Section 1: Linear regression implementation

1.

Multiplying by is like summing the vector and we get a scalar as expected.

2.

A graph of a line

Description automatically generated with medium confidence

3.

A graph of a test

Description automatically generated with medium confidence

We can see that when the leaning rate is too small we don’t see any learning because every step doesnt change the gradients enough. When the learning rate is too high we can see a spiked curve because the update steps miss the minima. When the learning rate is right (e.g lr = 0.001) e get a smooth decaying curve converging to the minima.

If we’re looking at lr = 0.001 it makes sense to increase the number of steps because it seems we didnt reach a platue and with more steps we can achieve lower loss and a better model. more steps can lead to overfitting so we should also implement early stopping.

If we’re looking at lr = 0.01 it doesn’t make much sense to increase the number of steps because it seems we already reached a platue and won’t gain any gains from that.

4.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Section | Train MSE | Valid MSE |
|  |  | Cross validated | |
| Dummy | 2 | 3.727 | 3.738 |

5.

A graph with a line graph and a chart with text

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Section | Train MSE | Valid MSE |
|  |  | Cross validated | |
| Dummy | 2 | 3.727 | 3.738 |
| Linear | 2 | 2.513 | 2.642 |

6. If we wouldn’t have normarlized the features beforehand it wouldn’t affect the Dummy model because it only cares about the labels (it takes avg of it) and not the features.

It would affect our model because without normalization dimensions may differ in scale considerably and we are using a single learning rate for all of them, which probably wouldn’t suit all the different scales.

7.

A graph with a line graph and text

Description automatically generated with medium confidence

Best reg-



8.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Section | Train MSE | Valid MSE |
|  |  | Cross validated | |
| Dummy | 2 | 3.727 | 3.738 |
| Linear | 2 | 2.513 | 2.642 |
| Lasso | 3 | 2.504 | 2.627 |

9. top 5 coefs in abs value

A black screen with white text

Description automatically generated

10.

A graph with a line

Description automatically generated

11. The magnitute of the coefficients are intresting because they tell us to what degree each feature is correleted (affects) with the result.

12. If we didn’t normalize It would probably affect the results. Also for the reason stated at q6 and also because the L1 regularization on unscaled features wouldn’t fit the different scales of the features and effictivly can cause features to be ignored or over considered.

13. Usually Lasso promotes sparse solutions but we see in our case where the optimal alpha is 0.01 the L1 regularization did not yet zeroed out the coefficients. When looking at the graph at tutorial 9 about different regularization behaviors we can assume that the low reg might had a weak affect on the coefficients, and so it would stay more or less the same.

Lets check by fitting a ridge regressor and plotting the coefficients:

A graph with a line

Description automatically generated

Seems about the same.

14. it’s important to apply re-normalization after the polynomial mapping because higher power monomials can cause features to have a value at different scale than the rest. If the orginal value is > 1 then the result can be very large and infulance regardless of their actual importance. If < 1 then values might vanish.

15.

A graph with a line graph and text

Description automatically generated with medium confidence

Optimal alpha



16.

A graph of a graph

Description automatically generated with medium confidence

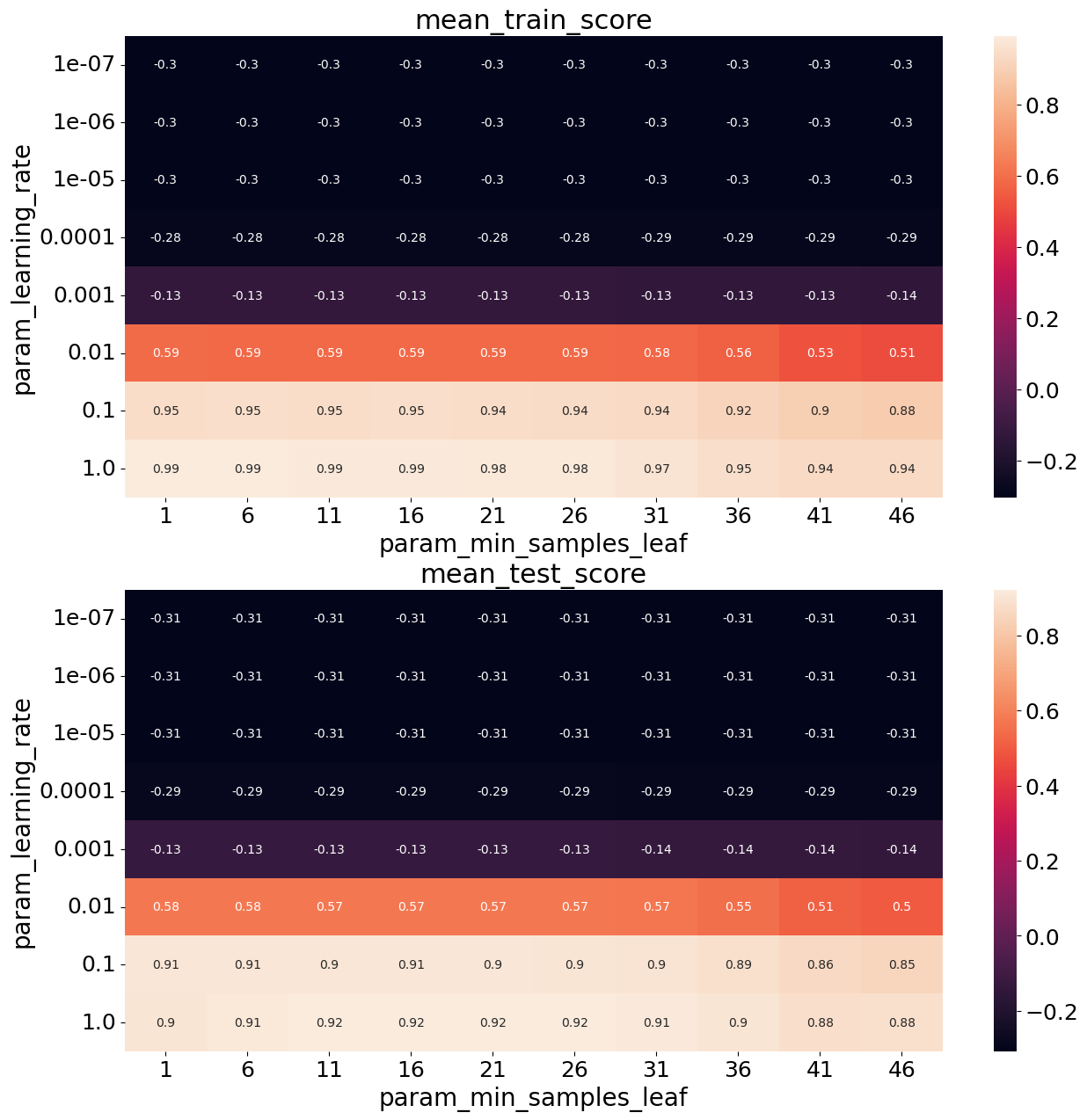
17.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Section | Train MSE | Valid MSE |
|  |  | Cross validated | |
| Dummy | 2 | 3.727 | 3.738 |
| Linear | 2 | 2.513 | 2.642 |
| Lasso | 3 | 2.504 | 2.627 |
| Polynomial Lasso | 4 | 3.360 | 3.414 |

18.

We did some unvariate analysis, used the built in GBR.feature\_importance and sklearns feature selection RFE and decided to use the columns:





Best validation score and params:



And best train score is 1 with learning\_rate = 1 and min\_samples = 1

19.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Section | Train MSE | Valid MSE |
|  |  | Cross validated | |
| Dummy | 2 | 3.727 | 3.738 |
| Linear | 2 | 2.513 | 2.642 |
| Lasso | 3 | 2.504 | 2.627 |
| Polynomial Lasso | 4 | 3.360 | 3.414 |
| GBM Regressor | 5 | 0.042 | 0.276 |

20.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Section | Train MSE | Valid MSE | Test MSE |
|  |  | Cross validated | | Retrained |
| Dummy | 2 | 3.727 | 3.738 | 5.195 |
| Linear | 2 | 2.513 | 2.642 | 3.491 |
| Lasso | 3 | 2.504 | 2.627 | 3.506 |
| Polynomial Lasso | 4 | 3.360 | 3.414 | 4.495 |
| GBM Regressor | 5 | 0.042 | 0.276 | 0.176 |

GBM is the top preformer among all the regressors (noting the feature selection we did).

The dummy model under performed all other models as expected (or wished) due to obvious under-fitting.

Lasso did slightly better the Linear at training time but did worse at test time, indicating possible over-fitting (though very close).

Polynomial Lasso didn’t do good probably because it had a very limited set of features to train on compared to the other models which used all the features.