Major 3 Report

Idan Albo – 206665051, Amit Grosman - 318170214

Part 1 – Linear Regression implementation

Q1.

The MSE loss definition for the non-homogeneous linear regressor is:

$$L(w,b)=\frac{1}{m}\sum_{i=1}^{m}(w^Tx_i+b-y_i)^2=\frac{1}{m}\|Xw+1_mb-y\|_2^2$$

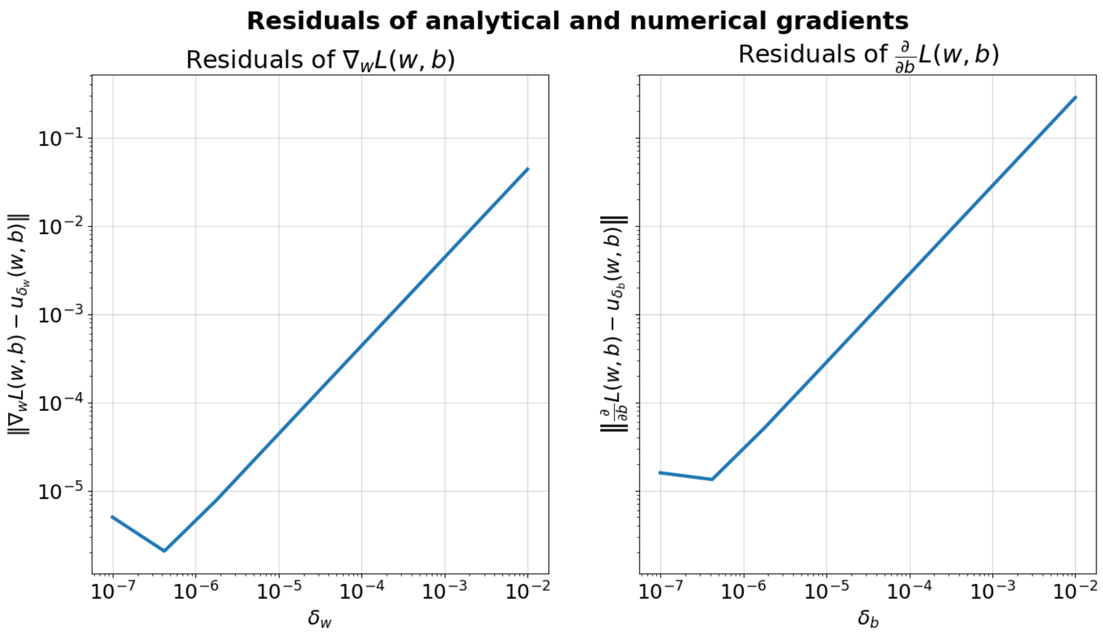
where $x=(x_1,\cdots,x_m)^T,\ \ y=(y_1,\cdots,y_m)^T,\ \ 1_m=(1,\cdots,1)^T\in\mathbb{R}^m$.

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

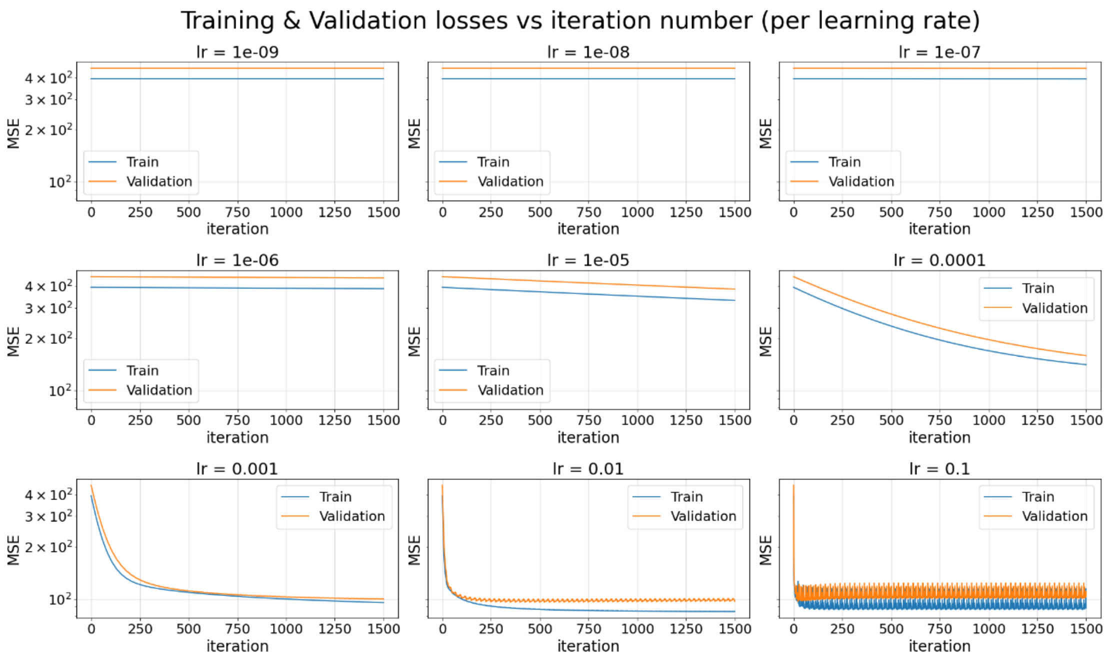
$$\frac{\partial}{\partial b}L(w,b)=\frac{\partial}{\partial b}\left(\frac{1}{m}\sum_{i=1}^{m}(w^Tx_i+b-y_i)^2\right)$$

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

Q2.



Q3.



We can see that as the learning rate decreases, the test and validation losses don't reach the minimum. This behavior can be explained by considering that we used bounded number of iterations, and for some learning rates it isn't enough to reach the minimum (or some area close enough to it).
Also, we can see that for higher values of learning rate, the algorithm reaches the minimum "area", after enough steps, but for higher values of learning rate, the step can skip the point that brings the loss to minimum, and "hovering" over it.

The optimal learning rate is 0.001, because the loss isn't fluctuating between iterations on the validation set (meaning that the learning rate isn't too high), but also big enough to converge to the minimum.

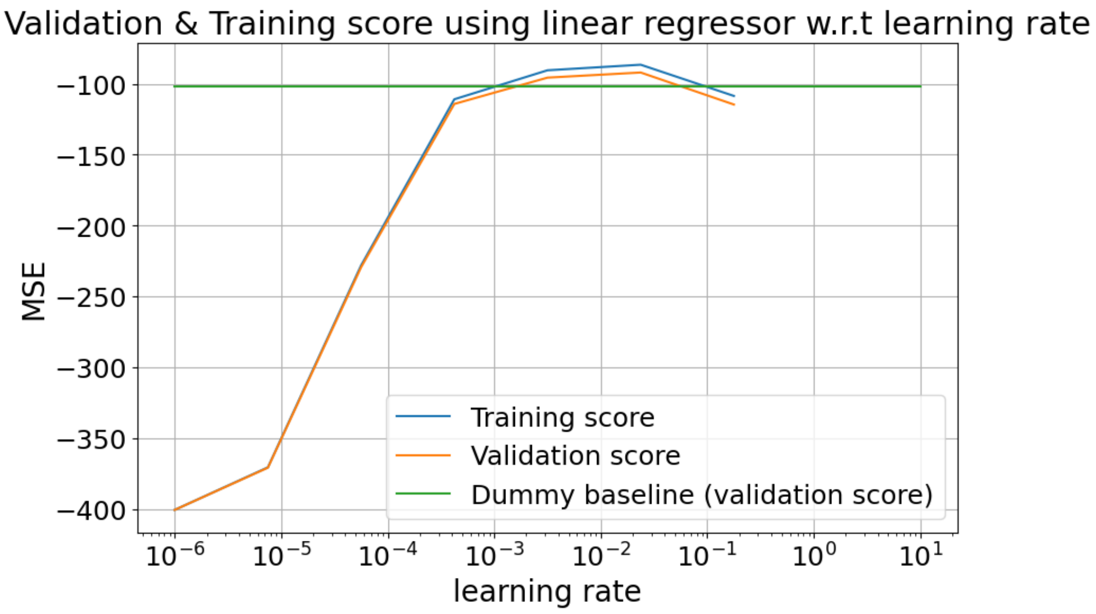
There is no need to increase the number of iterations (gradient steps), because for the learning rate stated above, the loss curve is optimal for the validation set, in a sense that the loss stop decreasing as number of iterations increases (meaning we reached the minimum in at most 1500 steps).

<u>Part 2 – Evaluation and Baseline</u>

Q4.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	101.62	101.79

Q5.



The optimal learning rate is 0.023, with validation error of 92.02.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	101.62	101.79
Linear	2	86.52	92.02

Q6.

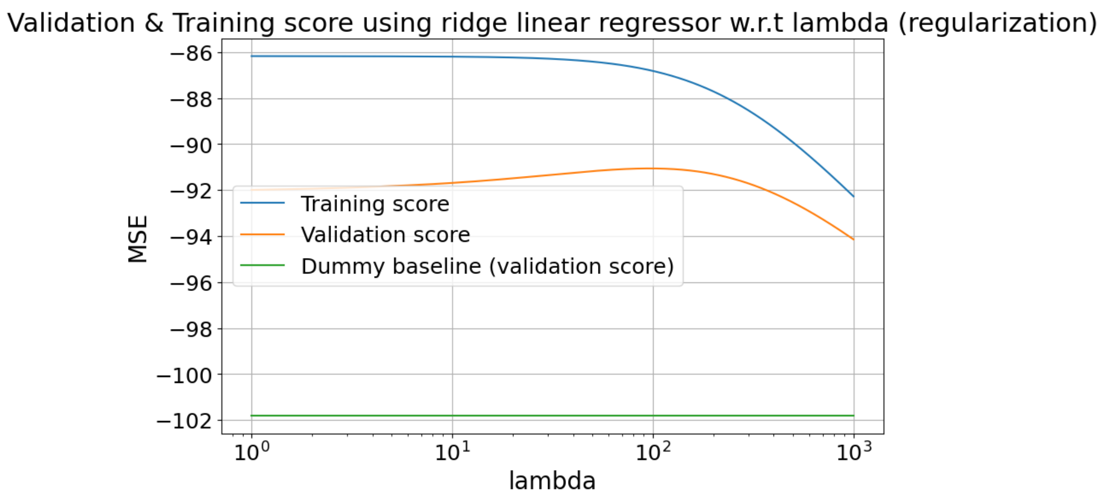
For the dummy regressor, the training performance will not change, because this regressor output the mean of the 'containment_level' label of the train set (which we did not normalize). Normalizing other labels will not change its mean, thus not changing the training performance.
For the linear regressor, the training performance will also not change (when there are no numerical errors), because ordinary least squares are invariant to scaling. More formally, we can model the normalizing procedure as multiplying the rows of the dataset (each row is a datum's features vector) with some "scaler" for the columns (the different features) and adding them some constant (that can very between features). In other words, some affine mapping for X.
Because each feature has different scalar associated with in the optimization problem:

$$\min_{w,b} \frac{1}{m} \|Xw + 1_m b - y\|_2^2$$

We can achieve the same minimal value for the loss function, with, or without the normalization for the features (However, a different set of w, b will reach minimal loss value for both cases), by choosing the correct scales for the features (that will be represented by w).
By having no constraints on w, b (or any regularization), we can simply alter them to reach the same result in the normalized / unnormalized cases.

<u>Part 3 – Ridge Linear Regression</u>

Q7.



The optimal λ for the regularization in the Rigid linear regressor is 97, with validation error of 91.06

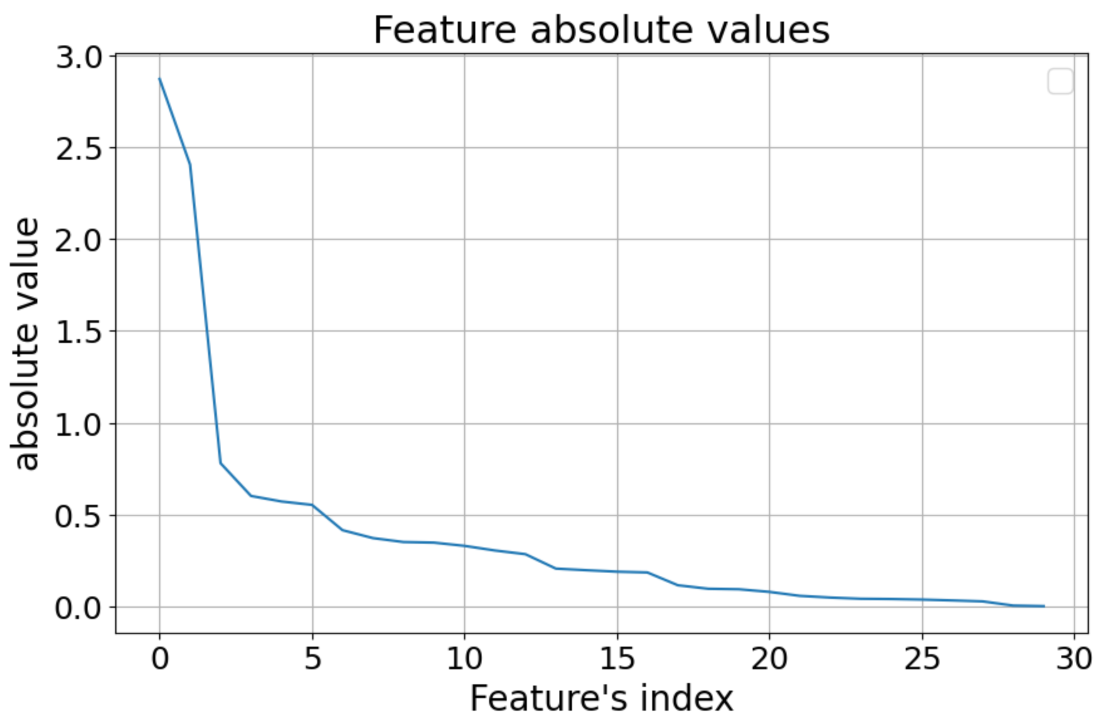
Q8.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	101.62	101.79
Linear	2	86.52	92.02
Ridge linear	3	86.79	91.06

Q9.

The features with the largest (absolute) coefficients in the resulting ridge linear regressor are:
1. PCR_01 – 2.87
2. sugar_levels – 2.4
3. sport_activity – 0.77
4. PCR_07 – 0.6
5. PCR_05 – 0.57

Q10.



Q11.

We considering the magnitude of the feature's coefficients as interesting, because they imply the importance of the feature when computing the prediction for a sample. In other words, the "heaviest" weights correspond to the most important features, because the sample's prediction is determined by a linear combination (with the weights as scalars) of its features values. Thus, larger magnitude result of a feature result in larger effect on the linear combination.

Q12.

For the Ridge linear regressor, the training performance of the Ridge model could have changed, differently than the linear regressor in Q6.
That can happen because for the Rigid linear regressor, the goal is the following optimization problem:

$$\min_{w} \|Xw - y\|_2^2 + \alpha \|w\|_2^2$$

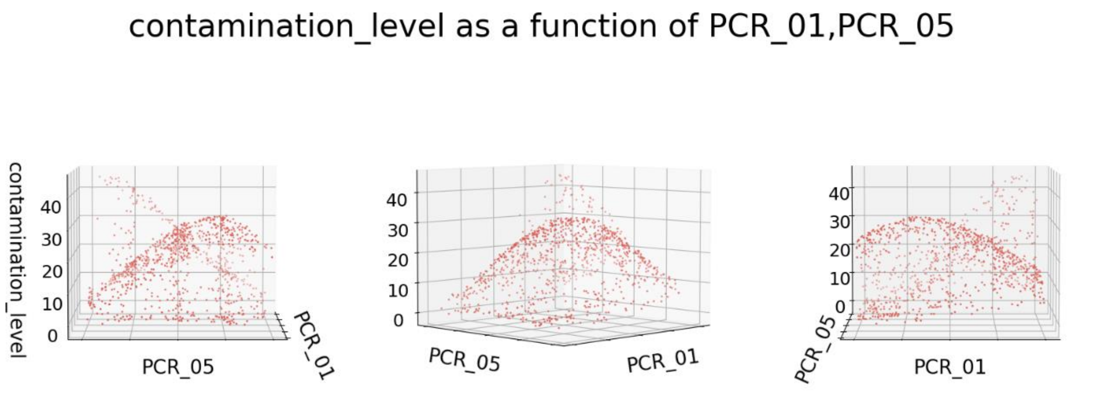
Meaning the different scaling for features in X (that result in different weights of the entries in w to "match" them, to reach the minimum of $\|Xw - y\|_2^2$ – similarly to the linear regressor) penalized differently because of different entries of w.
Thus, different global minimums are possible when changing the scaling of X (which in turn, affect w).

Q13.

We would expect the coefficients of the trained model to change when using the Lasso regressor, in a way that amplifies the importance of some features, while decreasing (almost zeroing) other features importance.
That is caused by using the $L1$ regularization method (the L1 norm of a vector defined as the sum of entries in absolute value – thus allowing the regressor to "drop" some features by zeroing their respected entries in w).

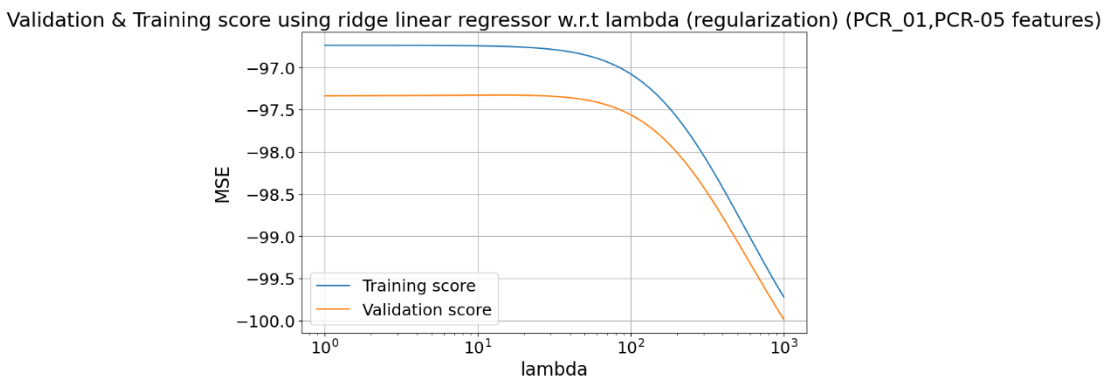
Part 4 – Polynomial fitting

Q14.



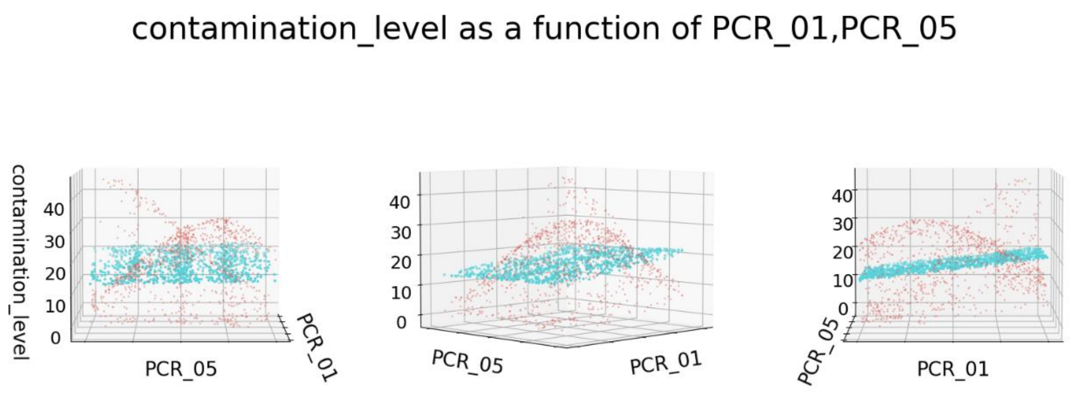
We can understand from the visualization that the data has some form (not totally randomized), and it seems that there are some extremums (it appears to be local maximum, with some additional noise that can be considered as additional maximum point, and minimum point in between). Therefore, we should expect that a polynomial of degree 3-4 (that can capture this extremums behavior) for the polynomial regressor of 'contamination_level' can achieve good results for this data set.

Q15.



The optimal λ for the regularization in the Rigid linear regressor (with features 'PCR_01','PCR_05') is 13, with validation error of 97.32.

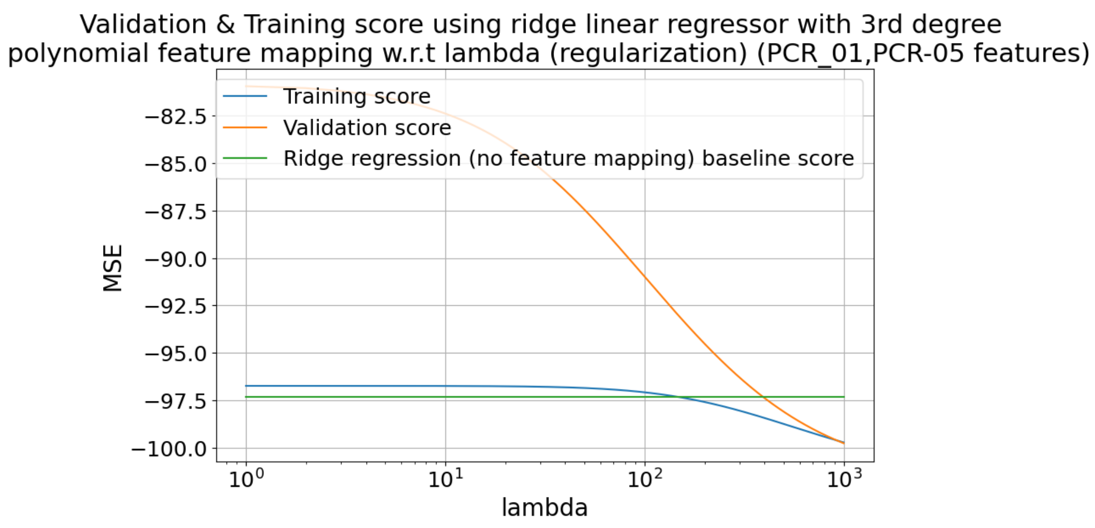
Q16.



Q17.

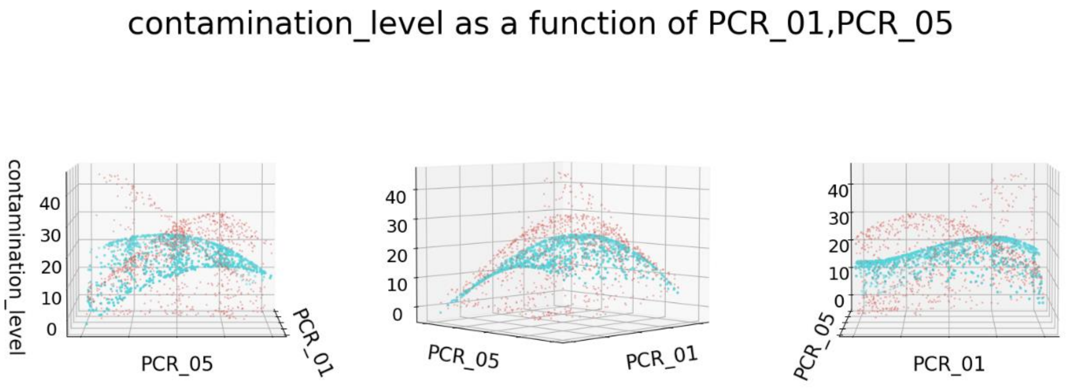
For the polynomial mapping, the training performance of the model could have changed, like the Rigid linear regressor Q12. That is because after mapping the features, the new features can get very high values and very low values. Consider 2 features x, y in the range $\left[\frac{1}{2}, \frac{3}{2}\right]$. If the x features tend to be lower than 1, while the y features tends to be higher than 1 (normalization can't fully "align" the features in the same interval & the same distribution), than the new feature x^k will be very large, while y^k will be small (closer to 0) – that we know that such differences can cause major changes (as explained in Q12). In we normalize the data after applying the feature mapping, we avoid this problem.

Q18.



The optimal λ for the regularization in the Rigid linear regressor (with 3rd degree polynomial feature mapping for 'PCR_01','PCR_05') is 1, with validation error of 80.94.

Q19.



Q20.

Long discussion & results for this section, using mathematical justifications + visualizations + MSEs results in 3-6 sentences.

The linear model does not have good results on this dataset, as we can visually see that the 'contamination_level' isn't a linear function of 'PCR_01' and 'PCR_05' features, thus leading to (relatively) high optimal loss, with validation score of 97~ - we can also see (using plot3d) that the optimal linear regressor does not capture the distribution of the actual data.
The linear model applied to the polynomial featured dataset have much better results than the previous model. Empirically, this model achieve better validation score, of 80~, and also we can see (using plot3d) that the optimal regressor much closer to describing the actual distribution (of the train data) – but still there is room for improvement (does not fully capture the second "sparse" manifold).

Part 5 – Fitting Gradient Boosted Machines

Task before Q21

These features were pre-processed by running them through preprare.py that we wrote in Major 1.
Each of the PCR features (1,2,3,4,5,9,10) were normalized using Min-Max Scaler because they are bounded in range.
The feature sugar_levels was normalized using the Standard Scaler because it has a similar distribution to a normal RV, which standard scaler can handle well.
The feature sport_activity was normalized using the Standard Scaler to be durable to outliers.

Q21.

The four loss functions for sklearn's GradientBoostingRegressor are Least Square Loss (MSE), Mean Absolute Error Loss (MAE), Huber Loss and Quantile Loss.
We will elaborate why each of them is relevant or not for our regression task:

MSE: this loss function is relevant for our regression task as it minimizes the MSE between the predicted value and the true target value, our goal is to minimize that difference hence why it is relevant.

MAE: like the MSE, this loss function minimizes the sum of absolute differences between the predicted value and the true target value (the L1 Loss), so it is relevant for our regression task because our goal is to minimize that difference.

Huber Loss: this Loss function is a combination between the first two loss functions above. It finds a compromise between the MSE and MAE by using the delta parameter to determine between the two. This loss function is relevant when there are outliers in the database, so it can be relevant to our regression task.

Quantile Loss: this Loss function is relevant if the goal is to predict quantiles of the target variable. It allows to specify the quantile value and then we can train the model to minimize the difference between the predicted quantile and the true quantile. We want to predict the target variable's value, so it is not relevant for our regression task.

Q22.

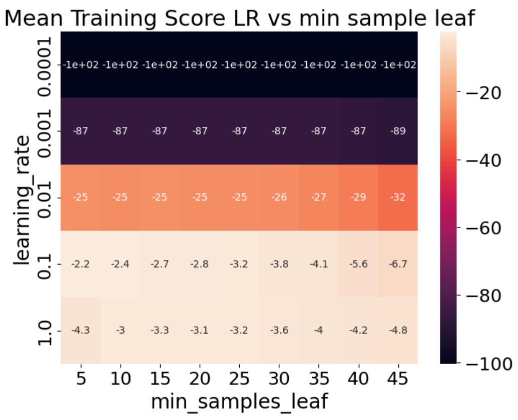
The main similarities between the Random Forest algorithm and the Gradient Boosted tress are that they both using weak learners from the decision trees hypothesis class, and that they both use bagging to train different trees and combining them (it'll discussed in the differences section about the "combining" method).

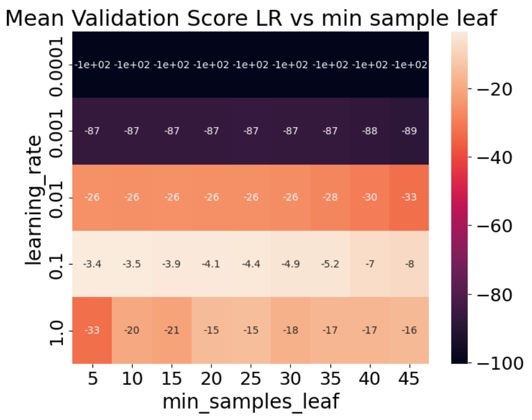
The main differences between the algorithms are how they are created and aggregated. In Gradient Boosted Trees, the trees are built iteratively, meaning they are built one after another while each new tree is built to improve on the deficiencies of the previous trees. In contrast to Random Forest, that builds the trees independently, each tree is a weak learner, but their results are aggregated into a single result (bagging, where the aggregation is the mean between all trees for a certain datum).

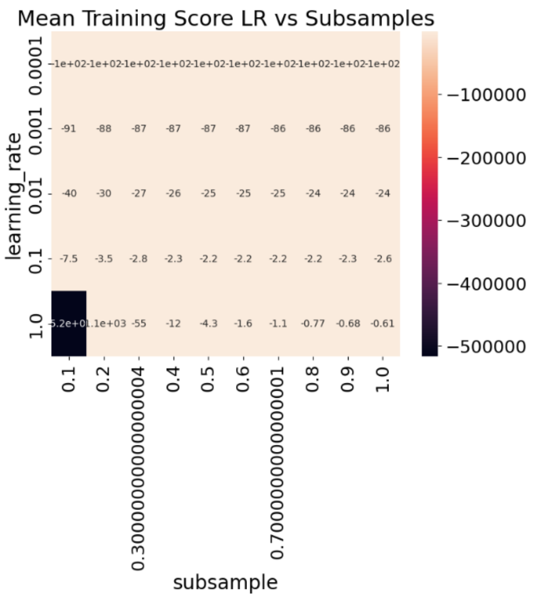
Q23.

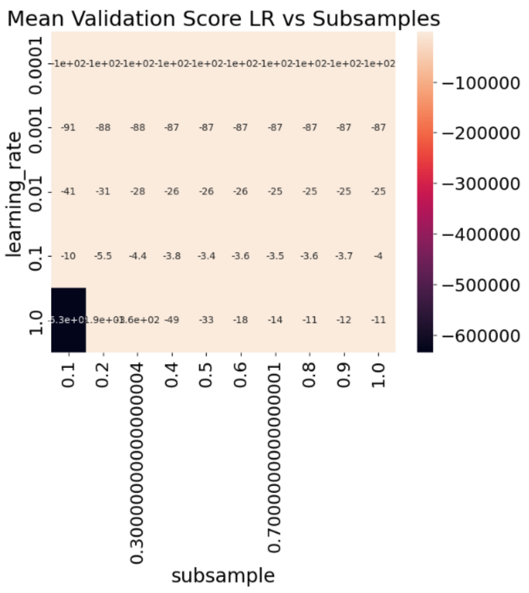
The optimal hyper-parameters are:
loss: huber
subsample: 0.5
min_samples_leaf: 5
learning_rate: 0.1
with optimal validation error of 3.39, and training error of 2.22.

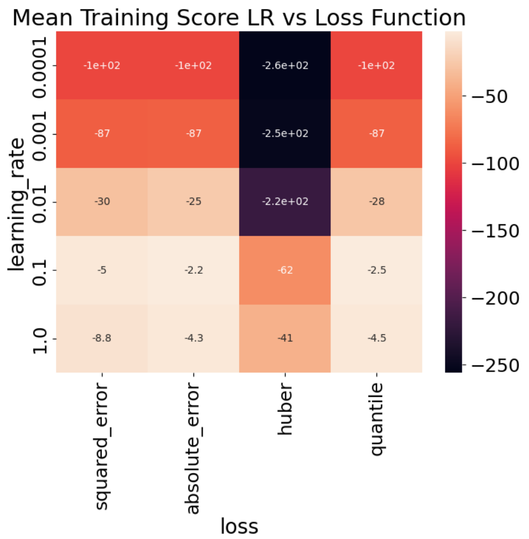
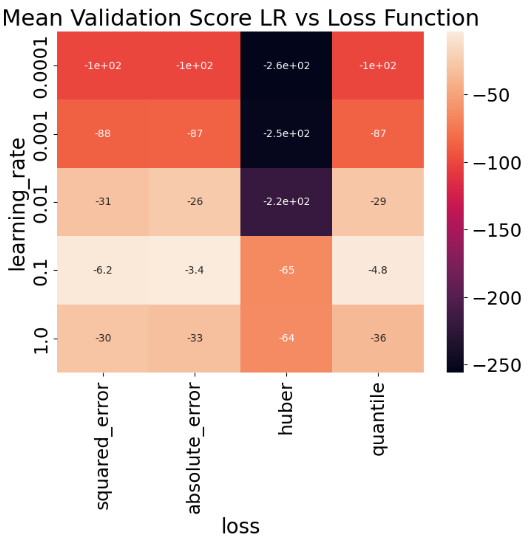
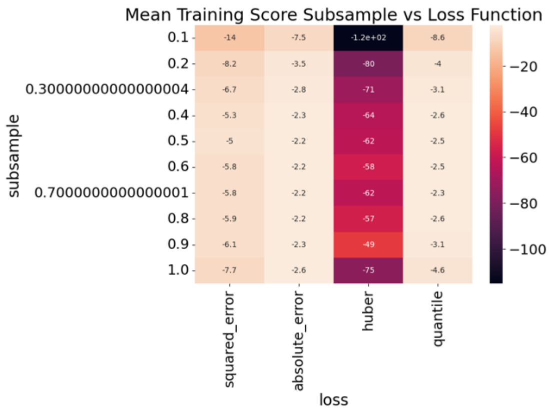
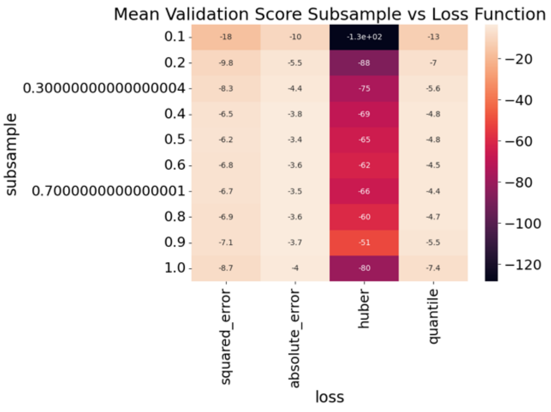
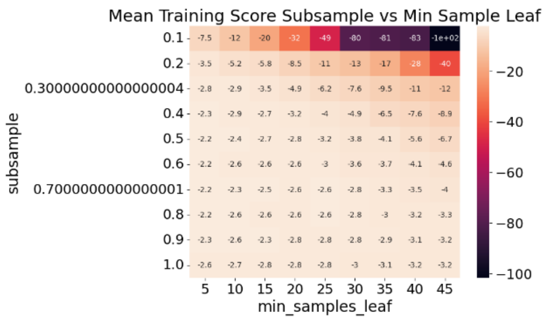
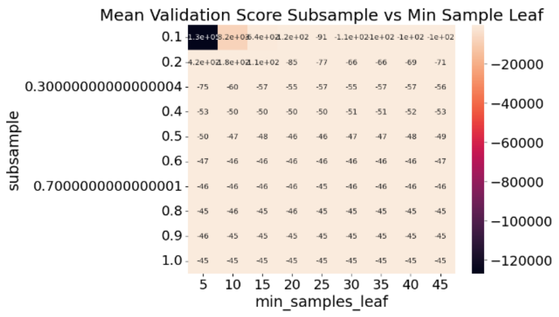
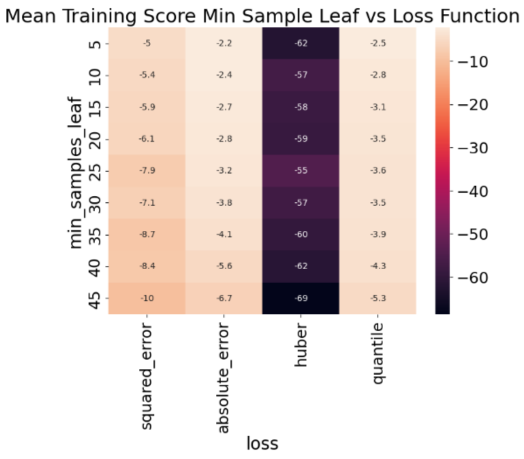
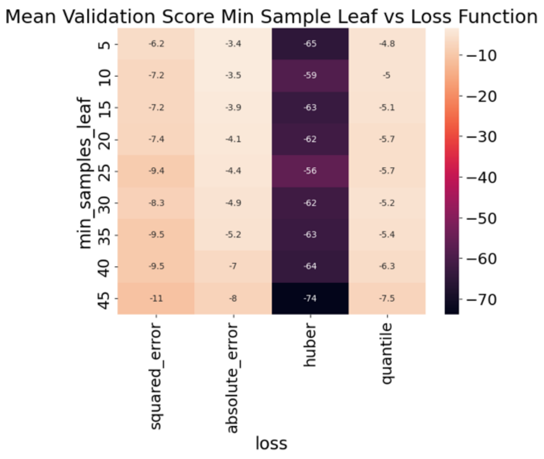
The following heatmaps are the combinations of 2 hyperparameters, with the other 2 fixed with the optimal values.










Mean Training Score LR vs Loss Function
learning_rate
loss
0.0001
0.001
0.01
0.1
1.0
squared_error
absolute_error
huber
quantile
-1e+02
-1e+02
-2.6e+02
-1e+02
-87
-87
-2.5e+02
-87
-30
-25
-2.2e+02
-28
-5
-2.2
-62
-2.5
-8.8
-4.3
-41
-4.5
−50
−100
−150
−200
−250

Mean Validation Score LR vs Loss Function
learning_rate
loss
0.0001
0.001
0.01
0.1
1.0
squared_error
absolute_error
huber
quantile
-1e+02
-1e+02
-2.6e+02
-1e+02
-88
-87
-2.5e+02
-87
-31
-26
-2.2e+02
-29
-6.2
-3.4
-65
-4.8
-30
-33
-64
-36
−50
−100
−150
−200
−250

Mean Training Score Subsample vs Loss Function
subsample
loss
0.1
0.2
0.30000000000000004
0.4
0.5
0.6
0.7000000000000001
0.8
0.9
1.0
squared_error
absolute_error
huber
quantile
−20
−40
−60
−80
−100

Mean Validation Score Subsample vs Loss Function
subsample
loss
0.1
0.2
0.30000000000000004
0.4
0.5
0.6
0.7000000000000001
0.8
0.9
1.0
squared_error
absolute_error
huber
quantile
−20
−40
−60
−80
−100
−120

Mean Training Score Subsample vs Min Sample Leaf
subsample
min_samples_leaf
0.1
0.2
0.30000000000000004
0.4
0.5
0.6
0.7000000000000001
0.8
0.9
1.0
5
10
15
20
25
30
35
40
45
−20
−40
−60
−80
−100

Mean Validation Score Subsample vs Min Sample Leaf
subsample
min_samples_leaf
0.1
0.2
0.30000000000000004
0.4
0.5
0.6
0.7000000000000001
0.8
0.9
1.0
5
10
15
20
25
30
35
40
45
−20000
−40000
−60000
−80000
−100000
−120000

Mean Training Score Min Sample Leaf vs Loss Function
min_samples_leaf
loss
5
10
15
20
25
30
35
40
45
squared_error
absolute_error
huber
quantile
−10
−20
−30
−40
−50
−60

Mean Validation Score Min Sample Leaf vs Loss Function
min_samples_leaf
loss
5
10
15
20
25
30
35
40
45
squared_error
absolute_error
huber
quantile
−10
−20
−30
−40
−50
−60
−70

Q24.

Model	Section	Train MSE	Valid MSE
		Cross validated	
Dummy	2	101.62	101.79
Linear	2	86.52	92.02
Ridge linear	3	86.79	91.06
GBM regressor	5	2.22	3.39

<u>Part 6 – Testing your models</u>

Q25.

Model	Section	Train MSE	Valid MSE	Test MSE
		Cross validated		Retrained
Dummy	2	101.62	101.79	98.29
Linear	2	86.52	92.02	90.07
Ridge linear	3	86.79	91.06	89.1
GBM regressor	5	2.22	3.39	2.8

The GBM regressor performed significantly better than the rest of the regressors on the test set.
We can conclude from the MSEs that the Dummy, Linear, and Ridge linear regressors tends to underfit this dataset (this data distribution).
In section 4 we saw a plot of the label as a function of 2 features, that explains those results – the label isn't some linear/affine function of the 2 features, which means that the Ridge linear regressor could not capture the data distribution very well. For the Dummy & Linear regressor we used more features than the selected 2 for the Ridge regressor, but this is still explainable by this plot (adding more features to non-linear distribution).
We can also conclude that our optimal regressor (and the others) does not overfit on this data distribution, because they achieved similar MSEs results for the train, validation and test set (meaning that the regressor didn't overfit the samples in the training set).
It's worth mentioning that the optimality of the GBM regressor was achieved by a lot more complicated way in tuning the hyper-parameters and deducting it's optimal hyper-parameters (in other words, tradeoff with training time). This procedure took a lot more time than the tunings for the other regressors, and we needed to assume (include some "knowledge") some bounded ranges and values of the hyper-parameters to consider before tuning (we had to limit our search, so the grid search will take reasonable amount of time).