Machine Learning

Excercise 1 - Regression

Group 19
Raphael Beyweiss, 1525113
Jakob Schafellner, 1404727

Regression Techniques

First model

Starting with a simple model to:

- get a sense of how challenging the problem is
- many more things might go wrong with complex models
- how much signal can we pull out using basic models

Linear Regression with Regularization

Why Regularization?

- Recall: Linear regression minimizes a loss function
- It chooses a coefficient for each feature variable
- Large coefficients can lead to overfitting
- Penalizing large coefficients: **Regularization**

Ridge Regression (1/2)

- Loss function = Least Squares loss function + $\alpha * \sum_{i=1}^{n} a_i^2$
- Alpha: Parameter we need to choose
- Picking alpha here is similar to picking k in k-NN
- Alpha controls model complexity
  - Alpha = 0: We get back LS (Can lead to overfitting)
  - Very high alpha: Can lead to underfitting

Ridge Regression (2/2)

Parameters:

- **Alpha**: regularization strength
- **fit_intercept**: weather to calculate an intercept for this model (e. g. not needed if centered)
- **normalize**: if regressor **X** will be normalized

Lasso Regression (1/2)

Loss function = Least Squares loss function + $\alpha * \sum_{i=1}^{n} |a_i|$

- Can be used to select important features of a dataset
- Shrinks the coefficients of less important features to exactly 0

Lasso Regression (2/2)

Parameters:

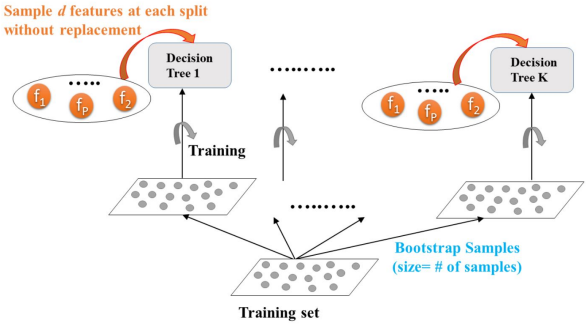
- **Alpha**: constant that multiplies the L1-Norm
- **fit_intercept**: weather to calculate an intercept for this model (e. g. not needed if centered)
- **normalize**: if regressor **X** will be normalized
- **positive:** forces the coefficients to be positive

Random Forest Regression (1/3)

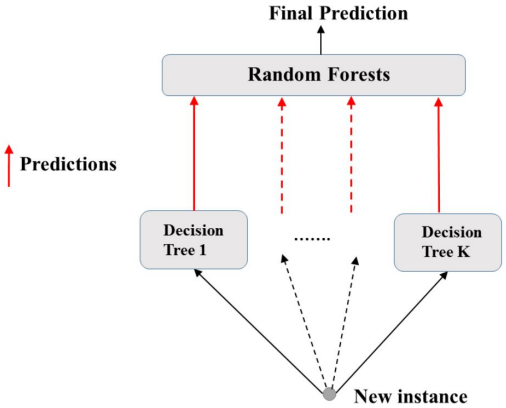
- **Base estimator: Decision Tree**
- Each estimator is trained on a different bootstrap sample having the same size as the training set
- RF introduces further randomization in the training of individual trees
- d features are sampled at each node without replacement (d < total number of features)
- Aggregates predictions through averaging

Random Forest Regression (2/3)

Training:



Prediction:

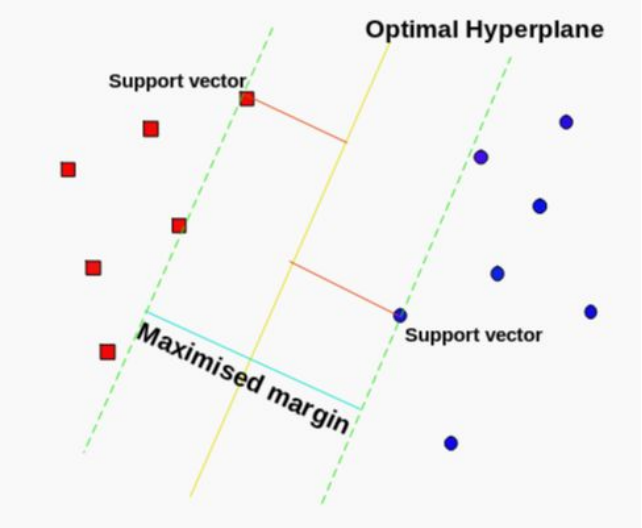


Random Forest Regression (3/3)

Parameters:

- **n_estimators:** number of trees in the forest
- **max_depth:** maximum depth of the tree
- **min_samples_split:** the min. number of samples required to split an internal node
- **min_samples_leaf:** the minimum number of samples required to be at a leaf node
- **min_weight_fraction_leaf:** the minimum weighted fraction of the sum total of weights
- **max_features:** the number of features to consider when looking for the best split
- **min_impurity_decrease:** A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

SVM Regression (1/2)



- SVM tries to find the widest possible separating margin
- SVM extends by using kernel tricks, transforming datasets into rich features space, so that complex problems can be still dealt with in the same “linear” fashion in the lifted hyper space.

SVM Regression (2/2)

Parameters:

- **c:** penalty parameter (higher => overfitting)
- **shrinking:** weather to use the shrinking heuristic
- **kernel:** specifies the kernel type

Boosting

- **Boosting:** Ensemble method combining several weak learners to form a strong learner.
- **Weak learner:** Model doing slightly better than random guessing (e. g. Decision Stump).

Boosting:

- Train an ensemble of predictors sequentially.
- Each predictor tries to correct its predecessor.
- Most popular boosting methods:
  - AdaBoost,
  - Gradient Boosting.

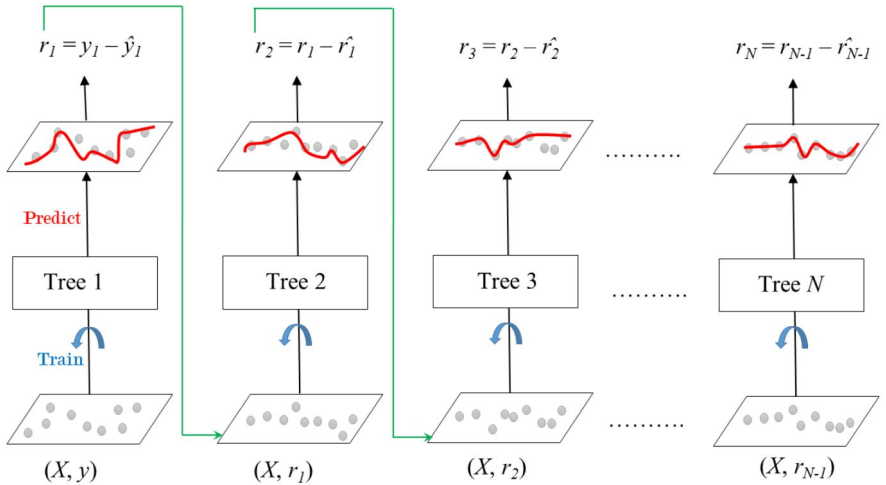
GradientBoostingRegressor (1/3)

- Sequential correction of predecessor's errors.
- Does not tweak the weights of training instances.
- Fit each predictor is trained using its predecessor's residual errors as labels.
- Gradient Boosted Trees: a CART is used as a base learner

GradientBoostingRegressor (2/3)

Prediction:

$$y_{pred} = y_1 + \eta r_1 + \ldots + \eta r_N$$



GradientBoostingRegressor (3/3)

Parameters:

- **n_estimators:** the number of boosting stages to perform
- **max_depth:** maximum depth of the individual regression estimators
- **max_features:** the number of features to consider when looking for the best split
- **learning_rate:** learning rate shrinks the contribution of each tree
- **subsample:** the fraction of samples to be used for fitting the individual base learners

Metrics

Metrics

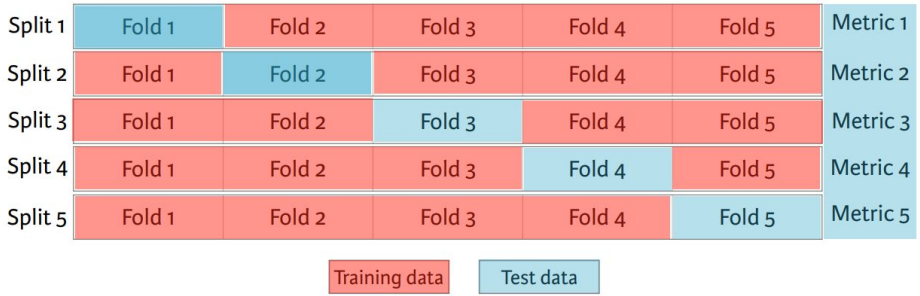
- **MAE** - Mean Absolute Error
- **MAPE** - Mean Absolute Percentage Error
- **RMSE** - Root Mean Square Error
- **R2-Score**

Scikit-Learn

GridSearchCV (1/2)

Cross-Validation:

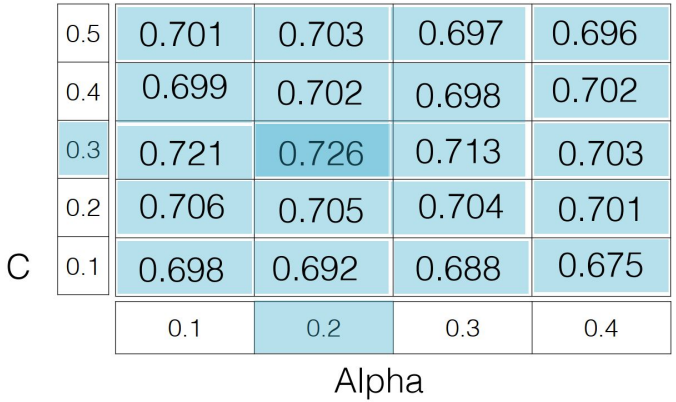
- Model performance is dependent on way the data is split
- Not representative of the model's ability to generalize
- Solution: Cross-validation!



GridSearchCV (2/2)

Hyperparameter tuning:

- Manually set a grid of discrete hyperparameter values.
- Set a metric for scoring model performance.
- Search exhaustively through the grid.
- For each set of hyperparameters, evaluate each model's CV score.
- The optimal hyperparameters are those of the model achieving the best CV score.



Pipeline

- Repeatable way to go from raw data to trained model
- Pipeline object takes sequential list of steps
- Output of one step is input to next step
- Transform: obj implementing .fit() and .transform()
- Flexible: a step can itself be another pipeline!

Train/test split

Hold-out set reasoning

- How well can the model perform on never before seen data?
- Using ALL data for cross-validation is not ideal
- Split data into training and hold-out set at the beginning
- Perform grid search cross-validation on training set
- Choose best hyperparameters and evaluate on hold-out set

Datasets

Bike Sharing (Kaggle) (1/4)

Shape: (8690, 15)

Features: int64, float64, object

No missing values

Bike Sharing (2/4)

Preprocessing:
Preprocessing is needed because data contains a string datetime object.

1. Convert data to ordinal value (counting number)
2. Extract DayOfWeek and WeekOfYear from data (performed better)

Removing correlations:

- id, workingday, temp, mnth (performed better without this features)

Bike Sharing (3/4)

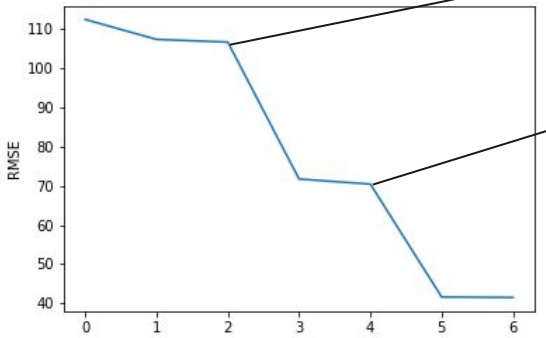
RMSE-Scores of various models:

- Ridge Regression: 135.898 RMSE
- Lasso Regression: 135.90 RMSE
- Random Forest: 41.53 RMSE
- SVM Regressor: 140.58 RMSE
- GradientBoostingRegressor: 40.93 RMSE

RMSE scores come from our hold out set validation.

Bike Sharing (Kaggle) (4/4)

RMSE History of Random Forest Model:



GridSearchCV parameter tuning

new Date, higher estimators,
finer parameter tuning

Student Performance (Kaggle) (1/4)

Shape: (198, 32)

Features: int64, float64, object

No missing values

Student Performance (Kaggle)(2/4)

Preprocessing:
Preprocessing is needed because data contains some nominal features.
For this features we implemented the following encoding schema:

- each possible value gets its own column, where it is 1 if the original feature appears otherwise 0

This method of encoding nominal features is called OneHotEncoding.

Student Performance (Kaggle)(3/4)

Preprocessing:

To remove unwanted correlations between the features we tried to remove some of them:

- removing 'Fedu', 'Fjob_teacher' improved the results

Encoding of nominal features:

- if the feature is binary the encoding creates only one feature (the other one is directly implied)

Student Performance (Kaggle)(4/4)

RMSE-Scores of various models:

- Ridge Regression: 4.715 RMSE
- Lasso Regression: 4.85 RMSE
- Random Forest: 4.668 RMSE
- SVM Regressor: 4.95 RMSE
- GradientBoostingRegressor: 4.71 RMSE

RMSE scores come from our hold out set validation.

Diamonds (1/4)

Shape: (53940, 10) -> 0,01 quantile downsampling to 10%

Features: int64, float64, object

No missing values

Diamonds (2/4)

Preprocessing:
Preprocessing is needed because data contains some ordinal features.
We had two possibilities to encode them:

- OneHotEncoding: each unique value gets its own feature
- label encoding: every value gets a number assigned (implies that the features have ratio-scale)

The OneHotEncoding approach turned out to perform better.

Diamonds (3/4)

Preprocessing:

Encoding of nominal features:

- if the feature is binary the encoding creates only one feature (the other one is directly implied)

Diamonds (4/4)

RMSE-Scores of various models:

- Ridge Regression: 1063.31 RMSE
- Lasso Regression: 1060.77 RMSE
- Random Forest: 684.62 RMSE
- SVM Regressor: 1495.66 RMSE
- GradientBoostingRegressor: 643.24RMSE

RMSE scores come from our hold out set validation.

Wine Quality (1/3)

Shape: (4898, 12)

Features: float64

No missing values

Wine Quality (2/3)

Quality:
Wine with Quality 3 and 9 is very rare. We used a stratified train/test split to ensure our model learns also about this rare qualities.

Preprocessing:
We tried as well to remove some correlated features, but we ended up with a very poor improvement for lasso/rigde regression and worse results for the other techiques.

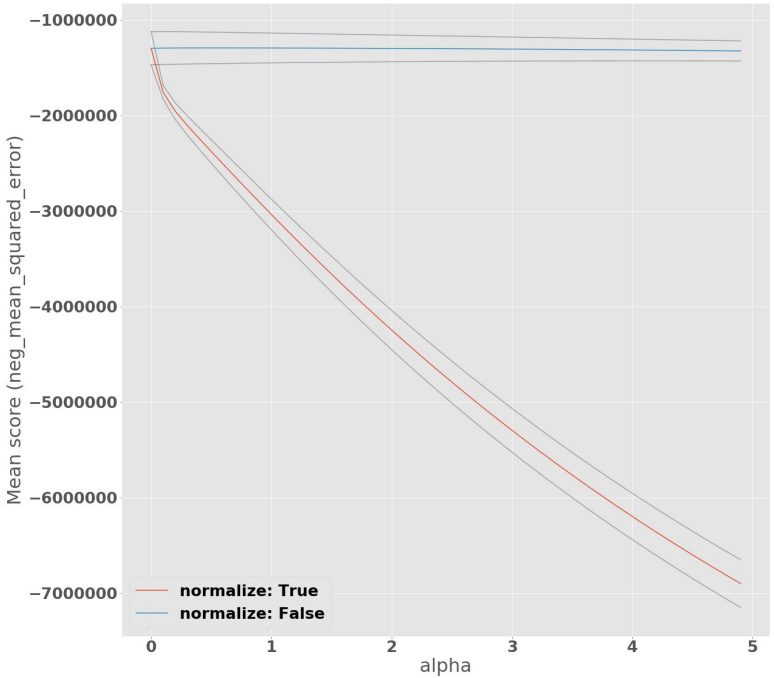
Wine Quality (3/3)

RMSE-Scores of various models:

- Ridge Regression: 0.75 RMSE
- Lasso Regression: 0.75 RMSE
- Random Forest: 0.60 RMSE
- SVM Regressor: 0.70 RMSE
- GradientBoostingRegressor: 0.58 RMSE

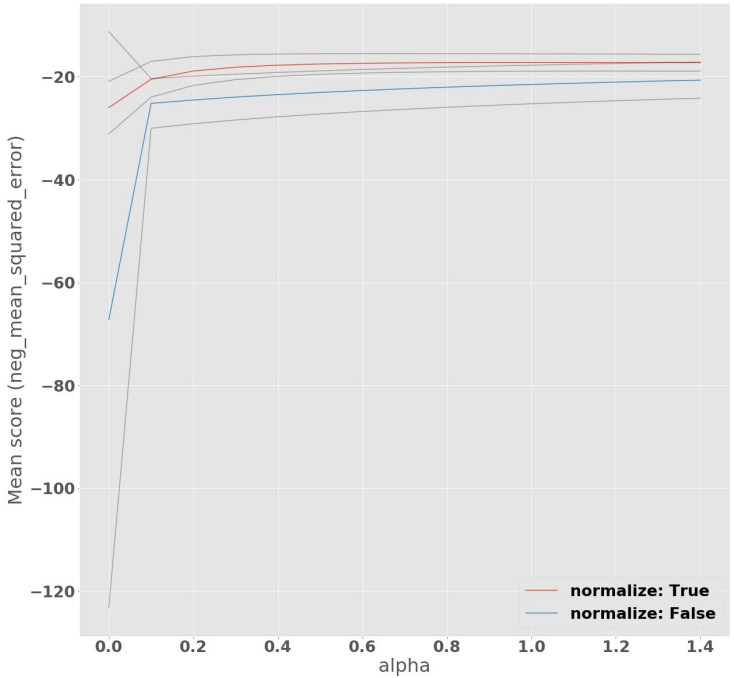
RMSE scores come from our hold out set validation.

Parameter Analysis Ridge Regression (d)



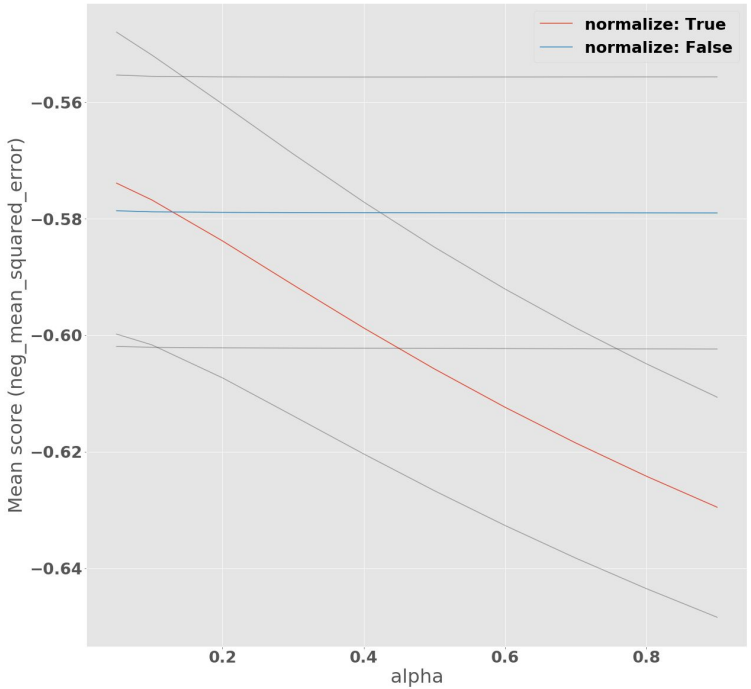
- Continuous decrease for increasing alpha
- Effect increases exponential if normalization is True

Parameter Analysis Ridge Regression (s_p)



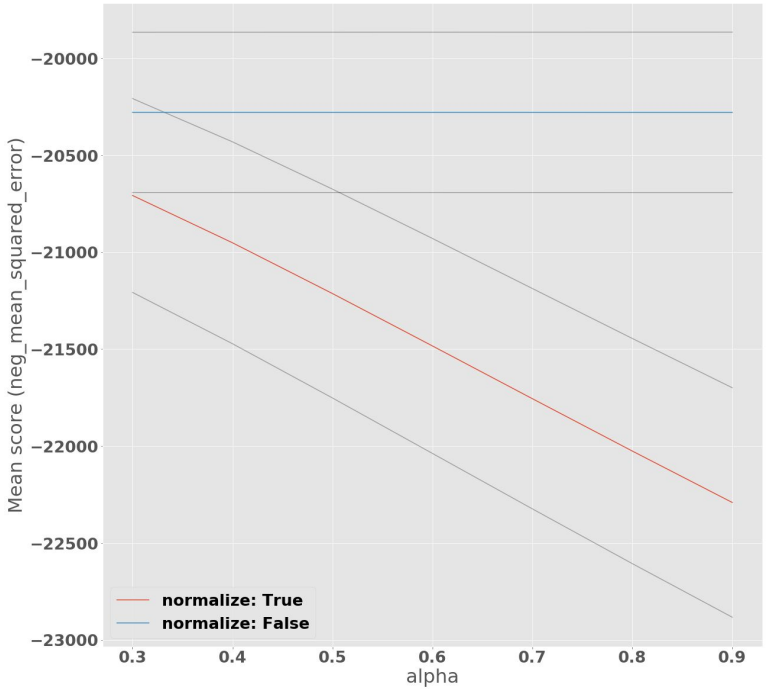
- Slightly better results with increasing alpha
- More stable and better results if normalization is True

Parameter Analysis Ridge Regression (w_w)



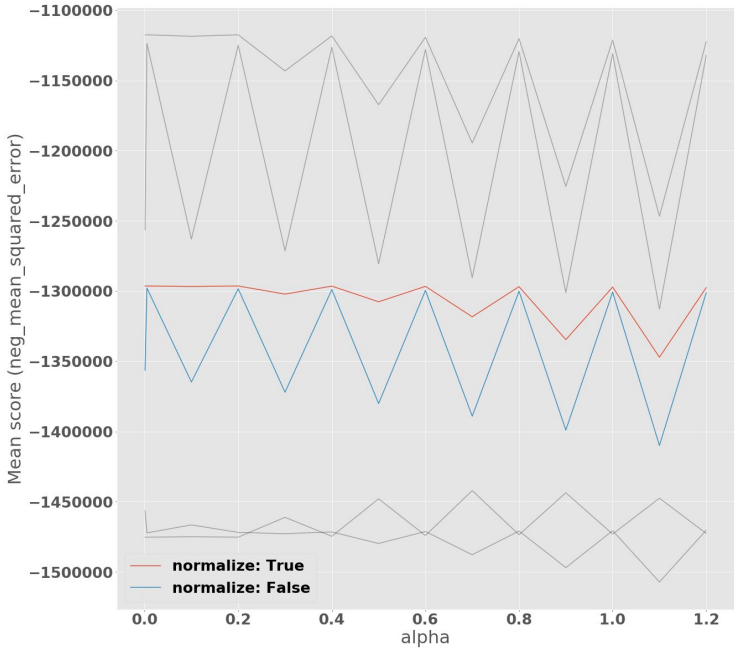
- If normalization is False there are no changes by increasing or decreasing alpha
- If normalization is True, higher alpha decreases the results, but with very small alpha the result is better than without normalization

Parameter Analysis Ridge Regression (b_s)



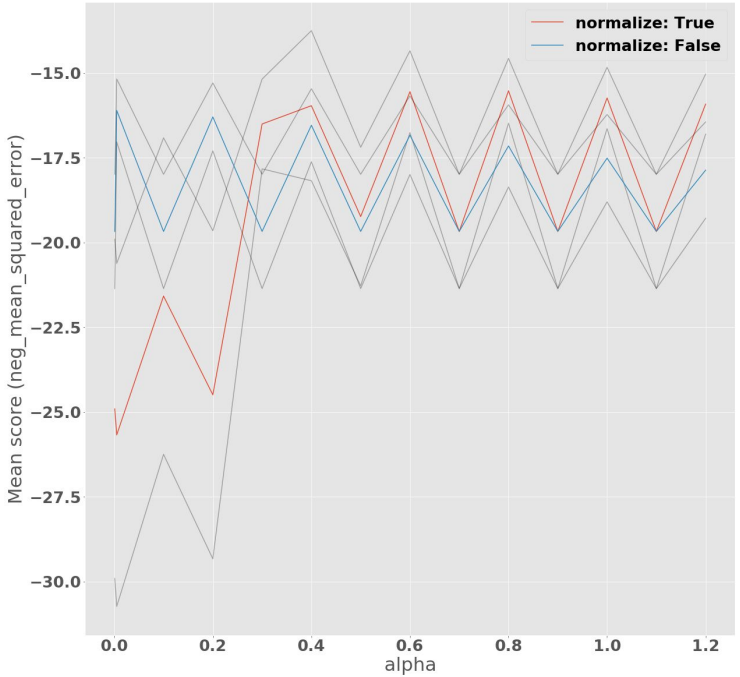
- If normalization is False there are no changes by increasing or decreasing alpha
- If normalization is True, higher alpha decreases the results, but even with very small alpha the result is not better than without normalization

Parameter Analysis Lasso Regression (d)



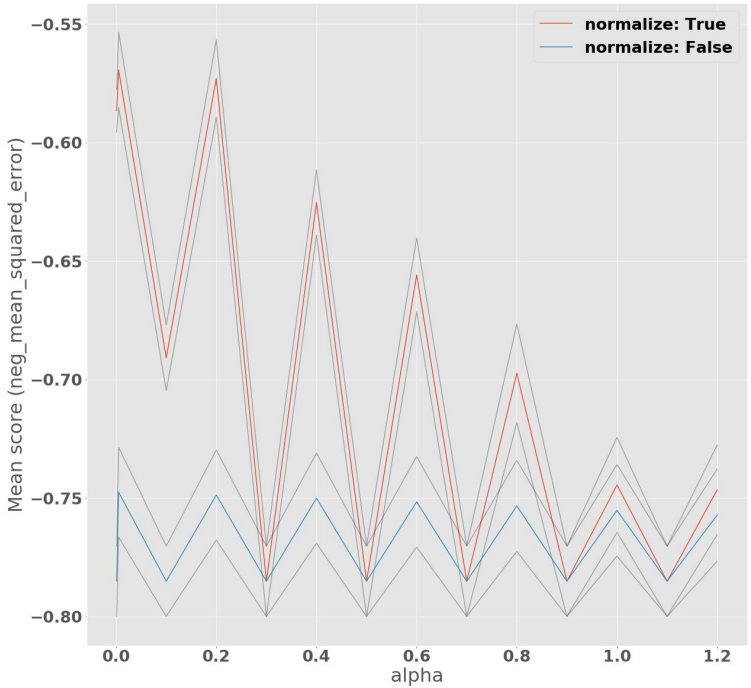
- oscillating behavior for increasing alpha
- the mean is slightly better with normalization, but the std is as well larger

Parameter Analysis Lasso Regression (s_p)



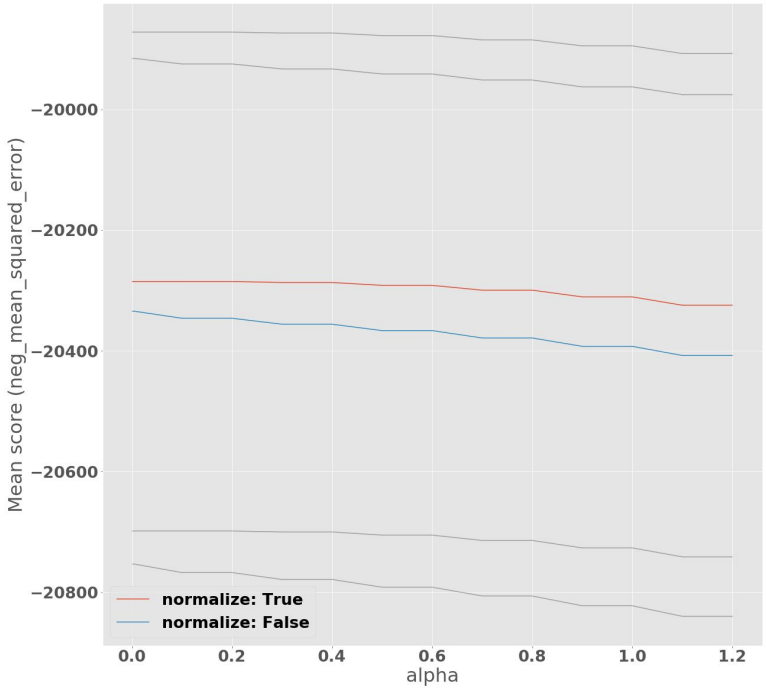
- oscillating behavior for increasing alpha
- with normalization True the results start very unstable and low with small alpha, but if alpha is bigger than 0.5/0.6 the std gets lower and the mean higher

Parameter Analysis Lasso Regression (w_w)



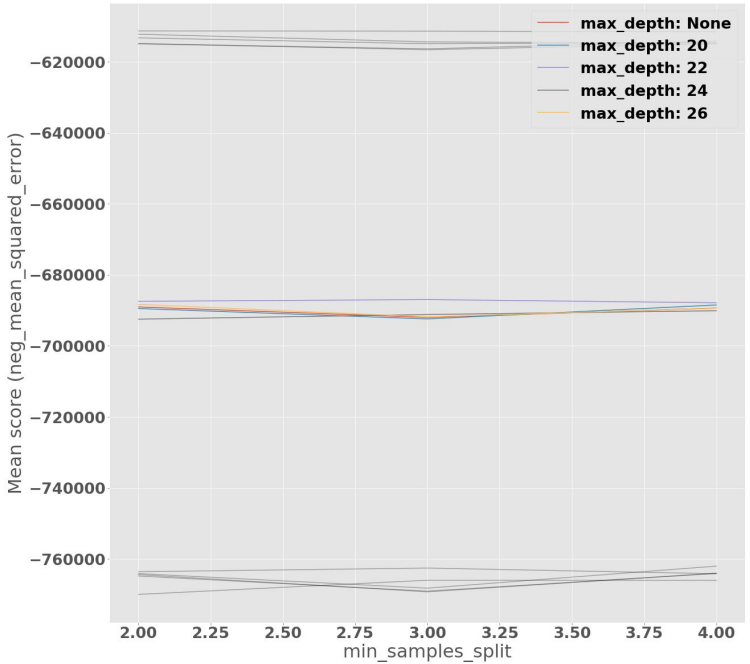
- oscillating behavior for increasing alpha
- If normalization is True alpha decreases the results in an oscillating behavior

Parameter Analysis Lasso Regression (b_s)



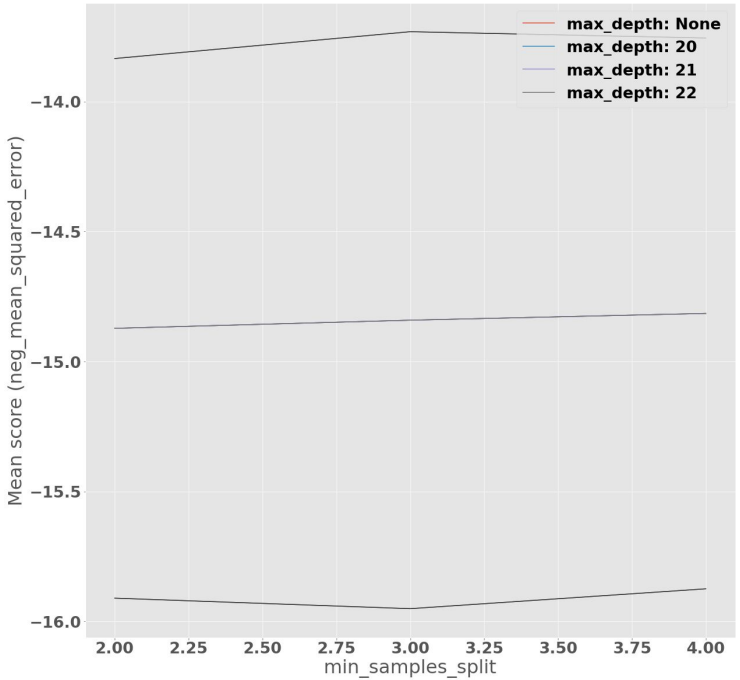
- slightly better results with normalization True
- small decrease for increasing alpha

Parameter Analysis RF (d)



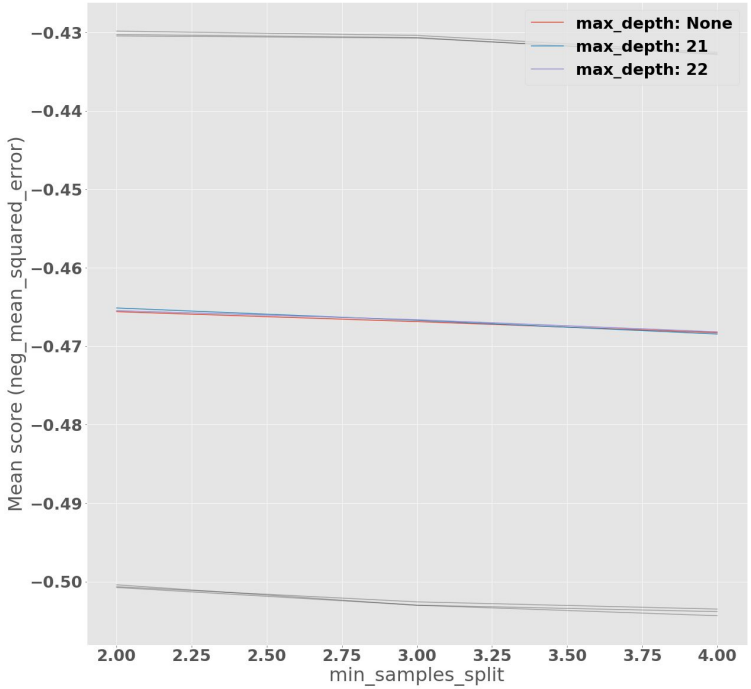
- almost no difference for different parameter setting, but most of the time stable and good results

Parameter Analysis RF (s_p)



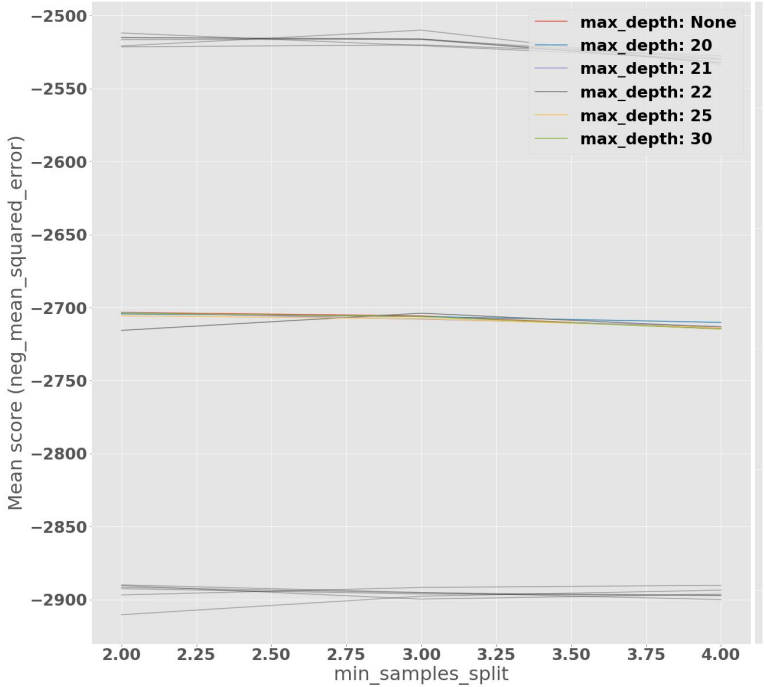
- almost no difference for different parameter setting, but most of the time stable and good results

Parameter Analysis RF (w_w)



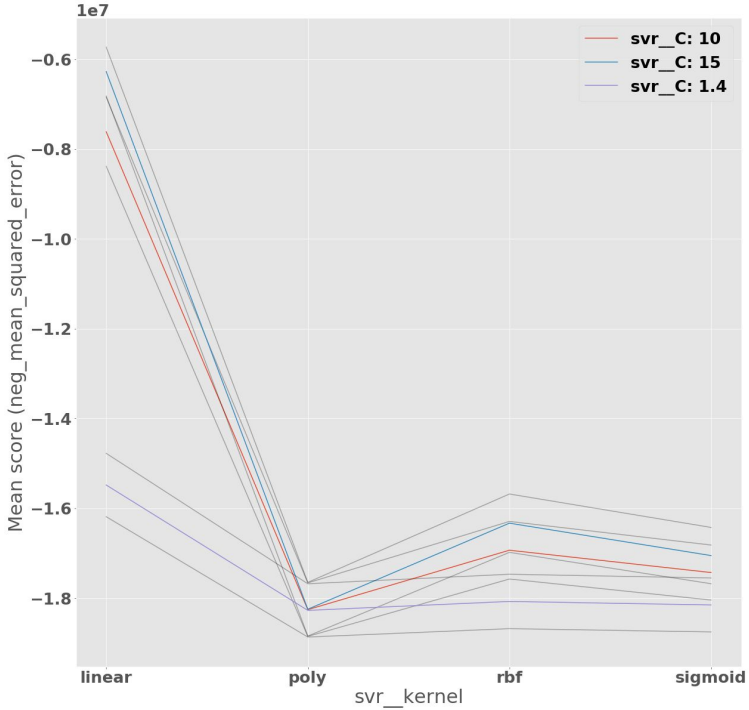
- almost no difference for different parameter setting, but most of the time stable and good results

Parameter Analysis RF (b_s)



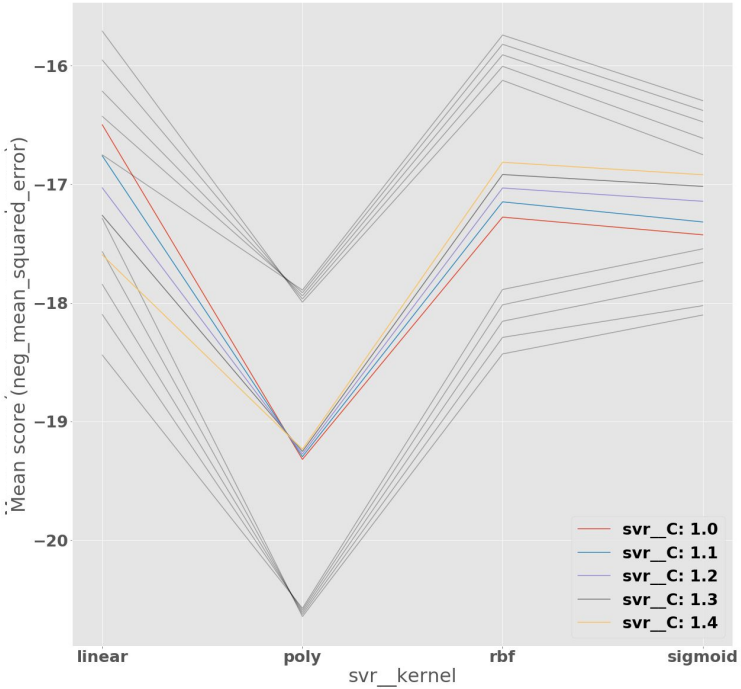
- almost no difference for different parameter setting, but most of the time stable and good results

Parameter Analysis SVR (d)



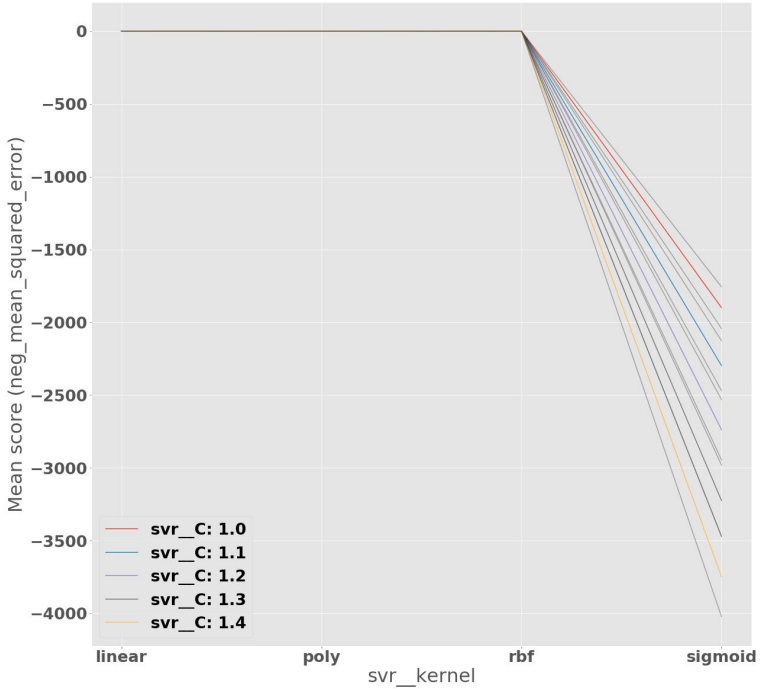
- Only good results with linear kernel
- higher C values increase the

Parameter Analysis SVR (s_p)



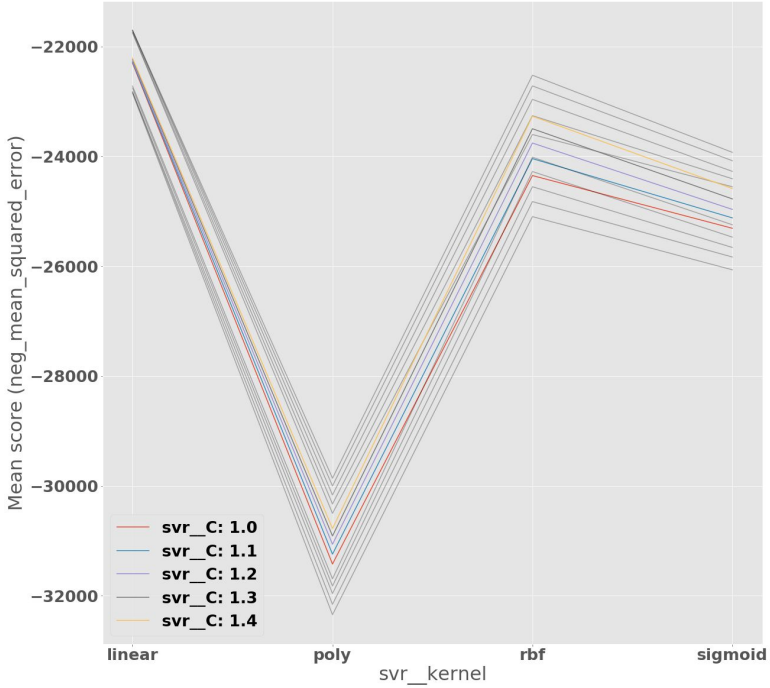
- bad results for poly kernel
- Best results for linear kernel, smaller values for C increase the results for linear kernel
- For rbf and sigmoid kernel the higher C values increase the results

Parameter Analysis SVR (w_w)



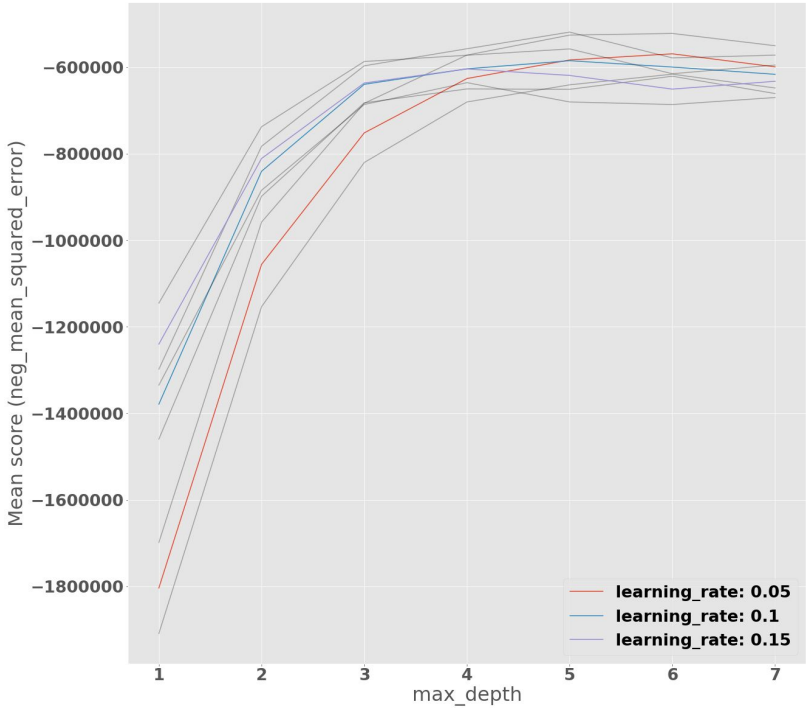
- Very very bad results for sigmoid kernel
- All other 3 kernel have good results

Parameter Analysis SVR (b_s)



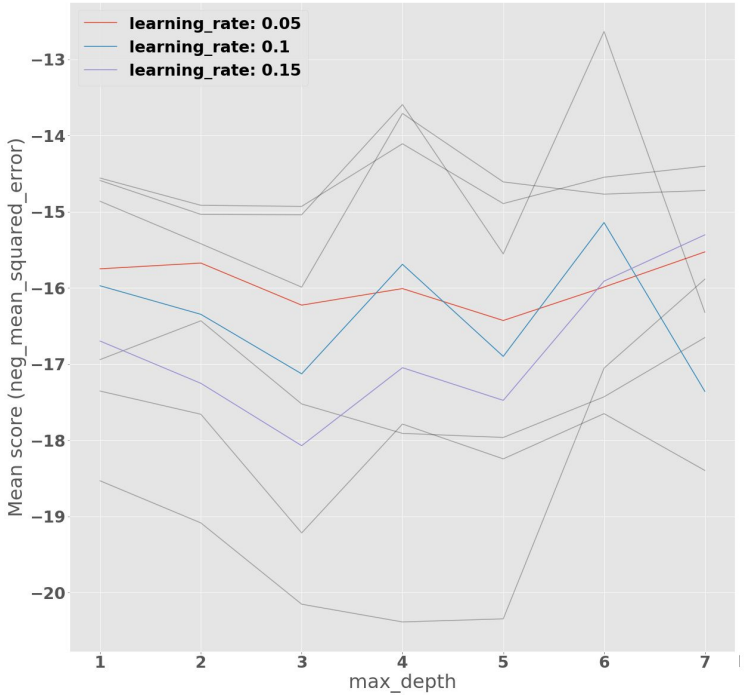
- Very bad results for poly kernel
- Best results with linear, very small difference for different C values

Parameter Analysis GBR (d)



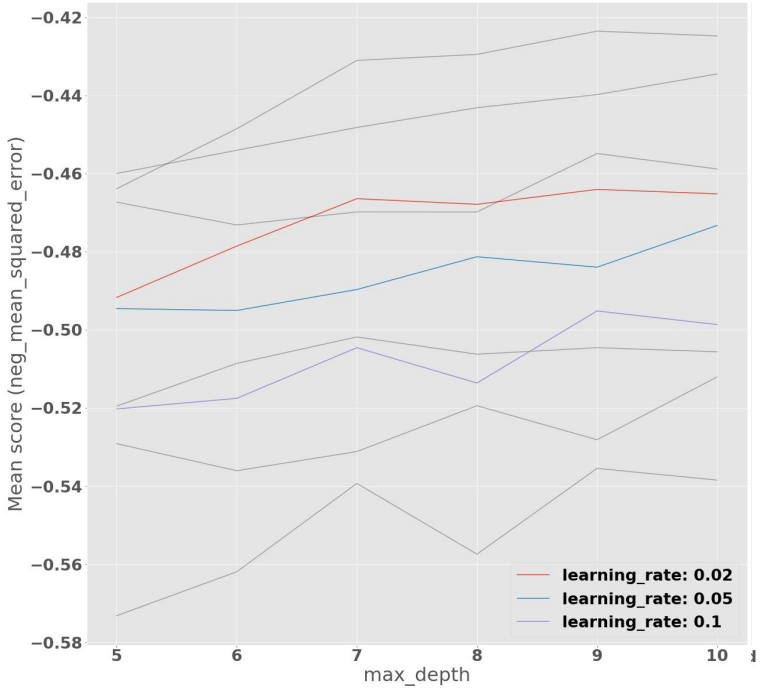
- Clearly the results are getting better by increasing the max depth until it reaches a threshold
- Smaller learning rate reduces the results with low max depth but with higher max depth the threshold appears to be later

Parameter Analysis GBR (s_p)



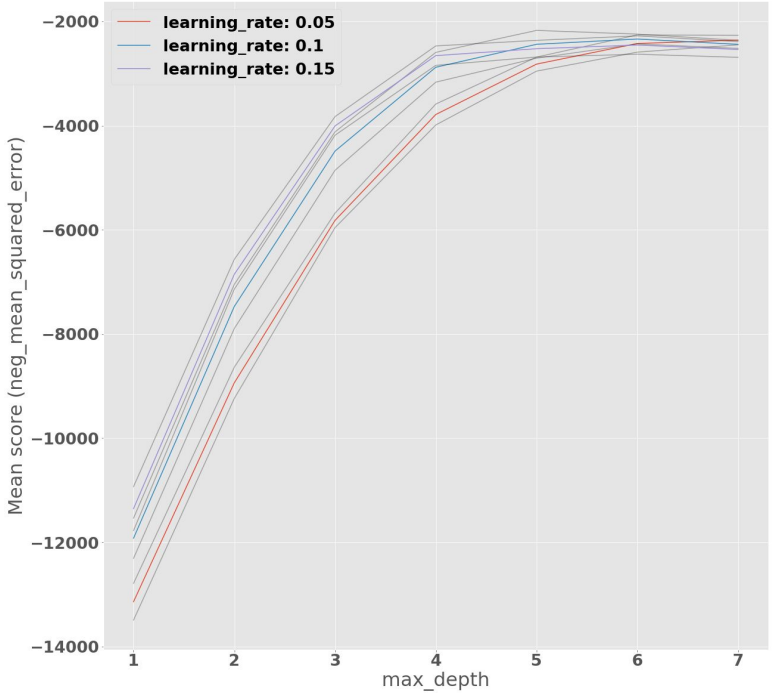
- oscillating behavior for increasing max depth
- also decreasing learning rate does not improve the results clearly

Parameter Analysis GBR (w_w)



- With increasing the max depth the results get better
- Lower learning rate improves the results slightly

Parameter Analysis GBR (b_s)



- Clearly the results are getting better by increasing the max depth until it reaches a threshold
- Smaller learning rate reduces the results with low max depth but with higher max depth the threshold appears to be later