

HW3 — ISYE6416

Ashish Dhiman — ashish.dhiman9@gatech.edu

May 7, 2023

Contents

1	Question 1	2
1.1	part a: Algorithm Efficiency	2
1.2	part b: Algorithm Robustness	2
1.3	part c: Algorithm Stability	2
1.4	part d: Algorithm Accuracy	2
1.5	part e: Worst case complexity of Quick Sort	2
1.6	part f: Bisection	4
1.7	part g: Matrix Multiplication Complexity	4
2	Question 2: Rank of Product of matrices	5
3	Question 3: k-means Algorithm	5
3.1	Centroid calculation	5
3.2	Cluster Assignment	6
4	Question 4: Spline and Smoothing Spline	7
4.1	part a: Expression of spline function	7
4.2	part b: Spline with noise	8
5	Question 5: Histogram and KDE	9
5.1	part a: 1D Histogram and KDE	9
5.2	part b: 2D Density estimate from Histogram	10
5.3	part c: 2D Density estimate from KDE	10
5.4	part d: Condition acc and amygdala on orientation	12
6	Question 6: Nonlinear Regression using spline	13
6.1	part a: Linear Regression	13
6.2	part b: Spline fitting	15
7	References:	15

1 Question 1

1.1 part a: Algorithm Efficiency

Efficiency of an algorithm refers to the amount of resources it needs to complete its designated task. Looking at the specific types of resources helps define the two types of efficiency measures:

1. **Time Complexity:** Concerned with the time an algorithm needs to complete a task
2. **Space Complexity:** Concerned with the amount of memory resources the algorithm expends

1.2 part b: Algorithm Robustness

Algorithm Robustness is the measure of algorithms to absorb and accept variability in input data and without failing. In other words, an algorithm is robust if it can accommodate large amounts of variability in the data, including invalid or unexpected inputs. An example of a robust algorithm is Linear Robust optimisation which allows for uncertainty in coefficients of LP constraints.

1.3 part c: Algorithm Stability

Algorithm Stability is the property where minor alterations in input do not cause a lot of variability in the output of the algorithm. The key difference b/w robustness and stability is that, while stability focuses on putting inertia on output, i.e. minimal delta in output, while robustness is focused more on maintaining feasibility of output.

1.4 part d: Algorithm Accuracy

Two types of algorithm accuracy are:

1. **Absolute acc.:** deviation between true and predicted value or

$$= |x_{true} - x_{pred}|$$

2. **Relative acc.:** ratio of absolute deviation to true value or

$$= \frac{|x_{true} - x_{pred}|}{x_{true}}$$

Sometimes approximate algorithms are preferable because they might be more efficient, or more stable and robust. for example: Stochastic Gradient vs gradient descent

1.5 part e: Worst case complexity of Quick Sort

Worst case scenario of Quick sort occurs for cases where pivot is at the extreme end. In such cases, the recursive algorithm does not create any gains.

Let $T(N)$ be time complexity to sort N elements
then for worst case i.e pivot on one end
 $T(N) = N + T(N - 1)$ (find pivot from N elements + sort rest $N-1$)
Similarly
 $T(N - 1) = (N - 1) + T(N - 2)$
 $\implies T(N) = N + N - 1 + N - 2 + \dots + 2 + T(1)$ (1)
With one element no sorting required, or $T(1) = 0$
 $\implies T(N) = N + N - 1 + \dots + 2$
 $= \frac{(N)(N + 1)}{2} - 1$
 $= O(N^2)$

Hence Proved

1.6 part f: Bisection

The problem boils down to finding roots of $f(x) = F(x) - 0.95$ where F is cdf of t distribution with 5 dof. The implementation of bisection algorithm is given below:

```
1 from scipy.stats import t
2
3 df = 5
4 start,end = 1.291,2.582
5
6 f= lambda x: t.cdf(x,df=df) - 0.95
7 f_start = f(start)
8 f_end = f(end)
9
10 print (f"Starting range : ({start:.4f},{end:.4f})")
11
12 c=0
13 while (end-start)>=10**(-4):
14     c+=1
15     mid = 0.5*(start+end)
16     f_mid = f(mid)
17     if f_mid >0:
18         end = mid
19     if f_mid<0:
20         start = mid
21     print (f"search range for iter={c} is ({start:.4f},{end:.4f})")
22
23 print (f"True Root of f:{t.ppf(0.95,df=df):.4f}")
```

The output for the code is given below:

```
1 Starting range : (1.2910,2.5820)
2 search range for iter=1 is (1.9365,2.5820)
3 search range for iter=2 is (1.9365,2.2592)
4 search range for iter=3 is (1.9365,2.0979)
5 search range for iter=4 is (1.9365,2.0172)
6 search range for iter=5 is (1.9768,2.0172)
7 search range for iter=6 is (1.9970,2.0172)
8 search range for iter=7 is (2.0071,2.0172)
9 search range for iter=8 is (2.0121,2.0172)
10 search range for iter=9 is (2.0147,2.0172)
11 search range for iter=10 is (2.0147,2.0159)
12 search range for iter=11 is (2.0147,2.0153)
13 search range for iter=12 is (2.0150,2.0153)
14 search range for iter=13 is (2.0150,2.0151)
15 search range for iter=14 is (2.0150,2.0151)
16 True Root of f:2.0150
```

1.7 part g: Matrix Multiplication Complexity

Let there be two matrices:

$$A = \begin{bmatrix} \cdot \end{bmatrix}_{n,m}, B = \begin{bmatrix} \cdot \end{bmatrix}_{m,k}$$

Now product AB can be seen as dot product between n rows of A and k columns of B . Each of these dot products in itself requires a multiplication of m elements. Therefore the time complexity of matrix product AB is $O(nmk)$.

Now If we both A and B are square matrices of dimensions (n, n) , the order becomes $O(n * n * n) = O(n^3)$

2 Question 2: Rank of Product of matrices

We are given two matrices:

$$A = []_{4,3}, B = []_{3,5}$$

Also it is given

$$\text{Rank}(A) = 2$$

Now we have:

$$r = \text{Rank}(AB) = \min\{\text{Rank}(A), \text{Rank}(B)\}$$

But

$$\text{Rank}(B) \leq \min(3, 5) \leq 3$$

$$\Rightarrow r = \begin{cases} 2 & \text{Rank}(B) \in \{2, 3\} \\ 1, & \text{Rank}(B) = 1 \end{cases}$$

- Example of B with Rank =1 so r = 1, $B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
- Example of B with Rank =2 so r = 2, $B = \begin{bmatrix} 1 & 0 & 2 & 0 & 2 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
- Example of B with Rank =2 so r = 3, $B = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

3 Question 3: k-means Algorithm

Given cost function,

$$J = \sum_{i=1}^m \sum_{j=1}^k c_{ij} \|x_i - \mu_j\|_2^2,$$

3.1 Centroid calculation

Given cluster assignment c_{ij} we want find cluster centroids μ_j that minimise the above cost function.

First order Optimality condition : $\nabla_{\mu_j} J = 0$

$$\begin{aligned} \nabla_{\mu_j} \left(\sum_{i=1}^m [c_{i1}|x_i - \mu_1|_2^2 + c_{i2}|x_i - \mu_2|_2^2 + \dots + c_{ik}|x_i - \mu_k|_2^2] \right) \\ \Rightarrow \nabla_{\mu_j} J = \sum_{i=1}^m -2 * c_{ij} (x_i - \mu_j) = 0 \quad (2) \\ \Rightarrow \sum_{i=1}^m (c_{ij} x_i) = \sum_{i=1}^m (c_{ij}) \mu_j \\ \Rightarrow \mu_j = \frac{\sum_{i=1}^m (c_{ij} x_i)}{\sum_{i=1}^m (c_{ij})} \end{aligned}$$

The first order condition is enough in this case because J is Convex in μ

3.2 Cluster Assignment

Now given cluster centroids μ_j we want cluster assignment c_{ij} that minimises the above cost function.

The above cost function has a minimum value of 0 and it is attained when the $c_{ij}=0$. However that would mean no cluster assignment, and we want one point to be assigned to one cluster. In other words:

$$\sum_j j = 1^k c_{ij} = 1$$

We also know that for a any given set of points $x_1, x_2, \dots x_n$

$$\sum_i |x_i - \bar{x}|_2^2 < \sum_i |x_i - x_j|_2^2$$

where \bar{x} is centroid of $x_1, x_2, \dots x_n$.

Thus putting the two properties together, we land at the optimal value of c_{ij} as :

$$c_{ij} = \begin{cases} 1 & \text{if } j = \arg \min_{j'} |x_i - \mu_{j'}|_2^2 \\ 0, & \text{otherwise} \end{cases}$$

4 Question 4: Spline and Smoothing Spline

4.1 part a: Expression of spline function

We want $s(x)$, such that it passes through (x_i, y_i) , $i = 0, 1, 2$ exactly.

Let s_{01} spline function between x_0 and x_1 , and similarly for s_{12} .

Also let $h = x_1 - x_0 = x_2 - x_1$

With also impose:

- s to be continuous, $s_{01}(x_1) = s_{12}(x_1)$
- s to be continuously differentiable, $s'_{01}(x_1) = s'_{12}(x_1)$
- s second derivative differentiable, $s''_{01}(x_1) = s''_{12}(x_1)$
- Fixed curvature at endpoints $s''_{01}(x_0) = s''_{12}(x_2) = 0$

Now given we are fitting cubic spline, we can write

$$s_{01}(x) = a_0 + b_0(x - x_0) + c_0(x - x_0)^2 + d_0(x - x_0)^3$$

$$s_{12}(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3$$

First derivative is

$$s'_{01}(x) = b_0 + 2c_0(x - x_0) + 3d_0(x - x_0)^2$$

$$s'_{12}(x) = b_1 + 2c_1(x - x_1) + 3d_1(x - x_1)^2$$

Second derivative is

$$s''_{01}(x) = 2c_0 + 6d_0(x - x_0)$$

$$s''_{12}(x) = 2c_1 + 6d_1(x - x_1)$$

Using $s''_{01}(x_0) = 0$ and $s''_{12}(x_2) = 0$

$$c_0 = 0$$

$$2c_1 = -6d_1h \implies c_1 = -3d_1h$$

Similarly we can use $s_{01}(x_1) = s_{12}(x_1)$

$$a_0 + b_0(x_1 - x_0) + c_0(x_1 - x_0)^2 + d_0(x_1 - x_0)^3 = a_1$$

$$\implies a_0 + b_0h + c_0h^2 + d_0h^3 = a_1$$

Also we can use $s'_{01}(x_1) = s'_{12}(x_1)$

$$b_0 + 2c_0(x_1 - x_0) + 3d_0(x_1 - x_0)^2 = b_1$$

$$\implies b_0 + 2c_0h + 3d_0h^2 = b_1$$

Also we can use $s''_{01}(x_1) = s''_{12}(x_1)$

$$2c_0 + 6d_0h = 2c_1$$

$$c_0 + 3d_0h = c_1$$

$$\implies c_1 = 3d_0h$$

Also we can use there is no noise, so

$$\implies a_0 = y_0$$

$$\implies a_1 = y_1$$

$$\implies y_2 = a_1 + b_1h + c_1h^2 + d_1h^3$$

(3)

Solving the above we get:

$$\begin{aligned}
a_0 &= y_0 \\
b_0 &= \frac{6y_1 - 5y_0 - y_2}{4h} \\
c_0 &= 0 \\
d_0 &= \frac{y_0 - 2y_1 + y_2}{4h^3} \\
a_1 &= y_1 \\
b_1 &= \frac{y_2 - y_0}{2h} \\
c_1 &= \frac{3(y_0 - 2y_1 + y_2)}{4h^2} \\
d_1 &= \frac{2y_1 - y_0 - y_2}{4h^3}
\end{aligned} \tag{4}$$

With the above coefficient values we have the spline functions parameterised in terms of input points.

4.2 part b: Spline with noise

We want

$$\min_s \quad \alpha \sum_{i=0}^2 (y_i - f_i)^2 + (1 - \alpha) \int_{x_0}^{x_2} [s''(x)]^2 dx$$

We know the solution of above equation is given by:

$$\sigma = M^{-1}Q[\alpha I + (1 - \alpha)IQ^T M^{-1}Q]^{-1}\alpha Iy$$

For alpha = 1/2, we have:

$$\sigma = M^{-1}Q[\frac{1}{2}I + \frac{1}{2}IQ^T M^{-1}Q]^{-1}\frac{1}{2}Iy$$

$$\sigma = [\frac{1}{2}M^{-1}Q + \frac{1}{2}M^{-1}QQ^T M^{-1}Q]\frac{1}{2}Iy$$

5 Question 5: Histogram and KDE

5.1 part a: 1D Histogram and KDE

Histogram for the two variables is:

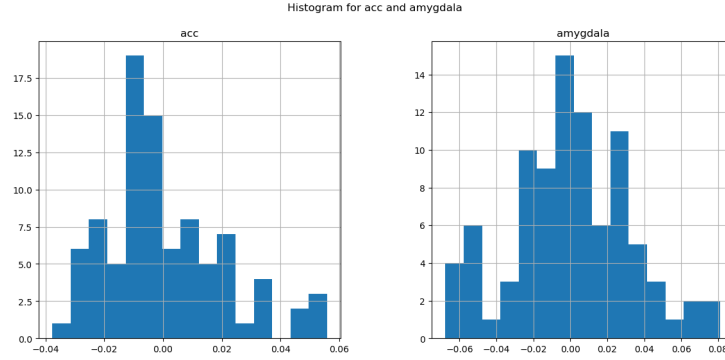


Figure 1: Histogram of acc(left) and amygdala (right)

Because there are 90 points in data, I have chosen number of bins as 15, with 6 points in each bin.

We can see that the two distributions are not normal, and have great thinning out between the center and the tails. Distribution of amygdala also looks more symmetric.

KDE for the two variables is:

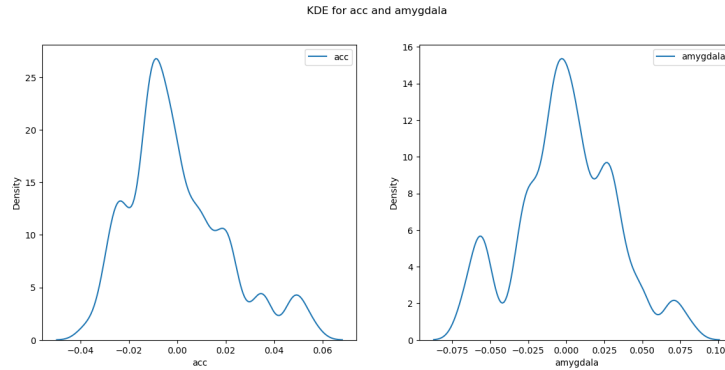


Figure 2: KDE with gaussian kernel of acc(left) and amygdala (right)

With proper tuning of bandwidth the distribution from KDE echoes the insights from Histogram, i.e thinning out before tails and relatively more symmetric density for amygdala.

5.2 part b: 2D Density estimate from Histogram

Since we have 90 points, I have picked number of bins for each variable as 6, giving a total of 36 bins. In my tuning I find this to be the optimum, because on average we get 2 to 3 points in each bin. Also a higher value becomes too granular, while lower value is too blurred.

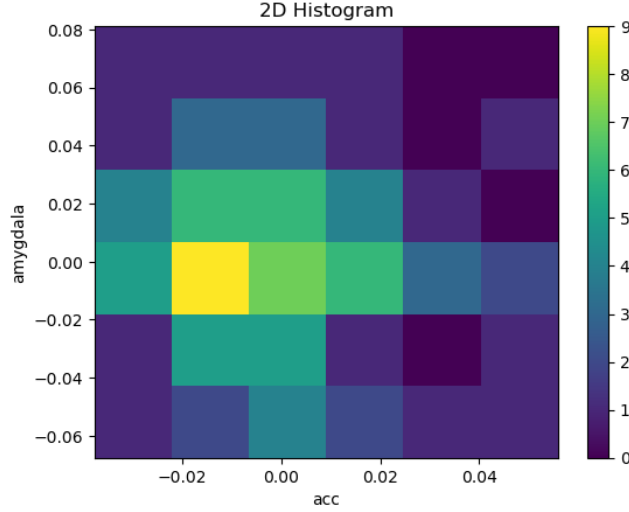


Figure 3: 2D Histogram

As we see the joint distribution is unimodal, with the location of mode being $acc \in (-0.02, 0.00)$ and $amygdala \in (-0.02, 0.00)$

5.3 part c: 2D Density estimate from KDE

The density plot below is given in the form of contour plots:

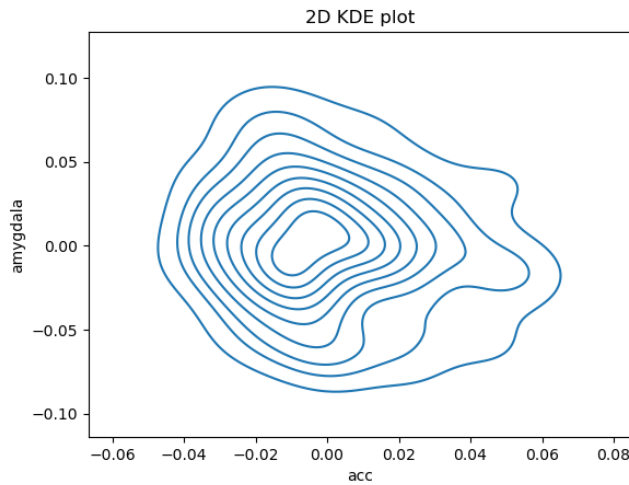


Figure 4: Joint density plot with KDE

Independence

Based on the joint density plot the two variables do seem independent. We can see that if I suppose I draw a horizontal line to get $\hat{P}(acc|amygdala = c)$ the distribution is roughly the same as the marginal KDE density from part a above. Similarly, if instead I imagine a vertical line, similar intuition holds.

The above assertion however is more strong near the mode, while need further qualifications around the tails, especially higher end of acc.

In contrast to above independent density plot, compare the plot of non-independent plot below:

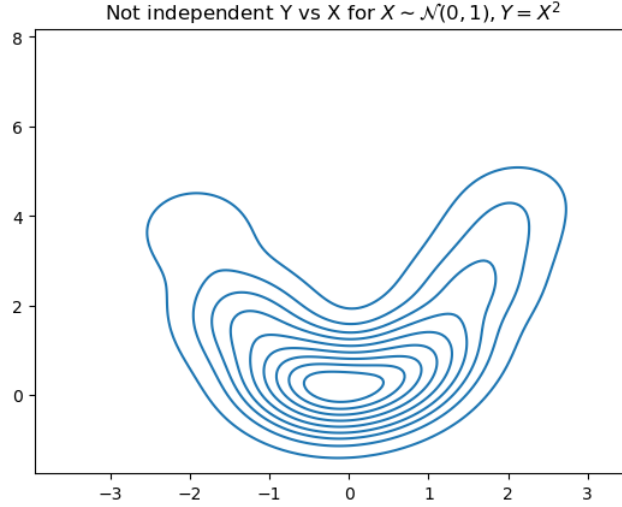


Figure 5: Joint density plot with KDE

The independence argument is further strengthened when looking at scatter plot, since there is little to none apparent correlation.

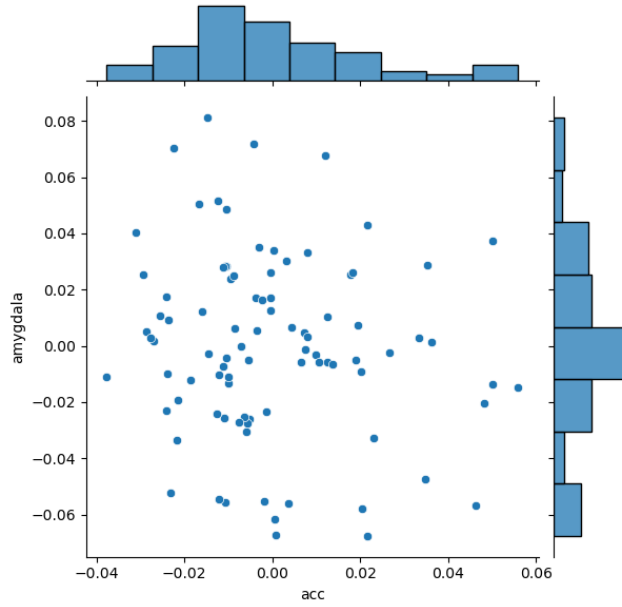


Figure 6: Scatter Plot of acc and amygdala

5.4 part d: Condition acc and amygdala on orientation

KDE Estimate of $\hat{p}(acc|orientation)$

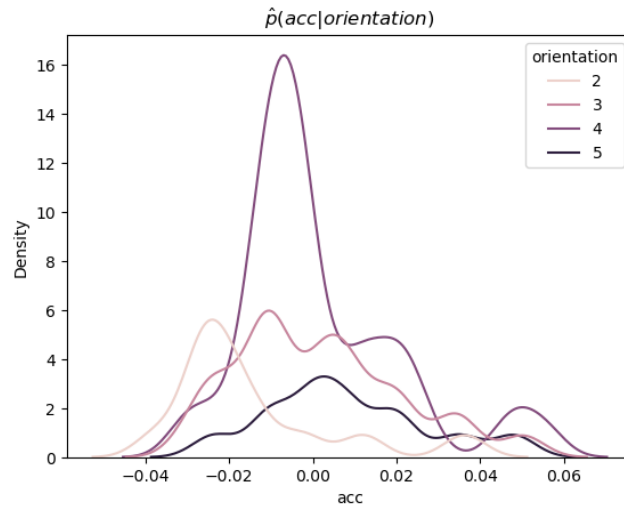


Figure 7: Conditional KDE for acc

From the looks of it seems for conservative political views or orientation =2, acc has a lower mode compared to other orientation values.

KDE Estimate of $\hat{p}(amygdala|orientation)$

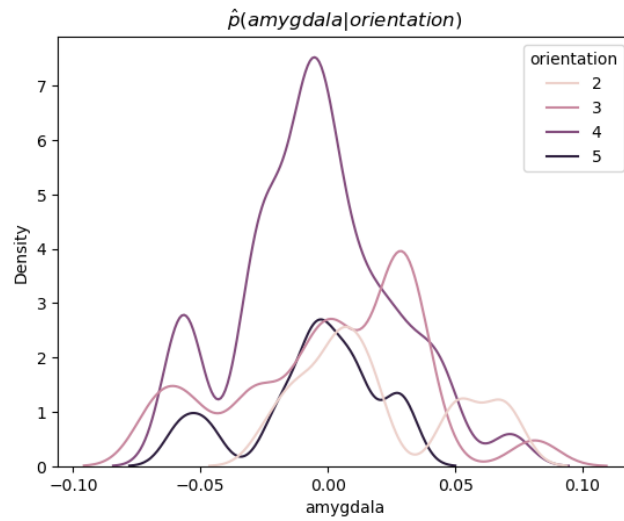


Figure 8: Conditional KDE for amygdala

From the looks of it seems for conservative political views or orientation =2, amygdala has a slightly higher mode compared to other orientation values.

Sample means conditioned on orientation

The conditional sample means of acc and amygdala on basis of Political Orientation is:

<i>orientation</i>	<i>Sample Mean</i>	
	amygdala	acc
2	0.019062	-0.014769
3	0.000588	0.001671
4	-0.004720	0.001310
5	-0.005692	0.008142

Table 1: Sample means of amygdala and acc conditioned on political orientation

Similar to KDE plot, we can see that the for conservative political orientation amygdala has higher values while acc has lower value.

6 Question 6: Nonlinear Regression using spline

6.1 part a: Linear Regression

I have fit an OLS regression after adding intercept term:

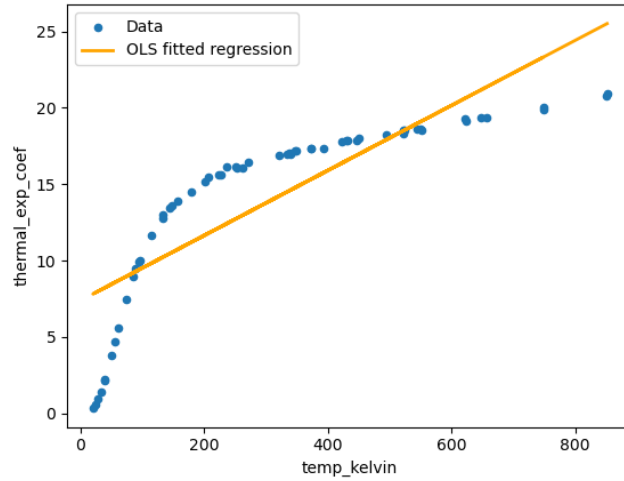


Figure 9: OLS Linear Regression

The model fit is:

$$\hat{y} = 7.3841 + 0.0213x$$

The fitting error is:

$$MSE = \frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = 10.89 \quad (n = 59)$$

The summary of fitted model is:

```

1
2
3
4 Dep. Variable:      thermal_exp_coef      R-squared:
5           0.686
6 Model:              OLS      Adj. R-squared:
7           0.681
8 Method:              Least Squares      F-statistic:
9           124.6
10 Date:              Fri, 24 Feb 2023      Prob (F-statistic):
11           5.70e-16
12 Time:              11:54:28      Log-Likelihood:
13           -153.14
14 No. Observations:      59      AIC:
15           310.3
16 Df Residuals:          57      BIC:
17           314.4
18 Df Model:              1
19 Covariance Type:      nonrobust
20
21
22
23
=====
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
=====
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
=====
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
=====
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
=====
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
=====
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
=====
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
=====
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
=====
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
=====
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
=====

```

6.2 part b: Spline fitting

The leave one out CV curve for smoothing hyperparameter λ is given below:

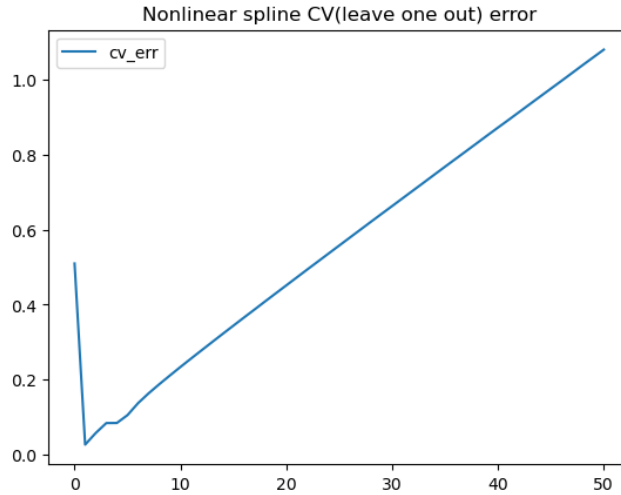


Figure 10: CV curve

From above $\lambda_{best} = 1$
The spline fit for this choice is:

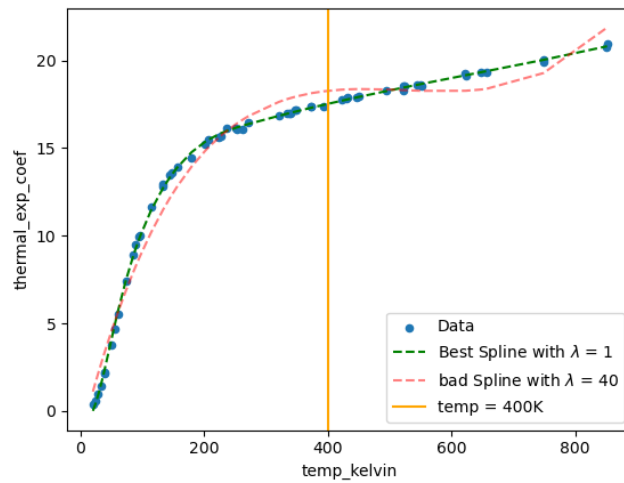


Figure 11: Spline fitting

For temp = 400, OLS fit = 15.90 For temp = 400, Spline fit = 17.52
From the graph hints that non linear spline might be a better fit.

7 References:

1. <https://www.baeldung.com/cs/quicksort-time-complexity-worst-case>
2. [https://en.wikipedia.org/wiki/Robustness_\(computer_science\)](https://en.wikipedia.org/wiki/Robustness_(computer_science))
3. <https://www.statlect.com/matrix-algebra/matrix-product-and-rank>
4. <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-me>

5. https://docs.scipy.org/doc/scipy/tutorial/interpolate/smoothing_splines.html#spline-smoothing-in-1-d