I got a permission from the TA in charge to submit alone.

Eva Poluliakhov 321882649

Section 1: Linear regression implementation

(For **this section only**, split your training set into a (new) training subset (80%)and a validation subset (20%))

Answer Q1:

Fix $i \in [m]$:

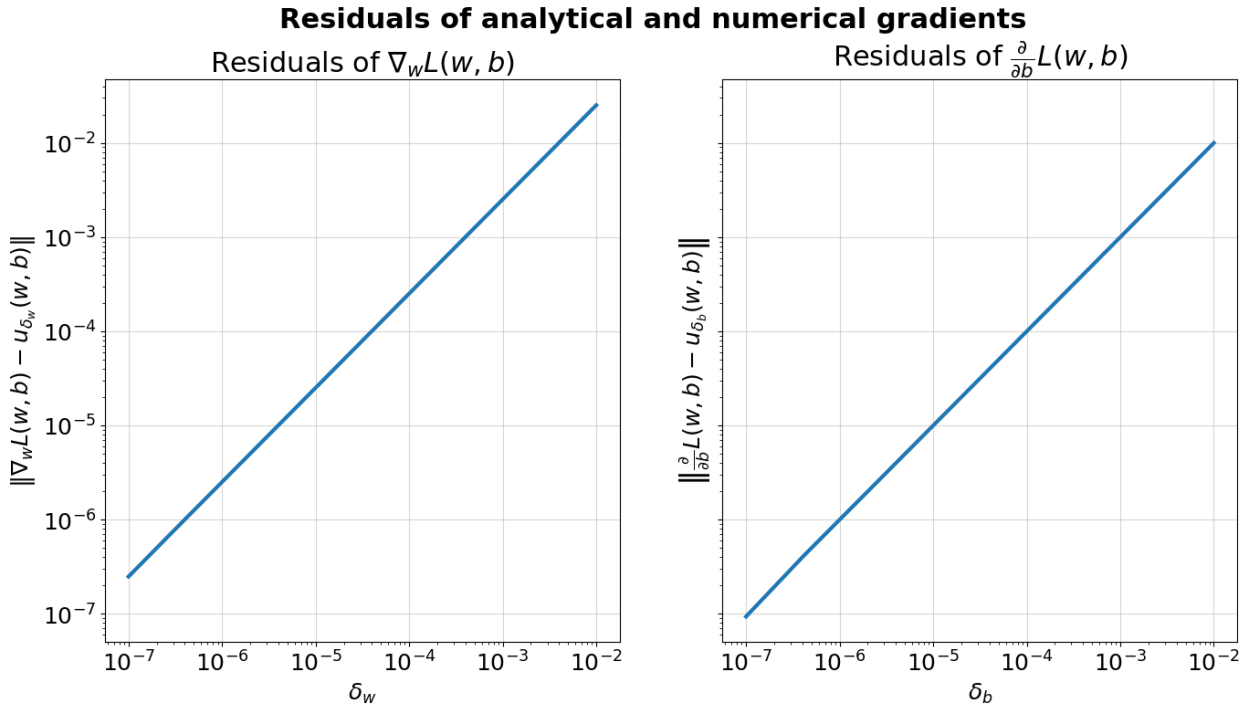
Using the chain rule: $\frac{\partial}{\partial b}\left(w^t x_i + 1_m \cdot b - y_i\right)^2 = 2 \cdot \left(w^t x_i + 1_m \cdot b - y_i\right) \cdot 1_m$

Thus using linearity of derivative:

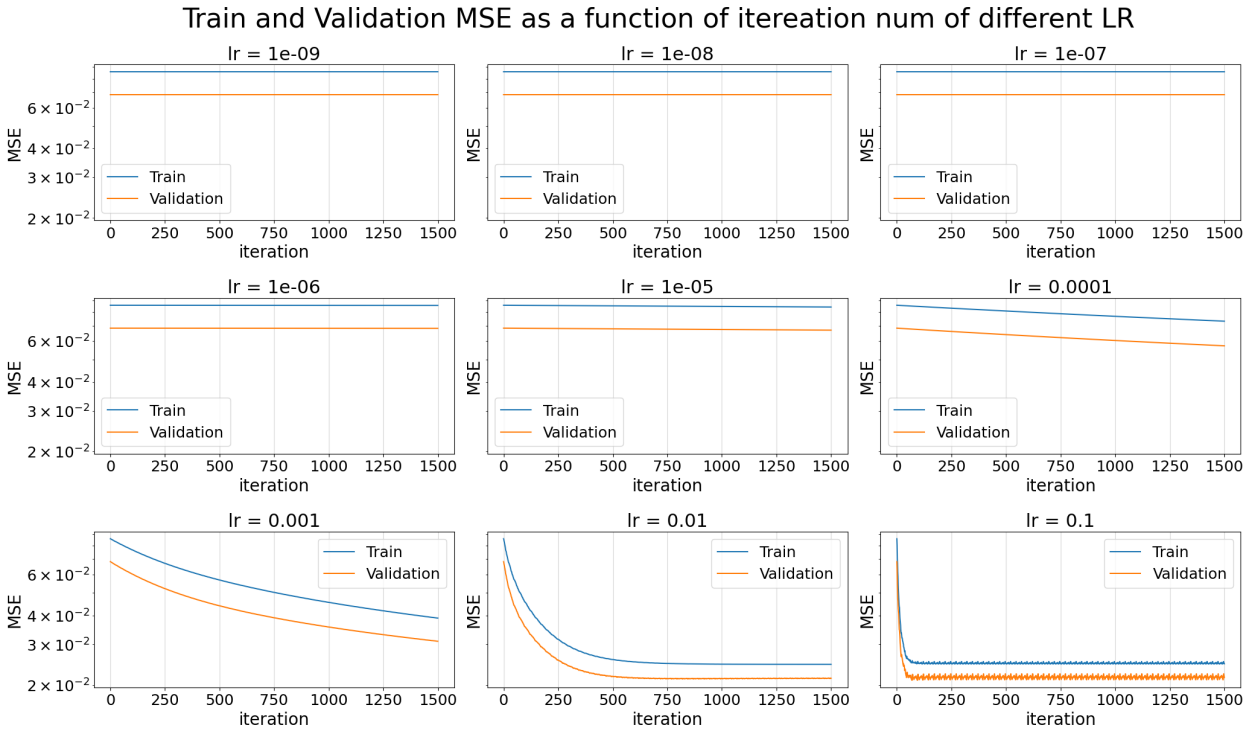
$\frac{\partial L}{\partial b}(w,b) = \frac{1}{m} \cdot \sum_{i=1}^{m} 2 \cdot (w^t x_i + 1_m \cdot b - y_i) 1_m = \frac{2}{m} \sum_{i=1}^{m} (w^t x_i + 1_m \cdot b - y_i) 1_m =$

$\frac{2}{\mathrm{m}} 1_{\mathrm{m}}^{\mathrm{T}} (Xw + 1_m \cdot b - y)$

Answer Q2:



Answer Q3:



It looks like that for very small learning rates (step size) $lr \leq 10^{-5}$ both the train and the validation sets have low accuracy. This could be because the step size is too small and GD algorithm needs more iterations than 1500.

When $10^{-5} < lr \leq 10^{-3}$ the MSE decreases (and the accuracy increases) although convergence is yet obtained. This could indicate the learning rate is fitting better but is still too small.

For $lr = 0.01$ MSE converge both for train and validation sets, after a small enough number of iterations.

Lastly, for $lr = 0.1$ the MSE decreases very fast for both train and validations sets but MSE doesn't converge, meaning the learning rate is too large and in advance the GD algorithm misses the minimum point.

All together the best learning rate is $lr = 0.01$. This is the only learning rate that converge in the default number of iterations.

It would not make sense to increase number of iterations because we found a learning rate that fits the train and validation sets and converges in the given number of iterations. Since we already achieved convergence, the gradient in each iteration starting from the convergence would be 0, hence the GD will not effect the next iterations.

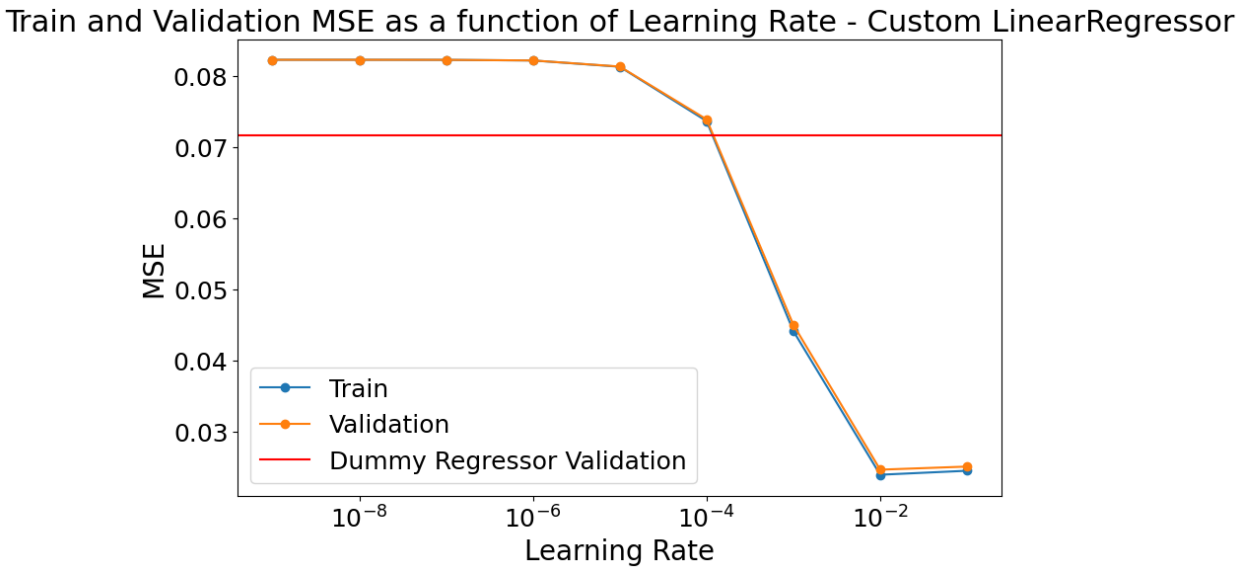
Section 2: Evaluation and Baseline

Answer Q4:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	0.0712949205807158	0.07165828897657857

Answer Q5:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	0.0712949205807158	0.07165828897657857
Linear	2	0.02388930002514985	0.024602601806949582



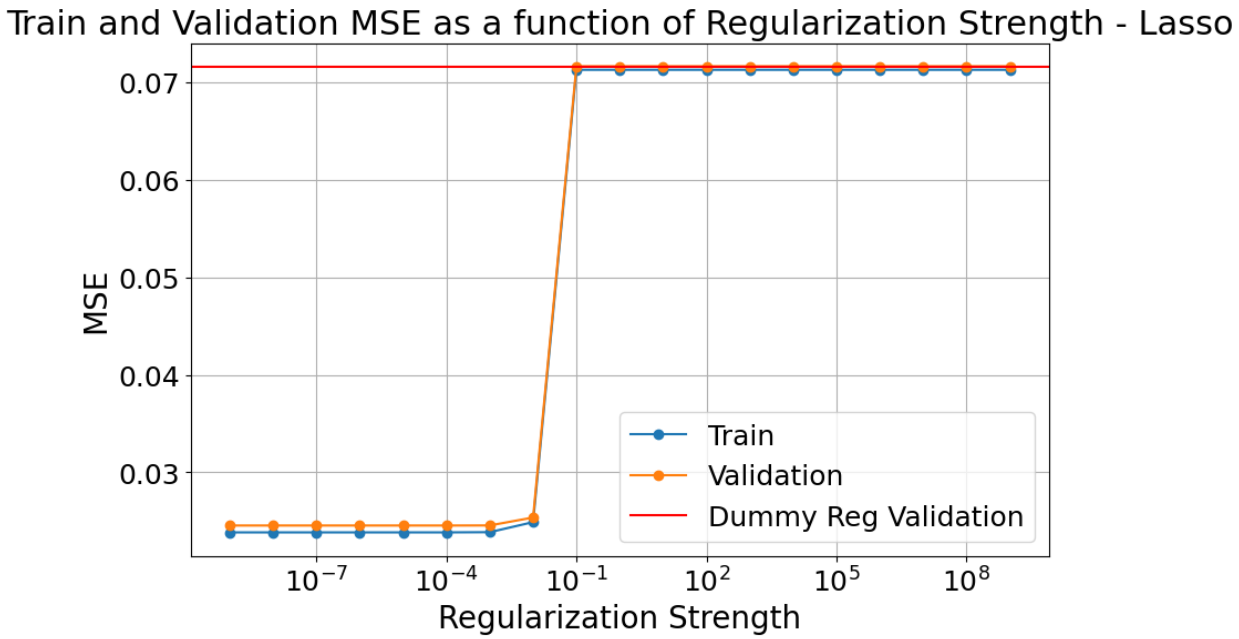

```
Optimal Learning Rate: 0.01
Validation Error: 0.024602601806949582
Train Error: 0.02388930002514985
```

Answer Q6:

Yes, without normalization the training performance would change.
First of all the loss function is MSE which is calculated by distance from the prediction to the correct label. When using raw data, outliers will have greater impact on the total error, although when normalizing, each error in prediction will have the same impact (on average) on the total error.
Moreover, un-even scaling could damage the GD algorithm performance. If there are outliers, we would want a greater step size in order to have convergence in appropriate number of iterations, but too large step size could damage the convergence in the matter of missing the minimum point.
Lastly, the best learning rate was chosen in hyperparameter tuning using cross-validation. When the data has high variance across folds, the validation error could be biased which could lead to choosing a different incorrect learning rate that we could less rely on it.
All together normalization is important in order to achieve the obtained training performance.

Section 3: Lasso linear regression

Answer Q7:




```
Optimal Regularization Strength (alpha): 0.0001
Validation Error: 0.02453804977770329
Train Error: 0.023834352618536225
```

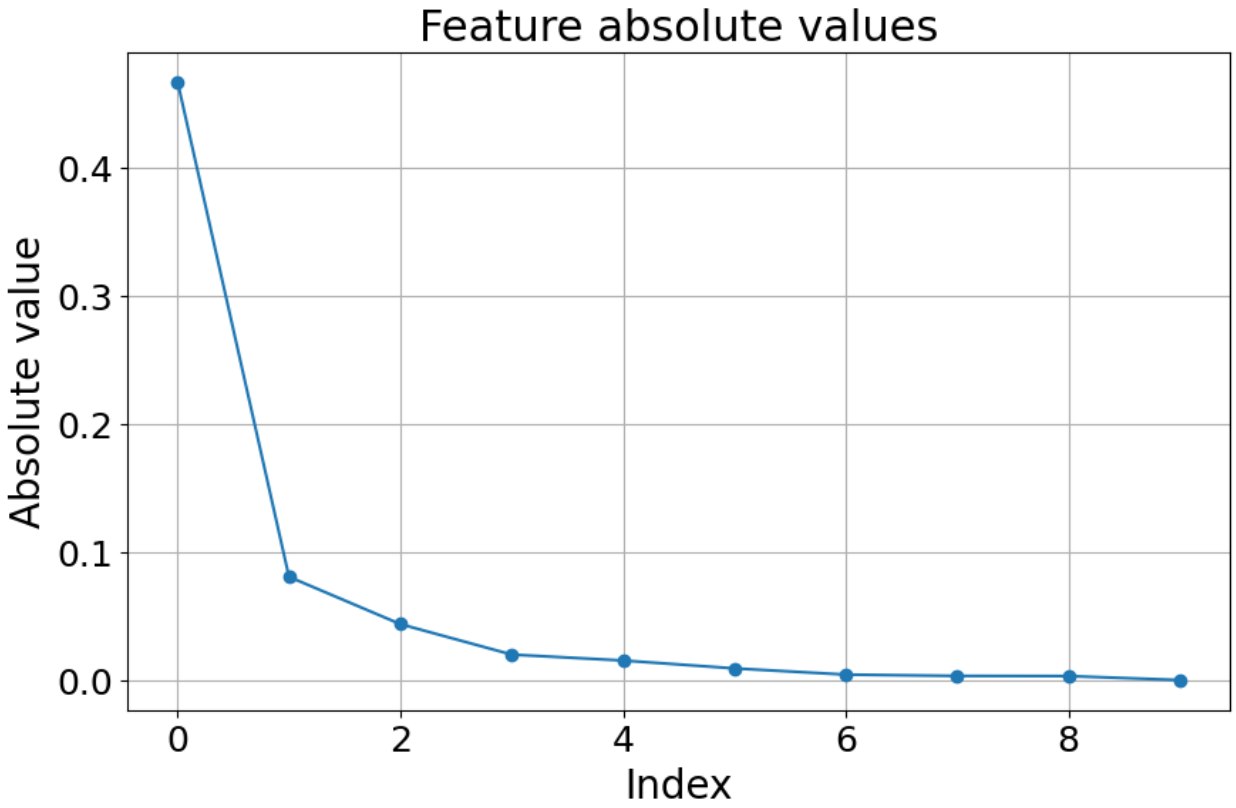
Answer Q8:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	0.0712949205807158	0.07165828897657857
Linear	2	0.02388930002514985	0.024602601806949582
Lasso Linear	3	0.023834352618536225	0.02453804977770329

Answer Q9:

```
Features with the largest coefficients (in absolute value):
  Feature  Coefficient
3  PCR_04     0.467117
1  PCR_02    -0.080873
5  PCR_06    -0.043937
8  PCR_09    -0.020178
9  PCR_10     0.015542
```

Answer Q10:



Answer Q11:

The lasso regressor will set a small or even zero coefficient for features that don't help the regression results enough. That means that the plot we generated in Q10 can give information about what features, compared to others, improve the fit and performance of the model and help us choose and focus on them.

Answer Q12:

If we chosen not to normalize features beforehand, the training performance of the Lasso model probably have changed. The lasso objective optimization is performed using GD algorithm. Just as mentioned before, in case of the outliers, a large learning rate might be chosen although it may disrupt convergence. Also we seen in the lecture that outliers may change and mislead the slop of the linear regressor, which wont classify the training data correctly. This is because features with larger values scale have more contribution than features with smaller value due to the square loss.

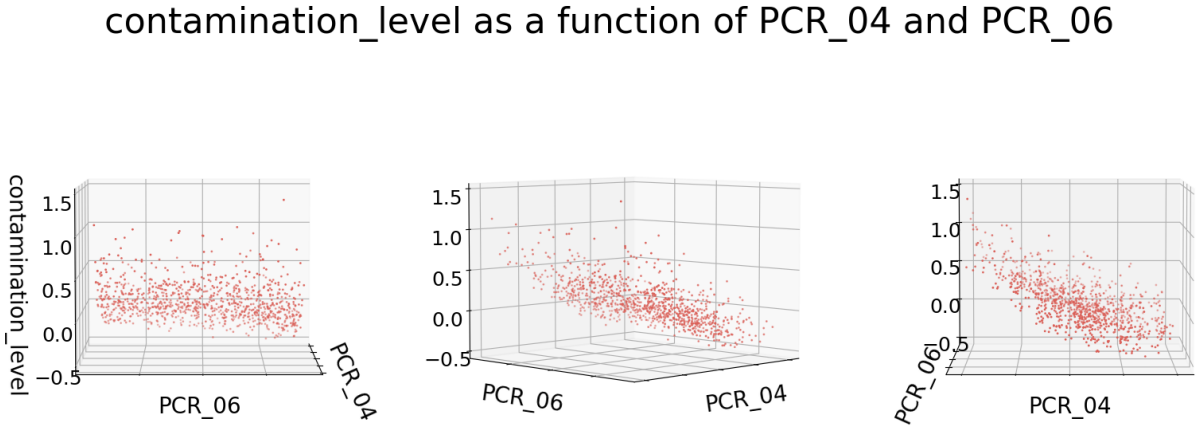
Answer Q13:

Without normalization, in the lasso objective, features with larger scales will get smaller coefficients and features with smaller scales will get bigger coefficients, in order to optimize the objective.

It is important to note that ridge and lasso regressors technically differ only by the norm each uses. That means that they're both regression is based on distance and both would be effected by un-normalized features. Thus, using ridge regressor instead of lasso regressor won't solve the problem of un-normalized features.

If we chose to use the ridge regressor, as we seen in the tutorial, lasso regression yields sparse solution although by using ridge regression, the irrelevant features coefficients would only shrink but as long as the regulation parameter not too large they would not equal zero. So the effect on the coefficient of the ridge regression would be less zero's and decay in the coefficient as the learning rate grows.

Section 4: Polynomial fitting (visualization)

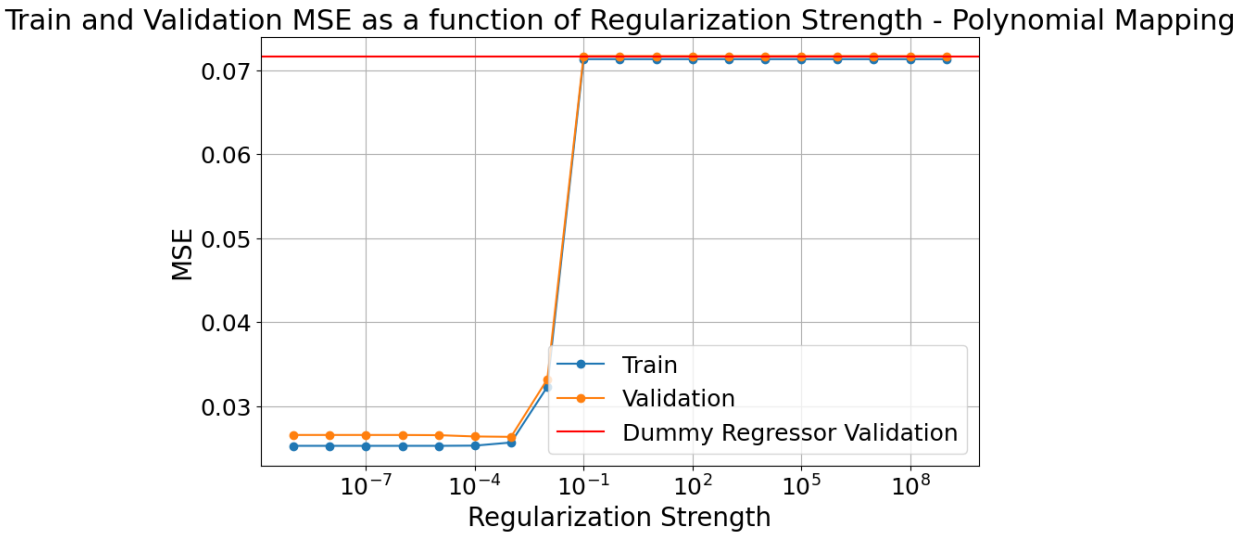


Answer Q14:

The product of the polynomial feature mapping is all the possible combinations of the features to the degree we set. For instance, in our case, denote some feature as X, the product includes the monomials x and x^4. It is clear that their image differs and has different range and scaling. Since after using the polynomial feature mapping we want to use the lasso regressor, it is important to normalize the features in order to give the correct coefficient to each feature, based on it's actual contribution to the loss and not to the contribution due to it's large (or small) scaling (as we discussed in Q12). Moreover, we might face again the problem of optimizing the lasso model loss using GD, with the same problems we have already discussed, such as convergence and tuning the correct learning rate, due to different scaling of features.

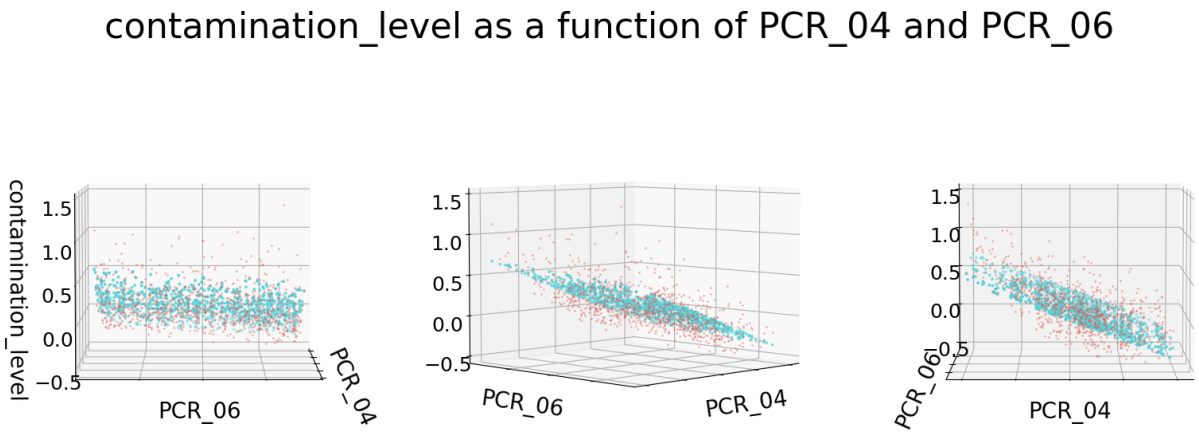
MORE IN THE NEXT PAGE

Answer Q15:




```
Optimal Regularization Strength (alpha): 0.001
Validation Error: 0.026332360843328784
Train Error: 0.025658676167310036
```

Answer Q16:

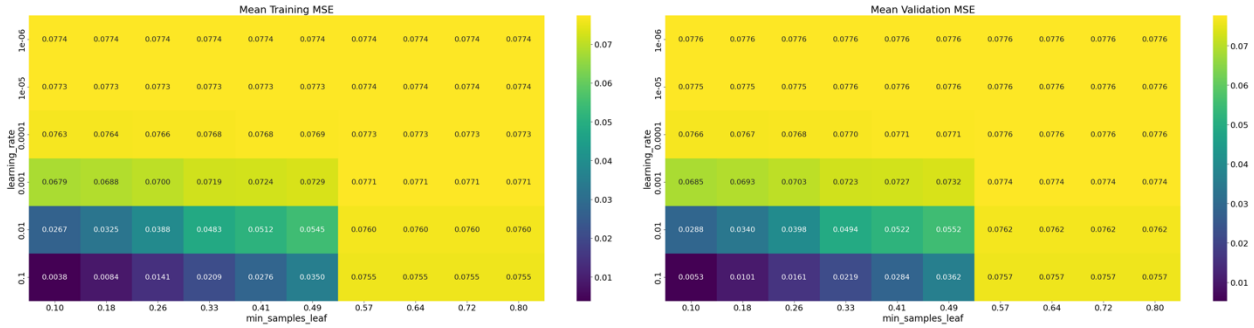


Answer Q17:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	0.0712949205807158	0.07165828897657857
Linear	2	0.02388930002514985	0.024602601806949582
Lasso Linear	3	0.023834352618536225	0.02453804977770329
Polynomial Lasso	4	0.025658676167310036	0.026332360843328784

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

Answer Q18:




```
Best parameters found: {'GBM__learning_rate': 0.01,
'GBM__min_samples_leaf': 0.05}
Best Train Error: 0.02216355423422791
Best Validation Error: 0.02412640336497494
```

Answer Q19:

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	0.0712949205807158	0.07165828897657857
Linear	2	0.02388930002514985	0.024602601806949582
Lasso Linear	3	0.023834352618536225	0.02453804977770329
Polynomial Lasso	4	0.025658676167310036	0.026332360843328784
GBM	5	0.023124238013163705	0.025538271366585923

Section 6: Testing your models

Answer Q20:

Model	Section	Train MSE	Valid MSE	Test MSE
		Cross validated		Retrained
Dummy	2	0.0712949205807158	0.07165828897657857	0.05582772158839282
Linear	2	0.0238893000251 4985	0.024602601806949582	0.02462522597776613
Lasso Linear	3	0.023834352618536225	0.02453804977770329	0.024529052086867103
Polynomial Lasso	4	0.025658676167310036	0.026332360843328784	0.026686267183952915
GBM	5	0.0221635542342 2791	0.02412640336497494	0.018962832765696276

Given the full details, it looks like the **GBM model performs the best** out of the other models. First, the MSE of the GBM is the lowest of all and is not underfitted or overfitted. If it was overfitted, we would have seen good performance on train and validation but poor performance on test, which did not occur. It is also not underfitted because it has a good performance (=low MSE) and it is able to capture the relation between features and prediction.

It is very prominent that the **dummy regressor underfits** the data, since it performs poorly both on cross validation and on test set.

Our custom **linear regressor and the lasso linear regressor** have similar results, and seem to **fit the data good** since on cross-validation and test set MSE do not differ, indicating the learnt models are not underfitting and not overfitting, although their accuracy is not as good as the GBM performance.

Lastly, the **polynomial lasso** has stable predictions and test MSE compared to itself cross-validation results although it has worse errors then the linear and the lasso linear regressors, meaning the model predicts the data well but not as good as the linear and lasso linear models. This could show a **slight overfit** in the learnt model compared to the lasso linear model, maybe due to using polynomial features the polynomial lasso model fits more complex patterns that damage it's ability to generalize.

We indeed expected the GBM to perform the best. First due to huber loss, which is less sensitive to outliers, secondly we tuned the learning rate that has an important impact in dealing with overfitting because the lower the lr the less each tree contributes to the final learnt model, and we also tuned the min_samples_leaf, allowing early stopping and fitting the model better. Moreover the GBM fit each tree based on the previous trees residuals, this allows the model to refine the areas where the performance was less good and to improve on them in the current iteration.