# 3.1 Basic Implementation

Relevant files: *Sec1.py, inpu.py, linear\_fit.py*

(Initial weight vector has been set as zero vector)

Output of file: Output\Out1.txt

For *maxit* stopping critera:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| α | Final MSE (train) | Final MAE (train) | Final MSE (validation) | Final MAE (validation) |
| 0.1 | Diverges | Diverges | Diverges | Diverges |
| 0.01 | Diverges | Diverges | Diverges | Diverges |
| 0.001 | 0.22282572493432173 | 0.3760290120346339 | 0.09405022775563182 | 0.09091900686248729 |

For *reltol* stopping criteria:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| α | Final MSE (train) | Final MAE (train) | Final MSE (validation) | Final MAE (validation) |
| 0.1 | Diverges | Diverges | Diverges | Diverges |
| 0.01 | Diverges | Diverges | Diverges | Diverges |
| 0.001 | 0.38881067790449847 | 0.5011810207678291 | 0.08217559439206809 | 0.0820887944933938 |

If alpha is not sufficiently small, gradient descent algorithm is not able to converge, i.e., MSE becomes arbitrarily large and no minima for cost function is found.

# 3.2 Ridge Regression

Relevant files: *Sec1.py, inpu.py, linear\_fit.py*

Output of file: Output\Out2.txt

Cost function:

Parameter update function:

α = 0.001

Stopping criteria = *maxit*

it = 1000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Final MSE (train) | Final MAE (train) | Final MSE (validation) | Final MAE (validation) |
| Linear | 0.22282572493432173 | 0.3760290120346339 | 0.09405022775563182 | 0.09091900686248729 |
| Ridge (λ= 5) | 0.14687606584412294 | 0.3002182441295067 | 0.10733115873786968 | 0.09781390398739045 |
| Ridge (λ= 25) | 0.16633349593063995 | 0.3214459237457869 | 0. 10302342975223235 | 0. 09538660399365138 |

gives lesser MSE and MAE than on validation set, which implies that for the given implementation, has high bias and has low bias and low variance.

# 3.3 Using Scikit-Learn Library

Relevant files: *Sec3.py, inpu.py*

Output of file: Output\Out3.txt

Comparison of sections 3.1 to 3.3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Name | Final MSE (train) | Final MAE (train) | Final MSE (validation) | Final MAE (validation) |
| 3.1 | Basic Linear | 0.22282572493432173 | 0.3760290120346339 | 0.09405022775563182 | 0.09091900686248729 |
| 3.2 | Ridge (λ= 5) | 0.14687606584412294 | 0.3002182441295067 | 0.10733115873786968 | 0.09781390398739045 |
| 3.3 | Using Scikit | 7.285247240934699 | 2.6991197159323446 | 1.474345543923734 | 0.44223740561431435 |

# 3.4 Feature Selection

Relevant files: *Sec4.py, inpu.py, linear\_fit.py*

Output of file: Output\Out4.txt

Linear regression

α = 0.001

Stopping criteria = *reltol*

bound = 0.000001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| #Features | Final MSE (train) | Final MAE (train) | Final MSE (validation) | Final MAE (validation) |
| 2490 | 0.38881067790449847 | 0.5011810207678291 | 0.08217559439206809 | 0.0820887944933938 |
| 10 (SelectKBest) | 1.2043872712872432 | 0.8475841489401378 | 0.2796538173445502 | 0.1590609557730694 |
| 10 (SelectFromModel) | 1.1962115989679691 | 0.8315110334012059 | 0.3886584911509137 | 0.17920046770939935 |

# 3.5 Classification

Relevant files: *Sec5.py, inpu.py*

Output of file: Output\Out5.txt

Update function:

For simplicity,

Which is the same update function as linear regression

# 3.6 Visualisation

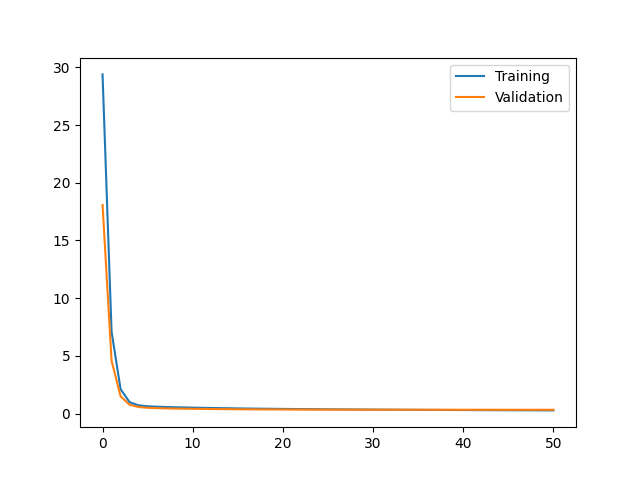
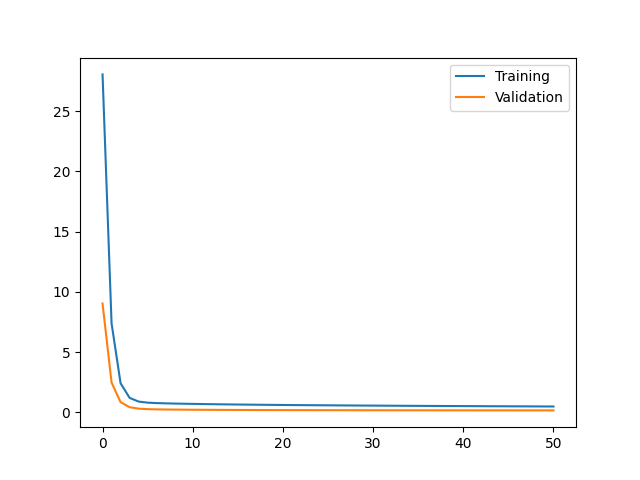
1. Plot training and validation MSE loss
2. Normalization
3. Diving data into different sized groups

it = 50

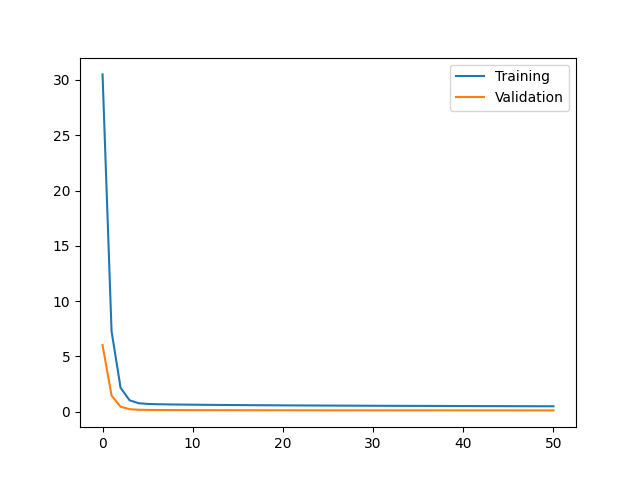
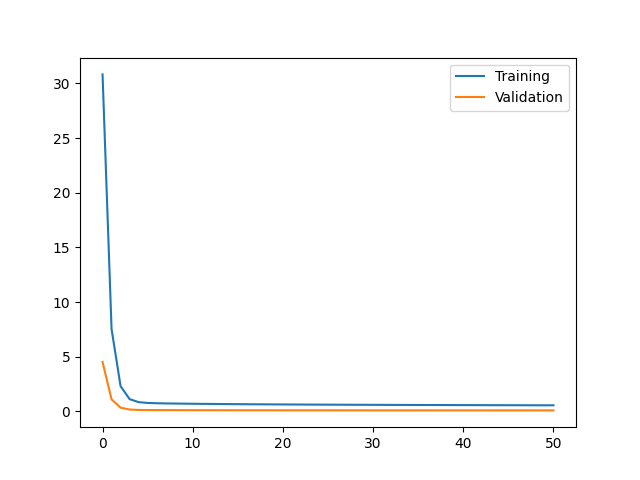
|  |  |  |
| --- | --- | --- |
| Fraction | Final MSE (Train)\* | Final MSE (Validation)\* |
| 0.25 | 0.398625380403785 | 0.30952612008992164 |
| 0.5 | 0.45743676633836533 | 0.18074135026665436 |
| 0.75 | 0.5348241543864707 | 0.11875836442482343 |
| 1 | 0.5520432975275971 | 0.08741084921216116 |

With reduced training set, in-sample errors are less since curve is easily able to fit, however out-sample errors are larger.

\* - Makes use of numpy.random, different values at each run

Fraction = 0.25 Fraction = 0.5

Fraction = 0.75 Fraction = 1

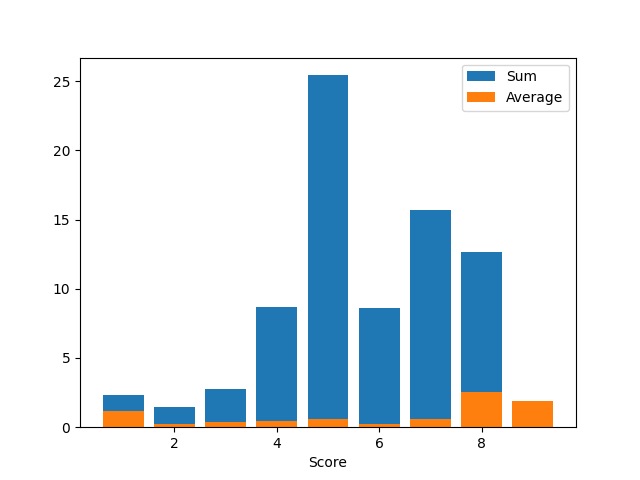
1. Dividing into two equal parts
2. For 3.1

Mean absolute difference = 1.457065325480676

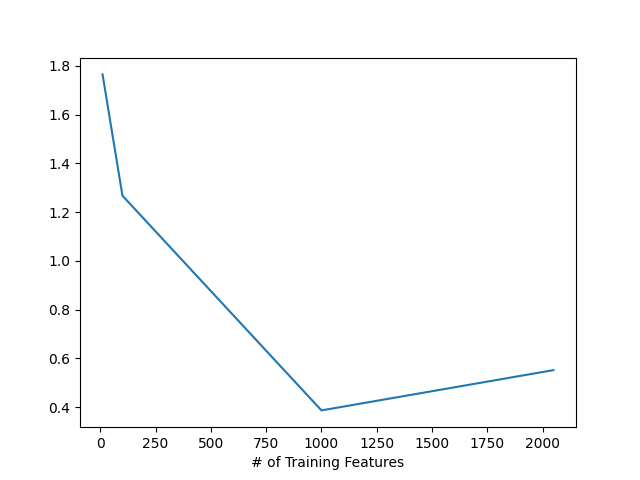
1. For 3.2

Mean absolute difference = 1.4632523576770717

1. Plotting against score values



1. More feature selection



# 3.7 Generalization Analysis

Relevant files: *Sec7.py, linear\_fit.py*

Output of file: Output\Out7.txt

|  |  |
| --- | --- |
| Dimension |  |
| 2 | 0.5457668016226359 |
| 5 | 0.45152483289549417 |
| 10 | 0.32685555703491354 |
| 100 | 1.4409368603683705 |

This shows us that generalization bound is a loose estimate for .

# 4 Evaluation

* Relevant files: *main.py*
* Dependencies: *linear\_fit.py*