REPORT
Assignment-2
Support Vector Machines

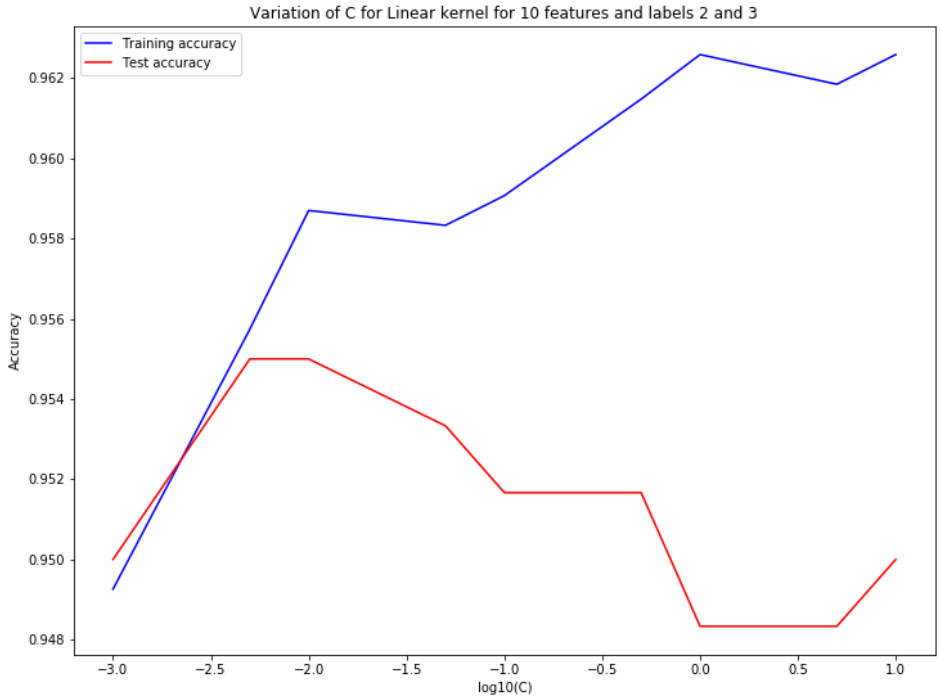
Made by: -
Aman Tiwari (2017EE10436)
Group-2

Case 1 – Binary Classification using LIBSVM

Linear Kernel

Labels Used – 2 and 3

No. of features – 10

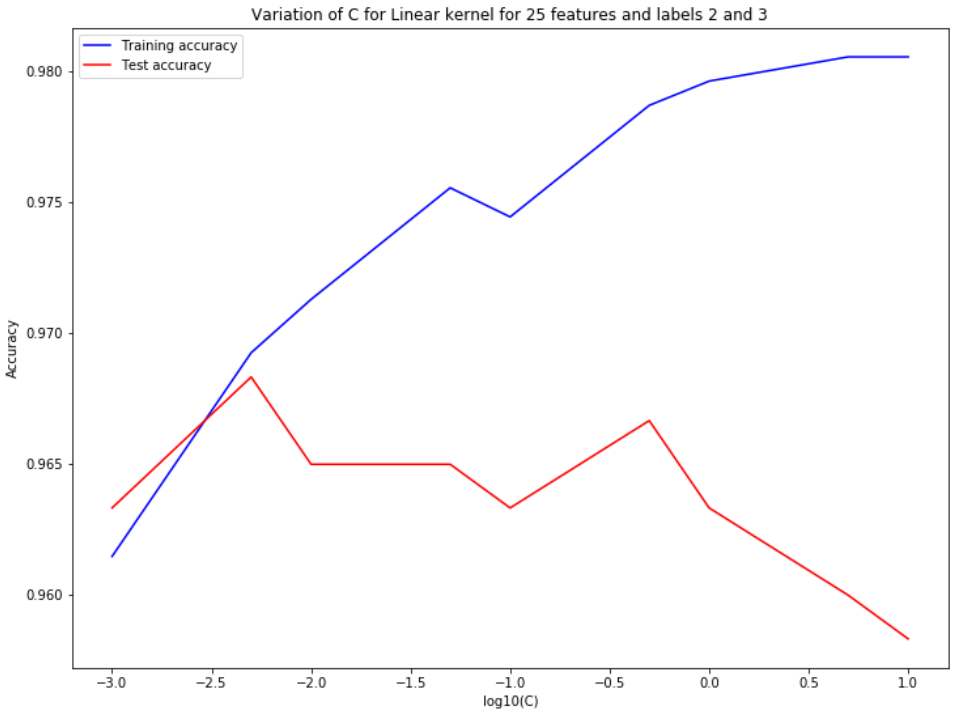


From the above graph, we see that for values of C below 0.005 underfitting occurs (training and test accuracy both increases on increasing C) and for values of C above 0.01 overfitting occurs (training accuracy increases, and test accuracy decreases on increasing C). Thus, the optimal value of C is around 0.01.

At C = 0.01,

Training Accuracy = 95.87% ; Test Accuracy = 95.49%

No. of features – 25



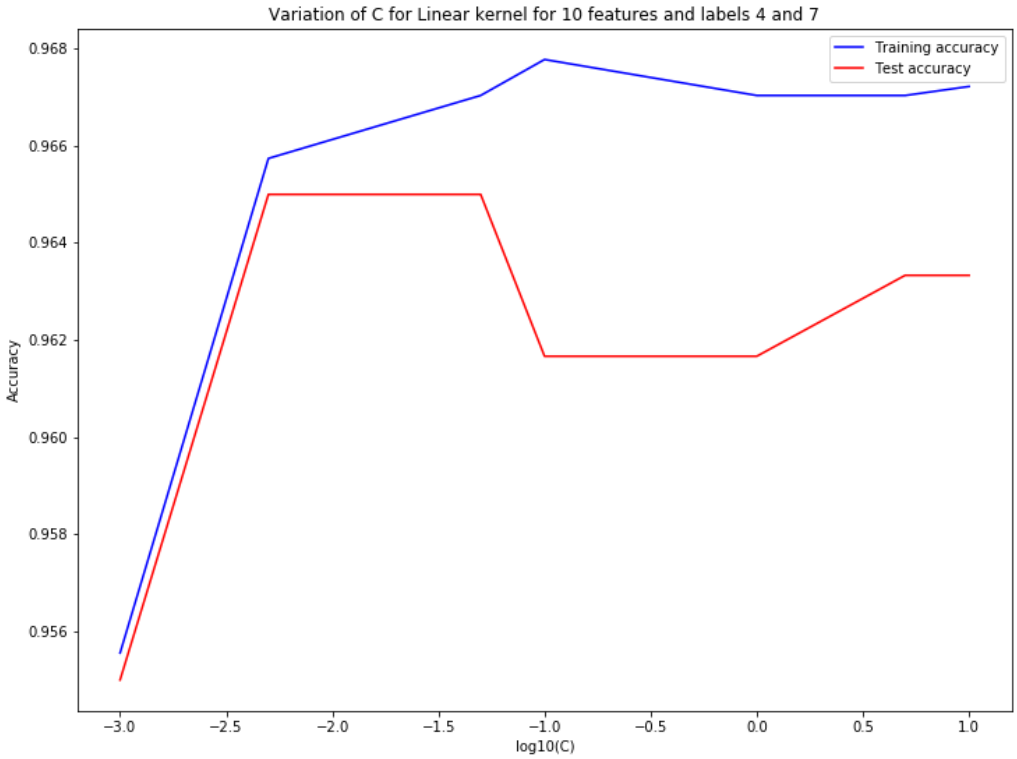
From the above graph, we see that for values of C below 0.005 underfitting occurs (training and test accuracy both increases on increasing C) and for values of C above 0.005 overfitting occurs (training accuracy increases, and test accuracy decreases on increasing C). Thus, the optimal value of C is around 0.005.

At C = 0.005,

Training Accuracy = 96.92% ; Test Accuracy = 96.83%

Labels Used – 4 and 7

No. of features – 10

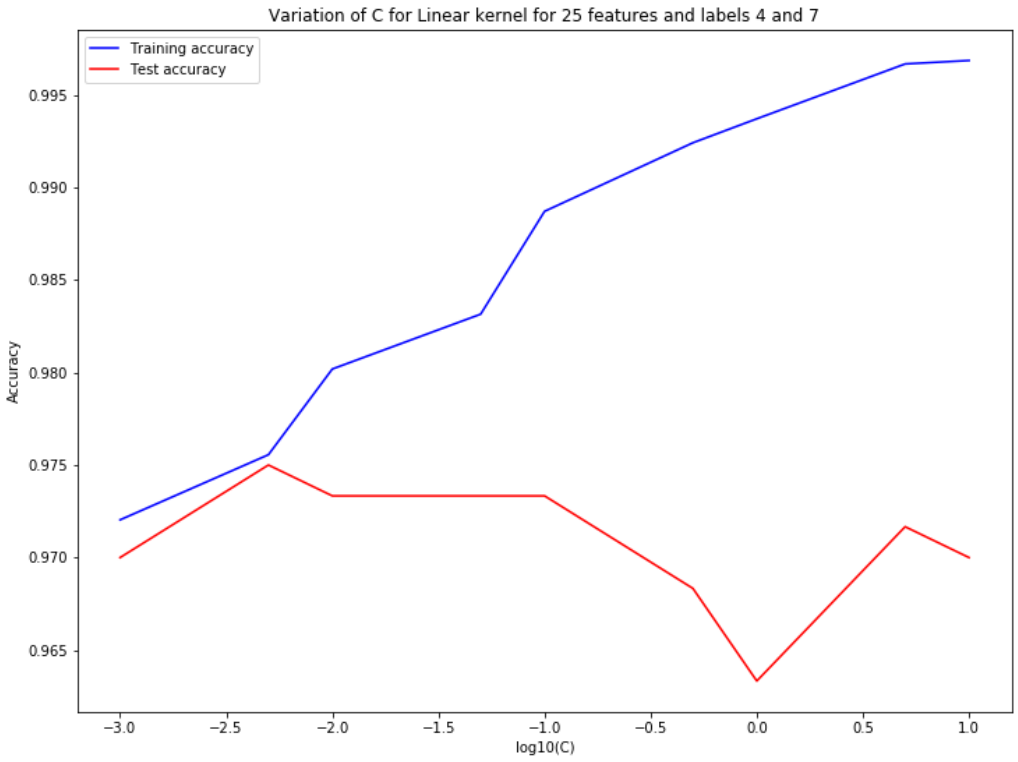


From the above graph, we see that for values of C below 0.005 underfitting occurs (training and test accuracy both increases on increasing C) and for values of C above 0.05 overfitting occurs (training accuracy increases, and test accuracy decreases on increasing C). Thus, the optimal value of C is around 0.05.

At C = 0.05,

Training Accuracy = 96.73% ; Test Accuracy = 96.49%

No. of features – 25



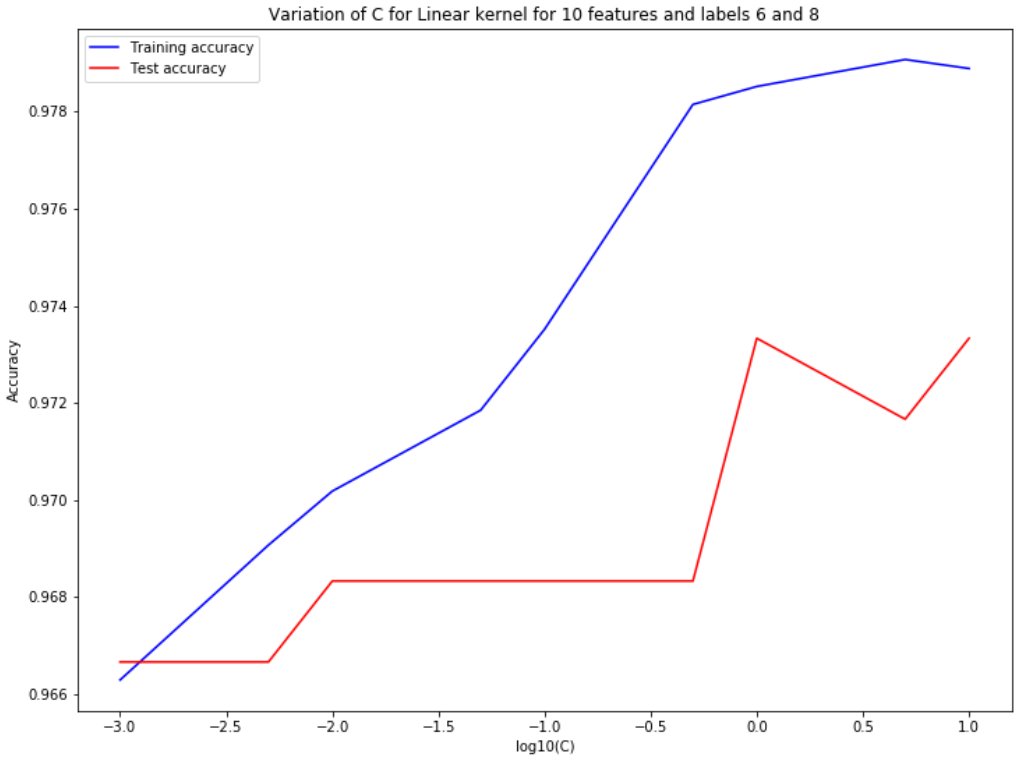
From the above graph, we see that for values of C below 0.005 underfitting occurs (training and test accuracy both increases on increasing C) and for values of C above 0.005 overfitting occurs (training accuracy increases, and test accuracy decreases on increasing C). Thus, the optimal value of C is around 0.005.

At C = 0.005,

Training Accuracy = 97.55% ; Test Accuracy = 97.50%

Labels Used – 6 and 8

No. of features – 10

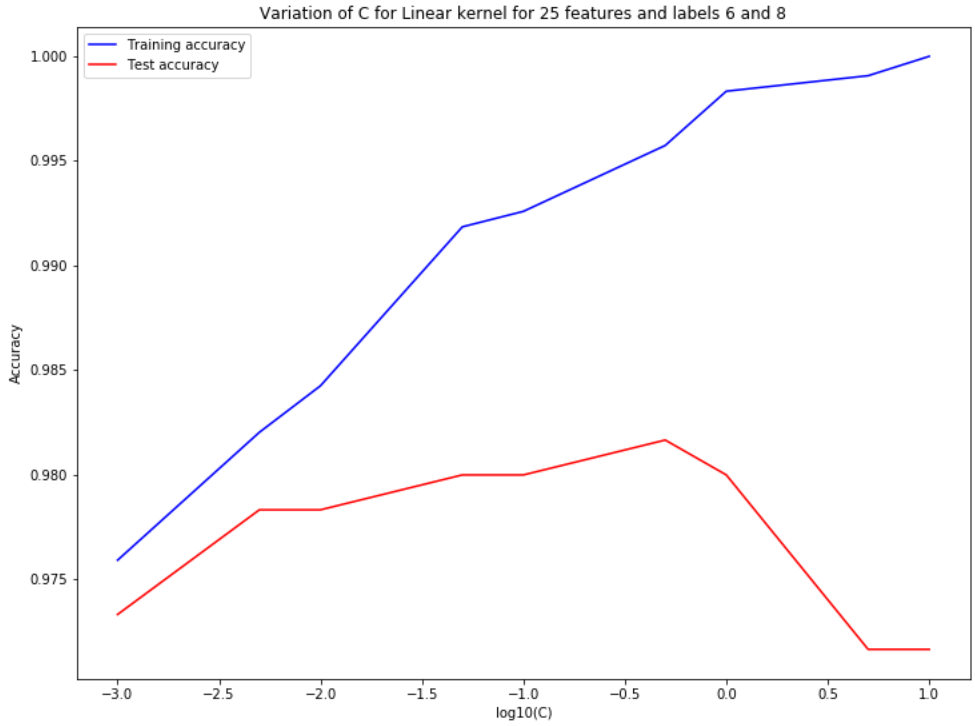


From the above graph, we see that for values of C below 1.0 underfitting occurs (training and test accuracy both increases on increasing C) and for values of C above 1.0 overfitting occurs (training accuracy increases, and test accuracy decreases on increasing C). Thus, the optimal value of C is around 1.0.

At C = 1.0,

Training Accuracy = 97.85% ; Test Accuracy = 97.33%

No. of features – 25



From the above graph, we see that for values of C below 0.5 underfitting occurs (training and test accuracy both increases on increasing C) and for values of C above 0.5 overfitting occurs (training accuracy increases, and test accuracy decreases on increasing C). Thus, the optimal value of C is around 0.5.

At C = 0.5, Training Accuracy = 99.57% ; Test Accuracy = 98.16%

Labels		10 features	25 features
2 and 3	Training Acc.	95.87%	96.92%
	Testing Acc.	95.49%	96.83%
	Optimal C	0.01	0.005
4 and 7	Training Acc.	96.73%	97.55%
	Testing Acc.	96.49%	97.50%
	Optimal C	0.05	0.005
6 and 8	Training Acc.	97.85%	99.57%
	Testing Acc.	97.33%	98.16%
	Optimal C	1.0	0.5

Table showing variation of Training/Test Accuracy and Optimal C for Linear Kernel for 3 different pair of classes (for 10 and 25 features)

From the above table, we observe that: -

1. Regarding the best hyperparameter, there is not much change in C by changing the pair of classes from 2 and 3 to 4 and 7, where in case of 25 features, it remains the same (=0.005) and in case of 10 features it changes from 0.01 to 0.05. But when the pair of classes is changed to 6 and 8, the optimal C changes a lot. In case of 25 features, it changes from 0.005 to 0.5 (100 times) and in case of 10 features, it changes from 0.05 to 1 (20 times change).

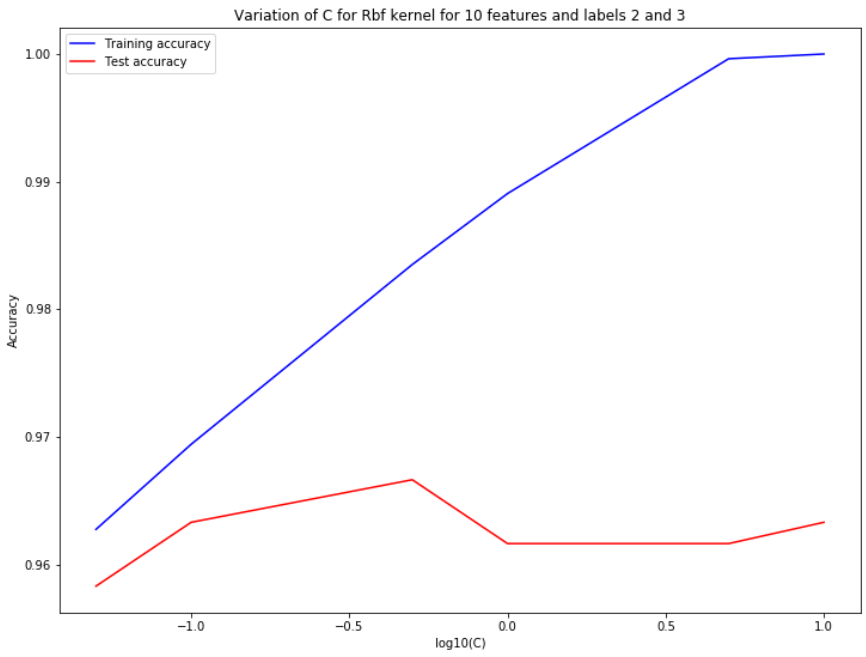
2. Thus, we conclude that for Linear kernel, the value of optimal hyperparameter depends largely on the pair of classes which are being chosen for the binary classification.

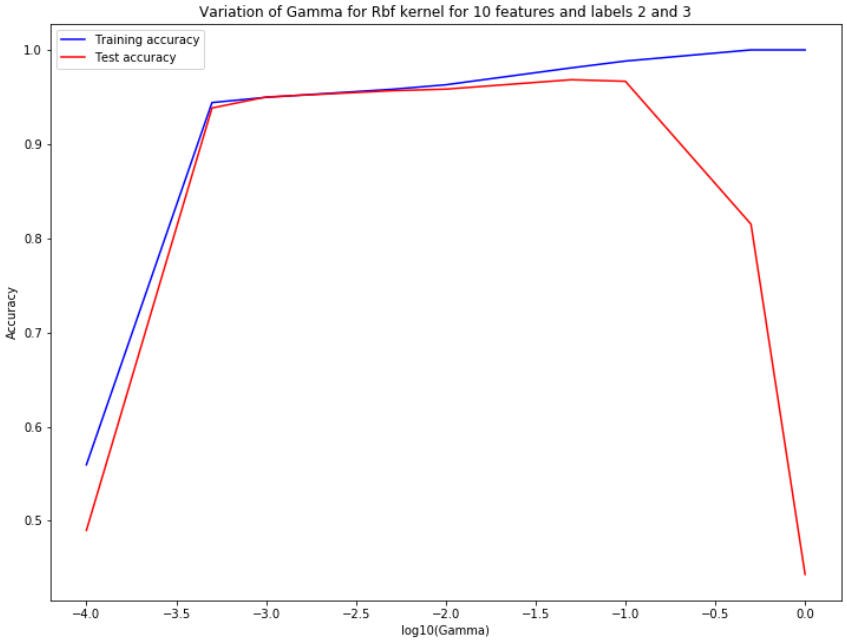
3. As we go from one pair of class to another, we see that the training/test accuracy does not change much (a maximum of 2.5% change is observed in the above table). Also, as we go from 10 features to 25 features, the training as well as the testing accuracy increases and the value of the optimal C decreases. Optimal C is defined as the C for which good fitting of model occurs.

Non-Linear Kernel (Gaussian)

Labels Used – 2 and 3

No. of features – 10





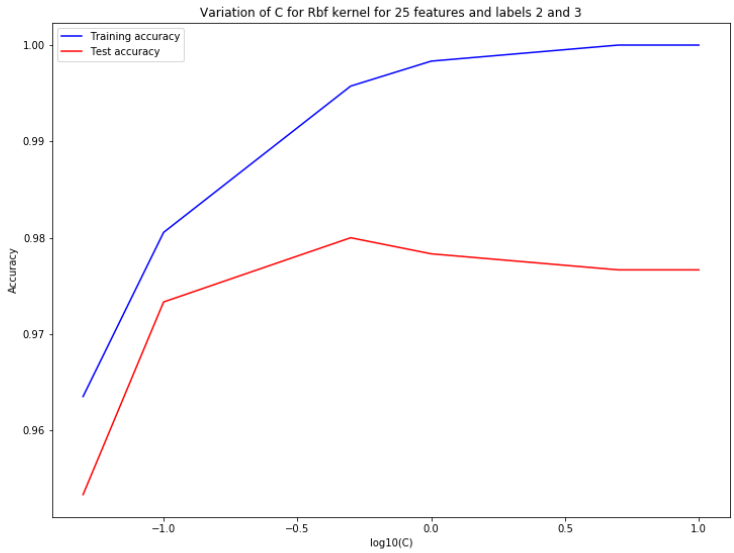
Optimal C = 0.5 (from graph), 0.47 (from Grid Search)

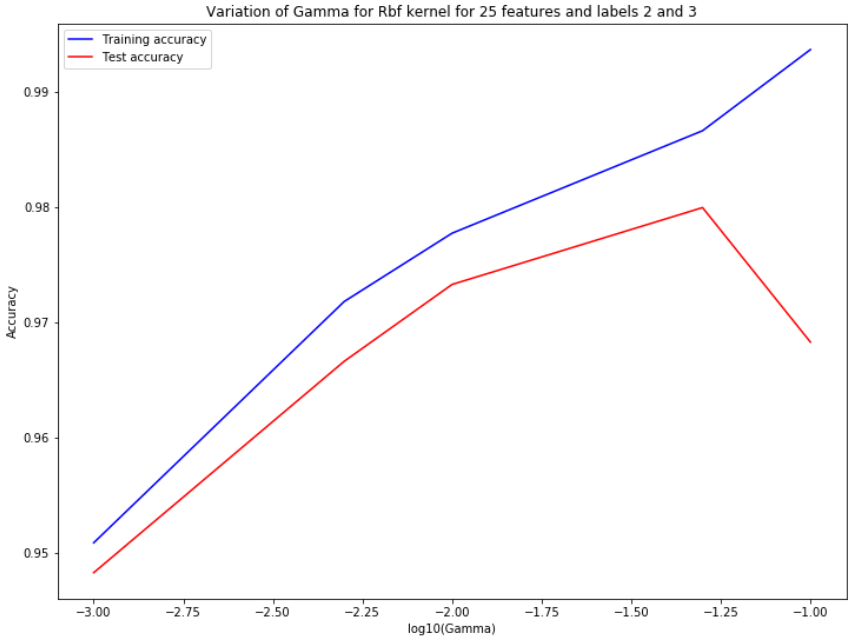
Optimal gamma = 0.1(from graph), 0.071 (from Grid Search)

At C=0.5 and gamma = 0.1,

Training Accuracy = 98.42% ; Test Accuracy = 96.66%

No. of features – 25





Optimal C = 0.5 (from graph), 0.27 (from Grid Search)

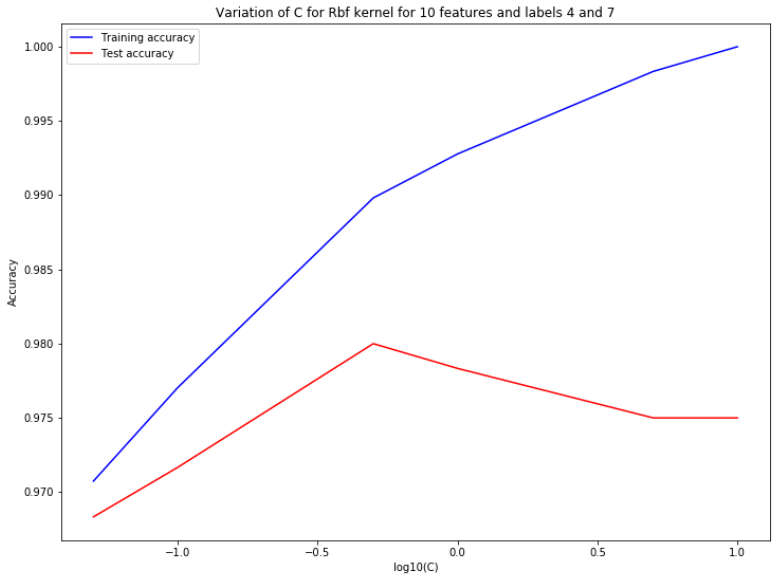
Optimal gamma = 0.05 (from graph), 0.02 (from Grid Search)

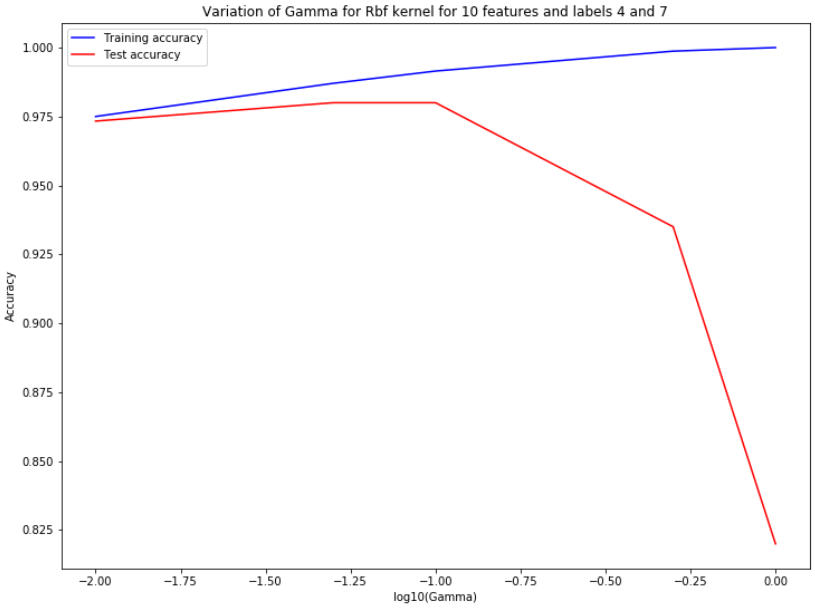
At C=0.5 and gamma = 0.05,

Training Accuracy = 99.375% ; Test Accuracy = 98.33%

Labels Used – 4 and 7

No. of features – 10





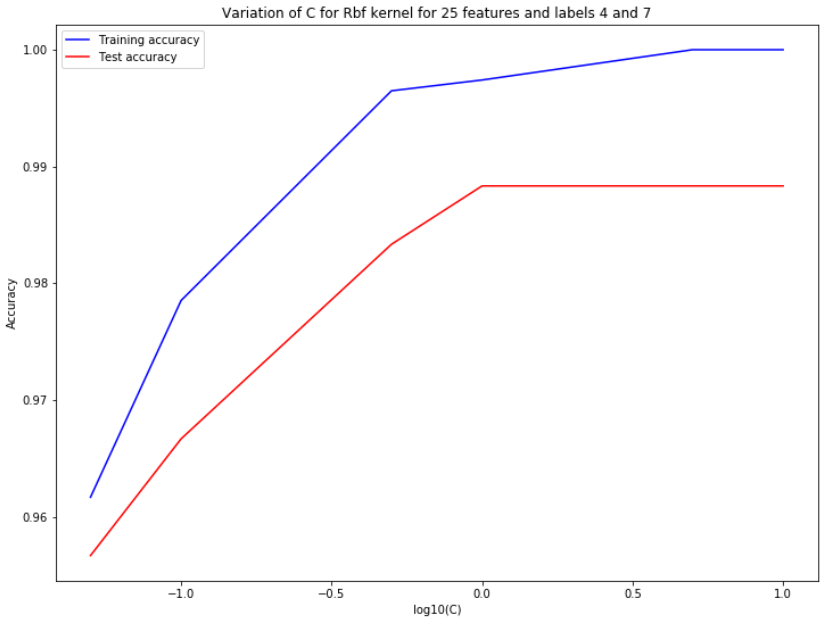
Optimal C = 0.5 (from graph), 0.84 (from Grid Search)

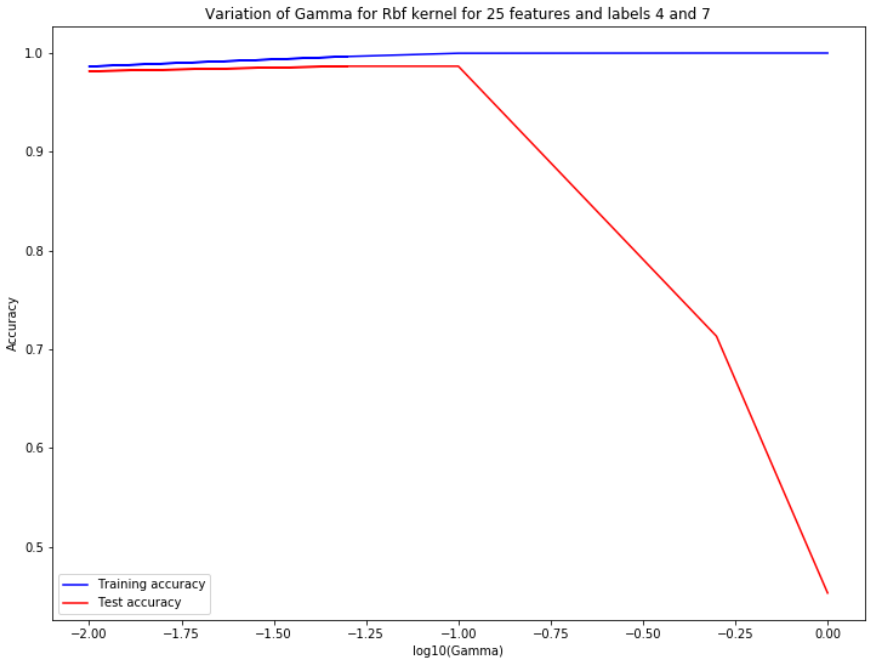
Optimal gamma = 0.1 (from graph), 0.1 (from Grid Search)

At C=0.5 and gamma = 0.1,

Training Accuracy = 98.98% ; Test Accuracy = 97.99%

No. of features – 25





Optimal C = 1 (from graph), 0.84 (from Grid Search)

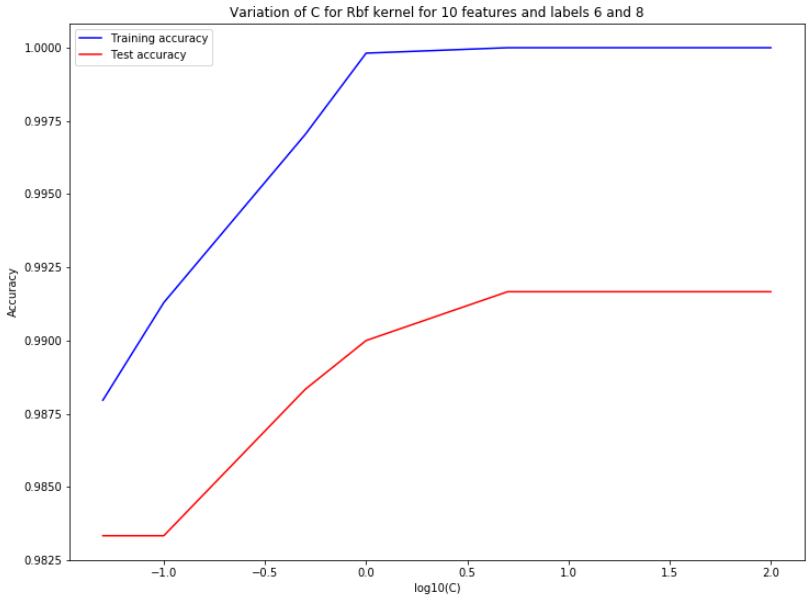
Optimal gamma = 0.1 (from graph), 0.06 (from Grid Search)

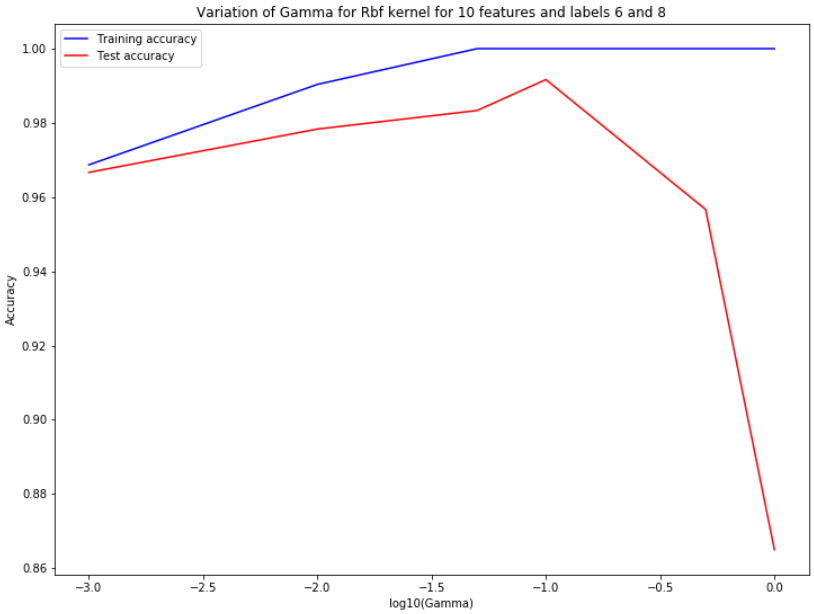
At C=1 and gamma = 0.1,

Training Accuracy = 100% ; Test Accuracy = 98.66%

Labels Used – 6 and 8

No. of features – 10





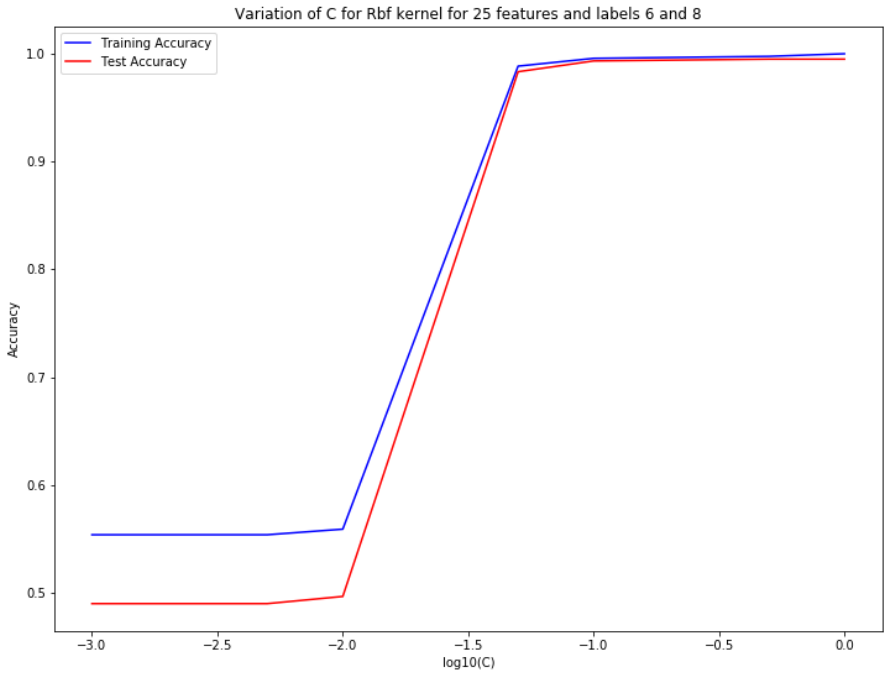
Optimal C = 5 (from graph), 2.69 (from Grid Search)

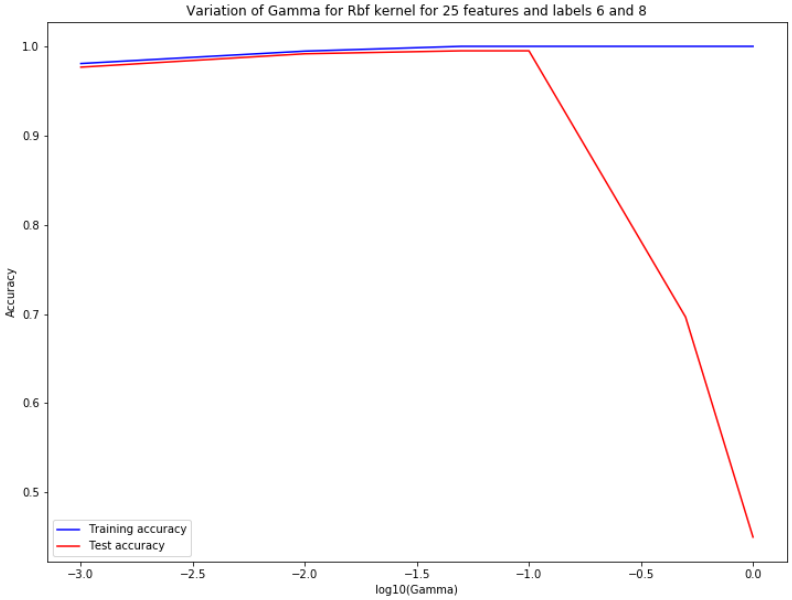
Optimal gamma = 0.1 (from graph), 0.088 (from Grid Search)

At C=5 and gamma = 0.1,

Training Accuracy = 100% ; Test Accuracy = 99.16%

No. of features – 25





Optimal C = 0.9 (from Grid Search)

Optimal gamma = 0.1 (from graph), 0.05 (from Grid Search)

At C=5 and gamma = 0.05,

Training Accuracy = 100% ; Test Accuracy = 99.16%

Labels		10 features	25 features
2 and 3	Training Acc.	98.42%	99.375%
	Testing Acc.	96.66%	98.33%
	Optimal C	0.5	0.5
	Optimal Gamma	0.1	0.05
4 and 7	Training Acc.	98.98%	100%
	Testing Acc.	97.99%	98.66%
	Optimal C	0.5	1
	Optimal Gamma	0.1	0.1
6 and 8	Training Acc.	100%	100%
	Testing Acc.	99.16%	99.16%
	Optimal C	5	5
	Optimal Gamma	0.1	0.05

Table showing variation of Training/Test Accuracy and Optimal C/Gamma for Rbf Kernel for 3 different pair of classes (for 10 and 25 features)

From the above table, we observe that: -

1. Regarding the best hyperparameter, there is not much change in C by changing the pair of classes from 2 and 3 to 4 and 7, where in case of 10 features, it remains the same (=0.5) and in case of 25 features it changes from 0.5 to 1. But when the pair of classes is changed to 6 and 8, the optimal C changes a lot. In case of 10 features, it changes from 0.5 to 5 (10 times) and in case of 25 features, it changes from 0.05 to 1 (20 times change).

2. The gamma remains constant for all the pair of classes taken when 10 features are taken and for 25 features, it changes only slightly (from 0.05 to 0.1) across the pair of classes as seen from above.

3. Thus, we conclude that for rbf kernel, the value of optimal hyperparameter C depends largely on the pair of classes which are being chosen for the binary classification, but the value of gamma does not depend on the pair of classes taken for binary classification.

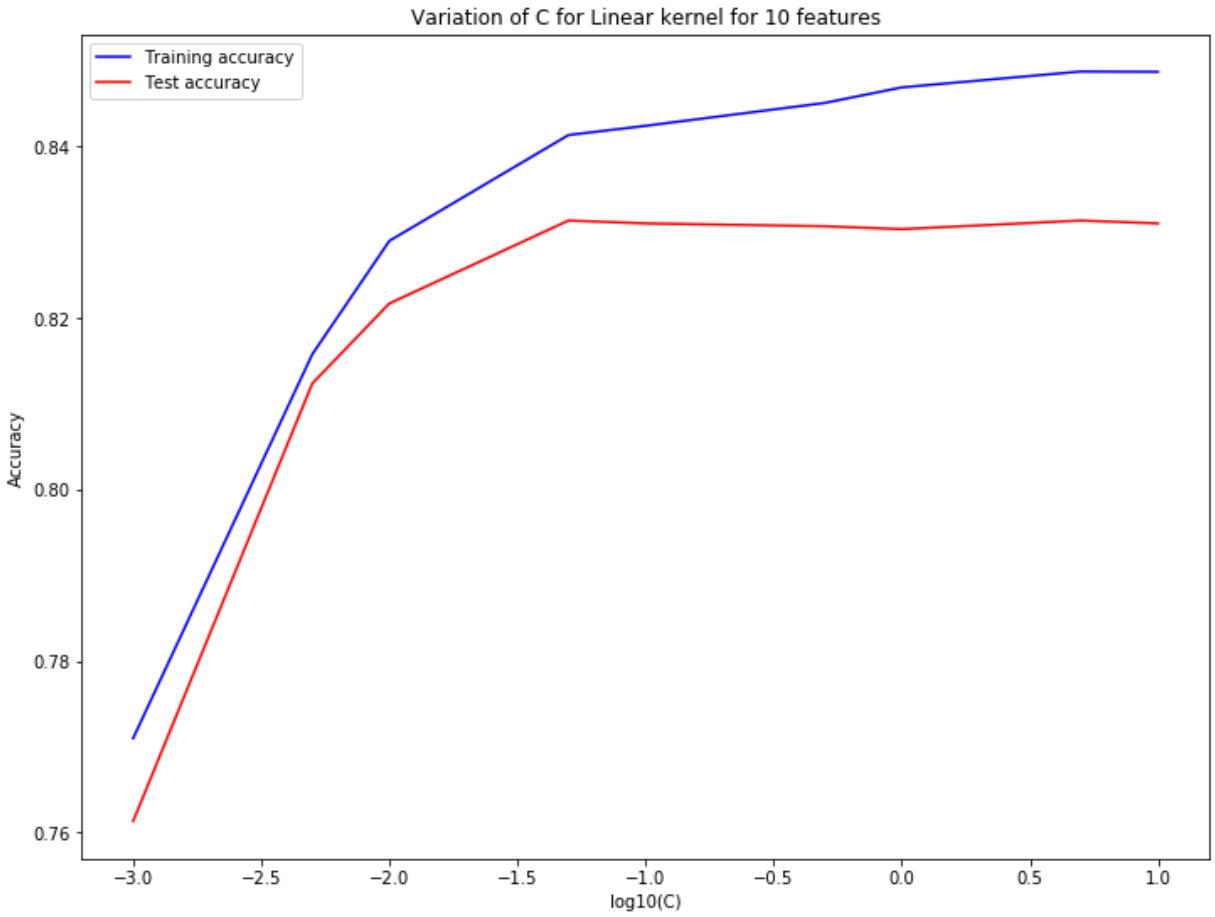
4. As we go from one pair of class to another, we see that the training/test accuracy does not change much (a maximum of 2.5% change is observed in the testing accuracy and a maximum of 1.5% change is observed in training accuracy).

5. As we move from 10 features to 25 features, the testing as well as the training accuracy increases. Also, the value of the optimal hyperparameters remains almost the same.

Case 2 – Multiclass classification using LIBSVM

Linear Kernel

No. of features – 10

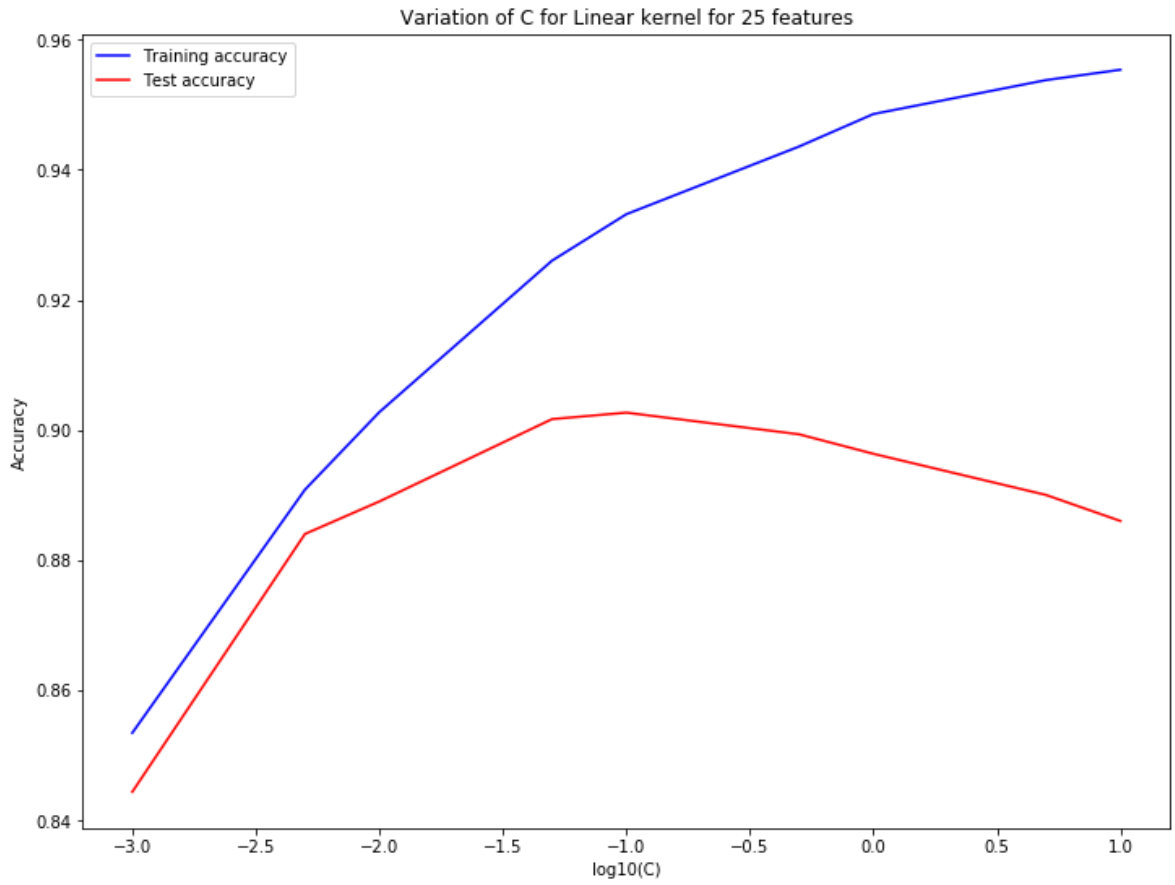


Seeing, the above graph for multiclass classification, we get the value of optimal C = 0.05, as above this model is overfitting and below this model is underfitting. This value of C is the same as derived from Grid Search.

At C = 0.05,

Training Accuracy = 84.12% ; Test Accuracy = 83.13%

No. of features – 25



Seeing, the above graph for multiclass classification, we get the value of optimal C = 0.1, as above this model is overfitting and below this model is underfitting. Value of C derived from Grid Search is 0.05

At C = 0.1,

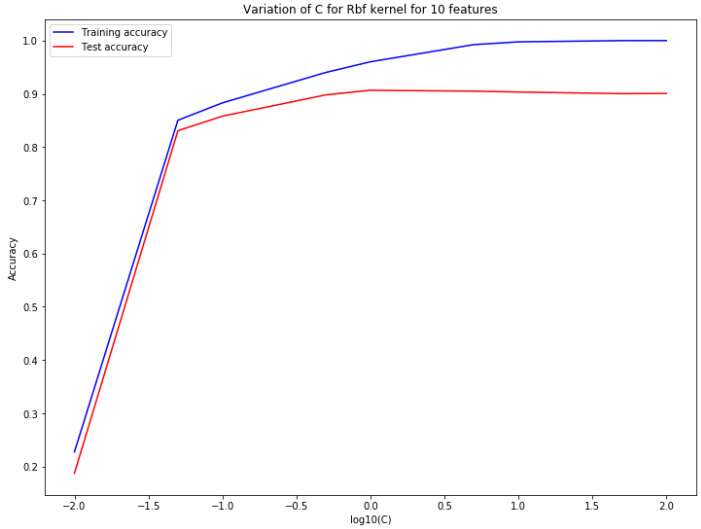
Training Accuracy = 93.31% ; Test Accuracy = 90.26%

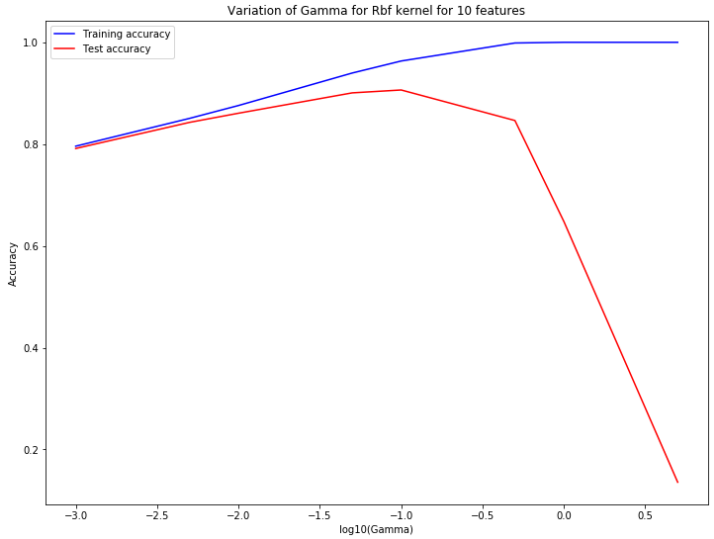
At C = 0.05,

Training Accuracy = 92.60% ; Test Accuracy = 90.16%

Non-Linear Kernel (Gaussian)

No. of features – 10





Optimal C = 1 (from graph), 1 (from Grid Search)

Optimal Gamma = 0.1 (from graph), 0.09 (from Grid Search)

For calculating the optimal C and Gamma, the concept of underfitting and overfitting has been used

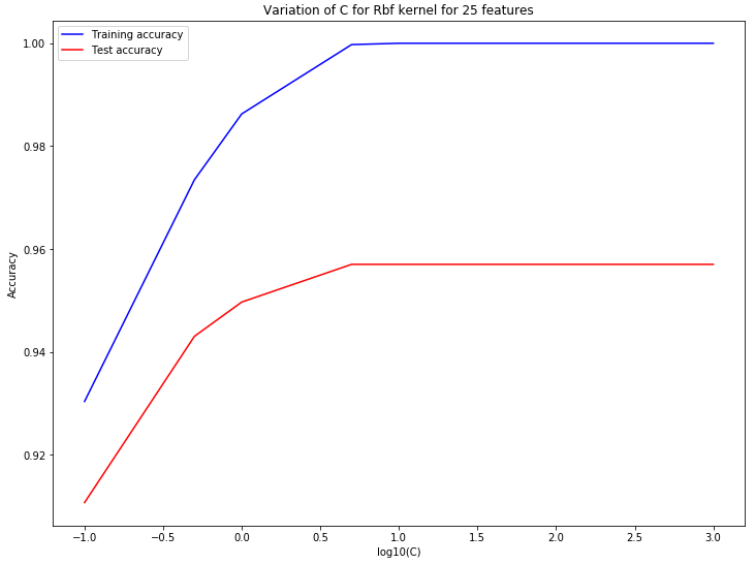
For C = 1 and Gamma = 0.1,

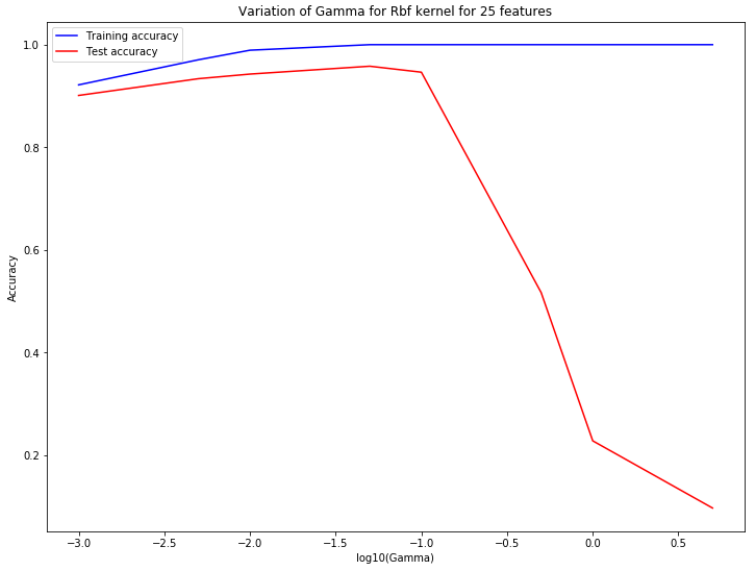
Train Accuracy = 96.32% ; Test Accuracy = 90.63%

For C = 1 and Gamma = 0.09,

Train Accuracy = 96.01% ; Test Accuracy = 90.66%

No. of features – 25





Optimal C = 5 (from graph), 10 (from Grid Search)

Optimal Gamma = 0.05 (from graph), 0.04 (from Grid Search)

For calculating the optimal C and Gamma, the concept of underfitting and overfitting has been used

For C = 5 and Gamma = 0.05,

Train Accuracy = 100% ; Test Accuracy = 95.79%

For C = 10 and Gamma = 0.04,

Train Accuracy = 100% ; Test Accuracy = 95.69%

		10 features	25 features
Linear	Training Acc.	84.12%	92.60%
	Testing Acc.	83.13%	90.16%
	Optimal C	0.05	0.05
Rbf	Training Acc.	93.62%	100%
	Testing Acc.	90.63%	95.79%
	Optimal C	1	5
	Optimal Gamma	0.1	0.05

From the above table, we observe that-

1. Firstly, the chosen library (LIBSVM in this case), uses one-vs-one method for multiclass classification. That is, it builds a separate classifier for all pairs of classes available and classifies on the basis of the combined result of all the classifiers.

2. As C decreases from its optimal value, underfitting starts occurring and when C starts increasing from its optimal value, overfitting starts occurring. Same happens with the increase or decrease of gamma from its optimal value.

3. For linear kernel, the optimal C remains almost the same as obtained in a majority of pairs in binary classification. This is because linear classifier is not able to classify multiple labelled classes having multiple features efficiently.

4. For rbf kernel, the optimal C changes a lot (5 here as opposed to 0.5 in the majority pairs in binary classification) as compared to binary classification but the optimal gamma remains almost the same (almost 0.1 in both the cases). This is because rbf kernel is very good at classifying points with multiple features, so it tries to put a heavy penalty on the points which are getting misclassified and thereby tries to draw a boundary same as that of a hard-margin SVM. Thus, this causes the optimal C value to differ a lot.

5. For linear kernel, as we go from 10 to 25 features, the training as well as test accuracy increases but the value of optimal C remains almost the same.

6. For rbf kernel, as we go from 10 to 25 features, the training as well as test accuracy increases. The value of optimal gamma remains almost the same whereas the value of optimal C increases slightly (5 times).

Case 3– Binary Classification using Convex Optimisation Package (CVX)

Linear Kernel

Labels used for classification – 0 and 1

Number of features - 10
Value of C derived after tuning it using Grid Search = **0.0018** (i.e. the value of C for which the accuracy becomes maximum for the first time, as before this underfitting occurs and above this overfitting starts occurring)

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy - 97.5%
No. of support vectors – 99
Support Vector Indices - [3 6 23 25 28 29 34 37 44 62 66 72 73 79 80 81 83 84 95 104 106 110 112 117 119 125 127 130 138 139 145 151 157 159 161 165 169 171 172 174 175 179 180 182 185 192 199 200 207 208 211 212 218 227 231 233 235 237 242 248 249 257 262 264 275 277 283 289 315 334 342 359 365 369 373 374 375 379 390 397 404 407 412 421 423 425 429 432 446 450 457 459 461 466 468 470 471 472 474]

CVX
Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy – 97.5%
No. of support vectors – 99
Support Vector Indices - [3 6 23 25 28 29 34 37 44 62 66 72 73 79 80 81 83 84 95 104 106 110 112 117 119 125 127 130 138 139 145 151 157 159 161 165 169 171 172 174 175 179 180 182 185 192 199 200 207 208 211 212 218 227 231 233 235 237 242 248 249 257 262 264 275 277 283 289 315 334 342 359 365 369 373 374 375 379 390 397 404 407 412 421 423 425 429 432 446 450 457 459 461 466 468 470 471 472 474]

Number of features – 25

Value of C derived after tuning it using Grid Search = **0.00122** (i.e. the value of C for which the accuracy becomes maximum for the first time, as before this underfitting occurs and above this overfitting starts occurring)

LIBSVM

Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy - 97.5%
No. of support vectors – 117
Support Vector Indices - [3 5 6 16 18 23 25 28 29 34 37 44 66 73 79 80 81 83 91 94 95 104 105 110 112 117 118 119 121 127 138 139 145 147 153 157 161 165 169 171 172 174 175 179 180 181 182 185 192 199 200 207 208 212 213 218 225 226 227 231 233 235 237 242 248 249 251 257 262 274 275 283 286 289 294 304 315 317 328 333 334 342 346 359 367 368 369 371 373 374 375 379 390 392 402 404 407 412 421 423 425 432 433 435 446 448 450 457 458 461 466 468 470 471 472 473 474]

CVX

Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy – 99.16%
No. of support vectors – 117
Support Vector Indices - [3 5 6 16 18 23 25 28 29 34 37 44 66 73 79 80 81 83 91 94 95 104 105 110 112 117 118 119 121 127 138 139 145 147 153 157 161 165 169 171 172 174 175 179 180 181 182 185 192 199 200 207 208 212 213 218 225 226 227 231 233 235 237 242 248 249 251 257 262 274 275 283 286 289 294 304 315 317 328 333 334 342 346 359 367 368 369 371 373 374 375 379 390 392 402 404 407 412 421 423 425 432 433 435 446 448 450 457 458 461 466 468 470 471 472 473 474]

Labels used for classification – 2 and 3

Number of features - 10
Value of C derived after tuning it using Grid Search = **0.221** (i.e. the value of C for which the accuracy becomes maximum for the first time, as before this underfitting occurs and above this overfitting starts occurring)

LIBSVM

Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 96.04%

Test Accuracy - 94.16%
No. of support vectors – 65
Support Vector Indices - [4 13 18 24 29 53 65 69 87 91 97 102 103 111 113 116 130 131 171 173 181 192 193 197 198 199 200 213 215 219 220 236 251 254 255 260 264 284 287 299 305 310 314 321 331 338 344 345 351 352 375 385 387 402 413 414 420 426 439 440 443 444 465 466 470]

CVX
Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 96.04%
Test Accuracy – 94.16%
No. of support vectors – 65
Support Vector Indices - [4 13 18 24 29 53 65 69 87 91 97 102 103 111 113 116 130 131 171 173 181 192 193 197 198 199 200 213 215 219 220 236 251 254 255 260 264 284 287 299 305 310 314 321 331 338 344 345 351 352 375 385 387 402 413 414 420 426 439 440 443 444 465 466 470]

Number of features – 25
Value of C derived after tuning it using Grid Search = **0.0058** (i.e. the value of C for which the accuracy becomes maximum for the first time, as before this underfitting occurs and above this overfitting starts occurring)

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 97.08%
Test Accuracy - 94.16%
No. of support vectors – 148
Support Vector Indices - [4 6 10 11 13 15 18 24 27 29 50 53 65 66 68 69 70 72 78 79 80 86 87 91 92 93 94 97 102 103 109 111 113 116 117 125 130 131 133 139 147 149 156 161 165 166 171 172 173 177 181 188 192 193 197 198 199 200 201 202 209 210 213 215 216 220 227 228 231 236 237 243 251 254 255 260 264 269 274 275 284 287 288 289 298 299 300 301 304 305 306 310 311 314 315 320 321 325 326 331 337 338 340 345 351 352 357 363 368 371 374 375 380 385 386 387 389 390 392 395 396 402 403 405 410 411 413 414 416 420 426 429 436 439 440 443 444 447 448 449 457 465 466 467 468 470 473 478]

CVX
Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 97.08%

Test Accuracy – 95%
No. of support vectors – 148
Support Vector Indices - [4 6 10 11 13 15 18 24 27 29 50 53 65 66 68 69 70 72 78 79 80 86 87 91 92 93 94 97 102 103 109 111 113 116 117 125 130 131 133 139 147 149 156 161 165 166 171 172 173 177 181 188 192 193 197 198 199 200 201 202 209 210 213 215 216 220 227 228 231 236 237 243 251 254 255 260 264 269 274 275 284 287 288 289 298 299 300 301 304 305 306 310 311 314 315 320 321 325 326 331 337 338 340 345 351 352 357 363 368 371 374 375 380 385 386 387 389 390 392 395 396 402 403 405 410 411 413 414 416 420 426 429 436 439 440 443 444 447 448 449 457 465 466 467 468 470 473 478]

Labels used for classification – 6 and 9

Number of features - 10
Value of C derived after tuning it using Grid Search = **0.0016** (i.e. the value of C for which the accuracy becomes maximum for the first time, as before this underfitting occurs and above this overfitting starts occurring)

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.58%
Test Accuracy - 99.16%
No. of support vectors – 214
Support Vector Indices - [0 4 12 13 15 17 21 22 27 28 29 30 31 32 39 40 43 46 47 48 49 52 53 56 60 62 63 65 66 68 69 70 71 74 75 77 81 83 87 88 89 91 93 94 95 100 102 104 107 112 113 115 119 120 121 124 125 131 133 135 136 138 139 140 144 145 146 148 149 150 151 154 155 157 160 164 167 171 172 175 176 177 178 182 186 188 191 192 194 196 197 203 204 206 210 211 212 213 218 221 222 226 228 230 233 238 241 244 246 248 251 255 257 260 265 269 274 275 276 277 278 279 282 283 286 287 291 294 295 296 297 304 305 308 309 311 313 314 315 318 322 326 328 329 333 334 336 337 340 341 343 344 349 351 354 355 356 357 360 361 364 369 372 374 376 378 380 383 390 395 397 398 399 400 402 403 410 413 414 415 416 417 420 422 423 424 426 428 429 430 431 435 436 441 443 444 446 447 450 454 455 456 459 463 464 465 466 468 471 472 474 475 476 479]

CVX
Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.58%

Test Accuracy – 99.16%
No. of support vectors – 213
Support Vector Indices - [0 4 12 13 15 17 21 22 27 28 29 30 31 32 39 40 43 46 47 48 49 52 53 56 60 62 63 65 66 68 69 70 71 75 77 81 83 87 88 89 91 93 94 95 100 102 104 107 112 113 115 119 120 121 124 125 131 133 135 136 138 139 140 144 145 146 148 149 150 151 154 155 157 160 164 167 171 172 175 176 177 178 182 186 188 191 192 194 196 197 203 204 206 210 211 212 213 218 221 222 226 228 230 233 238 241 244 246 248 251 255 257 260 265 269 274 275 276 277 278 279 282 283 286 287 291 294 295 296 297 304 305 308 309 311 313 314 315 318 322 326 328 329 333 334 336 337 340 341 343 344 349 351 354 355 356 357 360 361 364 369 372 374 376 378 380 383 390 395 397 398 399 400 402 403 410 413 414 415 416 417 420 422 423 424 426 428 429 430 431 435 436 441 443 444 446 447 450 454 455 456 459 463 464 465 466 468 471 472 474 475 476 479]

Number of features – 25
Value of C derived after tuning it using Grid Search = **0.00095** (i.e. the value of C for which the accuracy becomes maximum for the first time, as before this underfitting occurs and above this overfitting starts occurring)

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy - 100%
No. of support vectors – 266
Support Vector Indices - [0 1 4 6 12 13 14 15 17 18 20 21 22 24 27 28 29 30 31 33 39 40 43 44 46 47 48 49 50 52 53 56 59 60 61 62 63 65 66 68 69 70 71 74 75 77 81 82 83 87 88 89 91 93 94 95 99 100 102 104 105 107 109 113 115 119 120 121 123 125 126 131 133 135 136 138 139 140 141 144 145 146 148 149 150 151 154 155 157 164 165 167 168 171 172 175 176 177 178 180 182 183 186 188 191 192 193 194 196 197 199 200 202 203 204 206 209 210 211 212 213 214 218 221 222 223 226 227 228 233 238 241 244 245 246 248 251 253 254 255 257 258 259 260 265 269 273 274 275 276 277 278 279 280 282 283 286 287 290 291 292 293 294 295 296 303 304 305 308 309 311 312 313 314 315 317 318 320 322 326 327 328 329 330 333 334 336 337 338 340 341 343 344 349 351 354 355 356 357 358 359 361 364 366 367 369 372 374 376 378 379 380 381 383 384 387 390 393 394 395 397 398 399 400 402 403 410 413 414 415 416 417 418 420 422 423 424 426 428 429 430 431 435 436 438 441 443 444 446 447 450 452 454 455 456 459 463 464 465 466 468 469 471 472 474 479]

CVX

Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy – 100%
No. of support vectors – 266
Support Vector Indices - [0 1 4 6 12 13 14 15 17 18 20 21 22 24 27 28 29 30 31 33 39 40 43 44 46 47 48 49 50 52 53 56 59 60 61 62 63 65 66 68 69 70 71 74 75 77 81 82 83 87 88 89 91 93 94 95 99 100 102 104 105 107 109 113 115 119 120 121 123 125 126 131 133 135 136 138 139 140 141 144 145 146 148 149 150 151 154 155 157 164 165 167 168 171 172 175 176 177 178 180 182 183 186 188 191 192 193 194 196 197 199 200 202 203 204 206 209 210 211 212 213 214 218 221 222 223 226 227 228 233 238 241 244 245 246 248 251 253 254 255 257 258 259 260 265 269 273 274 275 276 277 278 279 280 282 283 286 287 290 291 292 293 294 295 296 303 304 305 308 309 311 312 313 314 315 317 318 320 322 326 327 328 329 330 333 334 336 337 338 340 341 343 344 349 351 354 355 356 357 358 359 361 364 366 367 369 372 374 376 378 379 380 381 383 384 387 390 393 394 395 397 398 399 400 402 403 410 413 414 415 416 417 418 420 422 423 424 426 428 429 430 431 435 436 438 441 443 444 446 447 450 452 454 455 456 459 463 464 465 466 468 469 471 472 474 479]

Thus, summarising the above data, we get the following table: -
For simplicity, Number of features used is 25 in the table

Labels		LIBSVM	CVX
0 and 1	Training Acc.	99.79%	99.79%
	Testing Acc.	97.5%	99.16%
	No. of SV	117	117
2 and 3	Training Acc.	97.08%	97.08%
	Testing Acc.	94.16%	95%
	No. of SV	148	148
6 and 9	Training Acc.	99.79%	99.79%
	Testing Acc.	100%	100%
	No. of SV	266	266

Non-Linear Kernel (Gaussian)

Labels used for classification – 2 and 3

Number of features – 10
Parameters using Grid Search: -
C = **0.47**; gamma = **0.071**

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 97.50%
Test Accuracy - 97.50%
No. of support vectors – 161
Support Vector Indices - [4 6 9 11 13 18 24 27 29 33 36 46 48 50 52 53 57 63 65 66 67 69 70 72 74 83 84 87 91 92 93 94 97 99 100 102 103 109 111 113 116 120 122 129 130 131 133 150 156 158 162 163 166 168 171 173 176 177 180 181 188 189 192 193 197 198 199 200 202 207 209 213 215 216 219 220 227 228 232 236 237 243 251 254 255 257 260 261 264 266 267 268 269 275 279 284 287 299 301 305 306 310 314 315 321 325 326 328 330 331 335 337 338 340 344 345 351 352 357 359 360 367 368 374 375 380 385 386 387 394 395 396 398 402 403 406 410 413 414 416 420 426 427 429 434 439 440 443 444 447 448 457 458 462 465 466 467 468 469 470 472]

CVX
Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 97.50%
Test Accuracy – 97.50%
No. of support vectors – 162
Support Vector Indices - [4 6 9 11 13 18 24 27 29 33 36 46 48 50 52 53 57 63 65 66 67 69 70 72 74 83 84 87 91 92 93 94 97 99 100 102 103 109 111 113 116 120 122 129 130 131 133 134 150 156 158 162 163 166 168 171 173 176 177 180 181 188 189 192 193 197 198 199 200 202 207 209 213 215 216 219 220 227 228 232 236 237 243 251 254 255 257 260 261 264 266 267 268 269 275 279 284 287 299 301 305 306 310 314 315 321 325 326 328 330 331 335 337 338 340 344 345 351 352 357 359 360 367 368 374 375 380 385 386 387 394 395 396 398 402 403 406 410 413 414 416 420 426 427 429 434 439 440 443 444 447 448 457 458 462 465 466 467 468 469 470 472]

Number of features – 25
Parameters using Grid Search: -
C = **0.27**; gamma = **0.02**

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)

Training Accuracy – 98.12%
Test Accuracy - 98.33%
No. of support vectors – 175
Support Vector Indices - [4 6 10 11 13 15 18 19 24 27 29 36 50 52 53 65 66 68 69 70 72 78 79 80 81 83 86 87 91 92 93 94 97 98 99 102 103 109 111 113 116 117 120 125 130 131 133 139 147 150 156 158 161 163 165 166 168 171 172 173 177 179 181 188 189 192 193 197 198 199 200 201 209 210 213 215 216 220 227 228 231 236 237 241 243 251 254 255 257 260 264 267 268 269 274 275 284 285 287 288 289 298 299 300 301 304 305 306 307 310 311 314 315 318 320 321 325 326 331 333 337 338 339 340 344 345 351 352 357 359 360 363 368 371 374 375 380 385 386 387 389 395 396 402 403 405 410 411 413 414 416 420 426 427 429 434 437 439 440 443 444 447 448 449 457 458 462 465 466 467 468 470 472 473 478]

CVX
Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 98.33%
Test Accuracy – 98.33%
No. of support vectors – 180
Support Vector Indices - [4 6 10 11 13 15 18 19 24 27 29 36 50 52 53 65 66 68 69 70 72 78 79 80 81 83 86 87 91 92 93 94 97 98 99 102 103 109 111 113 116 117 119 120 125 130 131 133 139 147 149 150 156 158 161 162 163 165 166 168 171 172 173 176 177 179 181 188 189 192 193 197 198 199 200 201 209 210 213 215 216 220 227 228 231 236 237 241 243 251 254 255 257 260 264 267 268 269 274 275 284 285 287 288 289 298 299 300 301 304 305 306 307 310 311 314 315 318 320 321 325 326 331 333 337 338 339 340 344 345 351 352 357 359 360 363 368 371 374 375 380 385 386 387 389 395 396 400 402 403 405 410 411 413 414 416 420 426 427 429 434 437 439 440 443 444 447 448 449 457 458 462 465 466 467 468 470 472 473 478]

Labels used for classification – 4 and 7

Number of features – 10
Parameters using Grid Search: -
C = **0.84**; gamma = **0.1**

LIBSVM
Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.16%
Test Accuracy - 97.5%
No. of support vectors – 145

Support Vector Indices - [5 12 18 22 30 32 33 34 36 39 40 42 44 45 46 48 51 52 53 55 56 62 65 69 70 75 76 79 81 83 92 93 96 101 102 107 108 109 110 117 119 120 124 126 130 131 132 133 135 139 140 149 155 156 158 160 162 163 169 171 180 188 189 190 193 194 196 199 200 203 211 216 218 226 231 234 238 243 246 248 250 251 258 261 263 264 266 269 274 276 280 293 295 297 302 304 306 309 310 312 313 315 320 321 323 328 331 333 335 336 341 345 353 363 370 371 375 377 379 387 399 413 414 415 420 425 429 430 431 435 438 439 442 444 446 448 451 452 461 468 470 475 476 478 479]

CVX

Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.58%
Test Accuracy – 96.66%
No. of support vectors – 147
Support Vector Indices - [5 12 18 22 30 32 33 34 36 39 40 42 44 45 46 48 51 52 53 55 56 62 65 69 70 75 76 79 81 83 92 93 96 101 102 107 108 109 110 117 119 120 124 126 130 131 132 133 135 139 140 149 155 156 158 160 162 163 169 171 180 188 189 190 193 194 196 199 200 203 211 216 218 226 231 234 238 243 246 248 250 251 258 261 263 264 266 269 274 276 280 293 295 297 300 302 304 306 309 310 312 313 315 320 321 323 328 331 333 335 336 341 345 353 363 370 371 375 377 379 387 399 413 414 415 420 425 429 430 431 434 435 438 439 442 444 446 448 451 452 461 468 470 475 476 478 479]

Number of features – 25
Parameters using Grid Search: -
C = **0.84**; gamma = **0.06**

LIBSVM

Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy - 99.16%
No. of support vectors – 190
Support Vector Indices - [2 5 7 12 13 16 18 21 22 30 32 33 34 38 39 40 42 44 45 46 48 51 52 53 55 62 65 69 70 76 79 81 83 92 93 96 98 100 101 102 106 107 108 109 110 114 117 118 119 120 124 126 130 131 132 133 135 136 139 140 143 148 149 151 155 156 158 160 162 163 164 166 169 171 174 180 182 185 188 189 190 193 194 196 199 200 203 211 215 216 218 219 224 225 226 231 234 235 238 243 246 248 250 251 258 261 263 264 265

269 271 273 274 275 276 277 280 281 292 293 294 295 297 298 302 304 306 309 310 315 318 321 323 327 328 330 331 332 335 341 343 345 350 353 355 363 370 371 377 379 381 387 388 393 394 399 401 404 406 411 413 414 415 425 429 430 431 434 435 437 438 439 441 442 444 445 447 448 449 452 457 458 462 464 468 470 475 476 478 479]

CVX

Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 99.79%
Test Accuracy – 99.16%
No. of support vectors – 199
Support Vector Indices - [2 5 7 12 13 16 18 21 22 30 32 33 34 37 38 39 40 42 44 45 46 48 51 52 53 55 56 62 65 69 70 76 79 81 83 92 93 96 98 100 101 102 105 106 107 108 109 110 114 117 118 119 120 124 126 130 131 132 133 135 136 139 140 143 148 149 151 155 156 158 160 162 163 164 166 169 171 174 180 182 185 188 189 190 193 194 196 199 200 203 211 215 216 218 219 224 225 226 230 231 234 235 238 243 246 248 250 251 254 258 261 263 264 265 269 271 273 274 275 276 277 279 280 281 292 293 294 295 297 298 302 304 306 309 310 315 318 321 323 327 328 330 331 332 335 341 343 345 350 353 355 363 370 371 377 379 381 387 388 393 394 399 401 404 406 411 413 414 415 420 425 429 430 431 434 435 437 438 439 441 442 444 445 447 448 449 451 452 457 458 461 462 464 468 470 475 476 478 479]

Labels used for classification – 6 and 8

Number of features – 10
Parameters using Grid Search: -
C = **2.69**; gamma = **0.088**

LIBSVM

Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 100%
Test Accuracy - 98.33%
No. of support vectors – 120
Support Vector Indices - [1 2 14 15 19 21 26 31 33 34 36 42 43 46 57 60 65 70 74 79 88 89 93 103 104 106 107 114 115 122 124 129 133 140 144 148 149 150 151 165 167 168 170 172 178 179 180 192 194 196 197 200 206 211 216 217 218 224 229 240 243 245 247 250 261 263 264 269 270 271 276 282 290 295 296 301 306 307 310 315 327 331 337 338 339 340 344 348 354 360 361 370 373 375 376 377 379 384 391 410 411 415 416 421 423 426 428 431 445 446 449 451 452 455 458 459 467 468 469 471]

CVX

Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 100%
Test Accuracy – 98.33%
No. of support vectors – 124
Support Vector Indices - [1 2 14 15 19 21 26 31 33 34 36 42 43 46 57 60 65 70 74 79 88 89 93 103 104 106 107 114 115 122 124 129 133 140 144 148 149 150 151 165 167 168 170 172 178 179 180 192 194 196 197 200 206 211 216 217 218 224 229 240 243 245 247 250 261 263 264 269 270 271 276 282 290 295 296 301 306 307 310 315 327 331 337 338 339 340 341 344 348 354 359 360 361 370 373 375 376 377 379 381 384 391 410 411 415 416 421 423 426 428 429 431 445 446 449 451 452 455 458 459 467 468 469 471]

Number of features – 25
Parameters using Grid Search: -
C = **0.9**; gamma = **0.05**

LIBSVM

Method used – Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 100%
Test Accuracy - 99.16%
No. of support vectors – 162
Support Vector Indices - [1 2 7 10 12 14 15 19 21 26 31 32 33 34 36 37 39 42 43 46 57 60 65 66 67 70 74 77 78 79 80 88 89 92 95 103 104 106 107 111 115 117 121 122 124 125 133 140 144 146 148 149 150 151 165 166 167 168 170 172 178 179 180 181 190 192 193 194 196 197 199 200 213 215 216 217 218 219 224 227 228 229 236 240 243 245 247 249 250 251 254 259 261 262 263 264 269 270 271 276 277 295 296 298 301 306 307 310 315 321 322 327 331 335 337 338 340 343 344 348 354 359 360 363 370 373 375 376 379 381 382 390 391 400 406 407 411 414 415 420 422 423 424 426 428 431 440 441 444 446 449 451 452 455 456 458 459 462 466 467 469 471]

CVX

Method used - Train, test splitting of data (Test size = 20% of total data)
Training Accuracy – 100%
Test Accuracy – 99.16%
No. of support vectors – 166
Support Vector Indices - [1 2 7 10 12 14 15 19 21 26 31 32 33 34 36 37 39 42 43 46 54 57 60 65 66 67 70 74 77 78 79 80 88 89 92 95 101

103 104 106 107 111 115 117 121 122 124 125 133 140 144 146 148 149 150 151 165 166 167 168 170 172 178 179 180 181 190 192 193 194 196 197 199 200 213 215 216 217 218 219 224 227 228 229 236 240 243 245 247 249 250 251 254 259 261 262 263 264 269 270 271 276 277 295 296 298 301 306 307 310 315 321 322 327 331 335 337 338 340 343 344 348 354 359 360 361 363 370 373 375 376 379 381 382 390 391 400 406 407 411 414 415 420 422 423 424 426 428 431 440 441 444 446 449 451 452 455 456 457 458 459 462 466 467 469 471]

Thus, summarising the above data, we get the following table: -
For simplicity, Number of features used is 25 in the table

Labels		LIBSVM	CVX
2 and 3	Training Acc.	98.12%	98.33%
	Testing Acc.	98.33%	98.33%
	No. of SV	175	180
4 and 7	Training Acc.	99.79%	99.79%
	Testing Acc.	99.16%	99.16%
	No. of SV	190	199
6 and 8	Training Acc.	100%	100%
	Testing Acc.	99.16%	99.16%
	No. of SV	162	166

From the above table we observe that –
1. The value of training and test accuracies as derived from the LIBSVM is almost the same as those derived using the CVXOPT package in both Linear as well as rbf kernels.

2. For linear kernel, the number of support vectors as obtained from both the libraries are exactly the same.

3. For rbf kernel, the number of support vectors as derived from CVXOPT package is slightly higher as compared to the number of support vectors derived from LIBSVM.

The following figures explain how the dual form is expressed into a format that can be fed into the convex optimisation package –

Since we will solve this optimization problem using the **CVXOPT** library in python we will need to match the solver's API which, according to the documentation is of the form:

$$\begin{aligned} & \min \frac{1}{2} x^T P x + q^T x \\ s.t. \ & Gx \leq h \\ & Ax = b \end{aligned}$$

Recall that the dual problem is expressed as:

$$\max_{\alpha} \sum_{i}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)} x^{(j)} >$$

Let $\mathbf{H}$ be a matrix such that $H_{i,j} = y^{(i)} y^{(j)} < x^{(i)} x^{(j)} >$, then the optimization becomes:

$$\begin{aligned} & \max_{\alpha} \sum_{i}^{m} \alpha_i - \frac{1}{2} \alpha^T \mathbf{H} \alpha \\ s.t. \ & \alpha_i \geq 0 \\ & \sum_{i}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

We convert the sums into vector form and multiply both the objective and the constraint by -1 which turns this into a minimization problem and reverses the inequality

$$\begin{aligned} & \min_{\alpha} \frac{1}{2} \alpha^T \mathbf{H} \alpha - 1^T \alpha \\ s.t. \ & -\alpha_i \leq 0 \\ s.t. \ & y^T \alpha = 0 \end{aligned}$$

We are now ready to convert our numpy arrays into the cvxopt format, using the same notation as in the documentation this gives

- $P := H$ a matrix of size $m \times m$
- $q := -\vec{1}$ a vector of size $m \times 1$
- $G := -diag[1]$ a diagonal matrix of -1s of size $m \times m$
- $h := \vec{0}$ a vector of zeros of size $m \times 1$
- $A := y$ the label vector of size $m \times 1$
- $b := 0$ a scalar

Computing the matrix $\mathbf{H}$ in vectorized form

Consider the simple example with 2 input samples $\{x^{(1)}, x^{(2)}\} \in \mathbb{R}^2$ which are two dimensional vectors. i.e. $x^{(1)} = (x_1^{(1)}, x_2^{(1)})^T$

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix}$$

We now proceed to creating a new matrix X' where each input sample x is multiplied by the corresponding output label y. This can be done easily in Numpy using vectorization and padding.

$$X' = \begin{bmatrix} x_1^{(1)}y^{(1)} & x_2^{(1)}y^{(1)} \\ x_1^{(2)}y^{(2)} & x_2^{(2)}y^{(2)} \end{bmatrix}$$

Finally we take the **matrix multiplication** of X' and its transpose giving $H = X'X'^T$

$$H = X'@X'^T = \begin{bmatrix} x_1^{(1)}y^{(1)} & x_2^{(1)}y^{(1)} \\ x_1^{(2)}y^{(2)} & x_2^{(2)}y^{(2)} \end{bmatrix} \begin{bmatrix} x_1^{(1)}y^{(1)} & x_1^{(2)}y^{(2)} \\ x_2^{(1)}y^{(1)} & x_2^{(2)}y^{(2)} \end{bmatrix}$$

$$H = \begin{bmatrix} x_1^{(1)}x_1^{(1)}y^{(1)}y^{(1)} + x_2^{(1)}x_2^{(1)}y^{(1)}y^{(1)} & x_1^{(1)}x_1^{(2)}y^{(1)}y^{(2)} + x_2^{(1)}x_2^{(2)}y^{(1)}y^{(2)} \\ x_1^{(2)}x_1^{(1)}y^{(2)}y^{(1)} + x_2^{(2)}x_2^{(1)}y^{(2)}y^{(1)} & x_1^{(2)}x_1^{(2)}y^{(2)}y^{(2)} + x_2^{(2)}x_2^{(2)}y^{(2)}y^{(2)} \end{bmatrix}$$

Thus, we get that H is nothing, but the kernel matrix multiplied by np.outer(y,y)

Now, for the case when the data is not fully separable, a penalty parameter C is introduced.

For the softmax margin SVM, recall that the optimization problem can be expressed as

$$\begin{aligned} &\max_{\alpha} \sum_{i}^{m} \alpha_i - \frac{1}{2}\alpha^T \mathbf{H} \alpha \\ s.t.\ \ &0 \leq \alpha_i \leq C \\ &\sum_{i}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

Which can be written in standard form as

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\alpha^T \mathbf{H} \alpha - 1^T \alpha \\ s.t.\ \ &-\alpha_i \leq 0 \\ &\alpha_i \leq C \\ &y^T \alpha = 0 \end{aligned}$$

This is almost the same problem as previously, except for the additional inequality constraint on α. We translate this new constraint into standard form by concatenating below matrix G a diagonal matrix of 1s of size $m \times m$. Similarly for the vector h to which the value of C is added m times.

After this the matrices, P, q, G ,h ,A and b are fed into the convex optimisation package and lagrange's multipliers are obtained. From the multipliers, y_predicted was calculated and if it was > 0 , it was classified as 1 and for less than 0 it was classified as -1

Part 2 of Assignment

In this part, since the number of data points is much greater than the number of features, the kernel that would work the best will be the rbf kernel.

The approach for this part was to firstly scale the data or normalize the features obtained (of both training and test set). Then a Grid Search was done on the training data keeping rbf as the kernel. The value of C and gamma which came out to be optimal from the Grid Search was C = 8.6 and gamma = 0.06.

When these values were tested by making use of cross validation, a training accuracy of = 99.994% and a test accuracy of 97.1399% was obtained.

Thus, the SVM was trained on training set using the above hyperparameters and then test set was run on it.