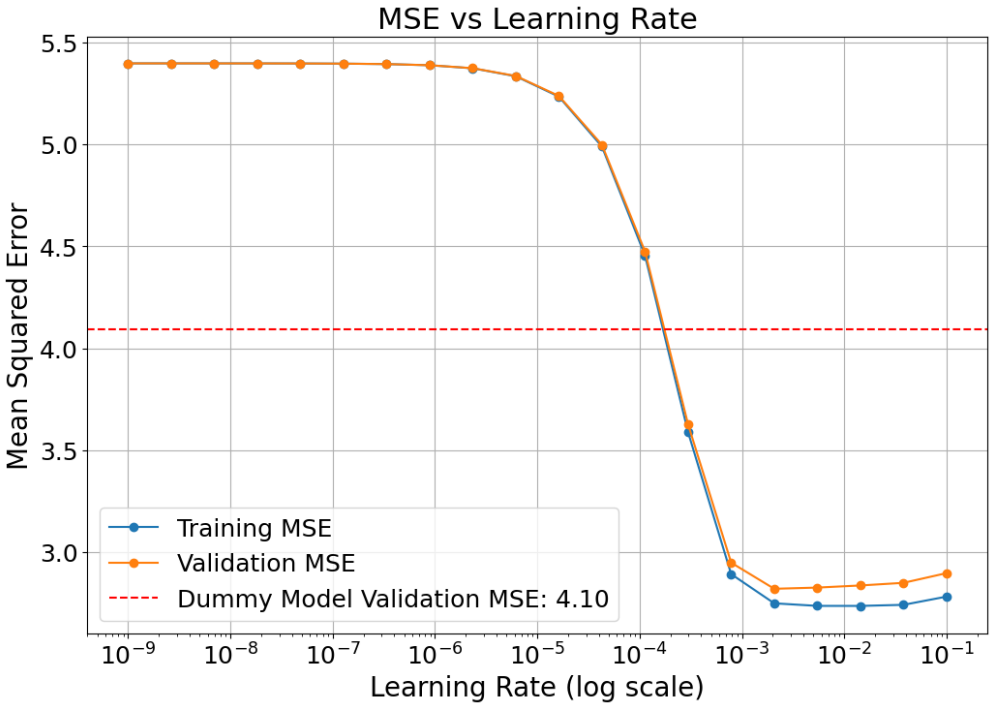
Simplest baseline

(Q4) The table for MSE per model is:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	4.078714238773915	4.09503407913296

Basic hyperparameter tuning

(Q5) The plots and values detailed in "the repeated tuning process" are:




```
Optimal learning rate:  0.002069138081114 79
Validation score:  2.820063743420568
```

And the updated table is:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	4.078714238773915	4.09503407913296
Linear	2	2.7487758023823132	2.820063743420568

(Q6) Had we chosen not to normalize features beforehand, we would get:

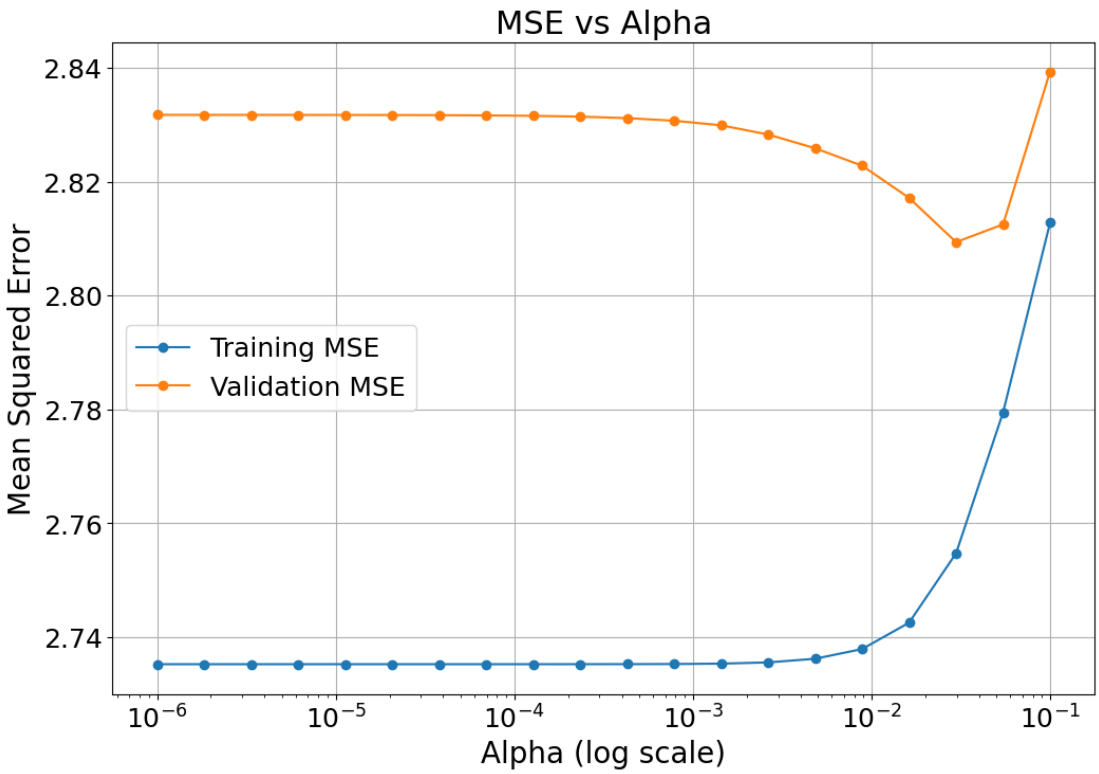
- Dummy model: as for the Dummy model, it doesn't care about the input features X_train because it makes predictions based only on the labels y_train, like their average. This means normalizing the features doesn't make any difference to how the model works or what results it gives.

Since the label's average stays the same no matter what you do with the features, the predictions will always be the same whether the data is normalized or not.

- Linear model: as for the Linear model, if we don't normalize the features, the training performance won't change (assuming no numerical errors and no regularization). Linear regression is a convex problem, so while the weights (=vector w) may differ without normalization, the solution will still minimize the loss function optimally, yielding the same predictions and performance.

Section 3: Lasso linear regression

(Q7) The plots and values detailed in "the repeated tuning process" are:




```
Optimal alpha:  0.029763514416313194
Lasso validation score:  2.8093780661720293
```

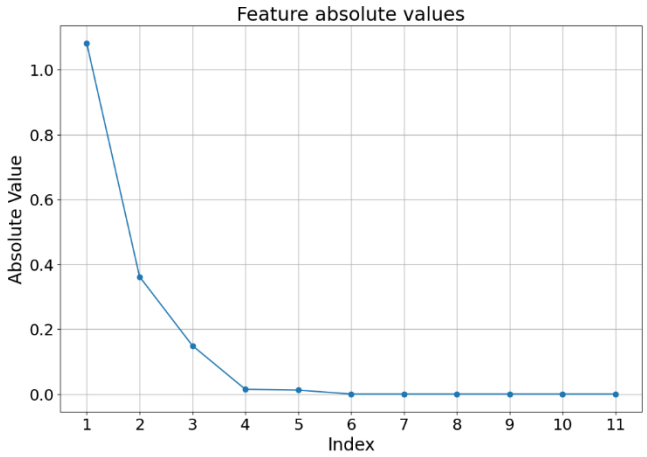
(Q8) And the updated table is:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	4.078714238773915	4.09503407913296
Linear	2	2.7487758023823132	2.820063743420568
Lasso Linear	3	2.754753173159186	2.8093780661720293

(Q9) The 5 features having the 5 largest coefficients (in absolute value) in the resulting regressor, from the largest to the smallest are:

```
happiness_score    1.082315
PCR_03             0.361828
PCR_06             0.149223
PCR_02             0.014576
PCR_10             0.012147
```

(Q10) The plot of the absolute values of the learned coefficients is:



(Q11) The magnitude of the coefficients is interesting because it reflects feature selection. Lasso uses L1-regularization, which can shrink some coefficients to exactly zero, like we got on our report, effectively removing less important features from the model. By examining the magnitudes, we can identify which features contribute most to the predictions and which are irrelevant, helping with dimensionality reduction.
Normalizing the features before applying Lasso ensures that all features contribute equally to the regularization process, allowing for a fair comparison of coefficient magnitudes and preventing features with larger scales from dominating the penalty, ultimately enhancing the model's ability to generalize well to unseen data.

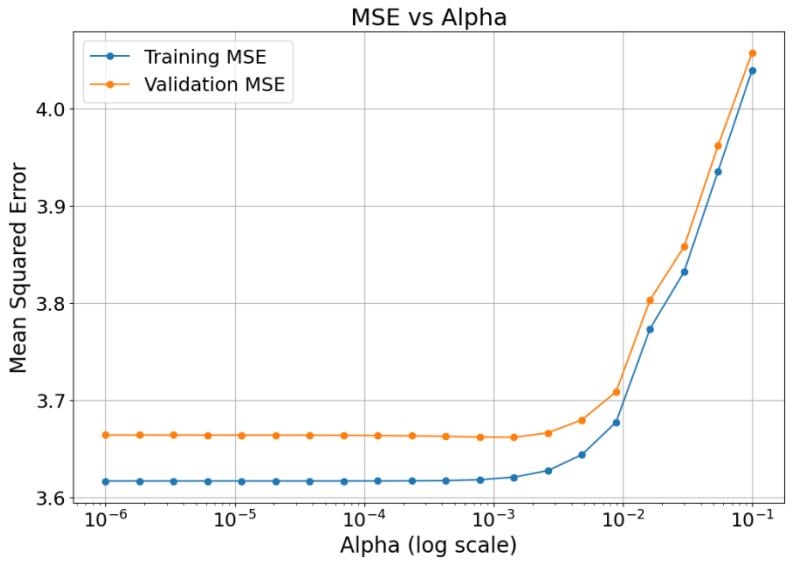
(Q12) If we hadn't normalized the features before running the Lasso regression, the training performance of the model could definitely be different. Lasso uses L1 regularization, which adds a penalty based on the size of the coefficients. If the features are on different scales, the ones with larger values could end up having a bigger impact on the model than they deserve, while smaller-scaled features might get pushed down more than they should.
This means the model might end up favoring certain features just because they have larger numbers, rather than based on how important they really are. So, without normalization, we might end up with a model that doesn't really capture the true relationships in the data and could perform poorly on new data.

(Q13) If we had used Ridge regression instead of Lasso regression, the coefficients of the trained model would likely be different in that Ridge would not shrink any coefficients to zero due to its use of L2 regularization, which penalizes the square of the coefficients. As a result, we would expect to see all coefficients remain non-zero, even for less important features, leading to a more uniform distribution of coefficients across the board. Ridge tends to reduce the magnitude of all coefficients more evenly, rather than selectively eliminating some as Lasso does, which means it does not perform automatic feature selection. This could be beneficial when many features contribute to the output, although it may also result in a more complex model.

Section 4: Polynomial fitting (visualization)

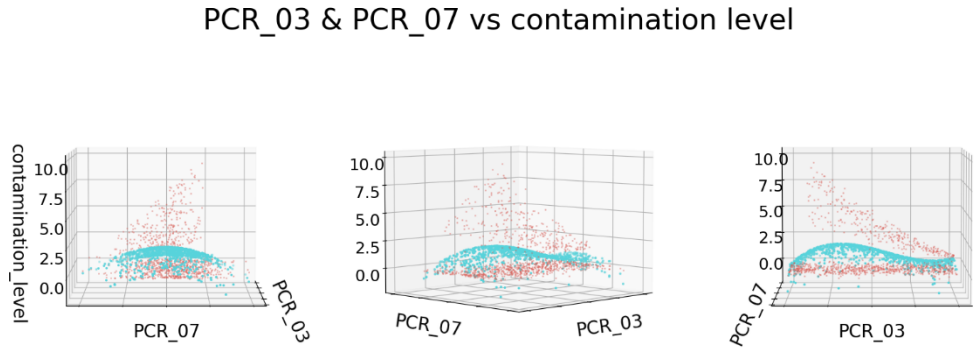
(Q14) Given a certain correlation between the normalized features, we want to maintain this relationship to assign appropriate weights to the model. When we map the features to a higher dimension, we may disrupt the relationship between them and change it exponentially, which could lead to a bias in favor of the feature with the larger values, giving it a higher weight than it had before the mapping. Therefore, normalization is required after mapping the features to a higher dimension to preserve the relationship between the features.

(Q15)




```
Optimal strength (alpha):  0.0014384498882876629
Optimal validation error:  3.6614743875130586
```

(Q16)

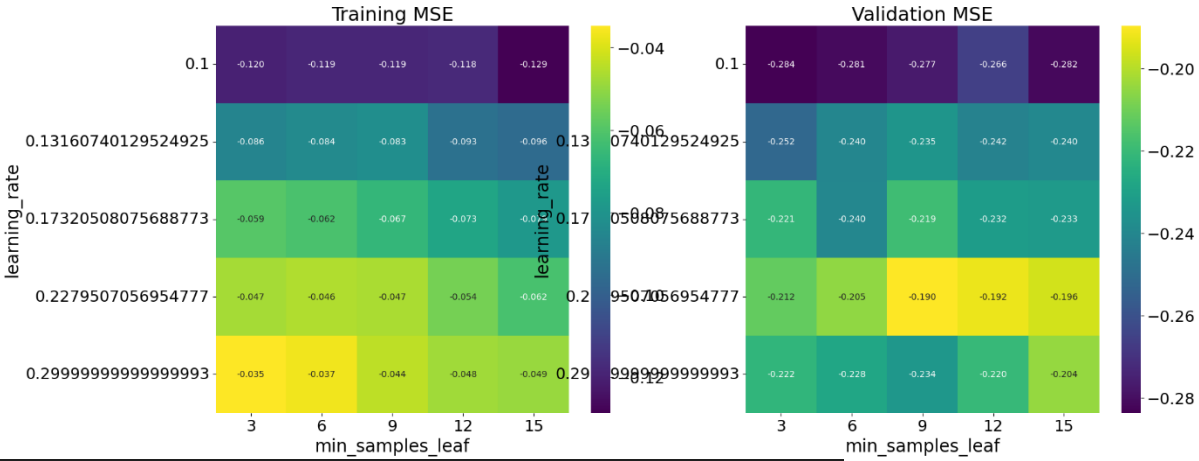


(Q17)

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	4.078714238773915	4.09503407913296
Linear	2	2.7487758023823132	2.820063743420568
Lasso Linear	3	2.754753173159186	2.8093780661720293
Polynomial Lasso	4	3.620445127100507	3.6614743875130586

Section 5: Fitting Gradient Boosted Machines (GBM) of the CovidScore

(Q18) plotting the heatmaps:




```
Optimal learning rate: 0.2279507056954777
Optimal min_samples_leaf: 9

Minimum Train MSE:  0.04677132038127515
Minimum Validation MSE:  0.18907504911141004
```

(Q19)

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	4.078714238773915	4.09503407913296
Linear	2	2.7487758023823132	2.820063743420568
Lasso Linear	3	2.754753173159186	2.8093780661720293
Polynomial Lasso	4	3.620445127100507	3.6614743875130586
GBM Regressor	5	0.04677132038127515	0.18907504911141004

Section 6: Testing your models

(Q20)

Model	Section	Train MSE	Valid MSE	Test MSE
		Cross validated		Retrained
Dummy	2	4.078714238773915	4.09503407913296	3.7741727683912414
Linear	2	2.7487758023823132	2.820063743420568	2.6505045808180716
Lasso Linear	3	2.754753173159186	2.8093780661720293	2.6451604777125284
Polynomial Lasso	4	3.620445127100507	3.6614743875130586	3.4276078892887076
GBM Regressor	5	0.04677132038127515	0.18907504911141004	0.17646621933131926

The GBM Regressor clearly provides the best results on the test set, achieving the lowest Test MSE of 0.17646. This aligns with expectations, as GBM combines the strengths of multiple models, assigning weights based on their performance to optimize predictions. Its ability to capture complex, non-linear patterns while avoiding overfitting makes it superior to the other models evaluated.

In contrast, the Dummy Regressor suffers from significant underfitting, as it predicts the average target value of the training set for all new data. This simplistic approach leads to the worst performance across all metrics, with the highest Test MSE.
Meanwhile, the Linear and Lasso Regressors shows similar performance, which can be explained by the fact that the optimal regularization parameter alpha for Lasso was found to be very small. This effectively reduces the impact of the regularization term, making the models very similar.
Mapping features to 3th-degree polynomials should improve performance in the Polynomial Lasso model, but as it introduced to only 2 features, it had less data to train on and therefore got higher Test MSE.
Overall, GBM's ability to combine multiple weak learners and assign weights to the best-performing ones makes it the most effective model in this scenario, as we have seen in class. It strikes the best balance between model complexity and generalization, leading to its exceptional results.