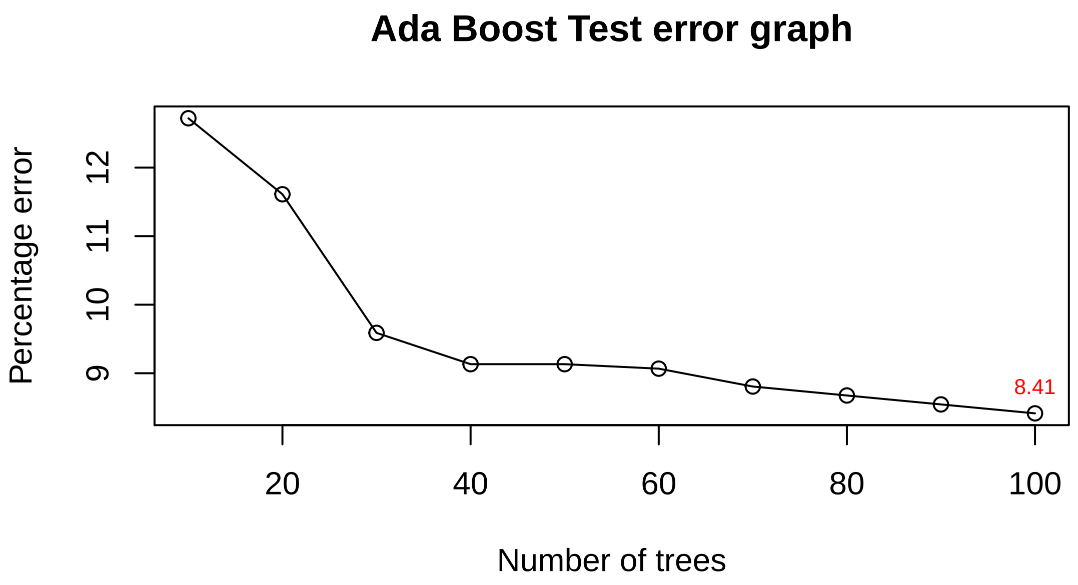
732A99/732A68/ TDDE01 Machine Learning Exam Help Block 2

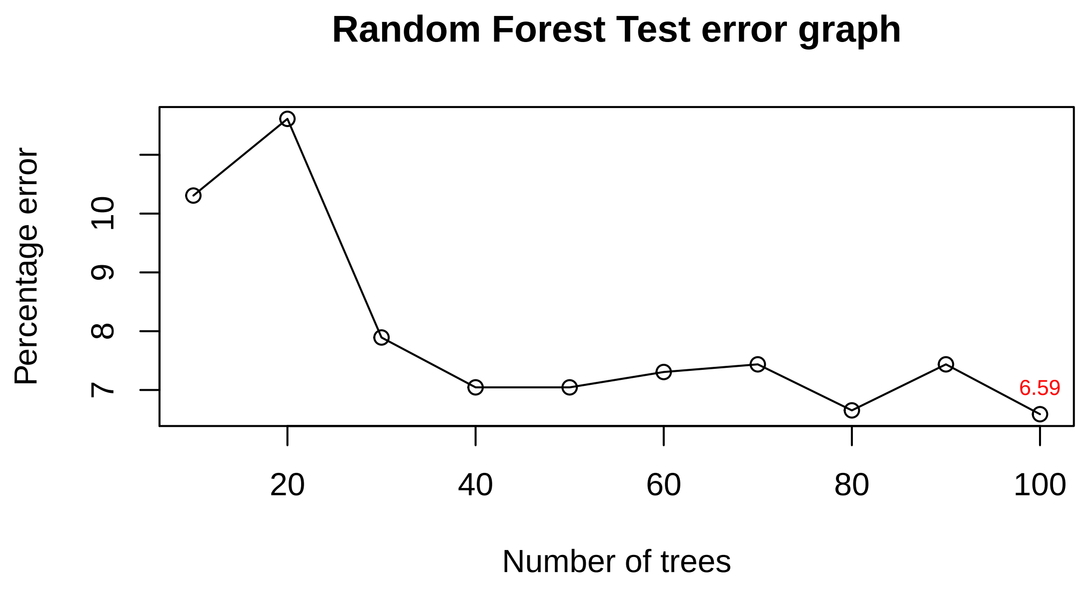
Vyshnavi Pisupati

12/01/2020

Block 2 Lab 1

1. ENSEMBLE METHODS

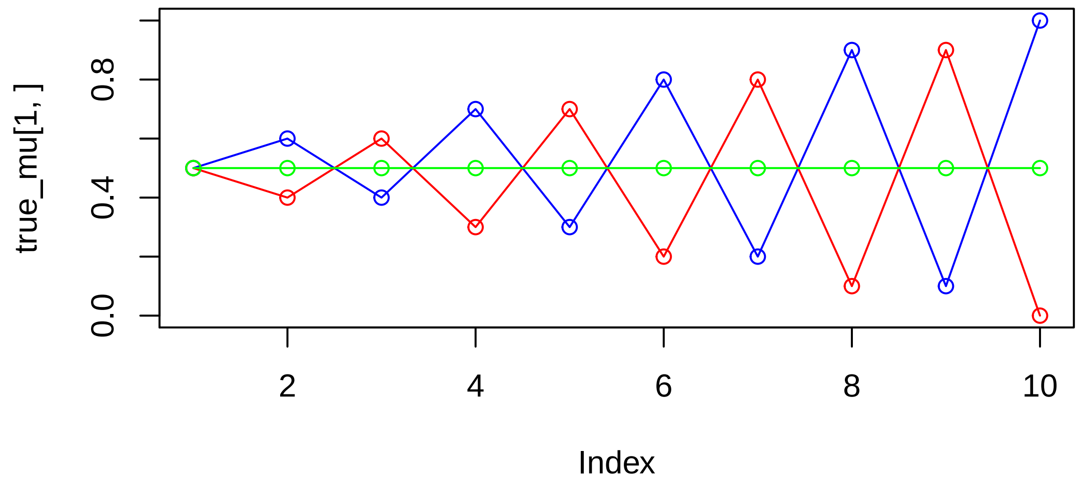




As seen from the above graphs, we can see that the error percentage for random forest is 6.59 which is less than the error percentage for ada boost model which is 8.41, though the number of trees considered in both the cases is 100.Hence random forest is the best model for this data.

2. MIXTURE MODELS

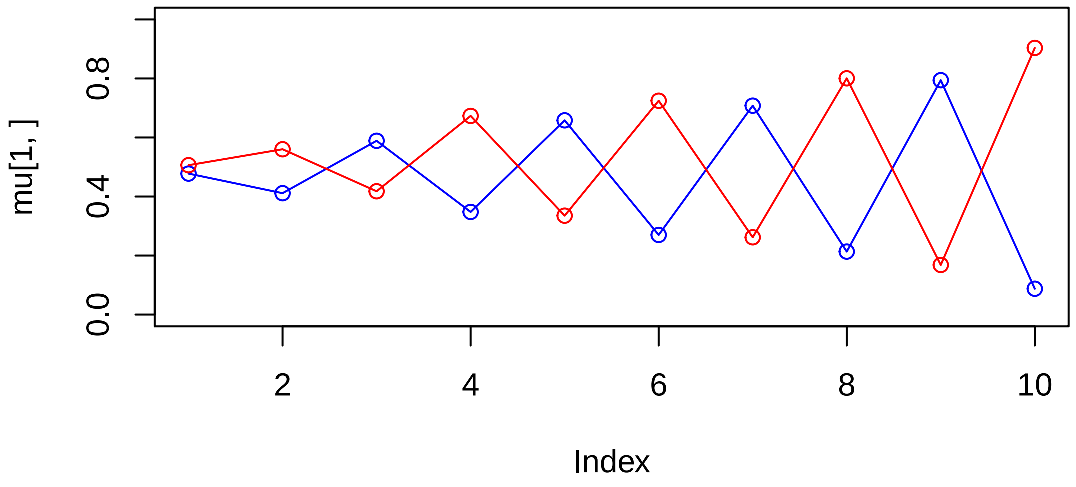
K=2



x
0.4992145
0.5007855

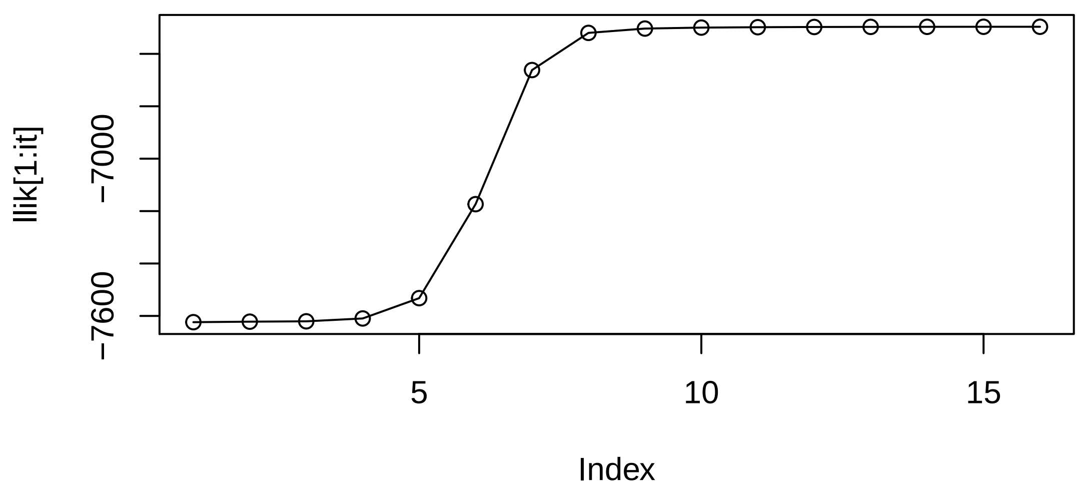
0.4924255	0.4939877	0.4935375	0.5042511	0.5040286	0.4987810	0.5012754	0.4971036	0.4982144	0.4987654
0.4929075	0.4993719	0.5088453	0.5068730	0.5016720	0.4929275	0.5077146	0.5095075	0.4924574	0.4992470

iteration: 1 log likelihood: -7623.873 iteration: 2 log likelihood: -7621.944 iteration: 3 log likelihood: -7620.533 iteration: 4 log likelihood: -7609.638 iteration: 5 log likelihood: -7532.2 iteration: 6 log likelihood: -7173.56 iteration: 7 log likelihood: -6661.821 iteration: 8 log likelihood: -6520.028 iteration: 9 log likelihood: -6503.563 iteration: 10 log likelihood: -6499.807 iteration: 11 log likelihood: -6498.296 iteration: 12 log likelihood: -6497.535 iteration: 13 log likelihood: -6497.12 iteration: 14 log likelihood: -6496.883 iteration: 15 log likelihood: -6496.745 iteration: 16 log likelihood: -6496.662

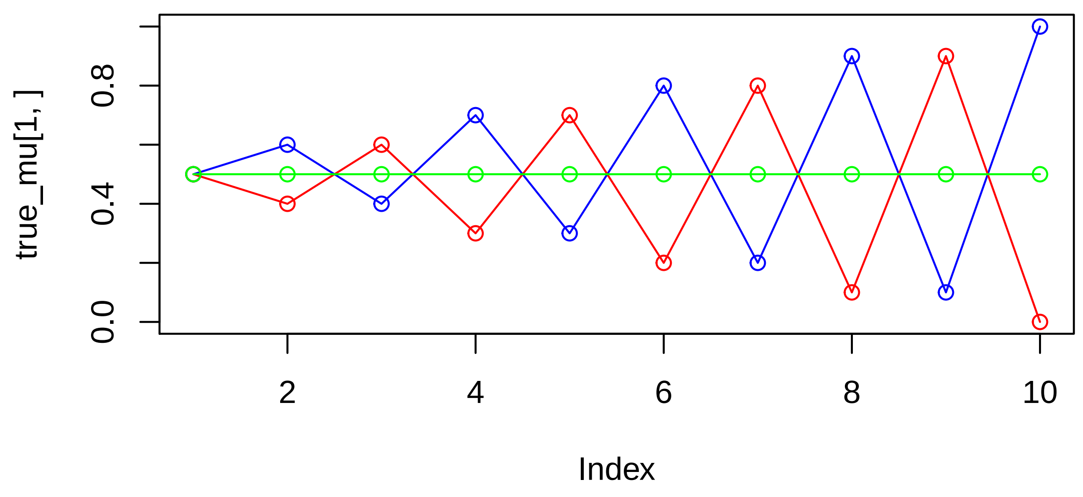


0.5	0.6	0.4	0.7	0.3	0.8	0.2	0.9	0.1	1.0
0.5	0.4	0.6	0.3	0.7	0.2	0.8	0.1	0.9	0.0
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

0.4777136	0.4113065	0.5888317	0.3477062	0.6580979	0.2698303	0.7078467	0.2134061	0.7941168	0.0875152
0.5061835	0.5601552	0.4177868	0.6731171	0.3350702	0.7245255	0.2617664	0.8004711	0.1681467	0.9035340



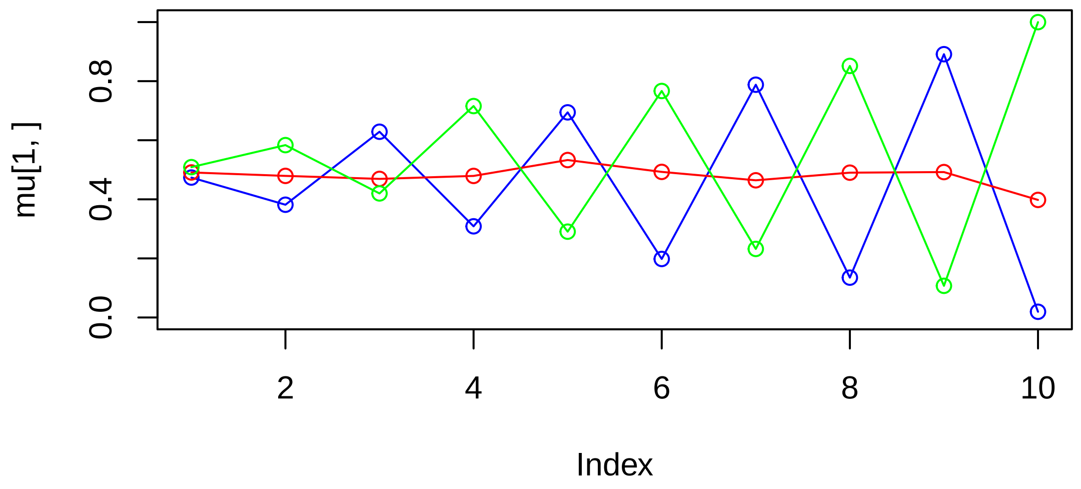
K=3



x
0.3326090
0.3336558
0.3337352

0.4939877	0.4935375	0.5042511	0.5040286	0.4987810	0.5012754	0.4971036	0.4982144	0.4987654	0.4929075
0.4993719	0.5088453	0.5068730	0.5016720	0.4929275	0.5077146	0.5095075	0.4924574	0.4992470	0.5008651
0.4975302	0.5077926	0.4939841	0.5059821	0.5063490	0.5041462	0.4929400	0.4992362	0.4943482	0.4903974

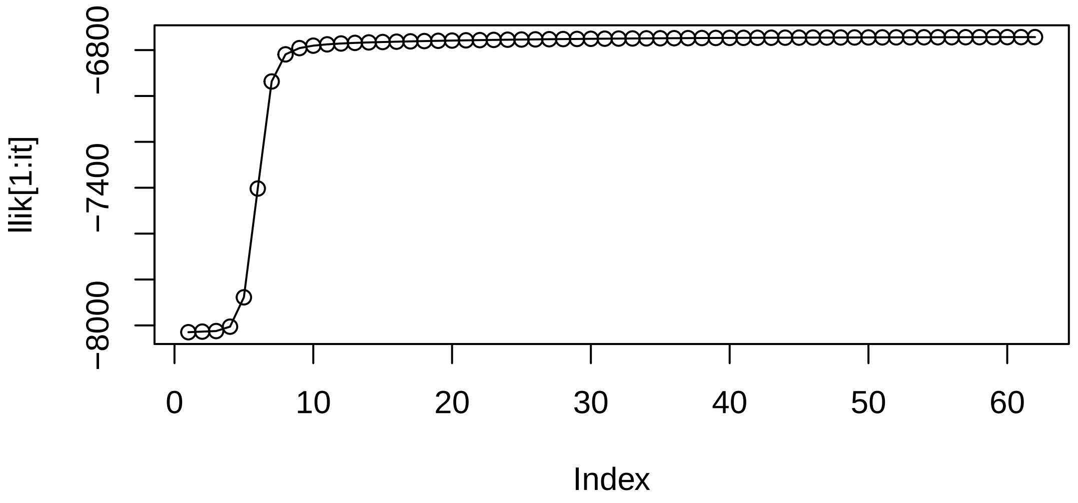
iteration: 1 log likelihood: -8029.723 iteration: 2 log likelihood: -8027.183 iteration: 3 log likelihood: -8024.696 iteration: 4 log likelihood: -8005.631 iteration: 5 log likelihood: -7877.606 iteration: 6 log likelihood: -7403.513 iteration: 7 log likelihood: -6936.919 iteration: 8 log likelihood: -6818.582 iteration: 9 log likelihood: -6791.377 iteration: 10 log likelihood: -6780.713 iteration: 11 log likelihood: -6774.958 iteration: 12 log likelihood: -6771.261 iteration: 13 log likelihood: -6768.606 iteration: 14 log likelihood: -6766.535 iteration: 15 log likelihood: -6764.815 iteration: 16 log likelihood: -6763.316 iteration: 17 log likelihood: -6761.967 iteration: 18 log likelihood: -6760.727 iteration: 19 log likelihood: -6759.572 iteration: 20 log likelihood: -6758.491 iteration: 21 log likelihood: -6757.475 iteration: 22 log likelihood: -6756.521 iteration: 23 log likelihood: -6755.625 iteration: 24 log likelihood: -6754.784 iteration: 25 log likelihood: -6753.996 iteration: 26 log likelihood: -6753.26 iteration: 27 log likelihood: -6752.571 iteration: 28 log likelihood: -6751.928 iteration: 29 log likelihood: -6751.328 iteration: 30 log likelihood: -6750.768 iteration: 31 log likelihood: -6750.246 iteration: 32 log likelihood: -6749.758 iteration: 33 log likelihood: -6749.304 iteration: 34 log likelihood: -6748.88 iteration: 35 log likelihood: -6748.484 iteration: 36 log likelihood: -6748.114 iteration: 37 log likelihood: -6747.767 iteration: 38 log likelihood: -6747.444 iteration: 39 log likelihood: -6747.14 iteration: 40 log likelihood: -6746.856 iteration: 41 log likelihood: -6746.589 iteration: 42 log likelihood: -6746.338 iteration: 43 log likelihood: -6746.102 iteration: 44 log likelihood: -6745.88 iteration: 45 log likelihood: -6745.67 iteration: 46 log likelihood: -6745.472 iteration: 47 log likelihood: -6745.285 iteration: 48 log likelihood: -6745.108 iteration: 49 log likelihood: -6744.939 iteration: 50 log likelihood: -6744.78 iteration: 51 log likelihood: -6744.627 iteration: 52 log likelihood: -6744.483 iteration: 53 log likelihood: -6744.344 iteration: 54 log likelihood: -6744.212 iteration: 55 log likelihood: -6744.086 iteration: 56 log likelihood: -6743.964 iteration: 57 log likelihood: -6743.848 iteration: 58 log likelihood: -6743.736 iteration: 59 log likelihood: -6743.628 iteration: 60 log likelihood: -6743.524 iteration: 61 log likelihood: -6743.423 iteration: 62 log likelihood: -6743.326



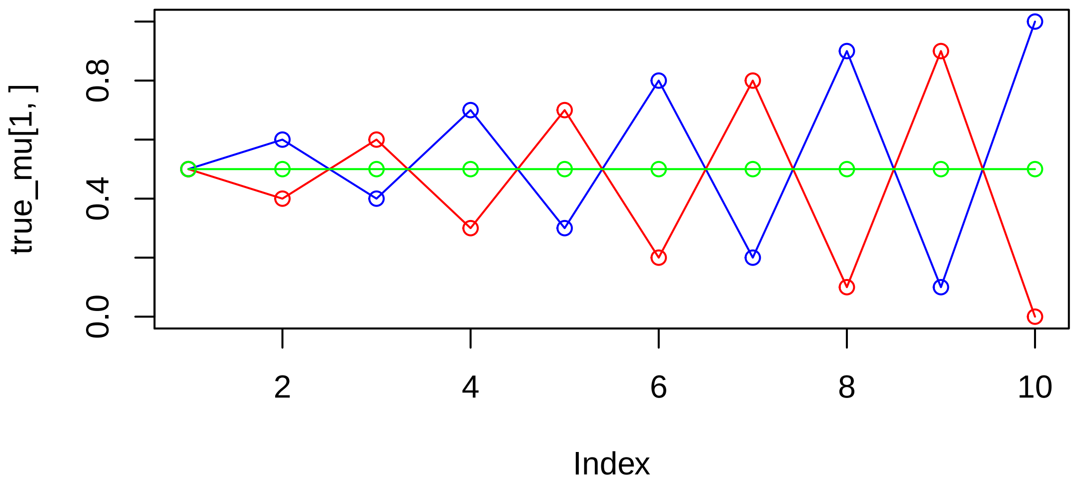
0.5	0.6	0.4	0.7	0.3	0.8	0.2	0.9	0.1	1.0

0.5	0.4	0.6	0.3	0.7	0.2	0.8	0.1	0.9	0.0
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

0.4737193	0.3817120	0.6288021	0.3086143	0.6943731	0.1980896	0.7879447	0.1349651	0.8912534	0.0193787
0.4909874	0.4793213	0.4691560	0.4791793	0.5329895	0.4928830	0.4643990	0.4902682	0.4922194	0.3979841
0.5089571	0.5834802	0.4199272	0.7157107	0.2905703	0.7667258	0.2320784	0.8516111	0.1072226	0.9998135



K=4

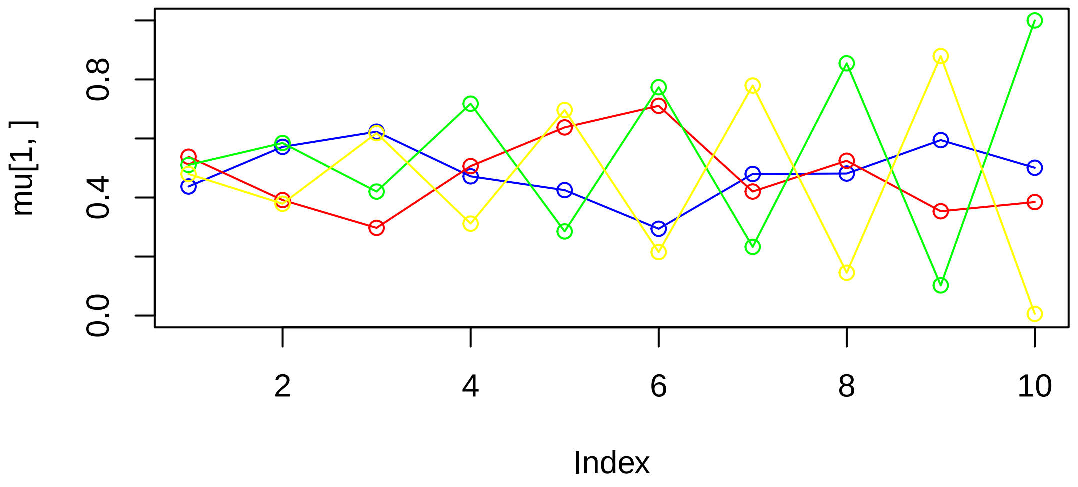


x
0.2491838
0.2499680
0.2500275
0.2508207

0.4935375	0.5042511	0.5040286	0.4987810	0.5012754	0.4971036	0.4982144	0.4987654	0.4929075	0.4993719
0.5088453	0.5068730	0.5016720	0.4929275	0.5077146	0.5095075	0.4924574	0.4992470	0.5008651	0.4975302
0.5077926	0.4939841	0.5059821	0.5063490	0.5041462	0.4929400	0.4992362	0.4943482	0.4903974	0.5089045
0.4909320	0.4982342	0.4961948	0.4967226	0.4984220	0.4960055	0.4997165	0.5090205	0.5057927	0.4947660

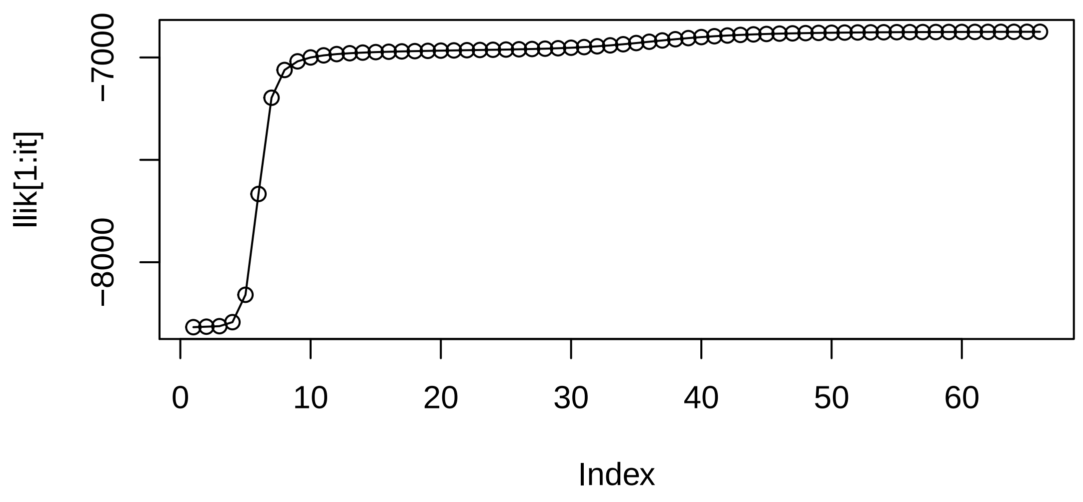
iteration: 1 log likelihood: -8317.187 iteration: 2 log likelihood: -8314.81 iteration: 3 log likelihood: -8312.256 iteration: 4 log likelihood: -8292.606 iteration: 5 log likelihood: -8159.059 iteration: 6 log likelihood: -7666.637 iteration: 7 log likelihood: -7196.701 iteration: 8 log likelihood: -7061.15 iteration: 9 log likelihood: -7018.948 iteration: 10 log likelihood: -6999.971 iteration: 11 log likelihood: -6989.735 iteration: 12 log likelihood: -6983.5 iteration: 13 log likelihood: -6979.315 iteration: 14 log likelihood: -6976.279 iteration: 15 log likelihood: -6973.932 iteration: 16 log likelihood: -6972.026 iteration: 17 log likelihood: -6970.415 iteration: 18 log likelihood: -6969.009 iteration: 19 log likelihood: -6967.751 iteration: 20 log likelihood: -6966.598 iteration: 21 log likelihood: -6965.517 iteration: 22 log likelihood: -6964.48 iteration: 23 log likelihood: -6963.457 iteration: 24 log likelihood: -6962.415 iteration: 25 log likelihood: -6961.313 iteration: 26 log likelihood: -6960.098 iteration: 27 log likelihood: -6958.703 iteration: 28 log likelihood: -6957.042 iteration: 29 log likelihood: -6955.01 iteration: 30 log likelihood: -6952.485 iteration: 31 log likelihood: -6949.342 iteration: 32 log likelihood: -6945.475 iteration: 33 log likelihood: -6940.834 iteration: 34 log likelihood: -6935.458 iteration: 35 log likelihood: -6929.501 iteration: 36 log likelihood: -6923.217 iteration: 37 log likelihood: -6916.917 iteration: 38 log likelihood: -6910.896 iteration: 39 log likelihood: -6905.381 iteration: 40 log likelihood: -6900.502 iteration: 41 log likelihood: -6896.299 iteration: 42 log likelihood:

```
-6892.745 iteration: 43 log likelihood: -6889.776 iteration: 44 log likelihood: -6887.313 iteration: 45 log
likelihood: -6885.273 iteration: 46 log likelihood: -6883.583 iteration: 47 log likelihood: -6882.178 iteration:
48 log likelihood: -6881.007 iteration: 49 log likelihood: -6880.024 iteration: 50 log likelihood: -6879.196
iteration: 51 log likelihood: -6878.494 iteration: 52 log likelihood: -6877.895 iteration: 53 log likelihood:
-6877.383 iteration: 54 log likelihood: -6876.941 iteration: 55 log likelihood: -6876.56 iteration: 56 log
likelihood: -6876.228 iteration: 57 log likelihood: -6875.939 iteration: 58 log likelihood: -6875.687 iteration:
59 log likelihood: -6875.465 iteration: 60 log likelihood: -6875.27 iteration: 61 log likelihood: -6875.098
iteration: 62 log likelihood: -6874.947 iteration: 63 log likelihood: -6874.813 iteration: 64 log likelihood:
-6874.694 iteration: 65 log likelihood: -6874.59 iteration: 66 log likelihood: -6874.497
```



0.5	0.6	0.4	0.7	0.3	0.8	0.2	0.9	0.1	1.0
0.5	0.4	0.6	0.3	0.7	0.2	0.8	0.1	0.9	0.0
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

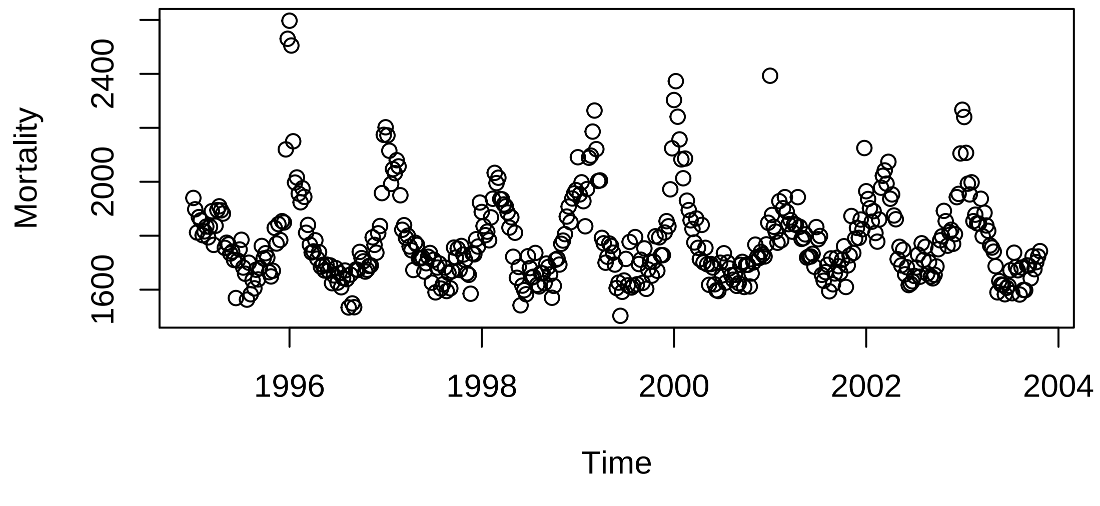
0.4372908	0.5716691	0.6230114	0.4717152	0.4251232	0.2940734	0.4797605	0.4812185	0.5945364	0.5008152
0.5381955	0.3913346	0.2971686	0.5062848	0.6375272	0.7107583	0.4202372	0.5246082	0.3534161	0.3845132
0.5102441	0.5846281	0.4200464	0.7178717	0.2850900	0.7735833	0.2327656	0.8546627	0.1022323	0.9999997
0.4797762	0.3788928	0.6181216	0.3114748	0.6964392	0.2149967	0.7793732	0.1450708	0.8791286	0.0058007

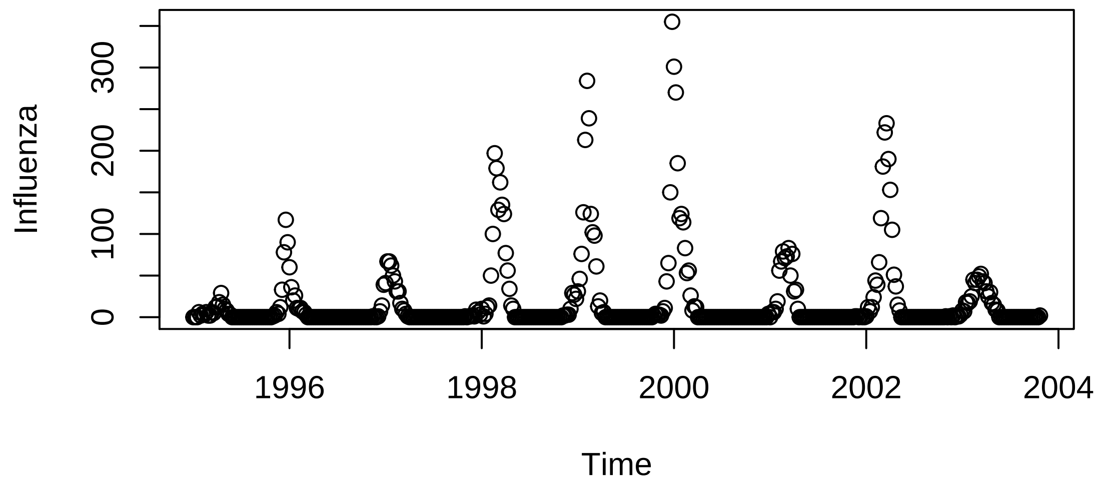


Block 2 Lab 2

Assignment 1. Using GAM and GLM to examine the mortality rates

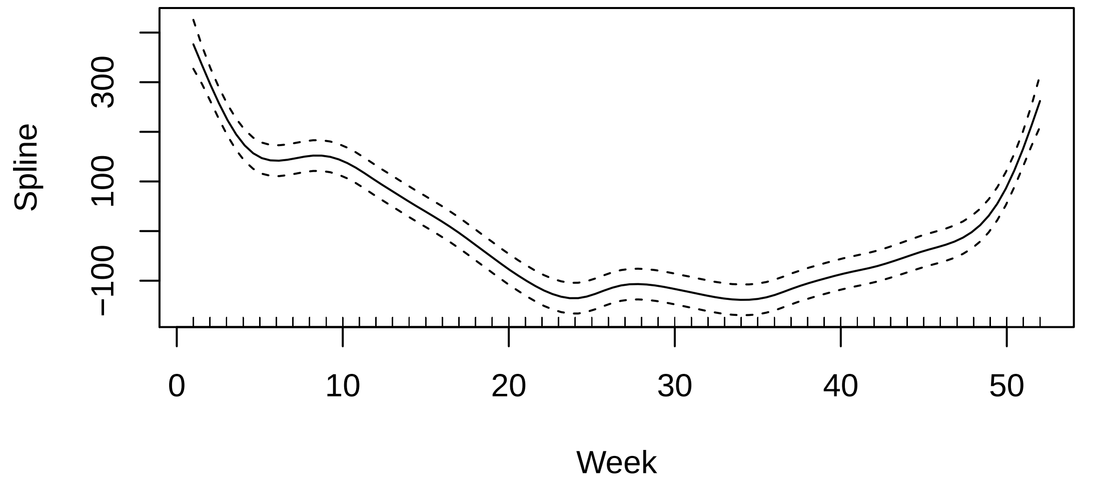
1.1 Plot to vizualize the data



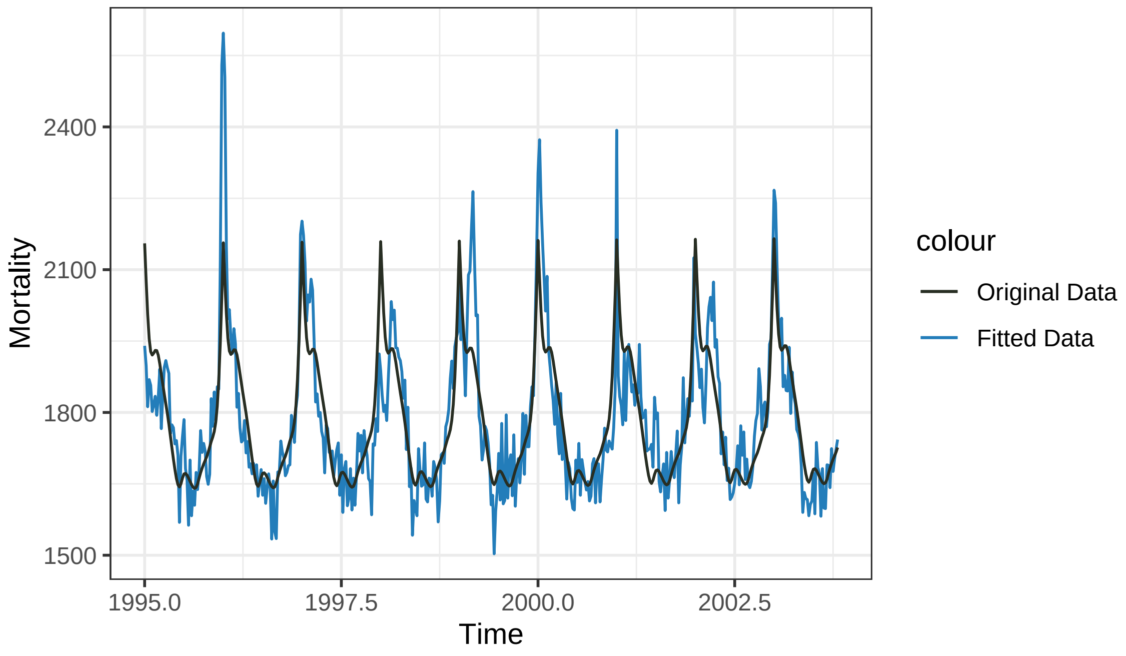


By looking at the data we see that mortality rates and influenza rates are higher at the end of a year and the begining of the suceeding year.We also see that the peaks in influenza match the peaks in the mortality rate.

1.2 Fitting a data set in GAM model



1.3 Analyzing the GAM model



Family: gaussian Link function: identity

Formula: Mortality ~ Year + s(Week, k = length(unique(influenza_data$Week)))

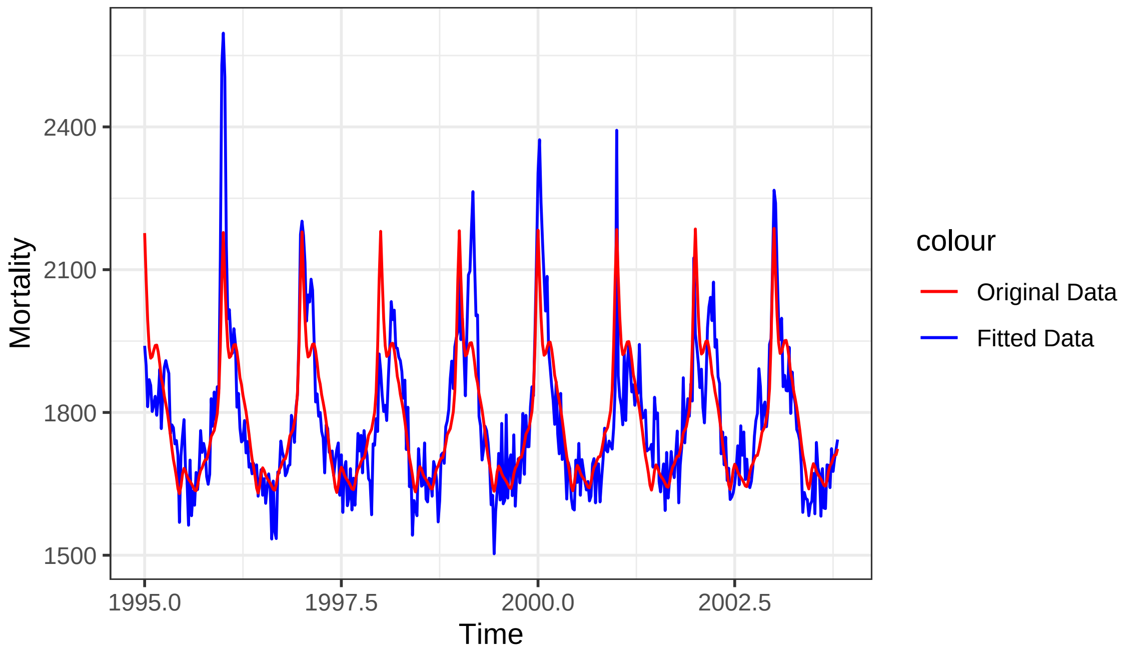
Parametric coefficients: Estimate Std. Error t value Pr(>|t|) (Intercept) -680.598 3367.760 -0.202 0.840 Year 1.233 1.685 0.732 0.465

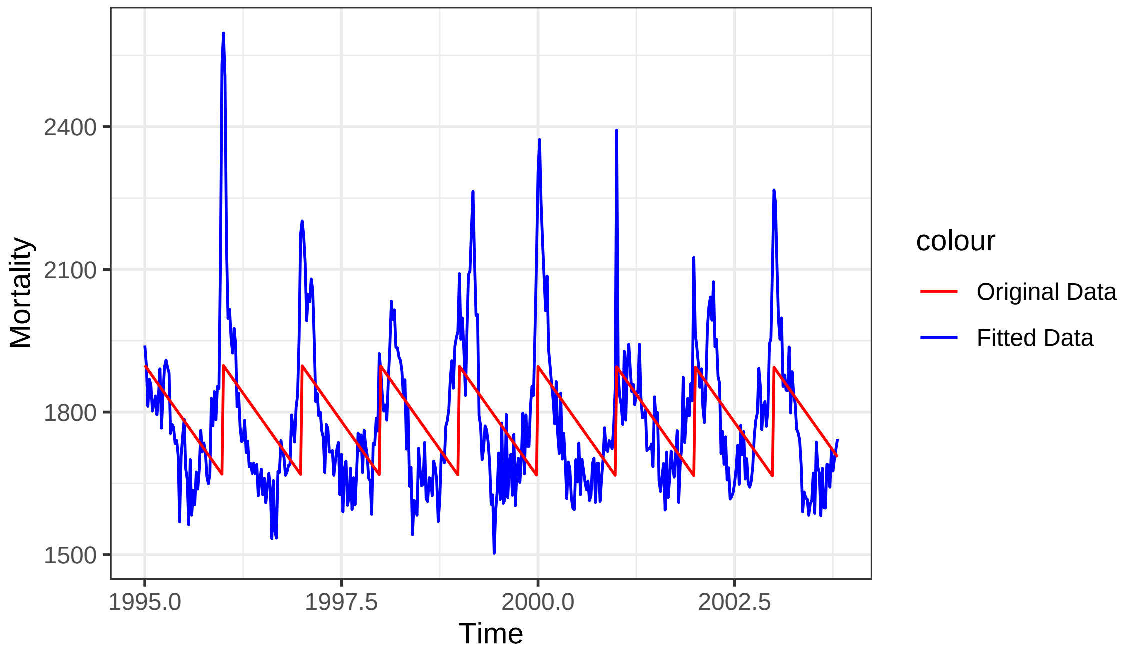
Approximate significance of smooth terms: edf Ref.df F p-value
s(Week) 14.32 17.87 53.86 <2e-16 *** — Signif. codes: 0 ‘’ ***0.001*** ’’ *0.01* ” 0.05 ‘.’ 0.1 ’ ’ 1

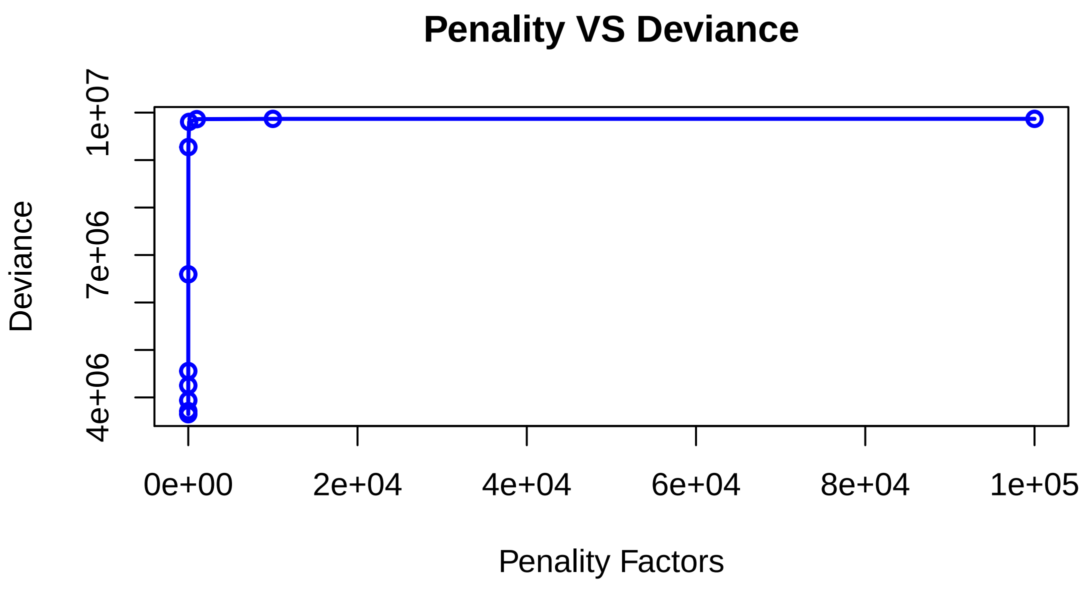
Rank: 52/53 R-sq.(adj) = 0.677 Deviance explained = 68.8% GCV = 8708.6 Scale est. = 8398.9 n = 459

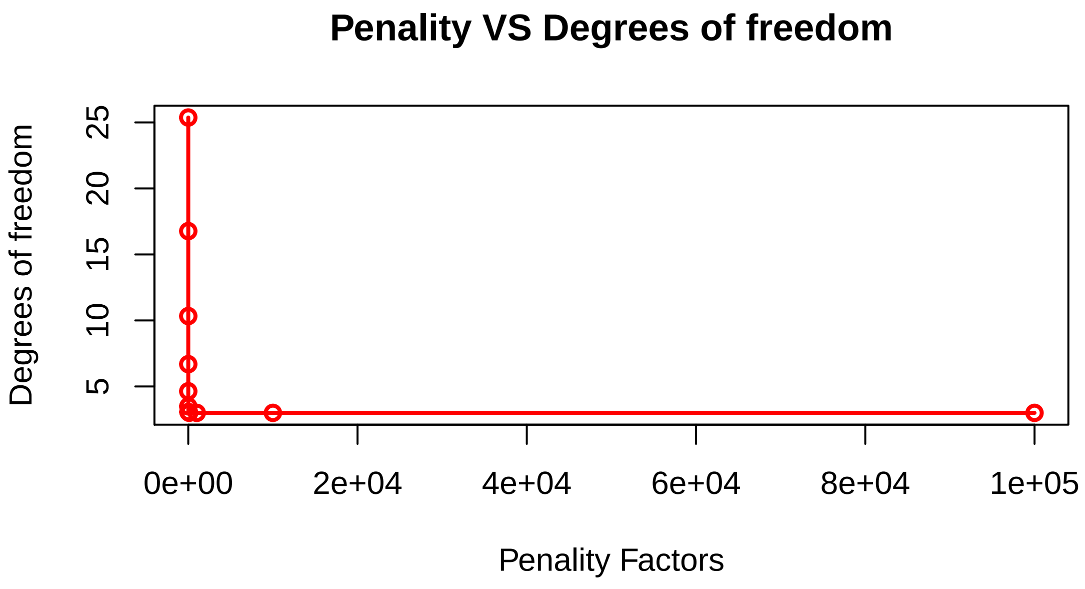
It can be seen form the graph that the mortality predictions are quite good except in the places where mortality rates peak. The predictions when the mortality rates are the highest are not as accurate when the mortality rates are low for some of the years. As seen from the p values of Year and Week, we observe that the Year has a very high p value which implies that this term is not significant. Also the p value of the Week variable is less indicating it is the most significant. We can also see from the plot of the spline component that it models the mortality rates as a function of week quite accurately.

1.4 Examining the relation with the penlity factor on the GAM model







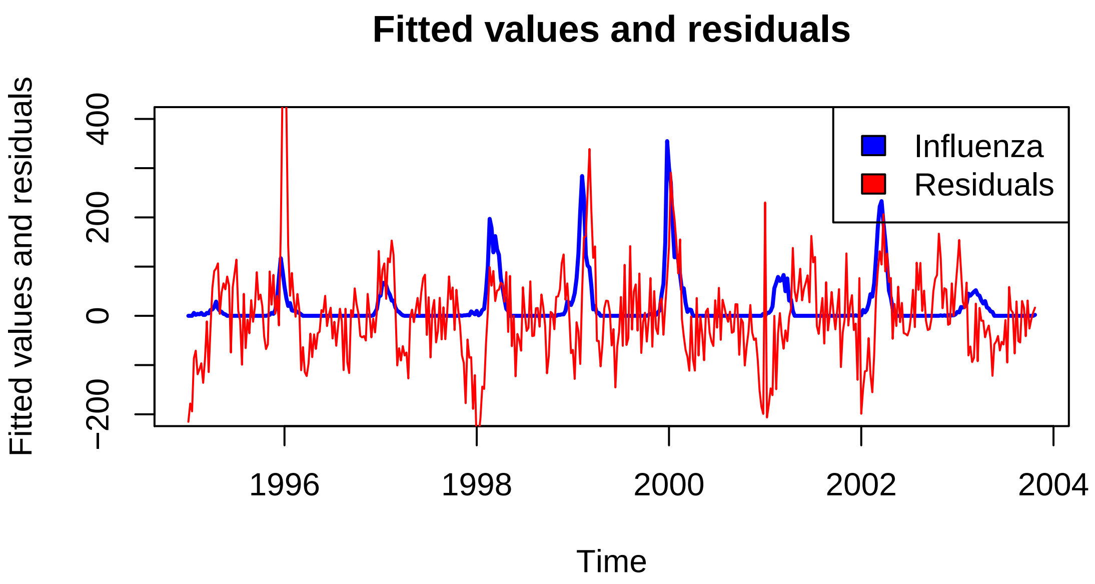


	GCV	Total.df	Deviance
Low Penalty GAM	8902.083	29.432229	3646841
high Penalty GAM	21783.909	3.000015	9868538

We can see that the GAM mode with low penalty is better fitted to our training data. Whereas the GAM model with a very high penalty is not very flexible. The estimated degrees of freedom has reduced to 1 when the model penalized the smooth term heavily to a simple linear relationship. (We can see from the plots of the spline components that increasing the smoothing penalty to a very high value ultimately reduces the spline function to a line) Also, the generalized cross validation score is better and the deviance of the model is lower when the penalty on the smoothing spline is low.

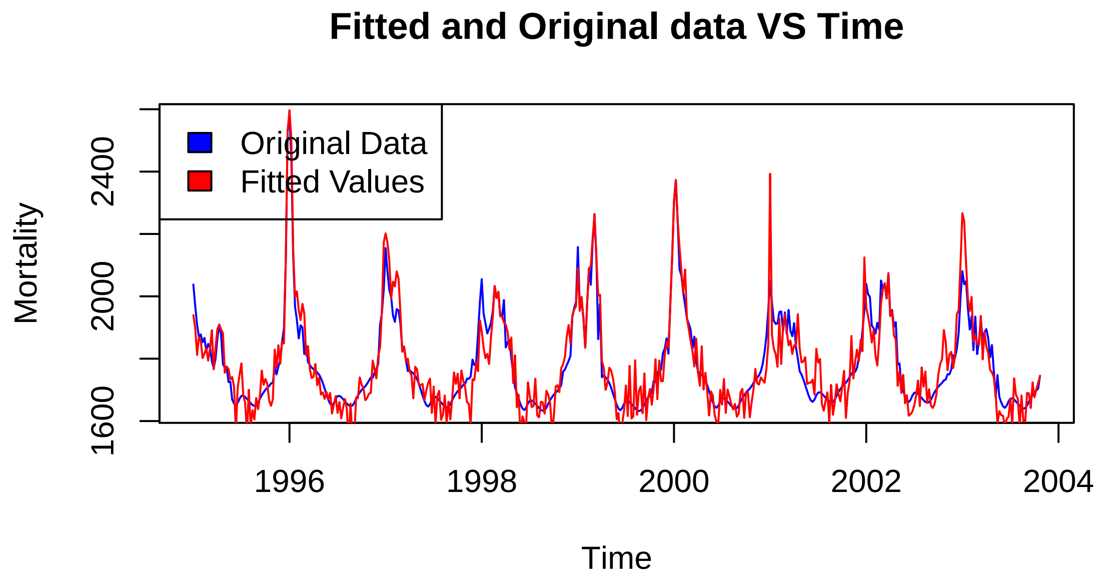
The two plots shown above explain how the penalty factor of the spline function affect the estimated deviance and estimated degrees of freedom of the GAM model in general for different values of penalty. The results confirm that increasing the penalty on the smoothing spline reduces its degrees of freedom making the predictions more linear. Also, increasing the penalty on the spline increases the deviance of the model and hence reduces the goodness of fit.

1.5 Plot of residuals and influenza values against time



The residual values of mortality rate predictions are the high (positive or negative) whenever there are outbreaks in influenza. We do see that the temporal patterns of the residuals are somewhat correlated to the outbreaks of influenza.

1.6 Mortality as an additive function of splines of year, week, and the number of confirmed cases of influenza



Family: gaussian Link function: identity

Formula: Mortality ~ s(Year, k = length(unique(influenza_data$Year)))+s(Week, k = length(unique(influenza_d$ata Week)))
+ s(Influenza, k = length(unique(influenza_data$Influenza)))

Parametric coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) 1783.765 3.198 557.8 <2e-16 *** — Signif. codes: 0 '' ***0.001*** '' *0.01* '' 0.05 '.' 0.1 ' ' 1

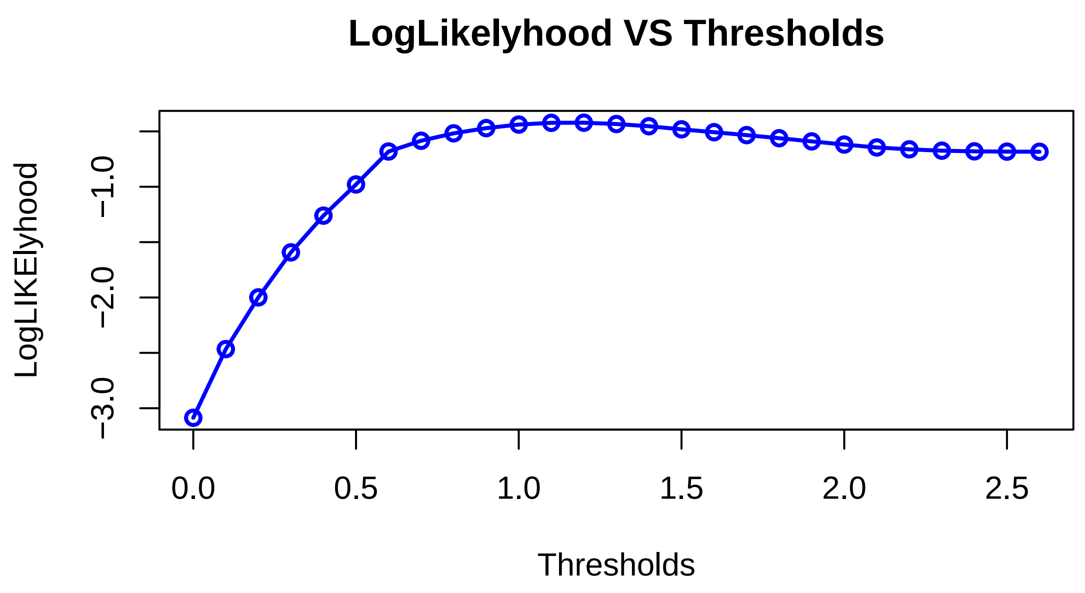
Approximate significance of smooth terms: edf Ref.df F p-value
s(Year) 4.587 5.592 1.500 0.178
s(Week) 14.431 17.990 18.763 <2e-16 ***s(Influenza) 70.094 72.998 5.622 <2e-16*** — Signif. codes: 0
'' ***0.001*** '' *0.01* '' 0.05 '.' 0.1 ' ' 1

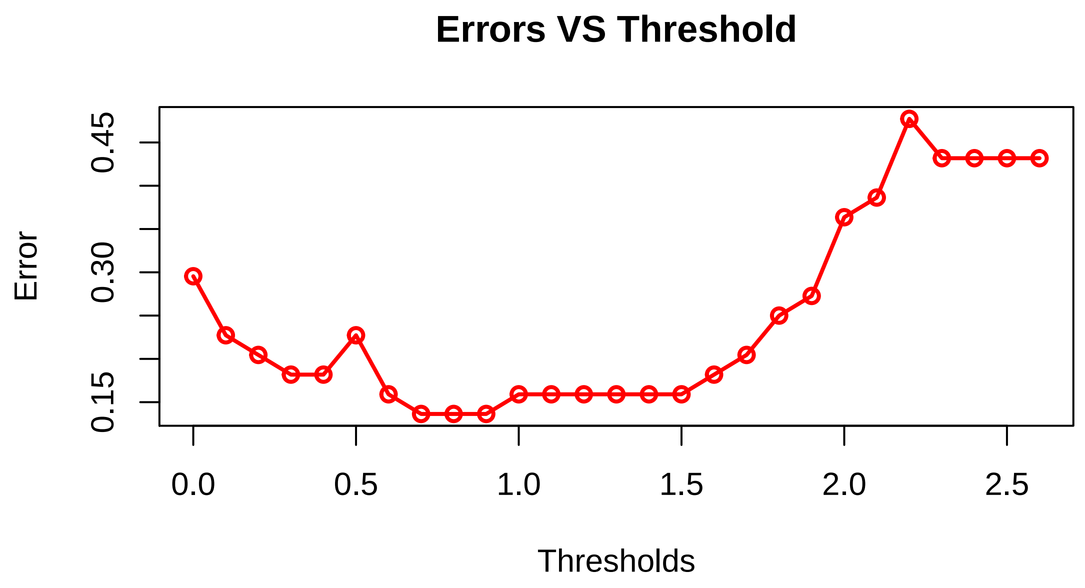
Rank: 134/144 R-sq.(adj) = 0.819 Deviance explained = 85.4% GCV = 5840.5 Scale est. = 4693.7 n = 459

	GCV	Total.df	Deviance
GAM with 3 splines	5840.177	97.58037	1731415
GAM with 1 spline	8708.581	19.86717	3718012

We can seen from the model summary that the spline components of week and influenza cases are both significant terms among other linear predictors. The generalized cross validation score of the GAM model has imcreased and the deviance reduced when compared to the GAM model with only one spline component of week. From the graph of actual and predicted mortality rates, it can be seen that the predictions have improved in the instances where the mortality rates peak. Hence, it can be concluded that this GAM model has the best overall fit.

12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505112Fold 1 :123456789101112131415161718192021222324252627 Fold 2 :123456789101112131415161718192021222324252627 Fold 3 :123456789101112131415161718192021222324252627 Fold 4 :123456789101112131415161718192021222324252627 Fold 5 :123456789101112131415161718192021222324252627 Fold 6 :123456789101112131415161718192021222324252627 Fold 7 :123456789101112131415161718192021222324252627 Fold 8 :123456789101112131415161718192021222324252627 Fold 9 :123456789101112131415161718192021222324252627 Fold 10 :123456789101112131415161718192021222324252627

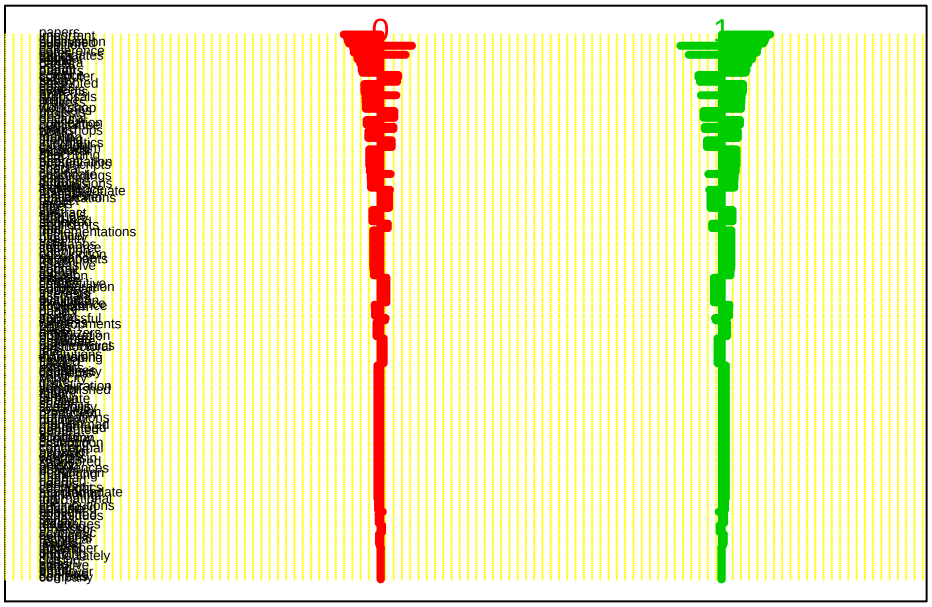




Analysis table :

Threshold	Non.zerofeatures	Likelyhood	Error
0.0	3428	-3.09	0.30
0.1	3162	-2.47	0.23
0.2	3024	-2.00	0.20
0.3	2996	-1.59	0.18
0.4	2920	-1.26	0.18
0.5	1829	-0.98	0.23
0.6	826	-0.68	0.16
0.7	615	-0.58	0.14
0.8	550	-0.52	0.14
0.9	243	-0.47	0.14
1.0	213	-0.44	0.16
1.1	128	-0.42	0.16
1.2	105	-0.42	0.16
1.3	70	-0.43	0.16
1.4	59	-0.45	0.16
1.5	41	-0.48	0.16
1.6	28	-0.51	0.18
1.7	18	-0.53	0.20
1.8	14	-0.56	0.25
1.9	12	-0.59	0.27
2.0	10	-0.62	0.36
2.1	7	-0.65	0.39
2.2	6	-0.66	0.48
2.3	6	-0.67	0.43
2.4	2	-0.68	0.43
2.5	1	-0.68	0.43
2.6	0	-0.68	0.43

Maximum Likelyhood 13Minumum Error 8



null device 1 id name 0-score 1-score [1,] 3036 papers -0.2151 0.2831 [2,] 2049 important -0.1986 0.2613 [3,] 1262 due -0.1941 0.2554 [4,] 4060 submission -0.1906 0.2508 [5,] 3364 published -0.1848 0.2432 [6,] 3187 position 0.1845 -0.2428 [7,] 596 call -0.1601 0.2107 [8,] 869 conference -0.1568 0.2064 [9,] 1045 dates -0.1568 0.2064 [10,] 607 candidates 0.1474 -0.1939 [11,] 4282 topics -0.1363 0.1793 [12,] 2990 original -0.1345 0.177

[13,] 599 camera -0.1186 0.156

[14,] 3433 ready -0.1186 0.156

[15,] 389 authors -0.1099 0.1446 [16,] 2588 march -0.1099 0.1446 [17,] 3022 pages -0.1099 0.1446 [18,] 3035 paper -0.1055 0.1389 [19,] 850 computer 0.1042 -0.1371 [20,] 3725 science 0.1042 -0.1371 [21,] 3671 salary 0.0975 -0.1284 [22,] 4177 team 0.0975 -0.1284 [23,] 681 chairs -0.095 0.125

[24,] 3243 presented -0.095 0.125

[25,] 1891 held -0.095 0.125

[26,] 329 aspects -0.095 0.125

[27,] 4129 systems -0.093 0.1224 [28,] 3125 phd 0.093 -0.1224 [29,] 3324 proposals -0.0888 0.1169 [30,] 2463 limited -0.0874 0.115

[31,] 2974 org -0.0874 0.115

[32,] 3286 process -0.086 0.1132 [33,] 283 april -0.086 0.1132 [34,] 4628 workshop -0.086 0.1132 [35,] 3188 positions 0.0818 -0.1077 [36,] 1233 doctoral 0.0818 -0.1077 [37,] 3191 post 0.0818 -0.1077 [38,] 3312 projects 0.0818 -0.1077 [39,] 810 committee -0.0814 0.107

[40,] 2889 notification -0.0814 0.107

[41,] 3274 privacy -0.0814 0.107

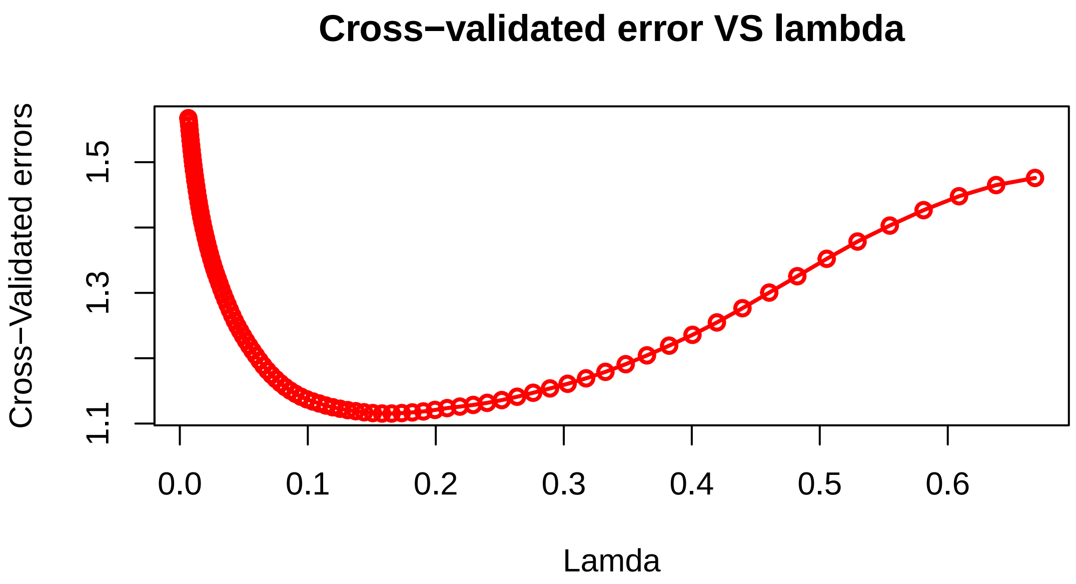
[42,] 3458 record 0.0762 -0.1002 [43,] 3891 skills 0.0762 -0.1002 [44,] 1643 forum -0.0717 0.0943 [45,] 3090 peer -0.0717 0.0943 [46,] 4629 workshops -0.0717 0.0943 [47,] 2561 making -0.0717 0.0943 [48,] 1477 excellent 0.0663 -0.0872 [49,] 2103 informatics 0.0663 -0.0872 [50,] 3992 starting 0.0663 -0.0872 [51,] 1007 curriculum 0.0663 -0.0872 [52,] 4500 versions -0.0662 0.0871 [53,] 92 acm -0.0662 0.0871 [54,] 1127 describing -0.0662 0.0871 [55,] 1698 fully -0.0662 0.0871 [56,] 3323 proposal -0.0662 0.0871 [57,] 2583 manuscripts -0.0662 0.0871 [58,] 3241 presentation -0.0662 0.0871 [59,] 4364 tutorial -0.0662 0.0871 [60,] 3836 short -0.0625 0.0822 [61,] 3952 special -0.0625 0.0822 [62,] 680 chair -0.0625 0.0822 [63,] 606 candidate 0.0602 -0.0792 [64,] 2305 june -0.0577 0.0759 [65,] 3285 proceedings -0.0577 0.0759 [66,] 1061 deadline -0.0561 0.0739 [67,] 2295 journal -0.0558 0.0735 [68,] 4061 submissions -0.0558 0.0735 [69,] 1501 experience 0.0558 -0.0734 [70,] 2058 include -0.0558 0.0734 [71,] 2438 letter 0.0508 -0.0669 [72,] 2442 levels 0.0508 -0.0669 [73,] 3311 project 0.0508 -0.0669 [74,] 3383 qualifications 0.0508 -0.0669 [75,] 3559 researcher 0.0508 -0.0669 [76,] 4176 teaching 0.0508 -0.0669 [77,] 4402 undergraduate 0.0508 -0.0669 [78,] 740 city 0.0508 -0.0669 [79,] 63 abstract -0.0481 0.0633 [80,] 3882 site -0.0481 0.0633 [81,] 4365 tutorials -0.0481 0.0633 [82,] 1563 february -0.0481 0.0633 [83,] 1594 final -0.0481 0.0633 [84,] 3589 reviewed -0.0481 0.0633 [85,] 4039 strong 0.0432 -0.0569 [86,] 267 applicants 0.0432 -0.0568 [87,] 2553 mail 0.0432 -0.0568 [88,] 2046 implementations -0.0429 0.0565 [89,] 2877 non -0.0429 0.0565 [90,] 501 bio -0.0429 0.0565 [91,] 2690 michael -0.0429 0.0565 [92,] 363 attention -0.0429 0.0565 [93,] 803 commerce -0.0429 0.0565 [94,] 2082 india -0.0429 0.0565 [95,] 3711 scenarios -0.0429 0.0565 [96,] 4342 trust -0.0429 0.0565 [97,] 4452 usability -0.0429 0.0565 [98,] 879 conjunction -0.0429 0.0565 [99,] 4449 url -0.0429 0.0565 [100,] 2433 length -0.0429 0.0565 [101,] 3051 participants -0.0429 0.0565 [102,] 3514 relevance -0.0429 0.0565 [103,] 3118 pervasive -0.0429 0.0565 [104,] 4451 usa -0.0429 0.0565 [105,] 4062 submit -0.0422 0.0555 [106,] 386 author -0.0389 0.0511 [107,] 1636 format -0.0389 0.0511 [108,] 1072 decision -0.0389 0.0511 [109,] 776 collaboration 0.0352 -0.0463 [110,] 831 competitive 0.0352 -0.0463 [111,] 1088 degree 0.0352 -0.0463 [112,] 2613 master 0.0352 -0.0463 [113,] 4529 vitae 0.0352 -0.0463 [114,] 1450 european 0.0352 -0.0463 [115,] 2837 needs 0.0352 -0.0463 [116,] 107 activities 0.0352 -0.0463 [117,] 336 assistant 0.0352 -0.0463 [118,] 2170 interests 0.0352 -0.0463 [119,] 1456 evaluation 0.0352 -0.0463 [120,] 1542 faculty 0.0352 -0.0463 [121,] 2150 intelligence -0.0342 0.045

[122,] 76 acceptance -0.0342 0.045

[123,] 3301 program -0.0326 0.0429 [124,] 2198 invited -0.0324 0.0426 [125,] 3021 page -0.0324 0.0426 [126,] 3386 quality -0.0324 0.0426 [127,] 4075 successful 0.0305 -0.0402 [128,] 272 apply 0.0267 -0.0351 [129,] 1149 developments -0.0238 0.0313 [130,] 4605 wireless -0.0238 0.0313 [131,] 2887 notes -0.0238 0.0313 [132,] 172 aims -0.0238 0.0313 [133,] 2219 issue -0.0238 0.0313 [134,] 2964 optimization -0.0238 0.0313 [135,] 2984 organizers -0.0238 0.0313 [136,] 340 associate 0.0191 -0.0252 [137,] 579 building 0.0191 -0.0252 [138,] 1797 graduate 0.0191 -0.0252 [139,] 2619 mathematics 0.0191 -0.0252 [140,] 1147 developing 0.0191 -0.0252 [141,] 1524 extension 0.0191 -0.0252 [142,] 2141 institutions 0.0191 -0.0252 [143,] 2251 java 0.0191 -0.0252 [144,]

2278 job 0.0191 -0.0252 [145,] 3194 postdoctoral 0.0191 -0.0252 [146,] 1591 filled 0.0191 -0.0252 [147,] 1702 funded 0.0191 -0.0252 [148,] 837 complexity -0.0183 0.0241 [149,] 1062 deadlines -0.0183 0.0241 [150,] 1490 exhibits -0.0183 0.0241 [151,] 2359 korea -0.0183 0.0241 [152,] 856 concepts -0.0183 0.0241 [153,] 1818 grid -0.0183 0.0241 [154,] 2197 invite -0.0183 0.0241 [155,] 3361 publicity -0.0183 0.0241 [156,] 4664 yang -0.0183 0.0241 [157,] 104 actions -0.0183 0.0241 [158,] 196 allowed -0.0183 0.0241 [159,] 455 behavior -0.0183 0.0241 [160,] 857 conceptual -0.0183 0.0241 [161,] 920 control -0.0183 0.0241 [162,] 967 covering -0.0183 0.0241 [163,] 1214 distribution -0.0183 0.0241 [164,] 1292 economy -0.0183 0.0241 [165,] 1343 elicitation -0.0183 0.0241 [166,] 1560 feature -0.0183 0.0241 [167,] 1721 game -0.0183 0.0241 [168,] 1745 generated -0.0183 0.0241 [169,] 1833 guaranteed -0.0183 0.0241 [170,] 2623 matter -0.0183 0.0241 [171,] 2746 mohammad -0.0183 0.0241 [172,] 2754 monday -0.0183 0.0241 [173,] 2888 notice -0.0183 0.0241 [174,] 2890 notifications -0.0183 0.0241 [175,] 3224 prediction -0.0183 0.0241 [176,] 3289 production -0.0183 0.0241 [177,] 3570 resource -0.0183 0.0241 [178,] 3703 scalability -0.0183 0.0241 [179,] 3802 sessions -0.0183 0.0241 [180,] 3943 spain -0.0183 0.0241 [181,] 3948 spatial -0.0183 0.0241 [182,] 4202 template -0.0183 0.0241 [183,] 4212 term -0.0183 0.0241 [184,] 4236 theory -0.0183 0.0241 [185,] 4363 tutor -0.0183 0.0241 [186,] 4435 unpublished -0.0183 0.0241 [187,] 4499 version -0.0183 0.0241 [188,] 4526 visualization -0.0183 0.0241 [189,] 84 accommodate -0.0183 0.0241 [190,] 940 copyright -0.0183 0.0241 [191,] 1291 economics -0.0183 0.0241 [192,] 1587 figures -0.0183 0.0241 [193,] 1859 hand -0.0183 0.0241 [194,] 1861 handled -0.0183 0.0241 [195,] 1965 huge -0.0183 0.0241 [196,] 2598 marketing -0.0183 0.0241 [197,] 2839 negotiation -0.0183 0.0241 [198,] 3169 policy -0.0183 0.0241 [199,] 3231 preferences -0.0183 0.0241 [200,] 3259 pricing -0.0183 0.0241 [201,] 3764 select -0.0183 0.0241 [202,] 3966 sponsored -0.0183 0.0241 [203,] 4490 venues -0.0183 0.0241 [204,] 4606 wisconsin -0.0183 0.0241 [205,] 2574 managing -0.0183 0.0241 [206,] 2757 monitoring -0.0183 0.0241 [207,] 2175 international -0.0181 0.0239 [208,] 919 contributions -0.0148 0.0195 [209,] 4268 title -0.0148 0.0195 [210,] 4281 topic -0.0148 0.0195 [211,] 134 advanced -0.0148 0.0195 [212,] 3957 specific -0.0148 0.0195 [213,] 3515 relevant 0.0135 -0.0178 [214,] 4064 submitted -0.0111 0.0147 [215,] 3794 series -0.0106 0.0139 [216,] 4185 techniques -0.0106 0.0139 [217,] 3588 review -0.0106 0.0139 [218,] 2005 ideas -0.0106 0.0139 [219,] 1144 develop 0.0098 -0.013 [220,] 2392 languages 0.0098 -0.013 [221,] 3295 professor 0.0098 -0.013 [222,] 67 academic 0.0096 -0.0126 [223,] 3800 services -0.0094 0.0124 [224,] 2847 networks -0.0091 0.012
[225,] 3582 results -0.0091 0.012
[226,] 4181 technical -0.0091 0.012
[227,] 2220 issues -0.0091 0.012
[228,] 2167 interest -0.0052 0.0068 [229,] 2894 november 0.0032 -0.0042 [230,] 2510 looking 0.0024 -0.0031 [231,] 115 adaptive 0.0024 -0.0031 [232,] 745 class 0.0024 -0.0031 [233,] 756 closing 0.0024 -0.0031 [234,] 1230 doc 0.0024 -0.0031 [235,] 2040 immediately 0.0024 -0.0031 [236,] 2945 ongoing 0.0024 -0.0031 [237,] 3991 start 0.0024 -0.0031 [238,] 1913 hiring 0.0024 -0.0031 [239,] 274 applying 0.0024 -0.0031 [240,] 1372 employer 0.0024 -0.0031 [241,] 1424 equal 0.0024 -0.0031 [242,] 1089 degrees 0.0024 -0.0031 [243,] 821 company 0.0024 -0.0031 The number of features selected 243The features selected papers important due submission published position call conference dates candidates

2.2 a Elastic net



The number of features selected 32 The best lambda after cross validation is : 0.1655087 Misclassification rate on Test data : 0.05

2.2b SVM with "Vallidot" kernel.

Setting default kernel parameters
Features selected : 43 Misclassification Error on Test data : 0.05

Comparing the elastic net and the SVM model with the nearest shrunken centroids, we see that the misclassification rates for all the three models are low here they are equal to 0.05.The nearest centroid selects 243 features while the elastic net selects 35 features and the SVM slects 43 features. So an elastic net is the best option as it has minimum number of features selected.

2.3 Benjamini-Hochberg method

	p_values	adj_p_values
papers	0.0000000	0.0000005
submission	0.0000000	0.0000019
position	0.0000000	0.0000129
published	0.0000002	0.0002157
important	0.0000003	0.0002860
call	0.0000004	0.0003122
conference	0.0000005	0.0003420
candidates	0.0000009	0.0005062
dates	0.0000014	0.0006576
paper	0.0000014	0.0006576
topics	0.0000051	0.0021665
limited	0.0000079	0.0030986

	p_values	adj_p_values
candidate	0.0000119	0.0043063
authors	0.0000215	0.0063314
camera	0.0000210	0.0063314
ready	0.0000210	0.0063314
phd	0.0000338	0.0091405
projects	0.0000350	0.0091405
org	0.0000374	0.0092605
chairs	0.0000586	0.0137773
due	0.0000649	0.0138683
original	0.0000649	0.0138683
notification	0.0000688	0.0140696
salary	0.0000797	0.0156184
record	0.0000909	0.0164390
skills	0.0000909	0.0164390
held	0.0001529	0.0266303
team	0.0001758	0.0295146
apply	0.0002166	0.0308409
committee	0.0002117	0.0308409
international	0.0002296	0.0308409
pages	0.0002007	0.0308409
proceedings	0.0002117	0.0308409
strong	0.0002246	0.0308409
workshop	0.0002007	0.0308409
degree	0.0003762	0.0453942
excellent	0.0003762	0.0453942
post	0.0003762	0.0453942
presented	0.0003765	0.0453942

The above displayed features correspod to the rejected hypothesis. This implies that these features are significant. From the result, it is observed that the Benjamini-Hochberg method reduces the errors due to the FPR by increasing the p-value of the falsely identified significant variable.

Appendix

```r
library(mboost)
library(randomForest)
RNGversion("3.5.1")
#reading the csv
spam_data<-read.csv2(file="D:/Perriod 2/Machine Learning/Lab/Block 2 lab1/spambase.csv",sep=";")
spam_data$Spam <- as.factor(spam_data$Spam)
#dividing into train and test
n<-nrow(spam_data)
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
id<-sample(1:n, floor(n*1/3))
spam_test<-spam_data[id,]
spam_train<-spam_data[-id,]
#building the model with adaboost and random forest for 10,20...,100
i<-10
```

```r
j<-1
ada_train_error<-c()
ada_test_error<-c()
rf_train_error<-c()
rf_test_error<-c()
while(i<=100){
  ####################################
  #adaboost model
  ada_boost_obj<-blackboost(formula = Spam~.,
                            data =spam_train,
                            family = AdaExp(),
                            control=boost_control(mstop=i))
  #predicting the values for both test and train with the ada_boost_obj
  ada_spam_train<-predict(ada_boost_obj,spam_train,type = "class")
  ada_spam_test<-predict(ada_boost_obj,spam_test,type = "class")
  #confusion matrix for train
  df1<-data.frame("True Values"=spam_train$Spam,
                  "Predicted Values"=ada_spam_train)
  ada_cm_train<-table(df1)
  #misclassification rate for train
  ada_train_error[j]=(1-sum(diag(ada_cm_train))/sum(ada_cm_train))*100
  #confusion matrix for test
  df2<-data.frame("True Values"=spam_test$Spam,
                  "Predicted Values"=ada_spam_test)
  ada_cm_test<-table(df2)
  #misclassification rate for test
  ada_test_error[j]=(1-(sum(diag(ada_cm_test))/sum(ada_cm_test)))*100

  ###############################################
  #random forest model
  rf_obj<-randomForest(formula = Spam~.,data =spam_train,ntree=j)
  #fitting the data for both train and test data
  rf_spam_train<-predict(rf_obj,spam_train,type="class")
  rf_spam_test<-predict(rf_obj,spam_test,type="class")
  #confusion matrix for train data
  df3<-data.frame("True Values"=spam_train$Spam,
                  "Predicted Values"=rf_spam_train)
  rf_cm_train<-table(df3)
  #misclassification for train
  rf_train_error[j]<-(1-sum(diag(rf_cm_train))/sum(rf_cm_train))*100
  #confusion matrix for test data
  df4<-data.frame("True Values"=spam_test$Spam,
                  "Predicted Values"=rf_spam_test)
  rf_cm_test<-table(df4)
  #misclassification for test
  rf_test_error[j]<-(1-sum(diag(rf_cm_test))/sum(rf_cm_test))*100
  j<-j+1
  i<-i+10
}
#ploting the error values
#ada boost
max_ada<-rep(NA,times=length(ada_test_error))
max_ada[which.min(ada_test_error)]<-
```

```r
  round(ada_test_error[which.min(ada_test_error)],2)
plot(x=seq(10,100,length.out = 10),y=ada_test_error,
     xlab = "Number of trees",
     ylab = "Percentage error",
     type="o",main = "Ada Boost Test error graph")
text(x=seq(10,100,length.out = 10),y=ada_test_error,
     labels = max_ada[max_ada != "NA"],
     pos = 3,
     cex = 0.7,
     col = 2)
#random forest
max_rf<-rep(NA,times=length(rf_test_error))
max_rf[which.min(rf_test_error)]<-
  round(rf_test_error[which.min(rf_test_error)],2)
plot(x=seq(10,100,length.out = 10),rf_test_error,
     xlab = "Number of trees",
     ylab = "Percentage error",
     type="o",main = "Random Forest Test error graph")
text(x=seq(10,100,length.out = 10),y=rf_test_error,
     labels = max_rf[max_rf != "NA"],
     pos = 3,
     cex = 0.7,
     col = 2)
RNGversion("3.5.1")
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)# true mixing coefficients
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=2 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
```

```r
for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}
knitr::kable(pi)
knitr::kable(mu)
for(it in 1:max_it) {
  Sys.sleep(0.5)
  # E-step: Computation of the fractional component for each data point
  for (n in 1:N){
    for(k in 1:K){
      lik <- sapply(seq(1:D), function(d) {
        dbinom(x[n, d], 1, mu[k, d])
      })
      z[n, k] <- prod(lik) * pi[k]
    }
    z[n, ] <- z[n, ] / sum(z[n, ])
  }

  #Log likelihood computation.
  for(n in 1:N) {
    for(k in 1:K) {
      llik[it] <- llik[it] +
        z[n, k] * (log(pi[k]) +
                     sum((x[n, ] * log(mu[k, ])) +
                           ((1 - x[n, ]) * log(1 - mu[k, ]))))
    }
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the lok likelihood has not changed significantly
  if (abs(llik[it - 1] - llik[it]) < min_change&&it!=1)
    break
  #M-step: ML parameter estimation from the data and fractional component assignments
  for(k in 1:K) {
    pi[k] <- sum(z[, k]) / N
    x_scaled <- x[, 1:D] * z[, k]
    mu[k, ] <- sapply(seq(1:D), function(d) {
      sum(x_scaled[, d]) / sum(z[, k])
    })
  }
}
plot(mu[1,], type="o", col="blue", ylim=c(0,1))
points(mu[2,], type="o", col="red")
knitr::kable(true_mu)
knitr::kable(mu)
plot(llik[1:it], type="o")
RNGversion("3.5.1")
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
```

```
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)# true mixing coefficients
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}
knitr::kable(pi)
knitr::kable(mu)
for(it in 1:max_it) {
  Sys.sleep(0.5)
  # E-step: Computation of the fractional component for each data point
  for (n in 1:N){
    for(k in 1:K){
      lik <- sapply(seq(1:D), function(d) {
        dbinom(x[n, d], 1, mu[k, d])
      })
      z[n, k] <- prod(lik) * pi[k]
    }
    z[n, ] <- z[n, ] / sum(z[n, ])
  }

  #Log likelihood computation.
  for(n in 1:N) {
    for(k in 1:K) {
      llik[it] <- llik[it] +
        z[n, k] * (log(pi[k]) +
        sum((x[n, ] * log(mu[k, ])) +
        ((1 - x[n, ]) * log(1 - mu[k, ]))))
    }
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the lok likelihood has not changed significantly
  if (abs(llik[it - 1] - llik[it]) < min_change&&it!=1)
    break
  #M-step: ML parameter estimation from the data and fractional component assignments
  for(k in 1:K) {
    pi[k] <- sum(z[, k]) / N
    x_scaled <- x[, 1:D] * z[, k]
    mu[k, ] <- sapply(seq(1:D), function(d) {
      sum(x_scaled[, d]) / sum(z[, k])
    })
  }
}
plot(mu[1,], type="o", col="blue", ylim=c(0,1))
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
knitr::kable(true_mu)
knitr::kable(mu)
plot(llik[1:it], type="o")
RNGversion("3.5.1")
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)# true mixing coefficients
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=4 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}
knitr::kable(pi)
knitr::kable(mu)
for(it in 1:max_it) {
  Sys.sleep(0.5)
  # E-step: Computation of the fractional component for each data point
  for (n in 1:N){
    for(k in 1:K){
      lik <- sapply(seq(1:D), function(d) {
        dbinom(x[n, d], 1, mu[k, d])
      })
      z[n, k] <- prod(lik) * pi[k]
    }
    z[n, ] <- z[n, ] / sum(z[n, ])
  }

  #Log likelihood computation.
  for(n in 1:N) {
    for(k in 1:K) {
      llik[it] <- llik[it] +
        z[n, k] * (log(pi[k]) +
        sum((x[n, ] * log(mu[k, ])) +
        ((1 - x[n, ]) * log(1 - mu[k, ]))))
    }
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the lok likelihood has not changed significantly
  if (abs(llik[it - 1] - llik[it]) < min_change&&it!=1)
    break
  #M-step: ML parameter estimation from the data and fractional component assignments
  for(k in 1:K) {
    pi[k] <- sum(z[, k]) / N
    x_scaled <- x[, 1:D] * z[, k]
    mu[k, ] <- sapply(seq(1:D), function(d) {
      sum(x_scaled[, d]) / sum(z[, k])
    })
  }
}
plot(mu[1,], type="o", col="blue", ylim=c(0,1))
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
points(mu[4,], type="o", col="yellow")
knitr::kable(true_mu)
knitr::kable(mu)
plot(llik[1:it], type="o")
influenza_data<-xlsx::read.xlsx("D:/Perriod 2/Machine Learning/Lab/Block 2 lab2/Influenza.xlsx",sheetNa
plot(influenza_data$Time,influenza_data$Mortality,xlab="Time",
     ylab="Mortality")
plot(influenza_data$Time,influenza_data$Influenza,xlab="Time",
     ylab="Influenza")
library(mgcv)
gam_model<-gam(Mortality~Year+s(Week,k=length(unique(influenza_data$Week))),
           data = influenza_data)
plot(gam_model,ylab="Spline")
# 1.3 plotting fitted VS original data
library(ggplot2)
```

```r
plot_df<-cbind(influenza_data,"fitted"=gam_model$fitted.values)
ggplot(plot_df) +
geom_line(aes(x = Time, y = Mortality, colour = "#282E23")) +
geom_line(aes(x = Time, y = fitted, colour = "#237DBA")) +
  theme_bw()+
scale_color_manual(labels = c("Original Data", "Fitted Data"),
values = c("#282E23", "#237DBA")) +
ylab("Mortality")
summary(gam_model)
# 1.4 Checking the gam model for differnt penalty factor
#Very small penalty factor
gam_modelp1<-gam(Mortality~Year+
                   s(Week,k=length(unique(influenza_data$Week)),sp=1e-5),
                 data = influenza_data)
#plotting the fitted VS actual data
plot_df1<-cbind(influenza_data,"fitted"=gam_modelp1$fitted.values)
ggplot(plot_df1) +
geom_line(aes(x = Time, y = Mortality, colour = "red")) +
geom_line(aes(x = Time, y = fitted, colour = "blue")) +
  theme_bw()+
scale_color_manual(labels = c("Original Data", "Fitted Data"),
values = c("red", "blue")) +
  ylab("Mortality")
#Very high penalty
gam_modelp2<-gam(Mortality~Year+
                   s(Week,k=length(unique(influenza_data$Week)),sp=1e+5),
                 data = influenza_data)
#plotting the fitted VS actual data
plot_df2<-cbind(influenza_data,"fitted"=gam_modelp2$fitted.values)
ggplot(plot_df2) +
geom_line(aes(x = Time, y = Mortality, colour = "red")) +
geom_line(aes(x = Time, y = fitted, colour = "blue")) +
  theme_bw()+
scale_color_manual(labels = c("Original Data", "Fitted Data"),
values = c("red", "blue")) +
  ylab("Mortality")
#Degrees of freedom and deviance variation with penalty factor
is<-cumprod(c(1e-5,rep(10,10)))
devi<-numeric()
df<-numeric()
j<-1
for(i in is){
  g<-gam(Mortality~Year+
           s(Week,k=length(unique(influenza_data$Week)),sp=i),
         data = influenza_data)
  devi[j]<-g$deviance
  df[j]<-sum(g$edf)
  j<-j+1
}
plot(x=is,
     y=devi,
     xlab="Penality Factors",
     ylab="Deviance",
     main="Penality VS Deviance",
     col="blue",lwd="2",type="o")
plot(x=is,
     y=df,
     xlab="Penality Factors",
     ylab="Degrees of freedom",
     main="Penality VS Degrees of freedom",
     col="red",lwd="2",type="o")
results <- data.frame(
GCV = c(gam_modelp1$gcv.ubre.dev, gam_modelp2$gcv.ubre.dev),
Total.df = c(sum(gam_modelp1$edf1), sum(gam_modelp2$edf1)),
Deviance = c(deviance(gam_modelp1), deviance(gam_modelp2)),
row.names = c("Low Penalty GAM", "high Penalty GAM"))
knitr::kable(results)
plot(x=influenza_data$Time,
     y=influenza_data$Influenza,
     xlab="Time",
     ylab="Fitted values and residuals",
     main="Fitted values and residuals",
     col="blue",lwd="2",type="l",ylim=c(-200,400))
points(x=influenza_data$Time,
       y=gam_model$residuals,
       col="red",lwd="1",type="l")
legend("topright",c("Influenza","Residuals"),fill=c("blue","red"))
# 1.6 splines of year, week, and the number of confirmed cases of influenza
new_gam_model<-gam(Mortality~s(Year,k=length(unique(influenza_data$Year)))+
                     s(Week,k=length(unique(influenza_data$Week)))+
                 s(Influenza,k=length(unique(influenza_data$Influenza))),
                   data=influenza_data)
# Plotted the fitted VS original data
plot(x=influenza_data$Time,
     y=new_gam_model$fitted.values,
     xlab="Time",
     ylab="Mortality",
     main="Fitted and Original data VS Time",
     col="blue",lwd="1",type="l")
points(x=influenza_data$Time,
       y=influenza_data$Mortality,
       col="red",lwd="1",type="l")
legend("topleft",c("Original Data","Fitted Values"),fill=c("blue","red"))
summary(new_gam_model)
results <- data.frame(
GCV = c(new_gam_model$gcv.ubre.dev, gam_model$gcv.ubre.dev),
Total.df = c(sum(new_gam_model$edf1), sum(gam_model$edf1)),
Deviance = c(deviance(new_gam_model), deviance(gam_model)),
row.names = c("GAM with 3 splines", "GAM with 1 spline"))
knitr::kable(results)
#Reading the csv file
mail_data<-read.csv2(file="D:/Perriod 2/Machine Learning/Lab/Block 2 lab2/data.csv",sep=";")
#dividing into train and test
n<-nrow(mail_data)
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
```

```r
id<-sample(1:n, floor(n*0.7))
mail_testt<-mail_data[-id,]
mail_trainn<-mail_data[id,]
#separating dependent and independent variable
library(pamr)
mail_train<-scale(mail_trainn)
mail_train[is.nan(mail_train)]<-0
mail_train<-as.data.frame(mail_train)
mail_train$Conference<-as.factor(mail_trainn$Conference)
x<-t(mail_train[,-4703])
y<-mail_train[[4703]]
mylist<-list(x=x,y=as.factor(y),
             geneid=as.character(1:nrow(x)),
             genenames=rownames(x))
pamr_model<-pamr.train(mylist,threshold = seq(0,5,0.1))
cv_model<-pamr.cv(pamr_model,mylist)
plot(x=cv_model$threshold,
     y=cv_model$loglik,
     xlab="Thresholds",
     ylab="LogLIKElyhood",
     main="LogLikelyhood VS Thresholds",
     col="blue",lwd="2",type="o")
plot(x=cv_model$threshold,
     y=cv_model$error,
     xlab="Thresholds",
     ylab="Error",
     main="Errors VS Threshold",
     col="red",lwd="2",type="o")
analysis<-data.frame("Threshold"=cv_model$threshold,
                     "Non-zerofeatures"=cv_model$size,
                     "Likelyhood"=round(cv_model$loglik,2),
                     "Error"=round(cv_model$error,2))
cat("Analysis table : ","\n")
knitr::kable(analysis)
#pamr.plotcv(cv_model)
cat("Maximum Likelyhood",which.max(cv_model$loglik))
cat("Minumum Error",which.min(cv_model$error))
# Centroid plot
pamr.plotcen(pamr_model,mylist,threshold=0.9)
dev.off()
#listing the 10 most contributing features
para_list<-pamr.listgenes(pamr_model,mylist,threshold=0.9,genenames=TRUE,fitcv = NULL)
cat("The number of features selected",paste(as.numeric(nrow(para_list)),collapse='\n'))
cat("The features selected","\n")
cat(paste(colnames(mail_trainn)[as.numeric(para_list[1:10,1])],collapse='\n'))
library(glmnet)
#finding best lambda using cross validation
cv_elastic_model<-cv.glmnet(x=as.matrix(mail_trainn[,-4703]),
                            y=as.factor(mail_trainn$Conference),
                            alpha=0.5,
                            family="binomial",
                            type.measure="deviance")
plot(x=cv_elastic_model$lambda,
     y=cv_elastic_model$cvm,
     xlab="Lamda",
     ylab="Cross-Validated errors",
     main="Cross-validated error VS lambda",
     type="o",
     col="red",
     lwd="2")
#best lambda with minimum CVM
best_lambda<-cv_elastic_model$lambda[which.min(cv_elastic_model$cvm)]
cat("The number of features selected",
    cv_elastic_model$nzero[which.min(cv_elastic_model$cvm)],"\n")
cat("The best lambda after cross validation is :",best_lambda,"\n")
#training the elastic model with best lambda
best_elastic_model<-glmnet(x=as.matrix(mail_trainn[,-4703]),
                           y=as.factor(mail_trainn$Conference),
                           alpha=0.5,
                           family="binomial",
                           lambda=best_lambda)
#reporting the test error
y_hut_elastic<-predict(best_elastic_model,
                       newx=as.matrix(mail_testt[,-4703]))
#cofusion matrix and misclassification rate
cm_elastic<-table(mail_testt$Conference,y_hut_elastic,
                  dnn = c("True Values","Predicted Values"))
mc_elastic<-sum(diag(cm_elastic))/sum(cm_elastic)
cat("Misclassification rate on Test data : ",mc_elastic)
# 2.3 Benjamini-Hochberg method
t_test<-lapply(mail_data[,-4703],
               function(x) t.test(x~mail_data[[4703]],data=mail_data,
                      alternative = c("two.sided"),
                      var.equal=FALSE))
#extracting the p values from list
p_values<-sapply(X=t_test,FUN=getElement, name = "p.value")
#adjusting the p value using the BH method
adj_p_values<-p.adjust(p_values,method="BH")
#sort the values in accending order
analyze<-as.data.frame(cbind(p_values,adj_p_values))
analyze<-analyze[order(analyze$adj_p_values),]
analyze_BH<-analyze[analyze$adj_p_values<0.05,]
knitr::kable(analyze_BH)
#########################################
p_values<-p_values[order(p_values)]
adj_p_values<-adj_p_values[order(adj_p_values)]
```