Mh223vk Mehdi Hmidi

Exercise 1

1) Tune the necessary hyperparameters, by for instance grid search.

(VG-exercise): Implementing your own kernel

For 1.1 I present both ANOVA using a python functions and using a Gramian Matrix

['ANOVA Using gram matrix'] »»»»»»»»»»»»»»»»»»»»»»»»

Best Parameter: degree: 5 sigma:30 C:1000

Best score:0.840625

['ANOVA using a python function'] »»»»»»»»»»»»»»»»»»»»»»»»

Best Parameter C: 1000

Best score: 0.821875

['linear'] »»»»»»»»»»»»»»»»»»»»»»»»

Best Parameter C: 10

Best score: 0.8078125

['rbf'] »»»»»»»»»»»»»»»»»»»»»»»»

Best Parameters: { 'C' : 1 , 'gamma' : 10}

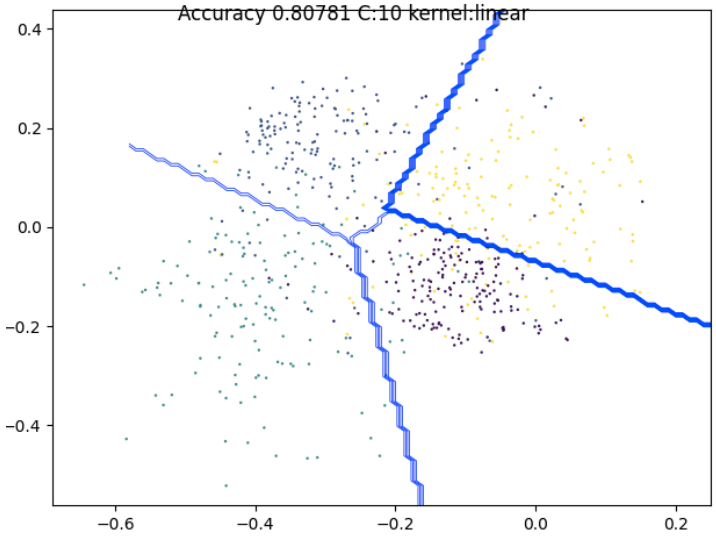
Best score: 0.8140625

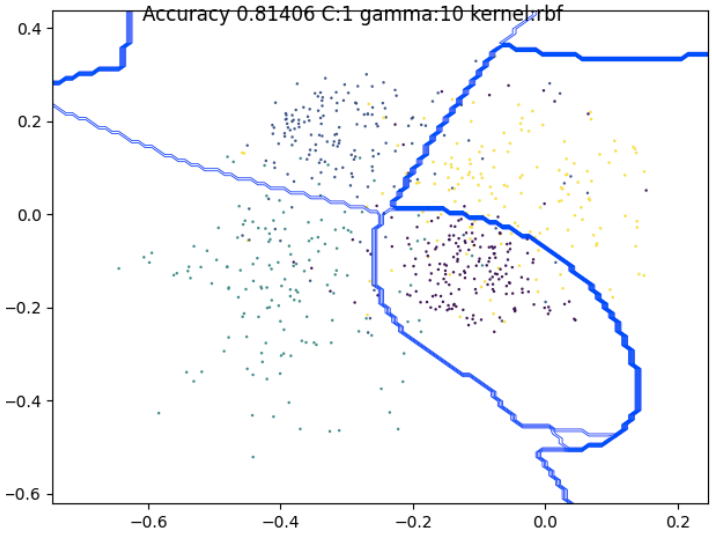
['poly'] »»»»»»»»»»»»»»»»»»»»»»»»

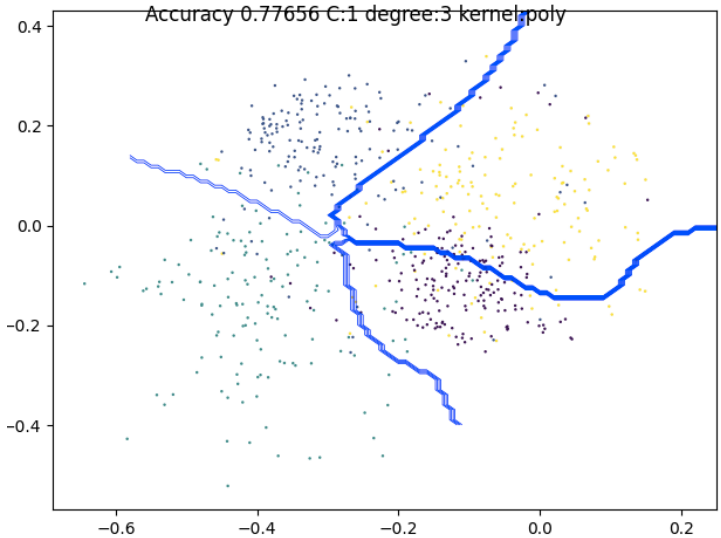
Best Parameters: { 'C' : 1 , 'degree' : 3}

Best score: 0.7765625

2) Produce a plot of the decision boundary for the best models together with the data.







Mh223vk Mehdi Hmidi

Exercise 2

********** **From Scratch** MultiClass One-vs-Rest SVM with rbf kernel

Best param learned C=5

Accuracy rate of my multiclass SVM in test set: 0.9476190476190476

Test set confusion matrix:

```
[[ 76   0   0   0   0   0   1   0   1   0]
 [  0 103   0   0   0   0   0   0   0   0]
 [  0   0  76   3   0   0   0   1   0   0]
 [  1   0   1  74   1   1   1   1   1   2]
 [  0   0   1   0  89   0   0   0   1   4]
 [  2   0   0   0   0  67   0   0   2   2]
 [  1   0   1   0   0   1  85   0   0   0]
 [  0   0   0   0   0   0   0  77   0   2]
 [  0   2   1   1   0   0   1   0  71   0]
 [  1   0   0   0   5   0   0   1   0  78]]
```

********** **SciKit-learn** MultiClass One-vs-Rest SVM with rbf kernel

Best params learned via GridSearch C=5, degree=3, gamma=0.01, tol=0.001

Accuracy of learned model: 0.9416666666666667

	precision	recall	f1-score	support
0	0.89	0.97	0.93	78
1	0.98	1.00	0.99	103
2	0.93	0.94	0.93	80
3	0.94	0.88	0.91	83
4	0.95	0.95	0.95	95
5	0.99	0.90	0.94	73
6	0.93	0.97	0.95	88
7	0.94	0.97	0.96	79
8	0.96	0.89	0.93	76
9	0.92	0.92	0.92	85

Test set confusion matrix:

```
[[ 76   0   0   0   0   0   1   0   1   0]
 [  0 103   0   0   0   0   0   0   0   0]
 [  2   0  75   1   0   0   0   2   0   0]
 [  1   0   3  73   1   0   2   1   0   2]
 [  0   1   0   0  90   0   1   0   1   2]
 [  2   0   1   1   1  66   0   0   1   1]
 [  2   0   1   0   0   0  85   0   0   0]
 [  0   0   0   0   0   0   0  77   0   2]
 [  0   1   1   3   0   1   2   0  68   0]
 [  2   0   0   0   3   0   0   2   0  78]]
```

Mh223vk Mehdi Hmidi

********** Binary One-vs-One SVM with rbf kernel
Accuracy of learned model 0.9380952380952381

	precision	recall	f1-score	support
0	0.90	0.97	0.94	78
1	0.99	1.00	1.00	103
2	0.89	0.96	0.92	80
3	0.92	0.87	0.89	83
4	0.97	0.93	0.95	95
5	0.96	0.89	0.92	73
6	0.94	0.95	0.95	88
7	0.95	0.97	0.96	79
8	0.96	0.89	0.93	76
9	0.90	0.92	0.91	85

Test set confusion matrix:
```
[[ 76   0   0   0   0   0   1   0   1   0]
 [  0 103   0   0   0   0   0   0   0   0]
 [  1   0  77   1   0   0   0   1   0   0]
 [  1   0   3  72   0   2   1   1   1   2]
 [  0   0   1   0  88   0   1   0   1   4]
 [  2   0   3   2   0  65   0   0   0   1]
 [  2   0   2   0   0   0  84   0   0   0]
 [  0   0   0   0   0   0   0  77   0   2]
 [  0   1   1   3   0   1   2   0  68   0]
 [  2   0   0   0   3   0   0   2   0  78]]
```

Mh223vk Mehdi Hmidi

Exercise 3

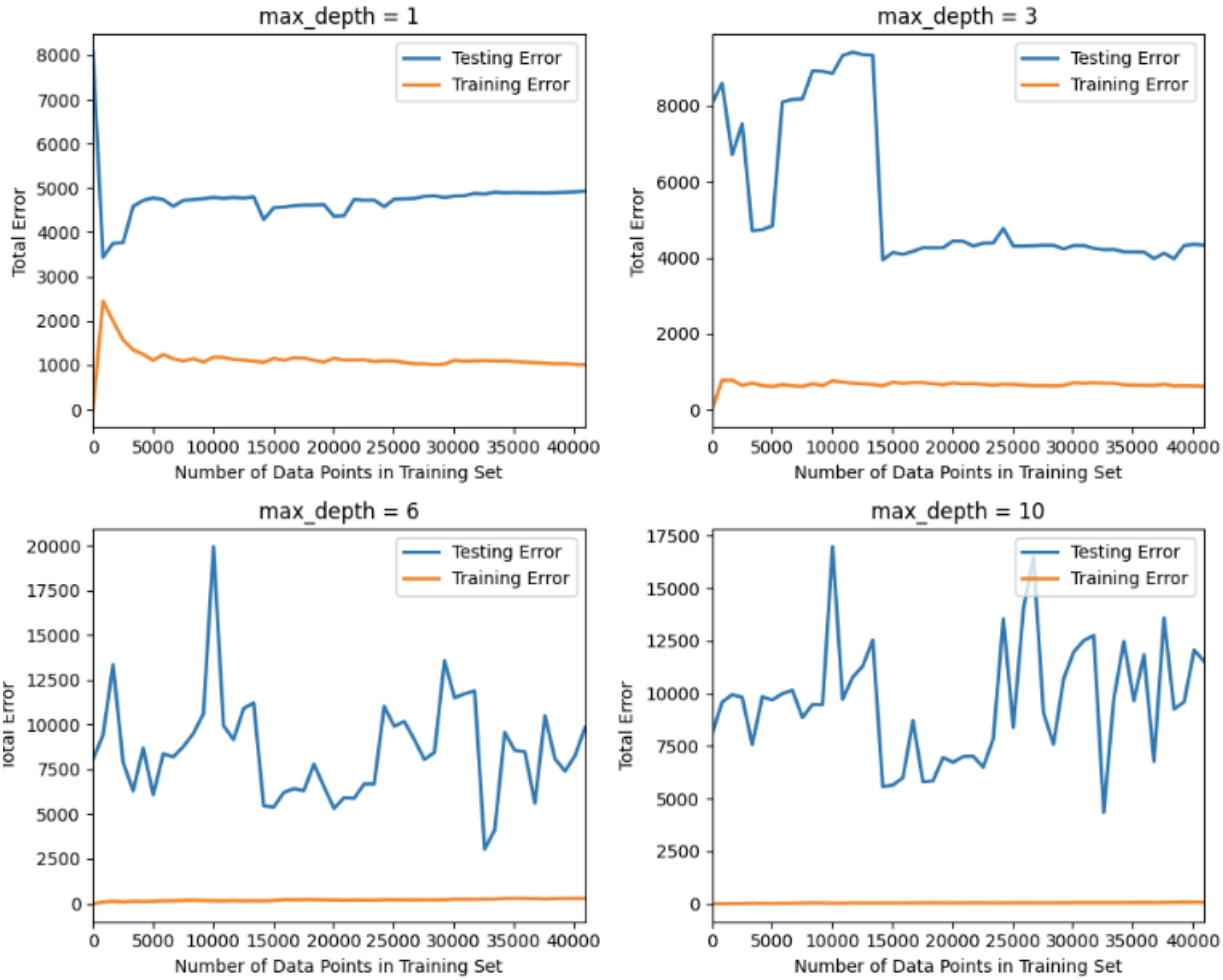
1)****** DecisionTreeRegressor kernel

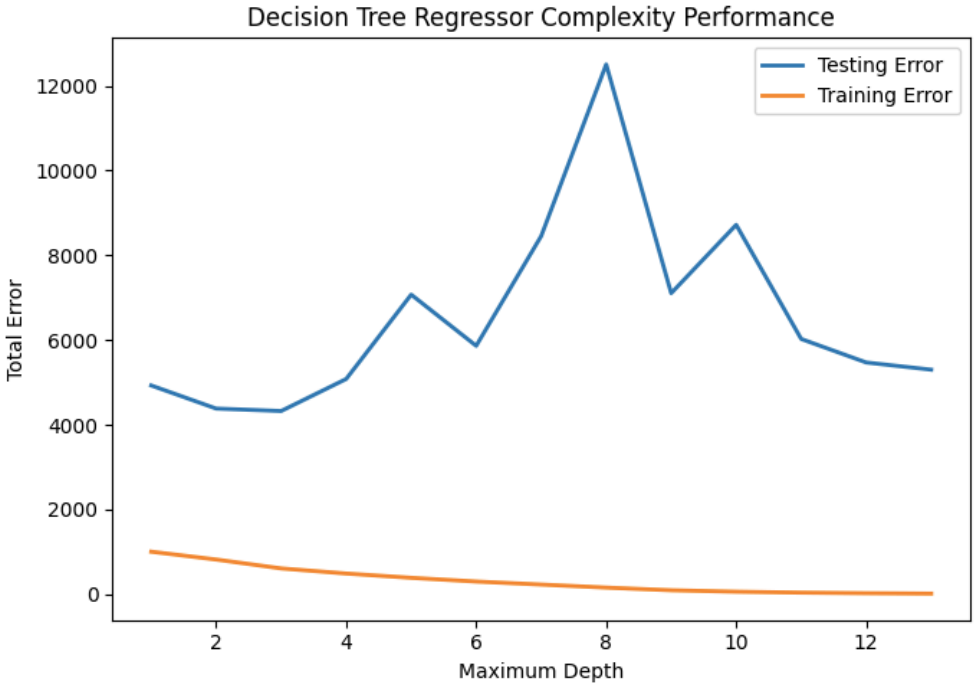
Best params learned via GridSearch max_depth=4

r2_score: **0.2801276852449447**

Train MSE **490.0561947243723**

Test MSE: **5082.224467309503**





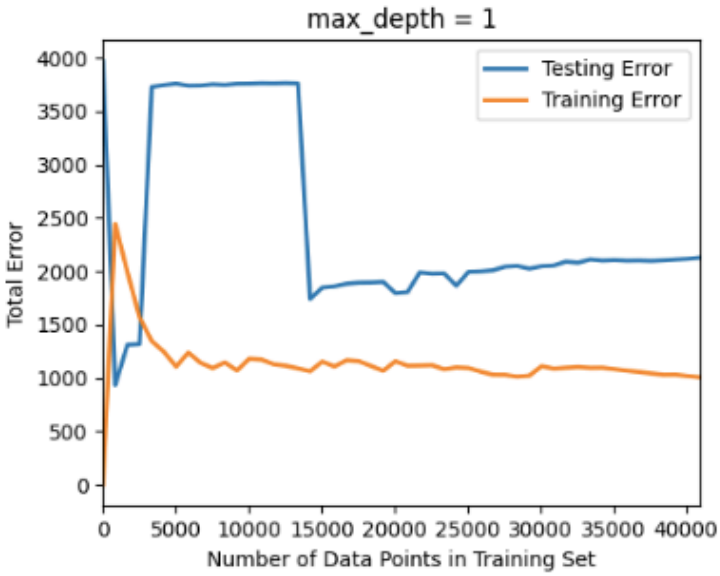
Mh223vk Mehdi Hmidi

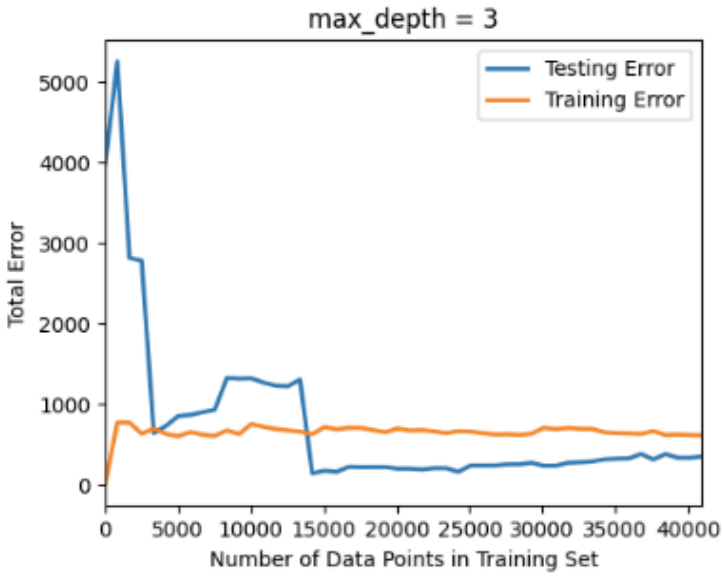
2) ****** RandomForestRegressor
r2_score = **-0.006782301072497665**
Train MSE **1163.0743017191662**
Test MSE: **7107.779447673054**

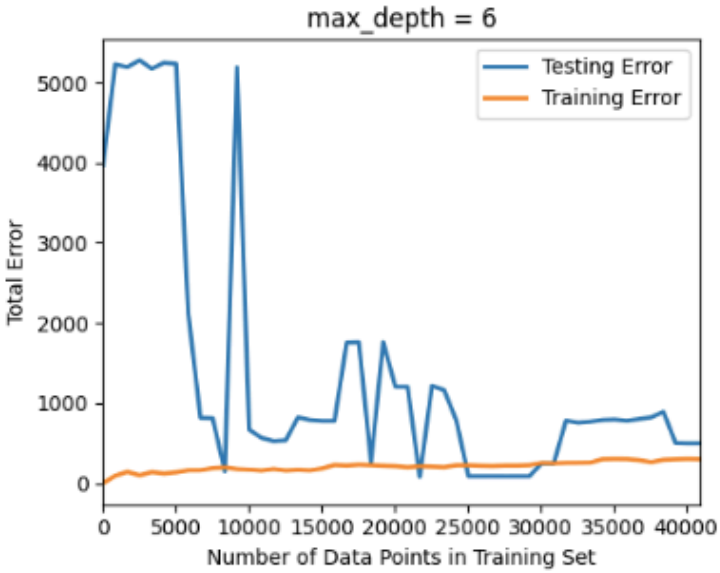
3)
The results do Improve as we can see:
****** DecisionTreeRegressor kernel
Best params learned via GridSearch max_depth=5
r2_score **0.9438733251746643**
Train MSE **388.8503894487183**
Test MSE **201.62340558013054**

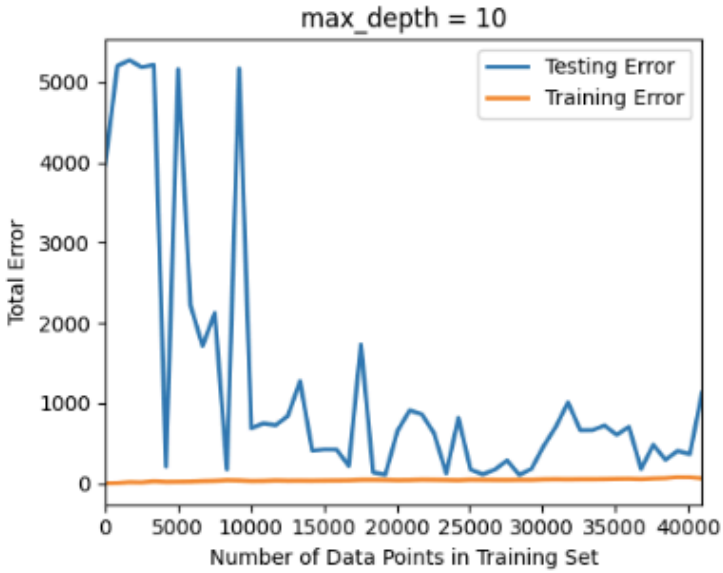
****** RandomForestRegressor
r2_score **0.07053893533560829**
Train MSE **1162.4472186665378**
Test MSE **3338.8955571473716**

In part 3.3 I restrict the X and y to those cases where h = 24 in the loaddata() method to result in the following graphs:





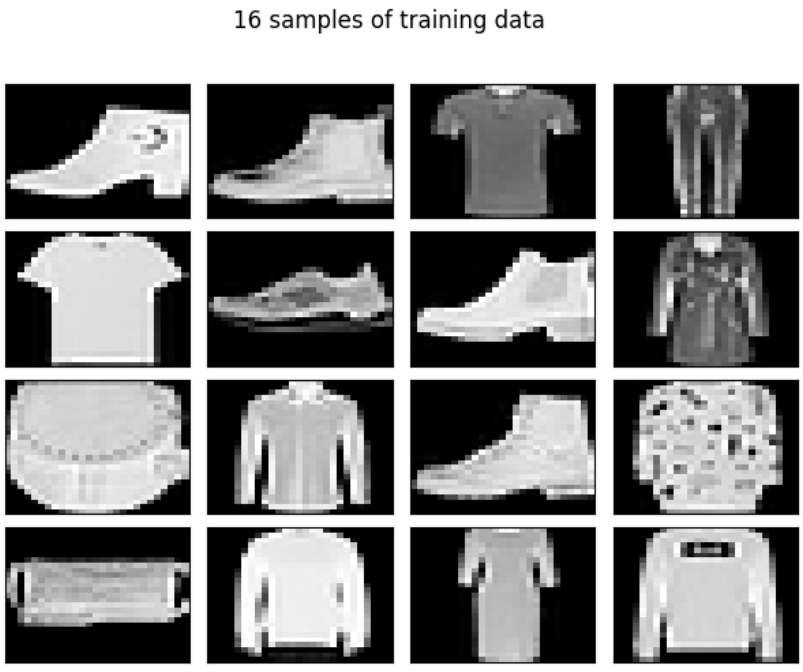




Mh223vk Mehdi Hmidi

Exercise 4

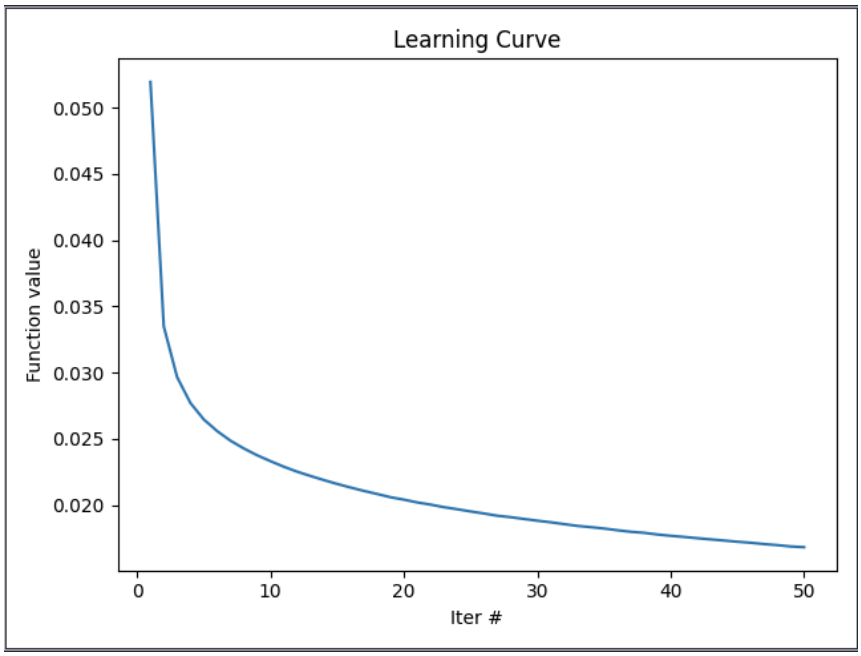
1) **Plot 16 random samples from the training set with the corresponding labels**.



Mh223vk Mehdi Hmidi

2) Train a multilayer perceptron to achieve at least 82% test accuracy.

Already at Epoch 25/50 we can reach loss: 0.0195 - mse: 0.0195 - accuracy: 0.8759
At Epoch 50 we achieve more than > 0.91



3) Plot the confusion matrix. Which are the easy/hard categories to classify? Are there any particular classes that often gets mixed together?

The model classified the “trouser” class 96% correctly but seemed to struggle quite a bit with the “shirt” class (~58% accurate).

T-shirts and shirts class are the most often to be confused together.