ELL409 REPORT

ASSIGNMENT 2

SARANSH AGARWAL 2019MT60763

PART 1 A)

BINARY CLASSIFICATION

Using LIBSVM

Using 25 Features

The Following code is used for Analysis for LIBSVM

```
PIPE = Pipeline([('scaler', StandardScaler()), ('SVM', svm.SVC(kernel='linear'))])
parameters = {'SVM__C':np.logspace(0, 1, 10)}
G = GridSearchCV(PIPE, param_grid=parameters, cv=5)

G.fit(text_x, text_t)
print ('Training score',G.score(text_x, text_t))
print ('Cross_validation score',G.score(tp_x, tp_t))
print (G.best_params_)
```

Figure 1: Wrapper Code

CLASSIFIER FOR CVX LIBRARY

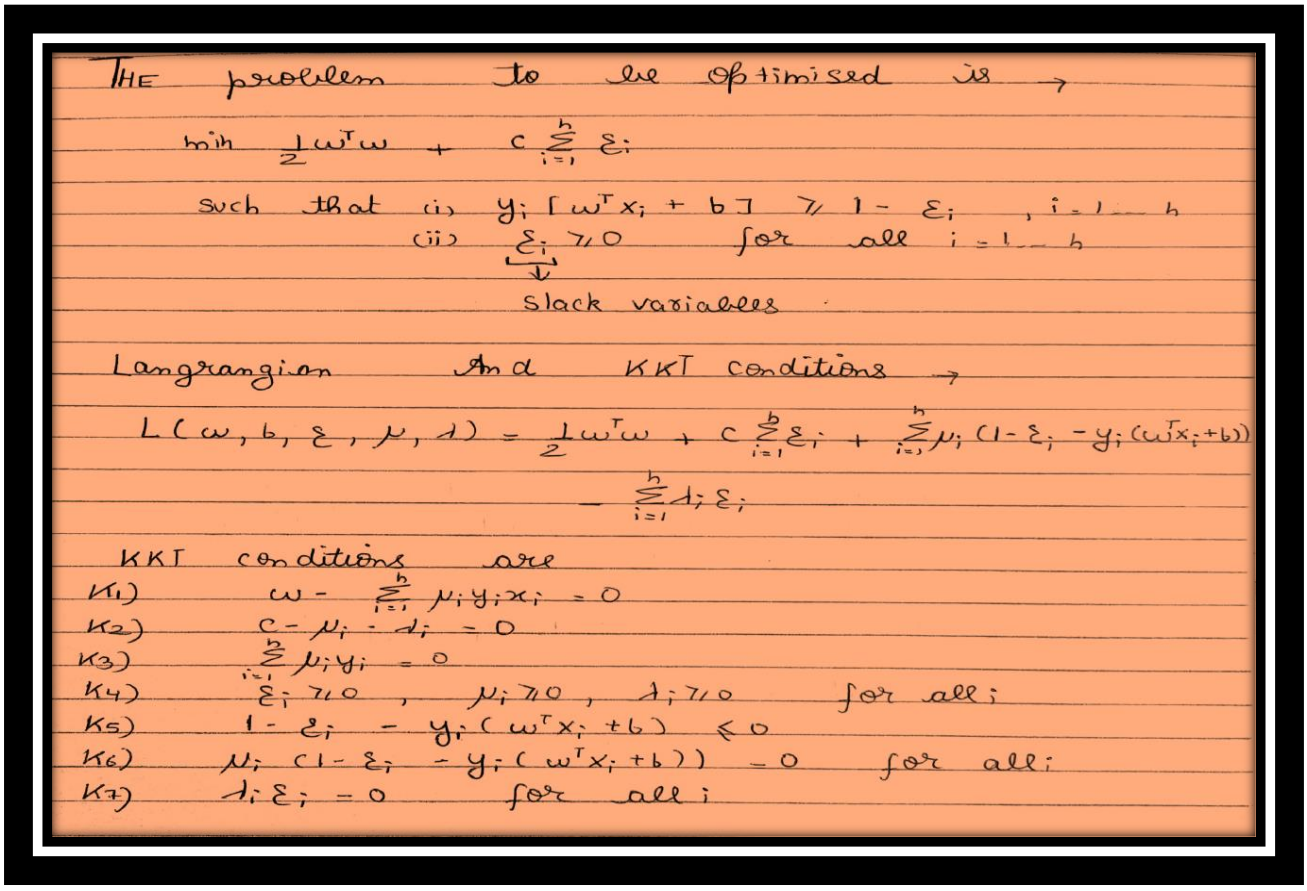

Figure 2: Optimisation problem and Lagrangian conditions

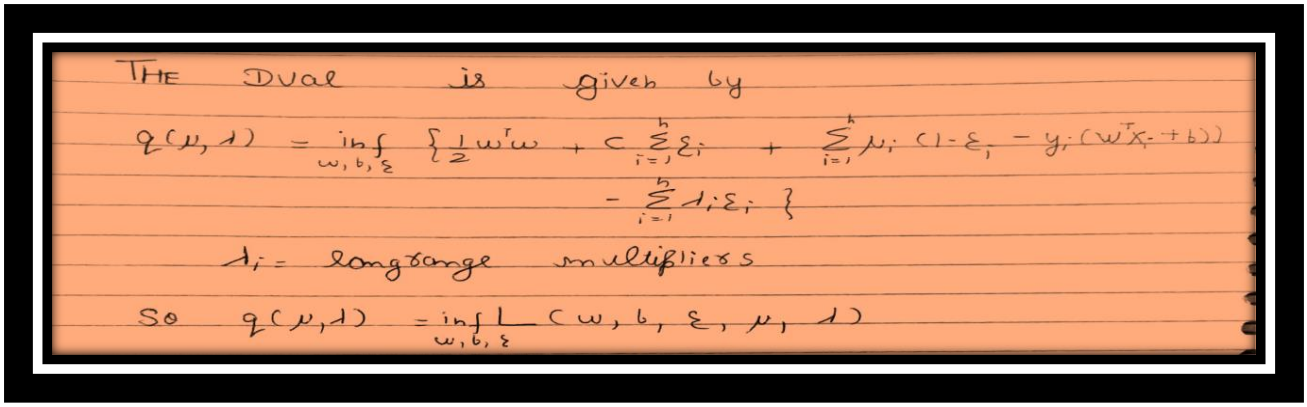

Figure 3: Dual form of optimisation problem

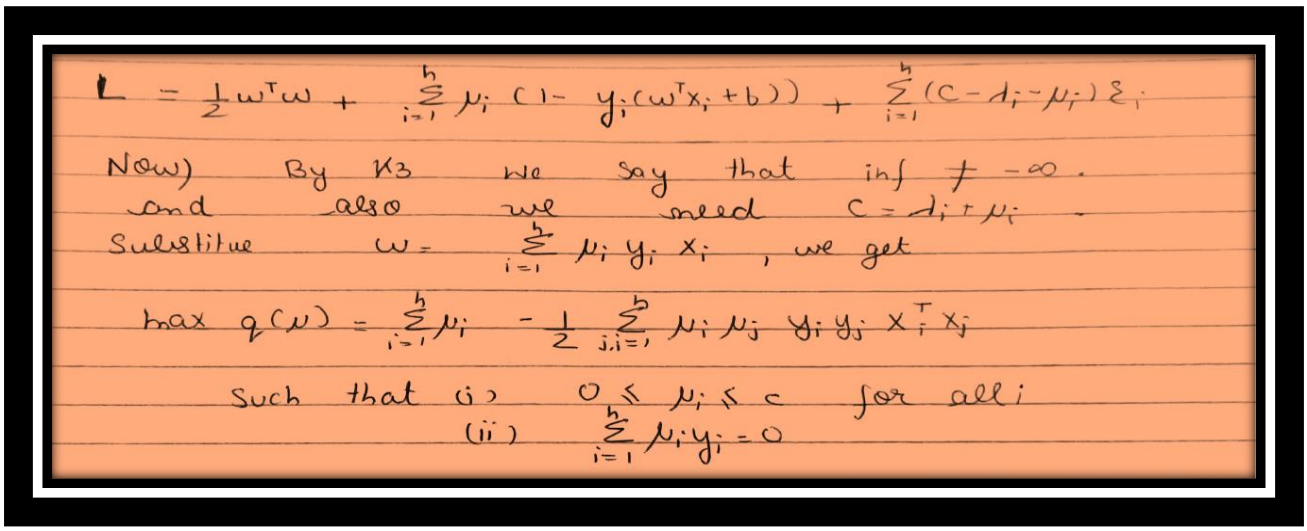

Figure 4: Solving dual using Lagrangian

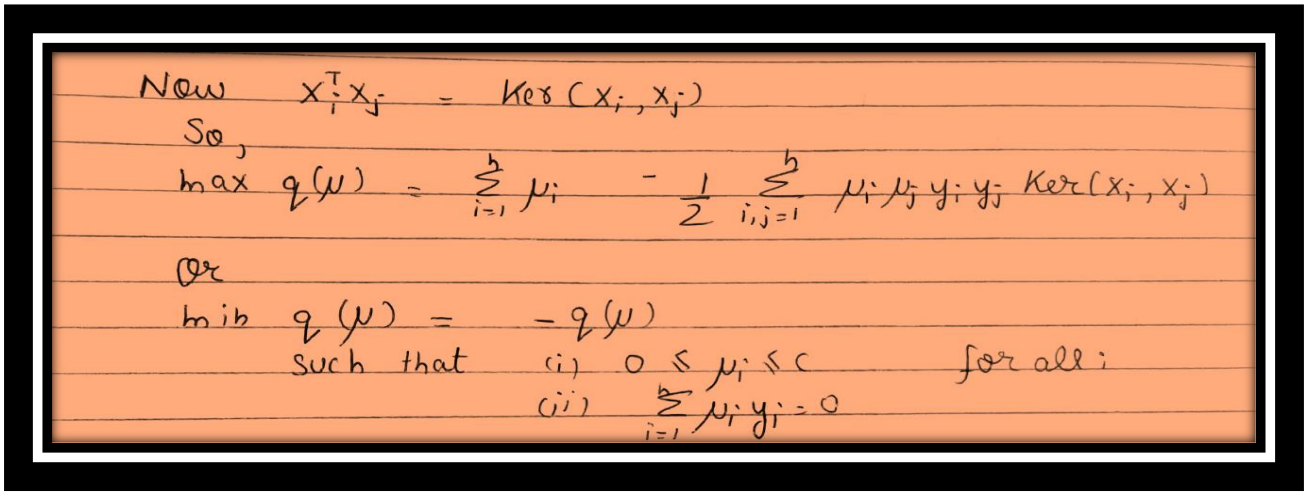

Figure 5: Converting in form to feed in CVX

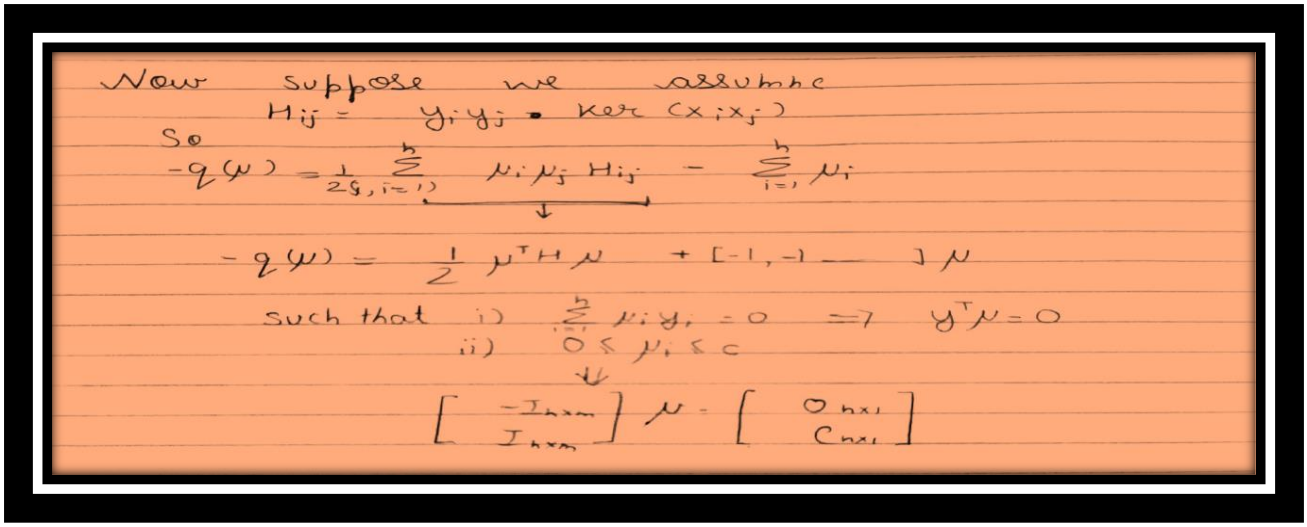

Figure 6: Converting in form to feed in CVX

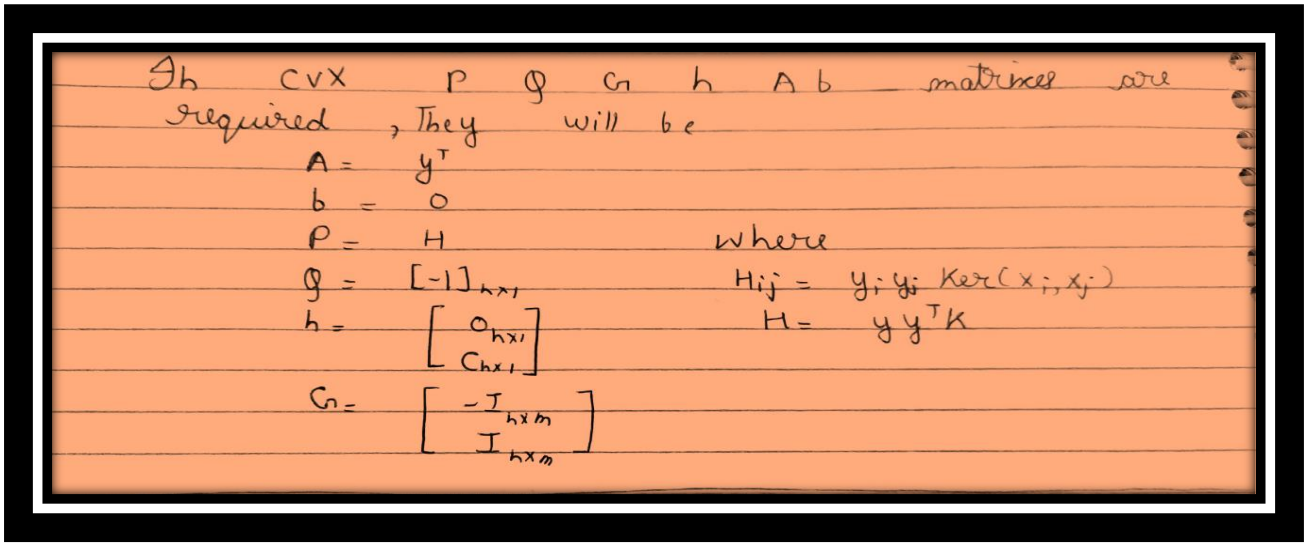

Figure 7: parameters to be pass in CVX

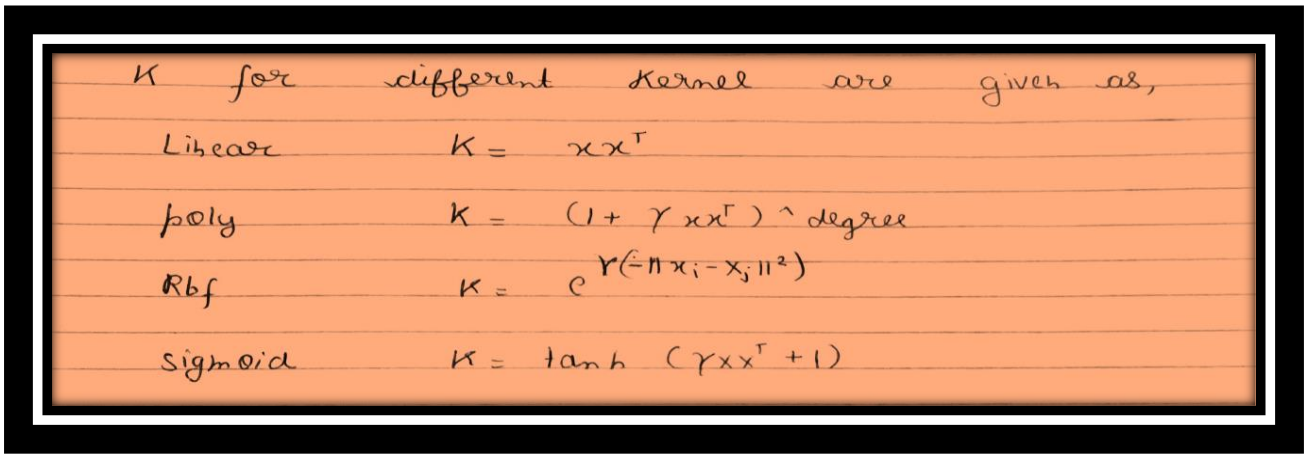

Figure 8: Kernel functions

The Following code is used for Analysis for CVX

```
def compute_K(kernel,X,gamma,degree):
    K = X.dot(np.transpose(X))
    if(kernel == 'poly'):
        K = (gamma*K+1)**degree
    elif(kernel == 'rbf'):
        u = np.diag(X.dot(np.transpose(X))).reshape((-1, 1))*np.ones((1, X.shape[0]))
        K = 2*K-u- np.diag(X.dot(np.transpose(X))).reshape((1, -1))*np.ones((X.shape[0], 1))
        K = np.exp(gamma*K)
    elif(kernel == 'sigmoid'):
        K = np.tanh(gamma*K+1)
    return K

def cvx_fiting(C,X,y,K):
    n = X.shape[0]
    y = y.reshape((-1,1)) * 1.0
    H = ((y.dot(np.transpose(y)))*K)
    Q = cvxopt.matrix(-np.ones((n,1)))
    p = cvxopt.matrix(H)
    G = cvxopt.matrix(np.concatenate((-np.eye(n), np.eye(n))))
    h = cvxopt.matrix(np.append(np.zeros((n,1)),(np.ones((n,1)))*C))
    A = cvxopt.matrix(np.transpose(y))
    b = cvxopt.matrix(0.0)
    cvxopt.solvers.options['show_progress'] = False
    sol=cvxopt.solvers.qp(p,Q,G,h,A,b)
    multipliers = np.array(sol['x'])
    return multipliers
```

Figure 9: Wrapper Code

HELPER FUNCTIONS

```
def get_scores(X,y,w,b):
    p = np.dot(X,w.T)+b
    m = y.shape[0]
    score = 0
    for j in range(m):
        if (p[j] >= 0):
            p[j] = 1
        else :
            p[j] = -1
    for i in range(m):
        if (p[i]*y[i]) > 0 :
            score=score+1
    return score/m

def weights(alpha,X,y):
    m,n = X.shape
    w = np.zeros(n)
    for i in range(X.shape[0]):
        w += alpha[i]*y[i]*X[i,:]
    return w
```

Figure 10: Wrapper Code

FOR CLASSES (0,1)

Linear Kernel

Training score: 1.0
Cross validation score: 0.9912168141592921
Best Hyperparameters: {'SVM__C': 1.0}

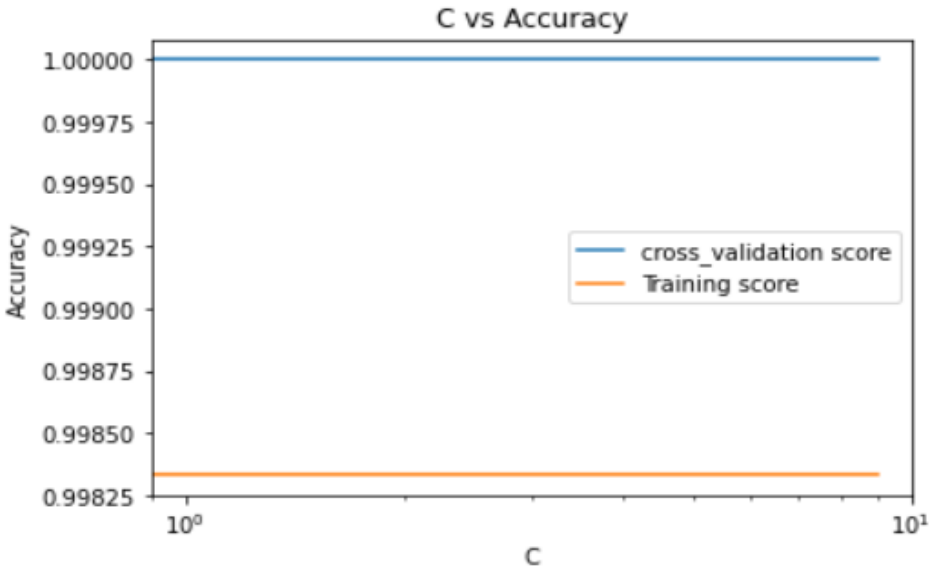

Figure 11: C vs Accuracy (Score) plot for linear kernel

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 31  37  45 117 158 222 243 272 296 312 335 365 391 400 476 582]
```

Training score: 1.0

Cross validation score: 0.9912168141592921

BY CVX

```
[ 31  37  45 117 158 222 243 272 296 312 335 365 391 400 476]
```

Training score: 1.0

Cross validation score: 0.998

RBF Kernel

Training score: 1.0
Cross validation score: 0.8451696165191741
Best Hyperparameters: {'SVM__C': 1.2915496650148839, 'SVM__gamma': 1.0}

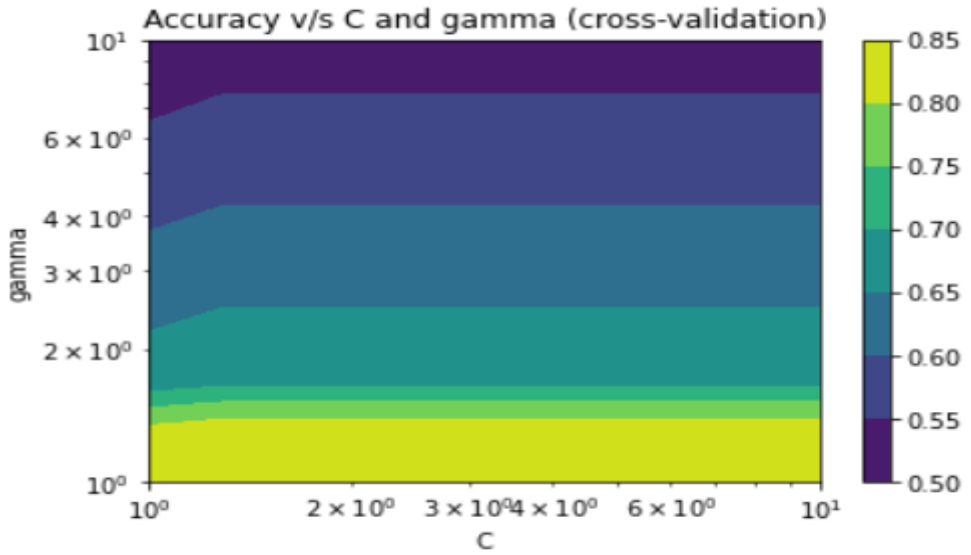

Figure 12: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

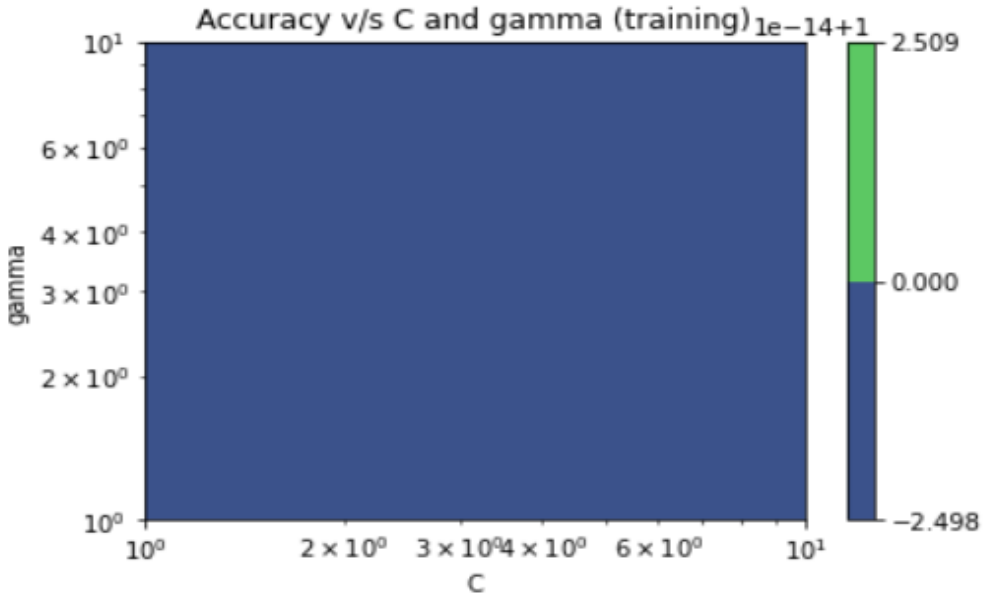

Figure 13: C and gamma vs Accuracy (Score) plot for RBF kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 324
 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
 344 345 346 347 348 349 350 351 352 353 354 356 357 358 359 360 361 362
 364 365 366 367 369 371 372 373 374 375 376 377 378 379 380 381 382 383
 384 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
 403 404 405 406 407 409 410 411 412 415 416 417 418 419 420 421 422 423
 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441
 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
 461 462 463 464 465 466 467 468 469 470 471 472 474 475 476 477 478 479
 481 482 483 484 485 486 488 491 492 493 494 495 496 497 498 500 501 502
 503 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 526 527 528 529 530 531 532 533 534 536 537 538 539 540 541
 542 543 544 545 546 547 548 550 551 552 553 554 555 556 557 558 559 560
 561 562 563 564 566 567 568 569 570 573 574 575 576 577 578 579 580 581
 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599
 600 601 602 603 604 605 606 607 608 609 611 613 614 615 616 617 618 619
 620 621 623 624 625 626 627 628 629 630]
```

Training score: 1.0
Cross validation score: 0.8451696165191741

BY CVX

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 324
 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
 344 345 346 347 348 349 350 351 352 353 354 356 357 358 359 360 361 362
 364 365 366 367 369 371 372 373 374 375 376 377 378 379 380 381 382 383
 384 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402
 403 404 405 406 407 409 410 411 412 415 416 417 418 419 420 421 422 423
 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441
 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478
 479 481 482 483 484 485 486 488 490 491 492 493 494 495 496 497 498 500
 501 502 503 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519
 520 521 522 523 524 526 527 528 529 530 531 532 533 534 536 537 538 539
 540 541 542 543 544 545 546 547 548 550 551 552 553 554 555 556 557 558
 559 560 561 562 563 564 566 567 568 569 570 573 574 575 576 577 578 579
 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597
 598 599 600 601 602 603 604 605 606 607 608 609 611 613 614 615 616 617
 618 619 620 621 623 624 625 626 627 628 629 630]
```

Training score: 0.72
Cross validation score: 0.69

Poly Kernel

Training score: 1.0
Cross validation score: 0.9912168141592921
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.0}

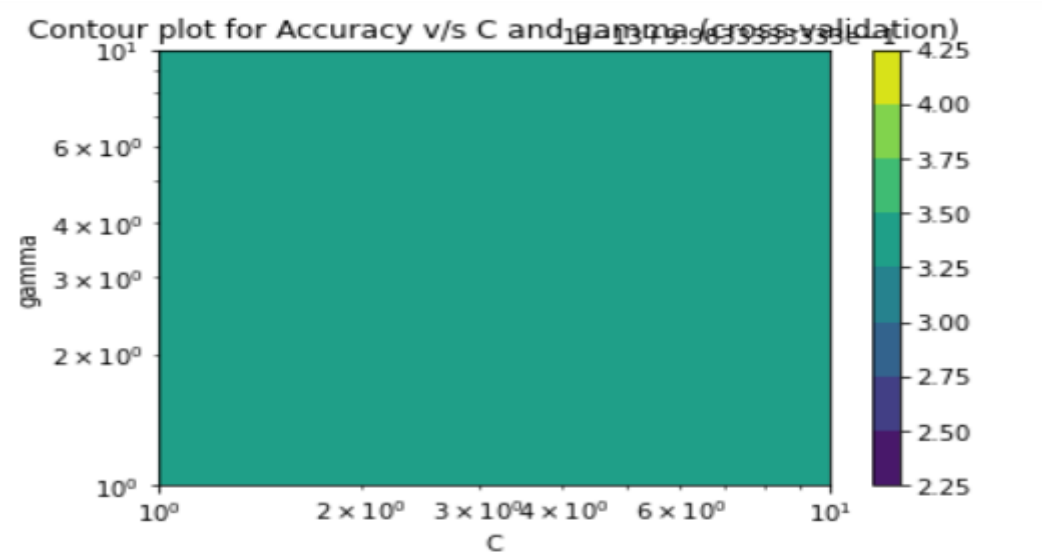

Figure 14: C and gamma vs Accuracy (Score) plot for Poly kernel for cross validation

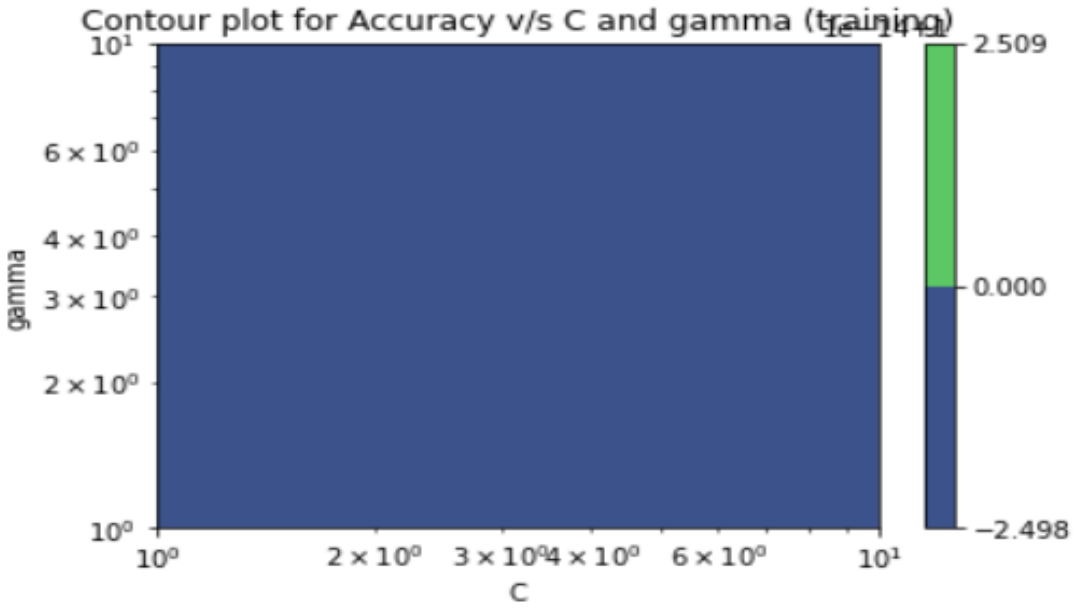

Figure 15: C and gamma vs Accuracy (Score) plot for Poly kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 31  37  45 117 158 222 243 272 296 312 335 365 391 400 476 582]
```

Training score: 1.0
Cross validation score: 0.9912168141592921

BY CVX

```
[ 31  37  45 117 158 222 243 272 296 312 335 365 391 400 476]
```

Training score: 1.0
Cross validation score: 0.998

FOR CLASSES (2,3)

Linear Kernel

Training score: 0.9765886287625418
Cross validation score: 0.9270431472081216
Best Hyperparameters: {'SVM__C': 7.742636826811269}

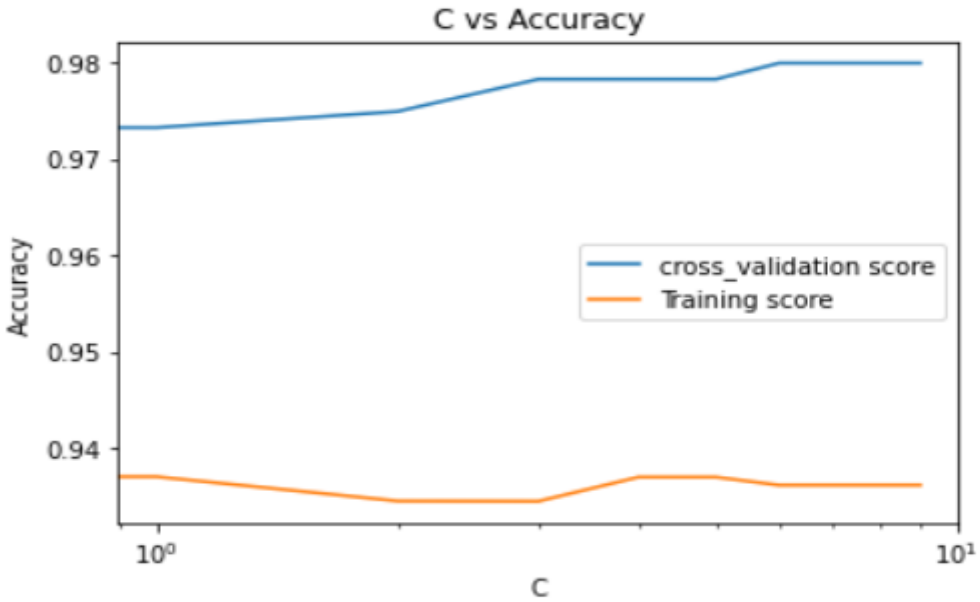

Figure 16: C and gamma vs Accuracy (Score) plot for Linear kernel

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 14  38  46  51  63  88 176 184 191 198 209 223 230 243 247 273 276 279
 282 284 285 286 291 303 315 316 323 331 347 359 360 362 368 383 395 406
 467 473 487 517 523 526 533 550 560 561 566]
```

Training score: 0.9765886287625418
Cross validation score: 0.9270431472081216

BY CSV

```
[ 14  38  46  51  63  88 176 184 191 198 209 223 230 243 247 273 276 279
 282 284 285 286 291 303 315 316 323 331 347 359 360 362 368 383 395 406
 467 473 487 517 523 526 533 550 560 561 566]
```

Training score: 0.90
Cross validation score: 0.92

RBF Kernel

Training score: 1.0
Cross validation score: 0.5215905245346869
Best Hyperparameters: {'SVM__C': 1.2915496650148839, 'SVM__gamma': 1.0}

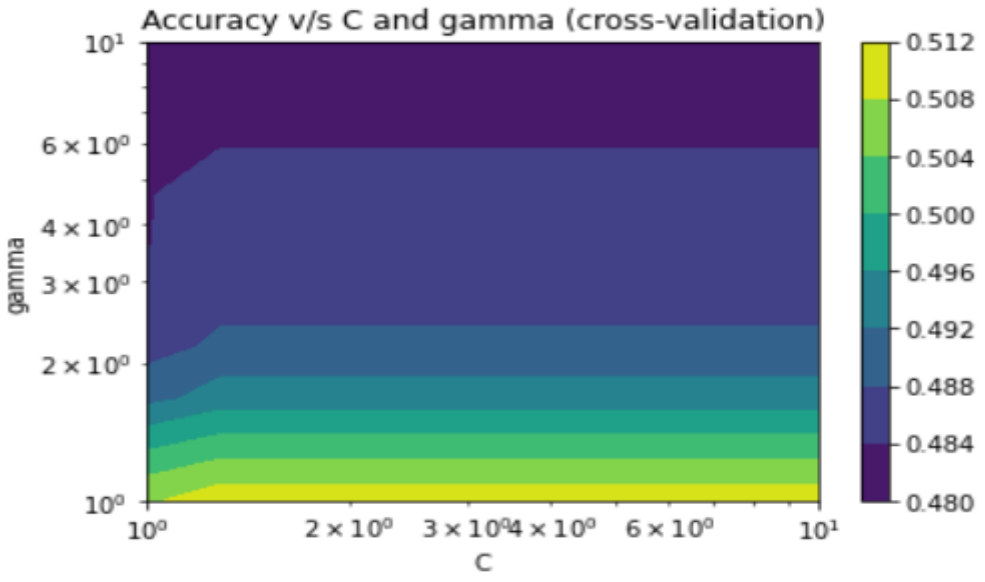

Figure 17: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

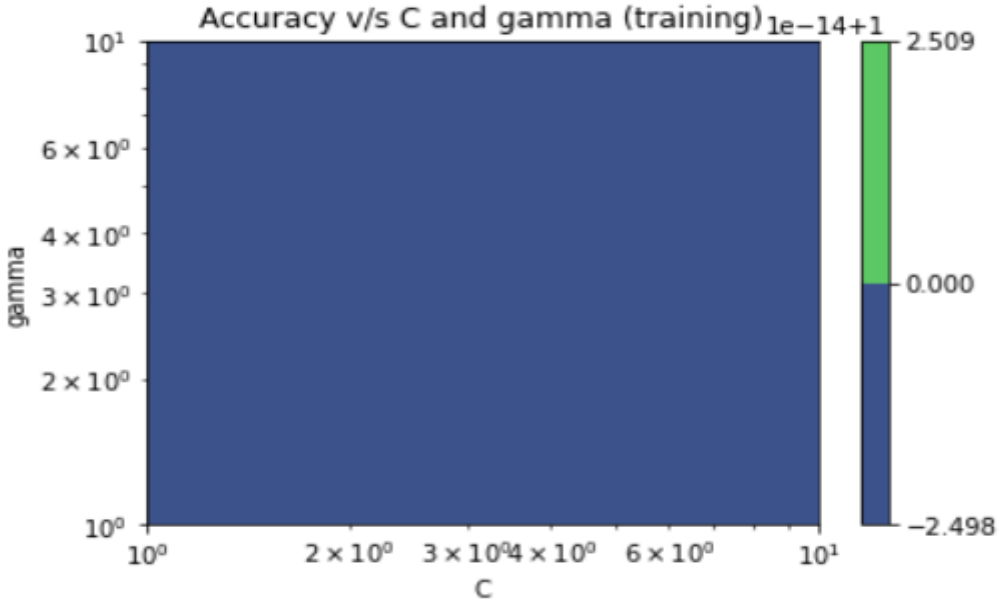

Figure 18: C and gamma vs Accuracy (Score) plot for RBF kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593
 594 595 596 597]
```

Training score: 1.0
Cross validation score: 0.5215905245346869

BY CSV

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593
 594 595 596 597]
```

Training score: 0.89
Cross validation score: 0.68

Poly Kernel

Training score: 1.0
Cross validation score: 0.953849407783418
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 5, 'SVM__gamma': 1.0}

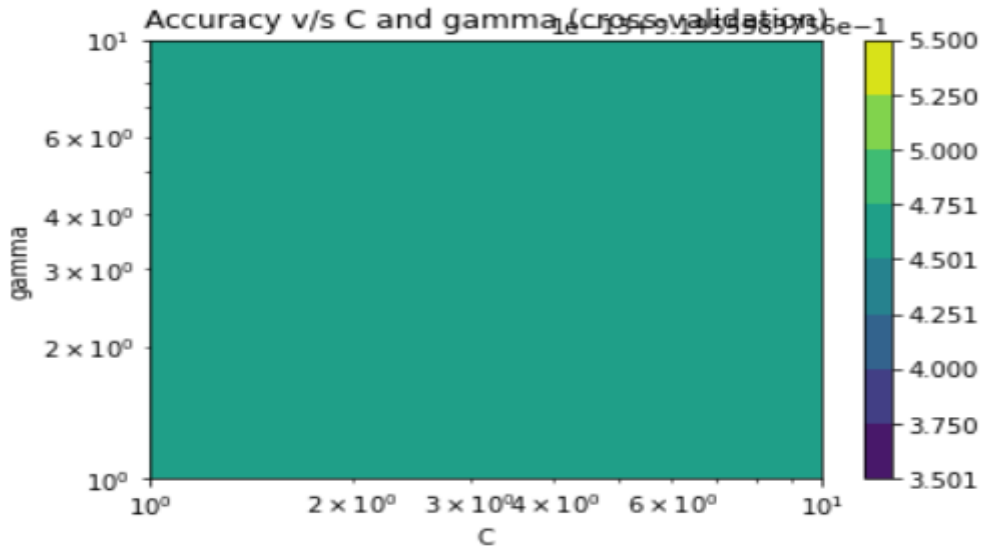

Figure 19: C and gamma vs Accuracy (Score) plot for Poly kernel for cross validation

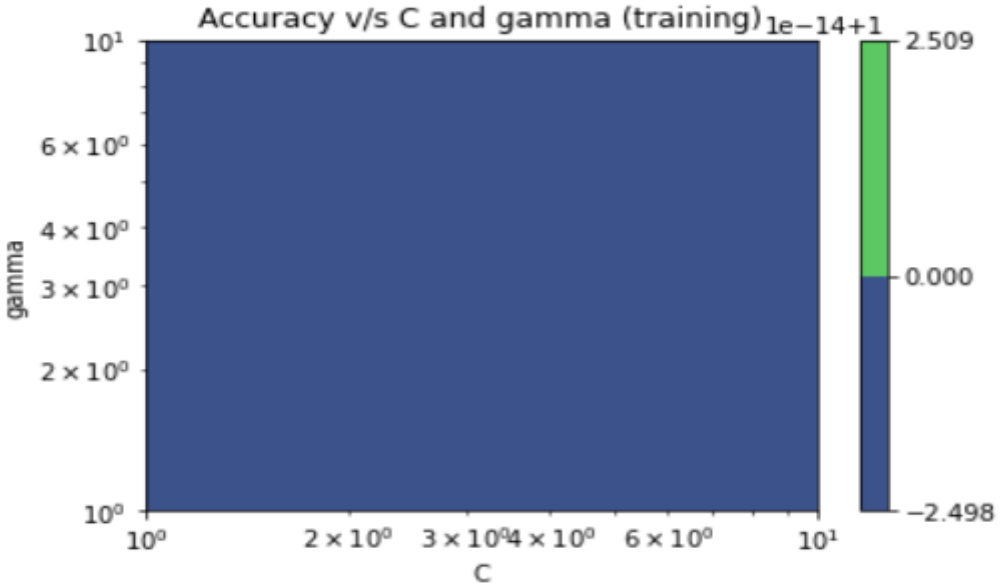

Figure 20: C and gamma vs Accuracy (Score) plot for Poly kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   2   3   8   9  10  14  17  20  23  24  30  36  37  41  43  45  50
  51  55  61  63  65  68  73  75  80  83  84  87  90  91  92 102 103 105
 107 108 111 112 118 119 126 130 135 138 139 143 149 153 156 161 162 163
 164 168 171 172 174 176 178 179 181 182 184 185 188 191 194 198 199 200
 201 202 203 208 209 211 213 216 218 220 221 223 226 227 228 229 230 232
 234 239 240 243 245 247 248 249 250 261 266 273 276 279 282 284 285 286
 287 288 290 291 294 298 299 300 302 303 308 310 311 312 315 316 318 320
 322 323 325 331 334 338 344 351 353 356 358 359 360 362 364 365 366 367
 368 369 370 371 376 380 381 383 385 387 390 394 395 400 403 404 406 417
 421 422 425 435 437 440 444 448 452 453 455 456 459 465 466 467 468 470
 472 473 474 475 479 482 487 488 489 491 493 497 499 508 511 512 516 517
 521 522 523 524 525 526 533 535 536 539 540 541 546 550 551 553 556 561
 566 576 580 581 582 583 586 589 591 596]
```

Training score: 1.0
Cross validation score: 0.953849407783418

BY CSV

```
[  2   3   8  10  14  17  20  23  24  30  36  37  41  43  45  50  51  55
  61  63  65  68  73  75  80  83  84  87  90  91  92 102 103 105 108 111
 112 118 119 126 130 135 138 139 143 149 153 161 162 163 168 172 174 176
 178 179 181 182 184 185 188 191 194 198 199 200 201 202 203 208 209 211
 213 216 218 220 221 223 224 226 227 228 229 230 232 234 239 240 243 245
 247 248 249 250 261 266 273 276 279 282 284 285 286 287 288 291 294 298
 299 300 303 308 310 311 312 315 316 318 320 322 323 325 328 331 334 338
 344 351 353 356 358 359 362 364 365 366 367 368 369 370 371 372 376 380
 381 383 385 387 390 394 400 403 404 406 417 421 422 424 425 435 437 440
 442 444 448 452 453 455 456 459 465 466 467 468 470 472 473 474 475 479
 482 487 488 489 491 493 497 498 499 508 511 512 516 517 521 522 523 524
 525 526 529 533 535 539 540 541 544 546 550 551 556 561 566 576 580 581
 582 583 586 589 591 596]
```

Training score: 0.516
Cross validation score: 0.517

FOR CLASSES (4,5)

Linear Kernel

Training score: 1.0
Cross validation score: 0.9719543522172486
Best Hyperparameters: {'SVM__C': 1.2915496650148839}

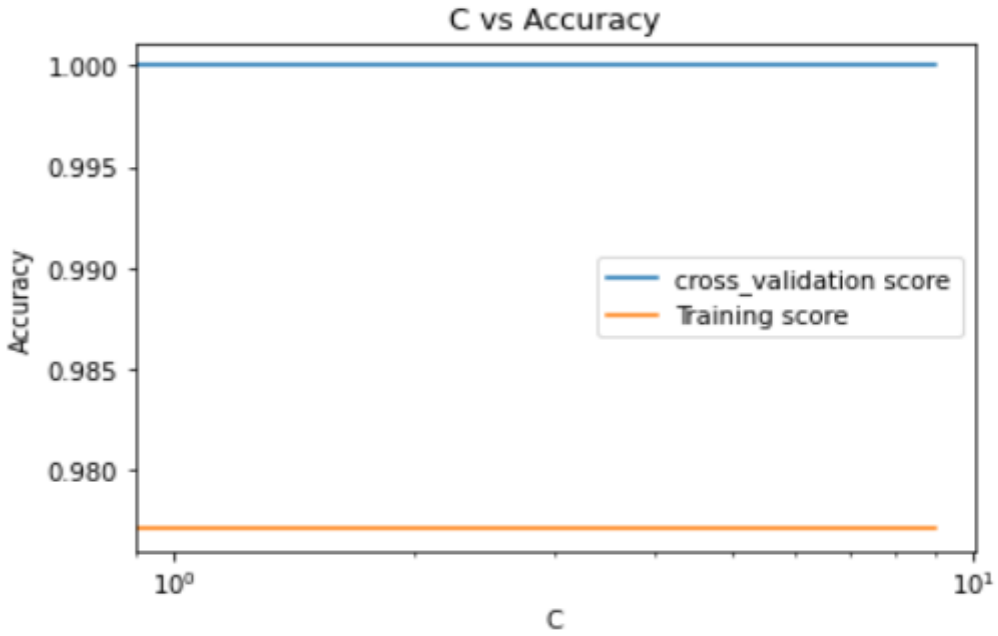

Figure 21: C and gamma vs Accuracy (Score) plot for Linear kernel

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  1  44  57 109 137 152 161 166 182 186 196 197 217 231 236 243 287 329
 330 347 357 378 395 414 433 444 477 481 529 532 544]
```

Training score: 1.0
Cross validation score: 0.9719543522172486

BY CSV

```
[  1  44  57 109 137 152 161 166 182 186 196 197 217 231 236 243 287 329
 330 347 357 378 395 414 433 444 477 481 529 532 544]
```

Training score: 1.0
Cross validation score: 0.984

RBF Kernel

Training score: 1.0
Cross validation score: 0.4958123953098827
Best Hyperparameters: {'SVM__C': 1.2915496650148839, 'SVM__gamma': 1.0}

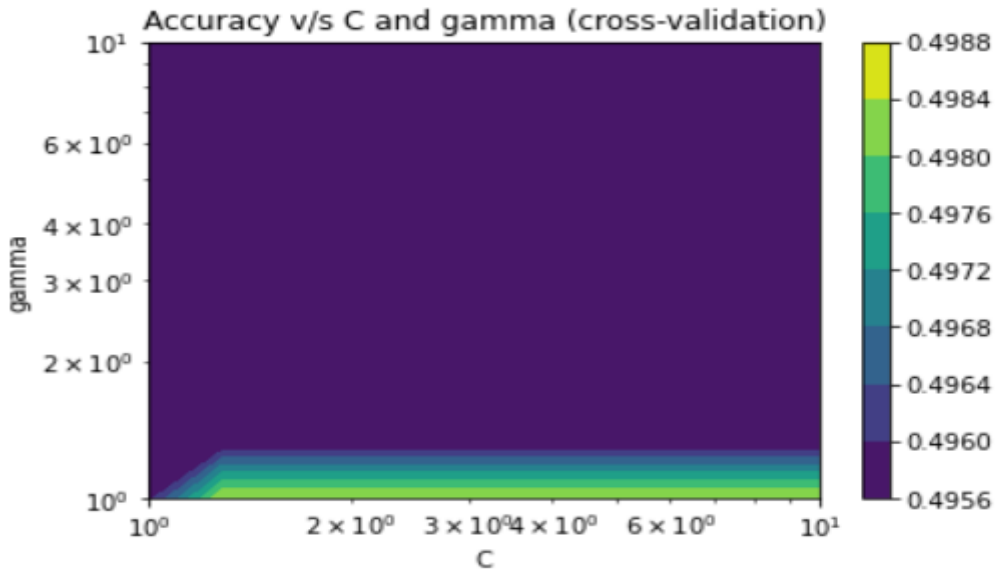

Figure 22: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

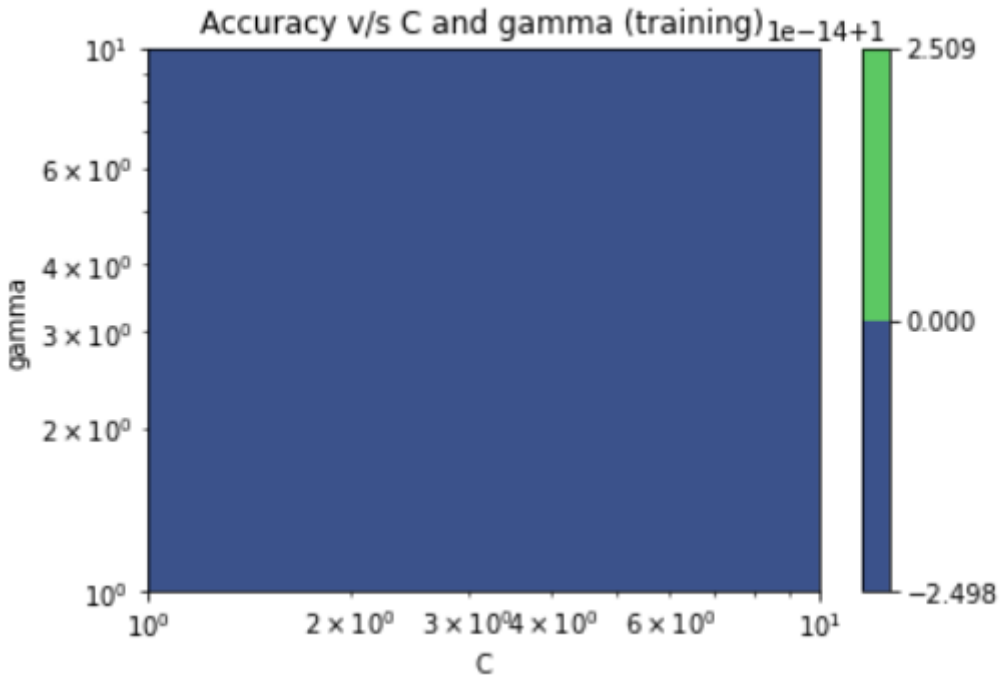

Figure 23: C and gamma vs Accuracy (Score) plot for RBF kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567]
```

Training score: 1.0
Cross validation score: 0.4958123953098827

BY CSV

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567]
```

Training score: 0.62
Cross validation score: 0.68

Poly Kernel

Training score: 1.0
Cross validation score: 0.9719543522172486
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.29}

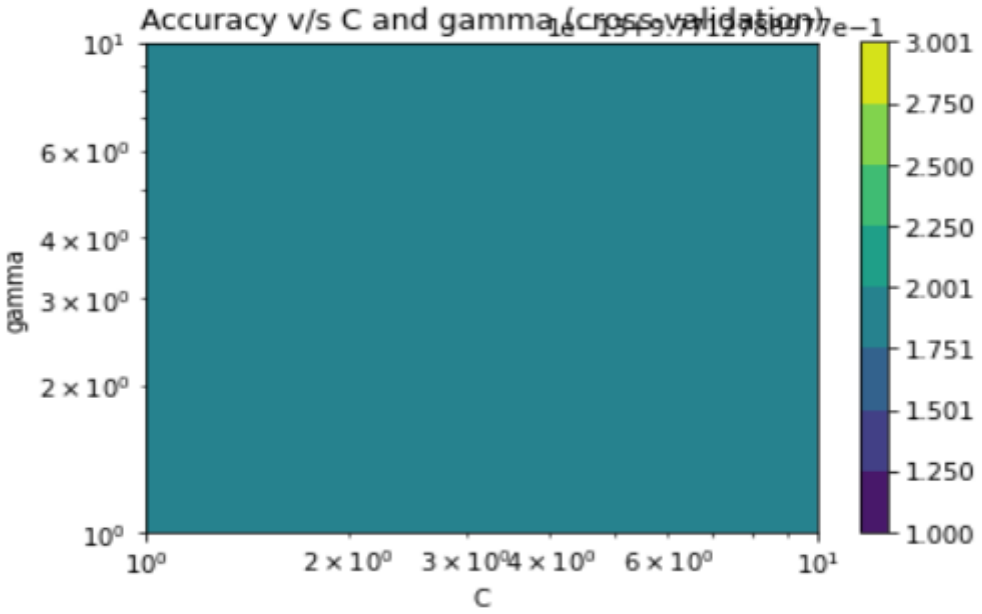

Figure 24: C and gamma vs Accuracy (Score) plot for Poly kernel for cross validation

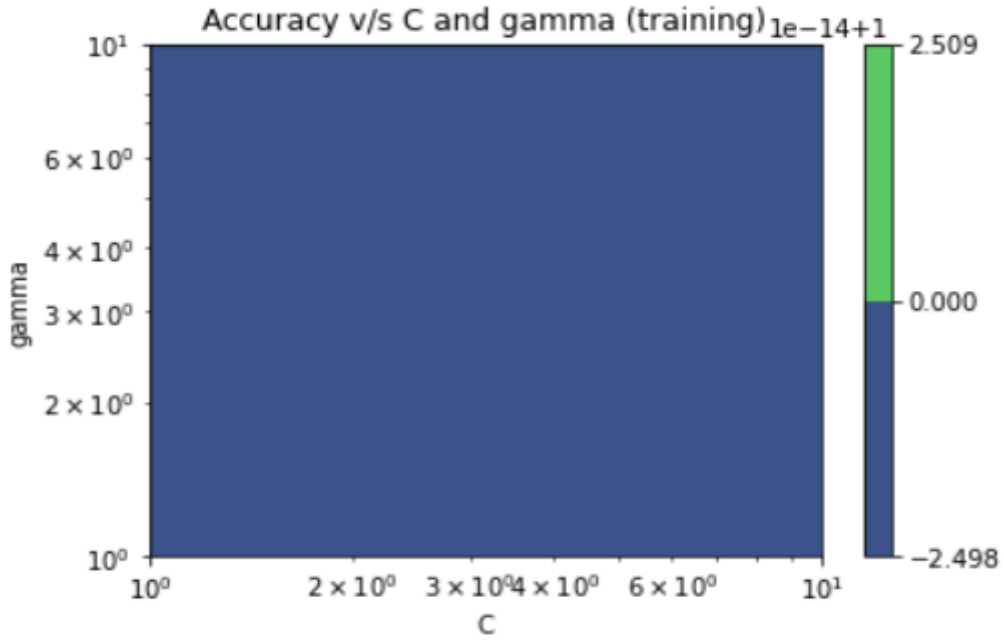

Figure 25: C and gamma vs Accuracy (Score) plot for Poly kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  1  44  57 109 137 152 161 166 182 186 196 197 217 231 236 243 287 329
 330 347 357 378 395 414 433 444 477 481 529 532 544]
```

Training score: 1.0
Cross validation score: 0.9719543522172486

BY CSV

```
[  1  44  57 109 137 152 161 166 182 186 196 197 217 231 236 243 287 329
 330 347 357 378 395 414 433 444 477 481 529 532 544]
```

Training score: 1.0
Cross validation score: 0.9717

Using 10 Features

FOR CLASSES (0,1)

Linear Kernel

Training score: 1.0
Cross validation score: 0.9975958702064897
Best Hyperparameters: {'SVM__C': 1.0}

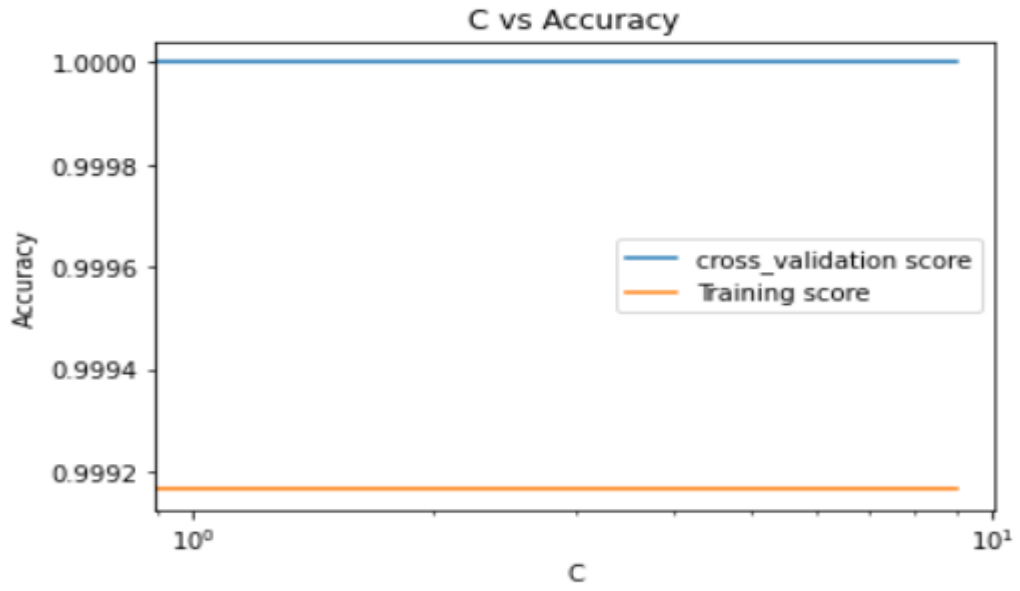

Figure 26: C and gamma vs Accuracy (Score) plot for Linear kernel

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 37  45 117 222 391 400 427 476 582]
```

Training score: 1.0
Cross validation score: 0.9975958702064897

BY CSV

```
[ 37  45 117 222 391 400 427 476 582]
```

Training score: 1.0
Cross validation score: 0.999

RBF Kernel

Training score: 1.0
Cross validation score: 0.9577507374631269
Best Hyperparameters: {'SVM__C': 1.6681005372000588, 'SVM__gamma': 1.0}

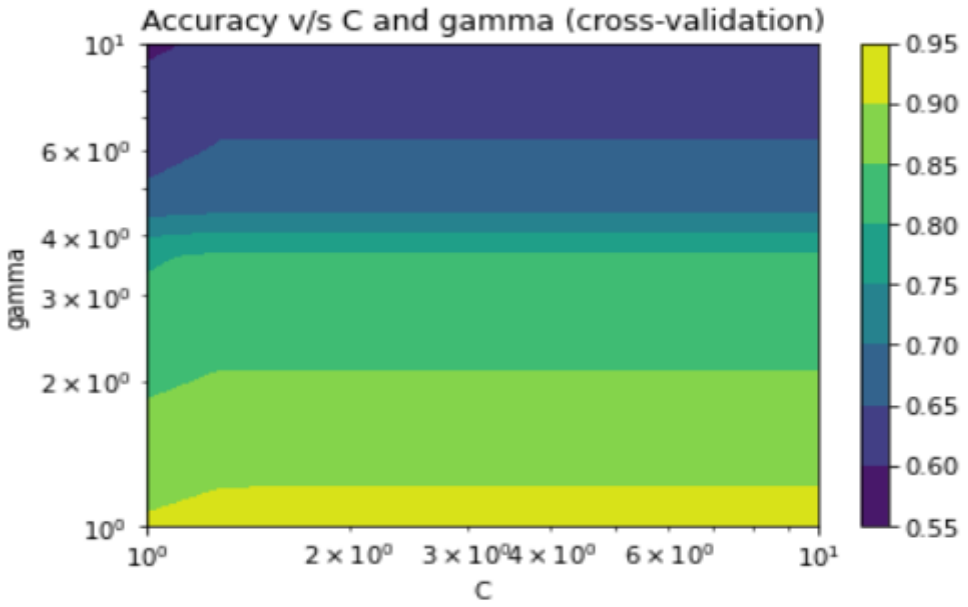

Figure 27: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

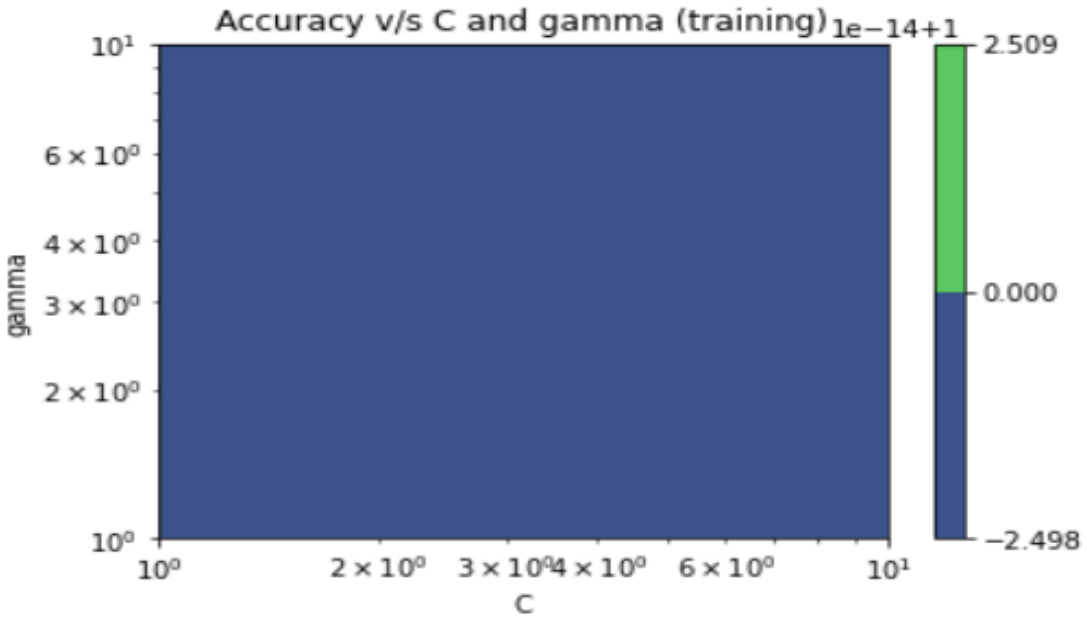

Figure 28: C and gamma vs Accuracy (Score) plot for RBF kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 312 313 314 315 317 318 319 320 321 322 324 325 326 327 328
 329 332 334 335 337 338 339 340 341 344 346 347 348 349 350 351 354 357
 360 361 362 365 367 369 371 372 373 374 375 376 377 378 379 380 381 383
 384 386 388 389 390 391 392 393 394 396 400 401 402 403 405 406 407 409
 410 411 412 415 417 420 423 424 425 426 427 428 430 431 433 434 435 436
 437 438 439 440 441 444 445 446 449 451 453 454 456 458 459 460 462 464
 466 467 469 470 471 473 474 475 476 477 478 479 483 485 492 493 494 497
 498 500 501 502 505 506 507 508 509 510 511 512 515 516 517 518 521 522
 523 526 532 533 534 537 538 539 540 541 544 546 547 550 551 552 553 554
 555 557 558 560 561 562 563 564 566 567 568 569 570 573 574 577 578 580
 582 585 586 588 589 591 592 593 594 595 596 599 600 601 602 603 604 605
 614 615 618 620 621 626 629]
```

Training score: 1.0
Cross validation score: 0.9577507374631269

BY CSV

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 312 313 314 315 317 318 319 320 321 322 324 325 326 327 328
 329 332 334 335 337 338 339 340 341 344 346 347 348 349 350 351 354 356
 357 359 360 361 362 365 366 367 369 371 372 373 374 375 376 377 378 379
 380 381 383 384 386 388 389 390 391 392 393 394 396 400 401 402 403 405
 406 407 409 410 411 412 415 417 420 423 424 425 426 427 428 430 431 433
 434 435 436 437 438 439 440 441 444 445 446 449 451 453 454 456 458 459
 460 462 464 466 467 469 470 471 473 474 475 476 477 478 479 483 485 492
 493 494 497 498 500 501 502 505 506 507 508 509 510 511 512 515 516 517
 518 521 522 523 526 532 533 534 537 538 539 540 541 544 546 547 550 551
 552 553 554 555 557 558 560 561 562 563 564 566 567 568 569 570 573 574
 577 578 580 582 585 586 588 589 591 592 593 594 595 596 599 600 601 602
 603 604 605 614 615 618 620 621 626 629]
```

Training score: 0.74
Cross validation score: 0.71

Poly Kernel

Training score: 1.0
Cross validation score: 0.9975958702064897
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.0}

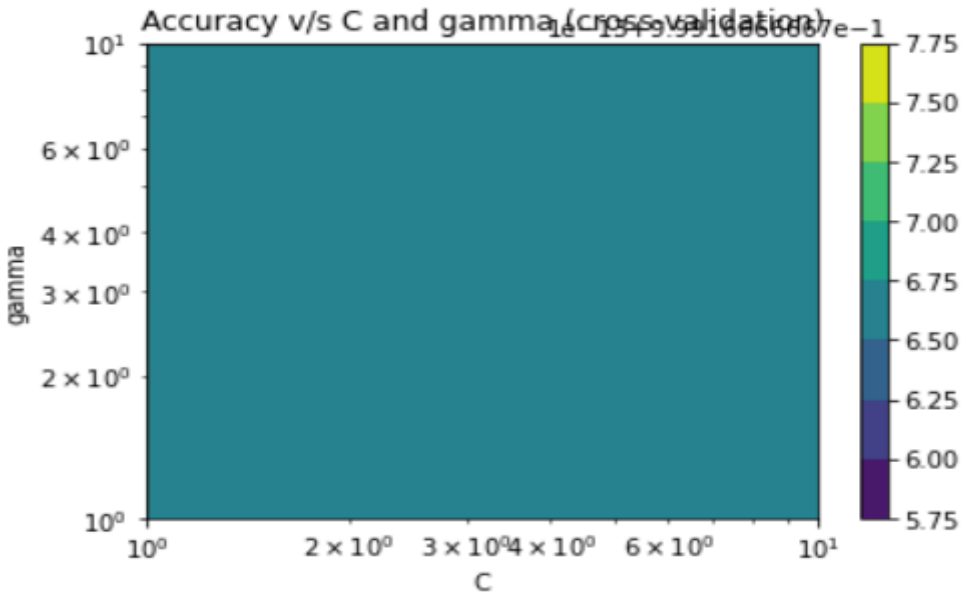

Figure 29: C and gamma vs Accuracy (Score) plot for Poly kernel for cross validation

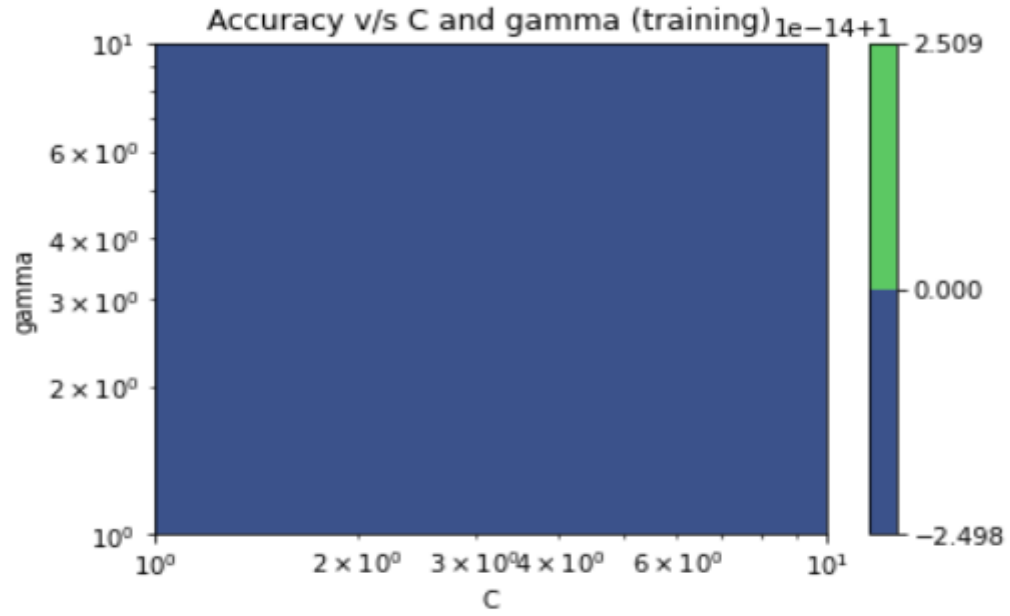

Figure 30: C and gamma vs Accuracy (Score) plot for Poly kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 37  45 117 222 391 400 427 476 582]
```

Training score: 1.0
Cross validation score: 0.9975958702064897

BY CSV

```
[ 37  45 117 222 391 400 427 476 582]
```

Training score: 1.0
Cross validation score: 0.999

FOR CLASSES (2,3)

Linear Kernel

Training score: 0.9632107023411371
Cross validation score: 0.9472208121827411
Best Hyperparameters: {'SVM__C': 1.0}

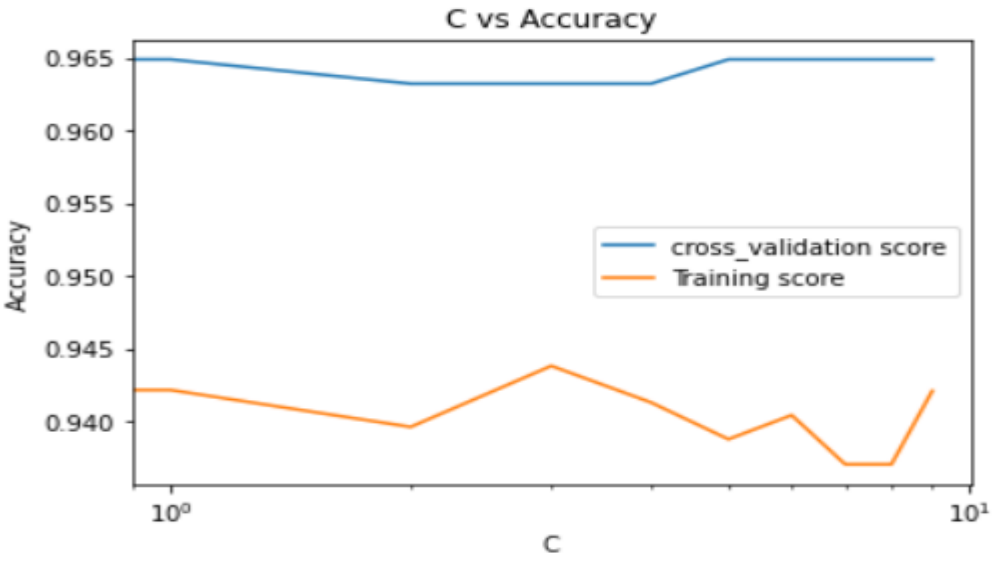

Figure 31: C and gamma vs Accuracy (Score) plot for Linear kernel

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 14  38  46  51  63 102 119 121 139 153 162 176 184 191 198 209 211 223
 230 234 247 248 273 276 279 282 285 286 287 290 291 294 308 315 316 323
 331 347 353 359 360 362 367 370 376 380 389 395 400 406 421 437 467 468
 473 487 523 526 541 550 553 560 561 572 589]
```

Training score: 0.9632107023411371
Cross validation score: 0.9472208121827411

BY CSV

```
[ 14  38  46  51  63 102 119 121 139 153 162 176 184 191 198 209 211 223
 230 234 247 248 273 276 279 282 285 286 287 290 291 294 308 315 316 323
 331 347 353 359 360 362 367 370 376 380 389 395 400 406 421 437 467 468
 473 487 523 526 541 550 553 560 561 572 589]
```

Training score: 0.918
Cross validation score: 0.910

RBF Kernel

Training score: 1.0
Cross validation score: 0.8393020304568529
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__gamma': 1.0}

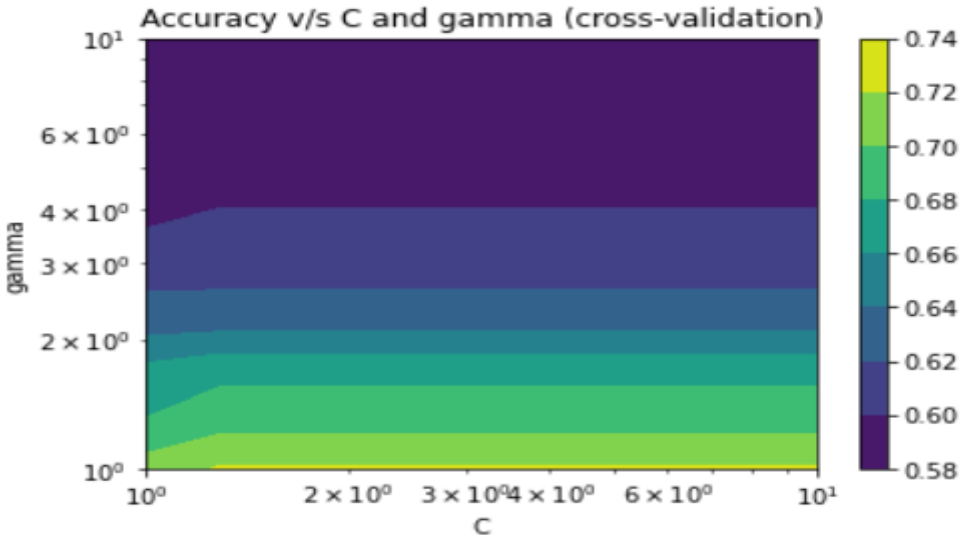

Figure 32: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

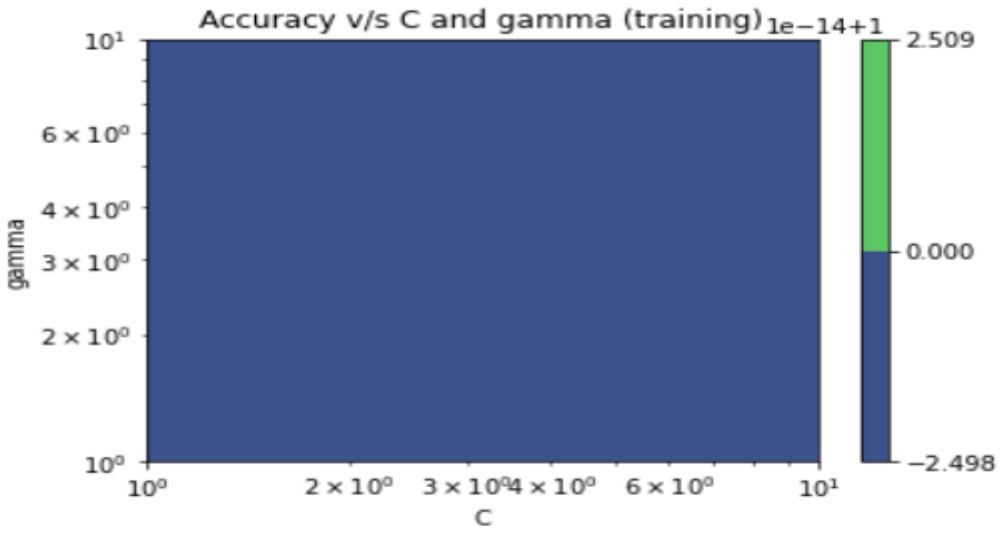

Figure 33: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593
 594 595 596 597]
```

Training score: 1.0
Cross validation score: 0.8393020304568529

BY CSV

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593
 594 595 596 597]
```

Training score: 0.92
Cross validation score: 0.70

Poly Kernel

Training score: 0.9632107023411371
Cross validation score: 0.9472208121827411
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.0}

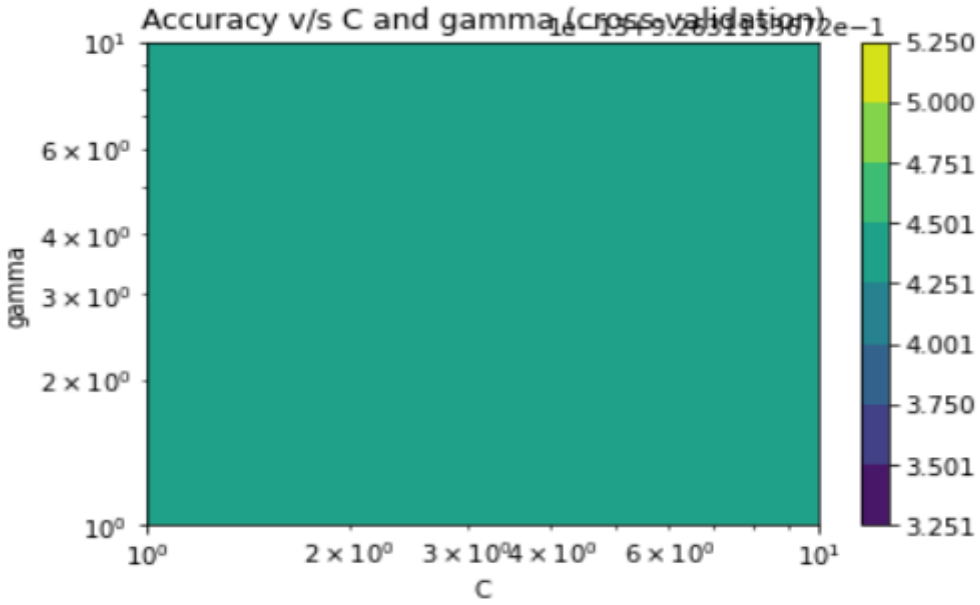

Figure 34: C and gamma vs Accuracy (Score) plot for Poly kernel for cross validation

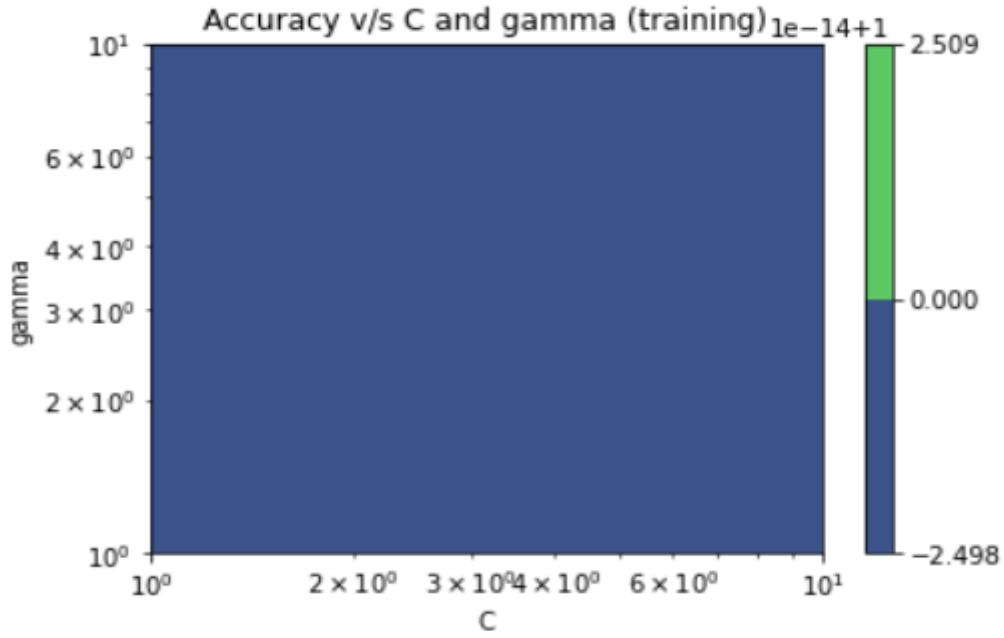

Figure 35: C and gamma vs Accuracy (Score) plot for Poly kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 14  38  46  51  63 102 119 121 139 153 162 176 184 191 198 209 211 223
 230 234 247 248 273 276 279 282 285 286 287 290 291 294 308 315 316 323
 331 347 353 359 360 362 367 370 376 380 389 395 400 406 421 437 467 468
 473 487 523 526 541 550 553 560 561 572 589]
```

Training score: 0.9632107023411371
Cross validation score: 0.9472208121827411

BY CSV

```
[ 14  38  46  51  63 102 119 121 139 153 162 176 184 191 198 209 211 223
 230 234 247 248 273 276 279 282 285 286 287 290 291 294 308 315 316 323
 331 347 353 359 360 362 367 370 376 380 389 395 400 406 421 437 467 468
 473 487 523 526 541 550 553 560 561 572 589]
```

Training score: 0.918
Cross validation score: 0.925

FOR CLASSES (4,5)

Linear Kernel

Training score: 0.9788732394366197
Cross validation score: 0.96693395834582
Best Hyperparameters: {'SVM__C': 1.0}

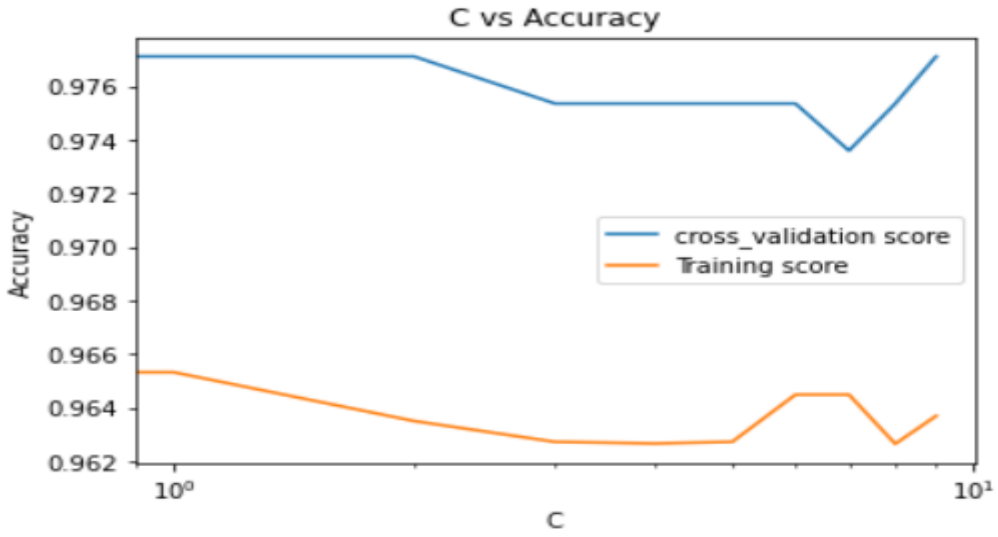

Figure 36: C and gamma vs Accuracy (Score) plot for Linear kernel

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 26  41  44  55  57 109 137 152 161 163 165 166 180 182 196 217 231 236
 243 262 269 287 291 302 320 328 330 348 354 357 372 378 382 384 388 413
 414 415 418 442 444 459 514 529 547 557]
```

Training score: 0.9788732394366197
Cross validation score: 0.96693395834582

BY CSV

```
[ 26  41  44  55  57 109 137 152 161 163 165 166 180 182 196 217 231 236
 243 262 269 287 291 302 320 328 330 348 354 357 372 378 382 384 388 413
 414 415 418 442 444 459 514 529 547 557]
```

Training score: 0.9788732394366197
Cross validation score: 0.968

RBF Kernel

Training score: 1.0
Cross validation score: 0.8356346945621
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__gamma': 1.0}

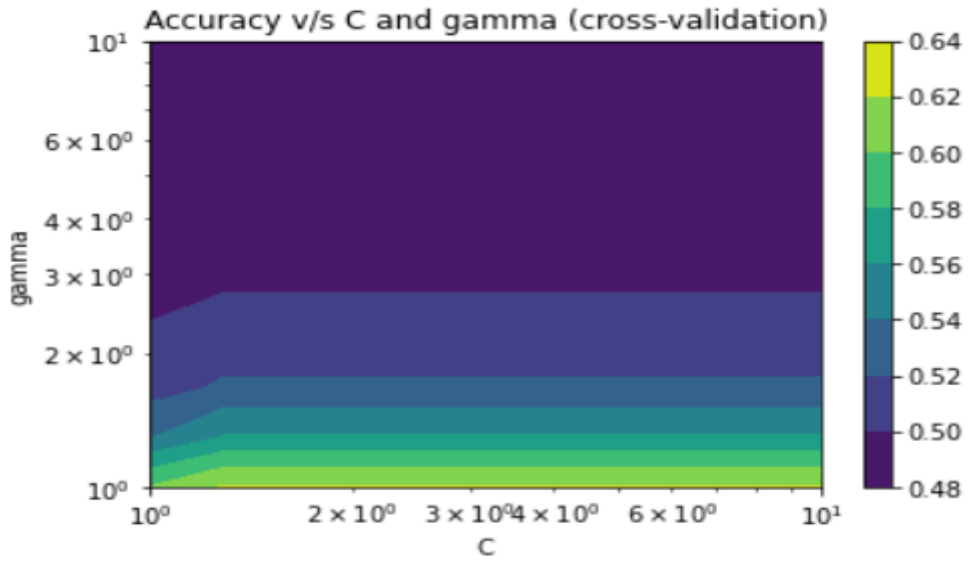

Figure 37: C and gamma vs Accuracy (Score) plot for RBF kernel for cross validation

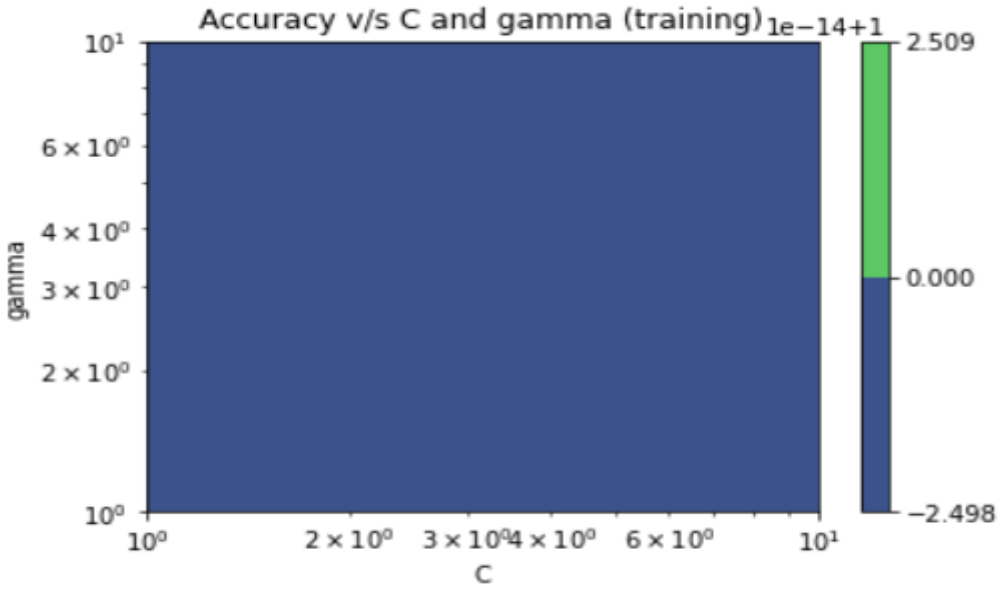

Figure 38: C and gamma vs Accuracy (Score) plot for RBF kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567]
```

Training score: 1.0
Cross validation score: 0.8356346945621

BY CSV

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567]
```

Training score: 0.64
Cross validation score: 0.67

Poly Kernel

Training score: 0.9788732394366197
Cross validation score: 0.96693395834582
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.0}

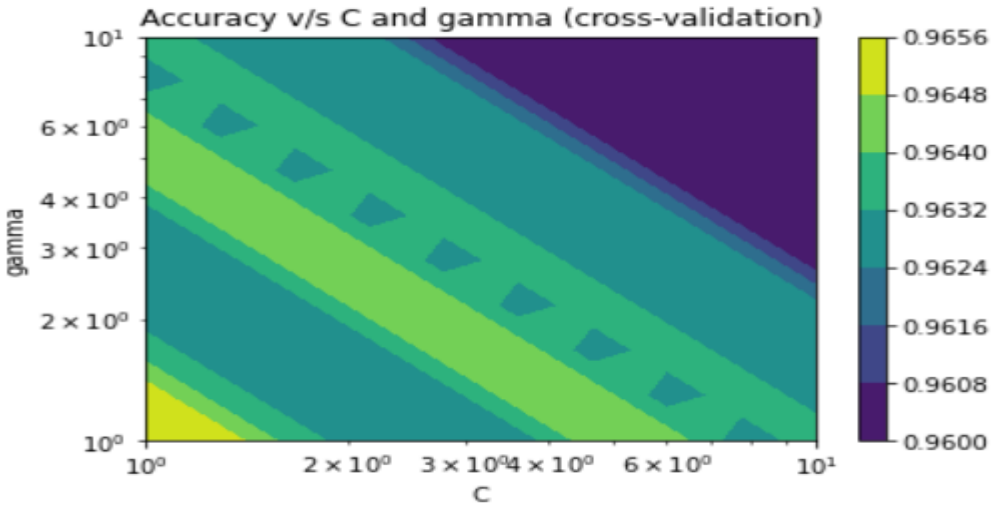

Figure 39: C and gamma vs Accuracy (Score) plot for Poly kernel for cross validation

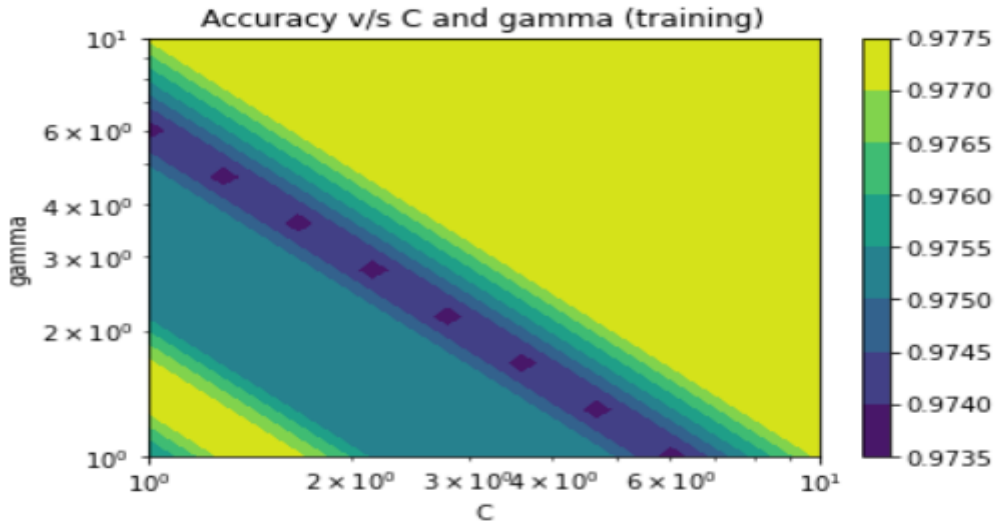

Figure 40: C and gamma vs Accuracy (Score) plot for Poly kernel for Training

COMPARISON BETWEEN CVX AND LIBSVM

SUPPORT VECTORS OBTAINED FROM TWO METHODS ARE AS FOLLOWS

BY LIBSVM

```
[ 26  41  44  55  57 109 137 152 161 163 165 166 180 182 196 217 231 236
 243 262 269 287 291 302 320 328 330 348 354 357 372 378 382 384 388 413
 414 415 418 442 444 459 514 529 547 557]
```

Training score: 0.9788732394366197
Cross validation score: 0.96693395834582

BY CSV

```
[ 26  41  44  55  57 109 137 152 161 163 165 166 180 182 196 217 231 236
 243 262 269 287 291 302 320 328 330 348 354 357 372 378 382 384 388 413
 414 415 418 442 444 459 514 529 547 557]
```

Training score: 0.97
Cross validation score: 0.93

COMPARISON FOR VARIOUS KERNEL FOR 10 FEATURES

Class 1	Class 2	KERNEL	TRAINING SCORE	CROSS VALIDATION SCORE	BEST C	BEST GAMMA
0	1	Linear	1	0.99	1	-
0	1	RBF	1	0.95	1.66	1
0	1	Poly	1	0.99	1	1
2	3	Linear	0.96	0.94	1	-
2	3	RBF	0.83	0.83	1	1
2	3	Poly	0.96	0.94	1	1
4	5	Linear	0.97	0.96	1	-
4	5	RBF	1	0.83	1	1
4	5	Poly	0.97	0.96	1	1

Figure 41: Comparison for various kernels

COMPARISON FOR VARIOUS KERNEL FOR 25 FEATURES

Class 1	Class 2	KERNEL	TRAINING SCORE	CROSS VALIDATION SCORE	BEST C	BEST GAMMA
0	1	Linear	1	0.99	1	-
0	1	RBF	1	0.84	1.29	1
0	1	Poly	1	0.99	1	1
2	3	Linear	0.97	0.92	7.74	-
2	3	RBF	1	0.52	1.29	1
2	3	Poly	1	0.95	1	1
4	5	Linear	1	0.97	1.29	-
4	5	RBF	1	0.49	1.29	1
4	5	Poly	1	0.97	1	1.29

Figure 42: Comparison for various kernels

CONCLUSION

Effect of increasing No of features

The above tables show the values of training error, cross validation error and values of best hyperparameters obtained using LIBSVM for different kernel functions

We can observe that training and cross validation scores are higher when we are using 25 features than 10 features. As number of features increases it provides a greater number of dimensions, so data of different classes can be used for better classification of data – points

Trends of Best Hyperparameters Upon changing pair of class

We can observe that when we are using 10 features the values of best hyperparameters remains almost same for different pair of classes, it doesn't vary much upon changing pair of classes.

While when we are using 25 features the values of best hyperparameters varies a little with the change in pair of classes.

OVERFITTING, UNDERFITTING AND GOOD FIT

Linear KERNEL using 25 features and classes = {0,1}

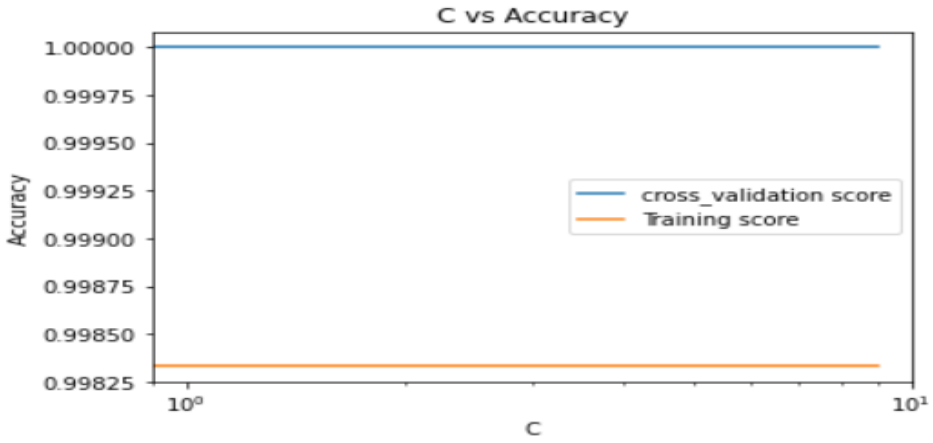

Figure 43: Variations of accuracy Vs C

RBF KERNEL using 25 features and classes = {0,1}

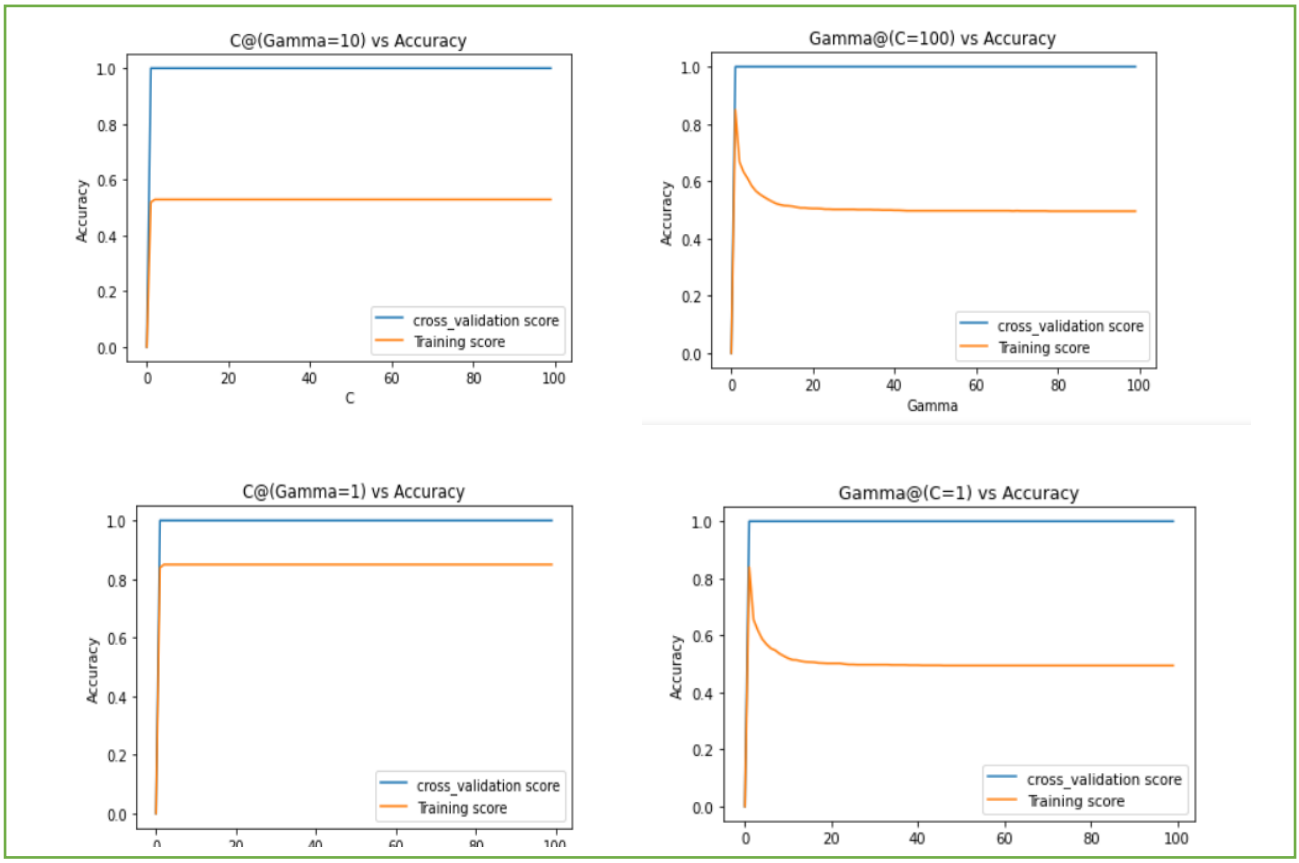

Figure 44: Variations of accuracy Vs C and Gamma

From the above graphs we can observe that low gamma and low C leads to underfittings then we get a sweet spot that is good fit then as we increase gamma and C the model starts to overfits.

POLY KERNEL using 25 features and classes = {0,1}

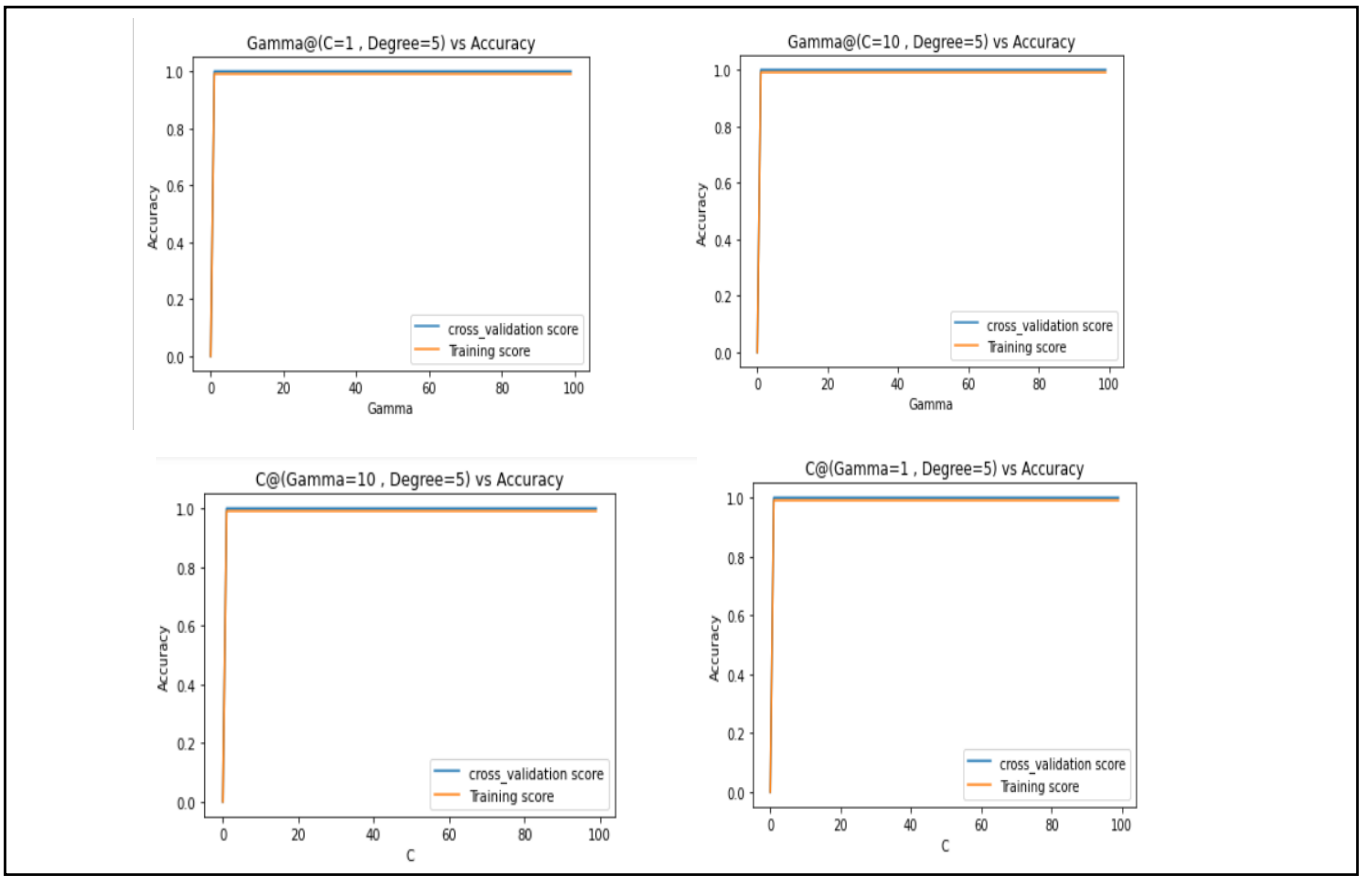

Figure 45: Variations of accuracy Vs C and Gamma

From the above graphs we can observe that low gamma and low C leads to underfittings then we get a sweet spot that is good fit then as we increase gamma and C the model starts to overfits.

Linear KERNEL using 10 features and classes = {0,1}

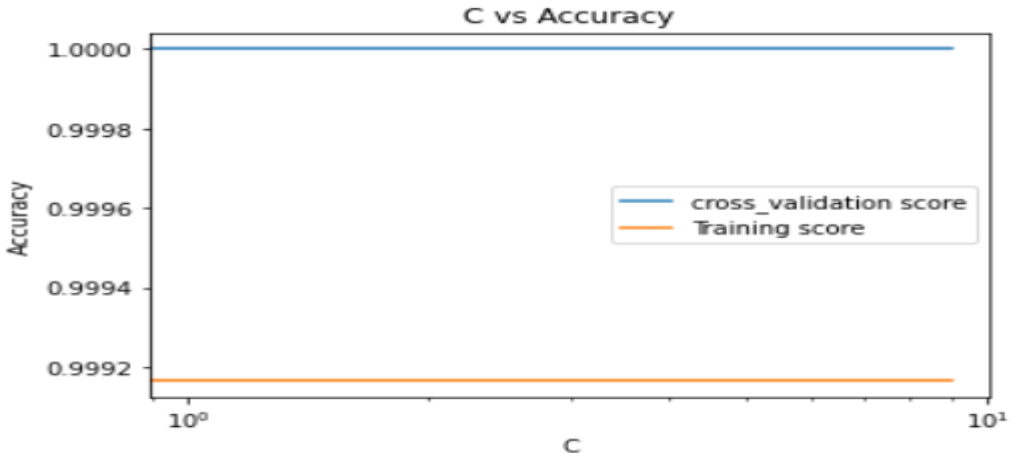

Figure 46: Variations of accuracy Vs C

RBF KERNEL using 10 features and classes = {0,1}

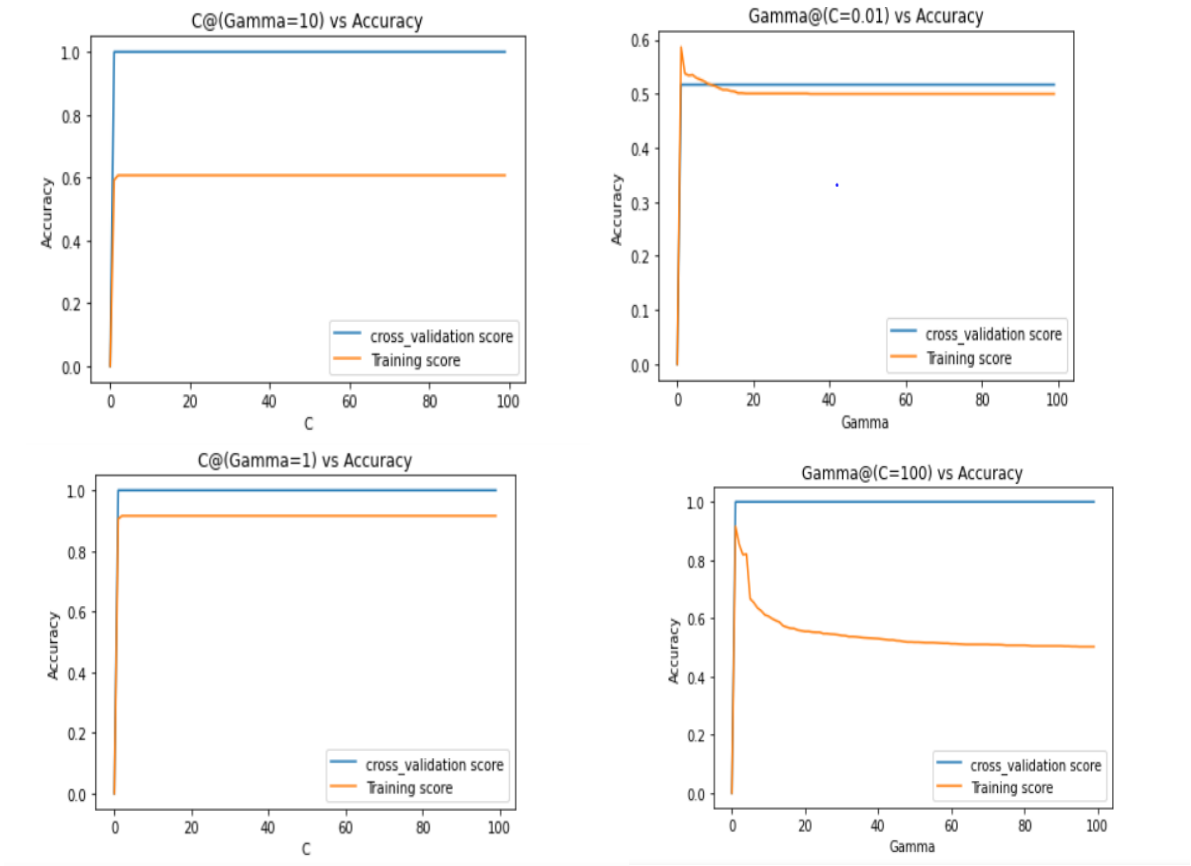

Figure 47: Variations of accuracy Vs C and Gamma

From the above graphs we can observe that low gamma and low C leads to underfittings then we get a sweet spot that is good fit then as we increase gamma and C the model starts to overfits.

POLY KERNEL using 10 features and classes = {0,1}

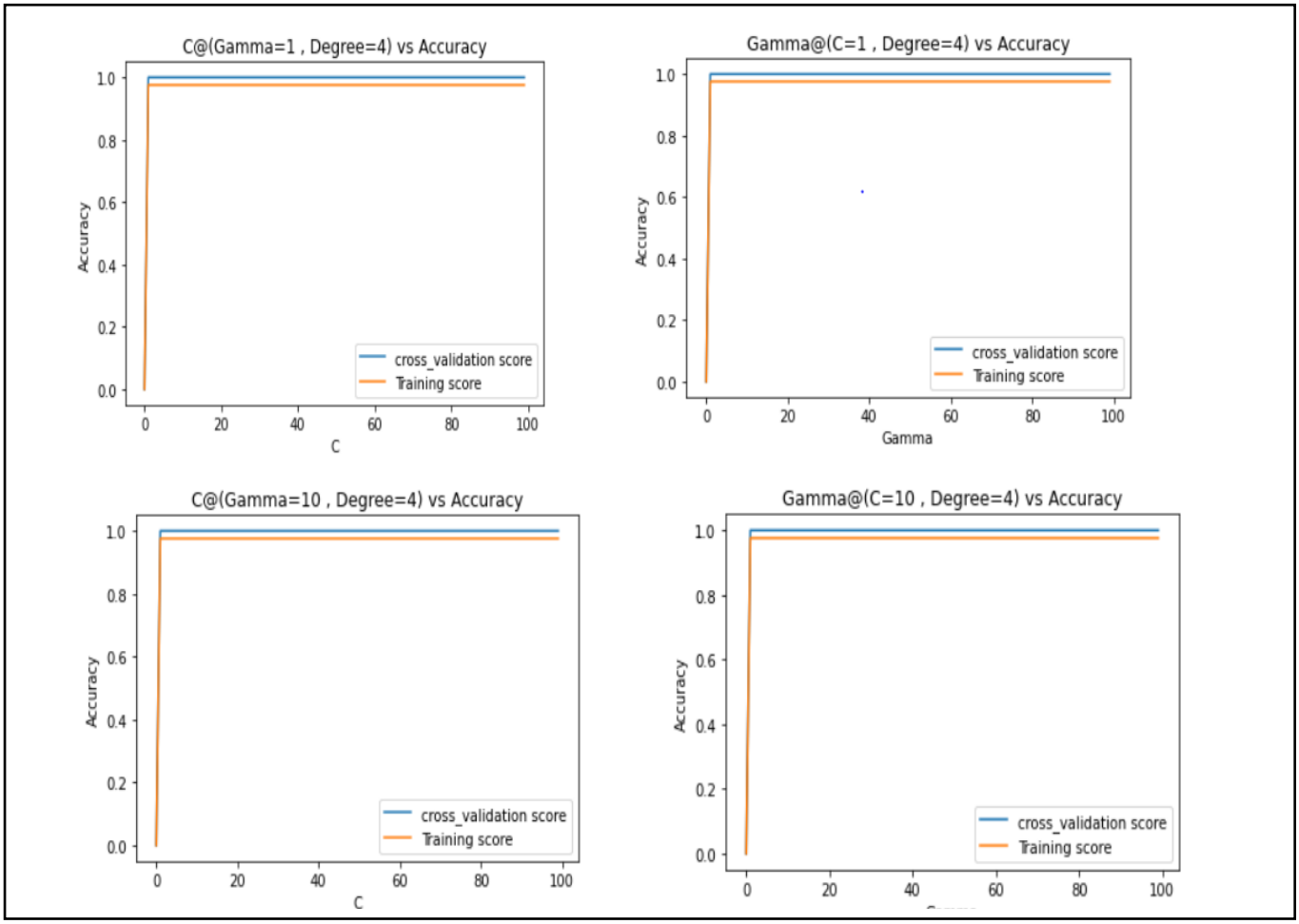

Figure 48: Variations of accuracy Vs C and Gamma

From the above graphs we can observe that low gamma and low C leads to underfittings then we get a sweet spot that is good fit then as we increase gamma and C the model starts to overfits.

MULTICLASS CLASSIFICATION

The LIBSVM USES ONE VS REST METHOD FOR MULTI CLASSIFICATION

Using 10 Features

The Following code is used for Analysis

```
PIPE = Pipeline([('scaler', StandardScaler()), ('SVM', svm.SVC(kernel='linear'))])
parameters = {'SVM__C':np.logspace(0, 1, 10)}
G = GridSearchCV(PIPE, param_grid=parameters, cv=5)

G.fit(text_x, text_t)
print ('Training score',G.score(text_x, text_t))
print ('Cross_validation score',G.score(tp_x, tp_t))
print (G.best_params_)
```

Figure 49: Wrapper code

Linear Kernel

Training score: 0.8867
Cross validation score: 0.812
Best Hyperparameters: {'SVM__C': 1.29}

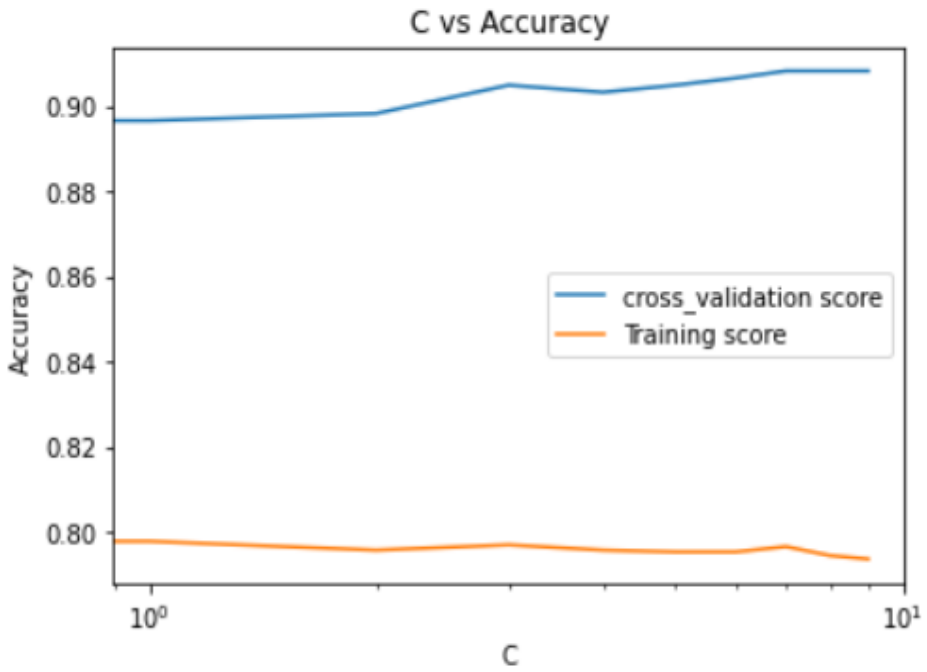

Figure 50: C vs Accuracy (Score) plot for linear kernel

RBF Kernel

Training score: 1.0
Cross validation score: 0.76
Best Hyperparameters: {'SVM__C': 1.2915496650148839, 'SVM__gamma': 1.0}

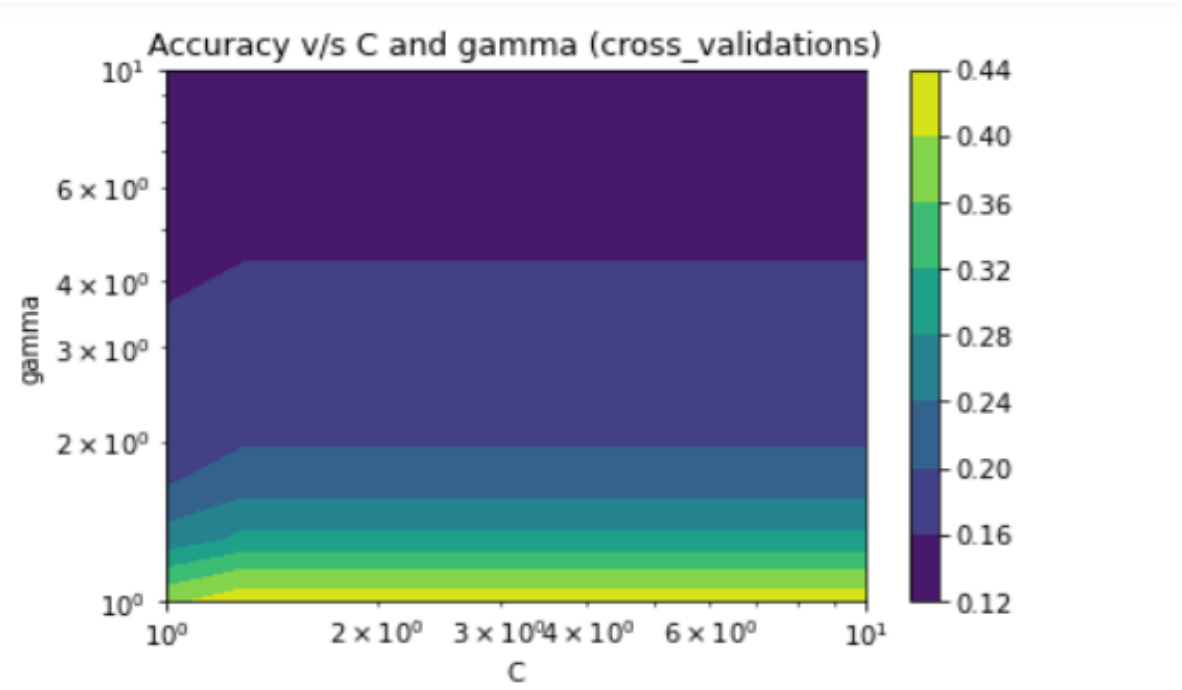

Figure 51: C and gamma vs Accuracy (Score) plot for rbf kernel for cross validation

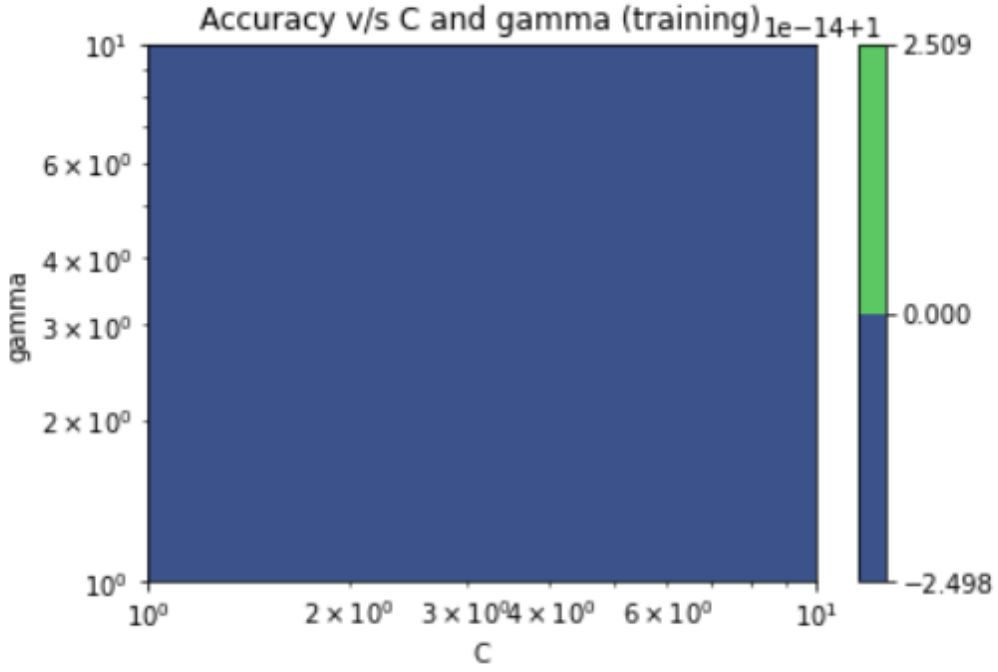

Figure 52: C and gamma vs Accuracy (Score) plot for rbf kernel for Training

Poly Kernel

Training score 0.886
Cross validation score 0.812
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.29}

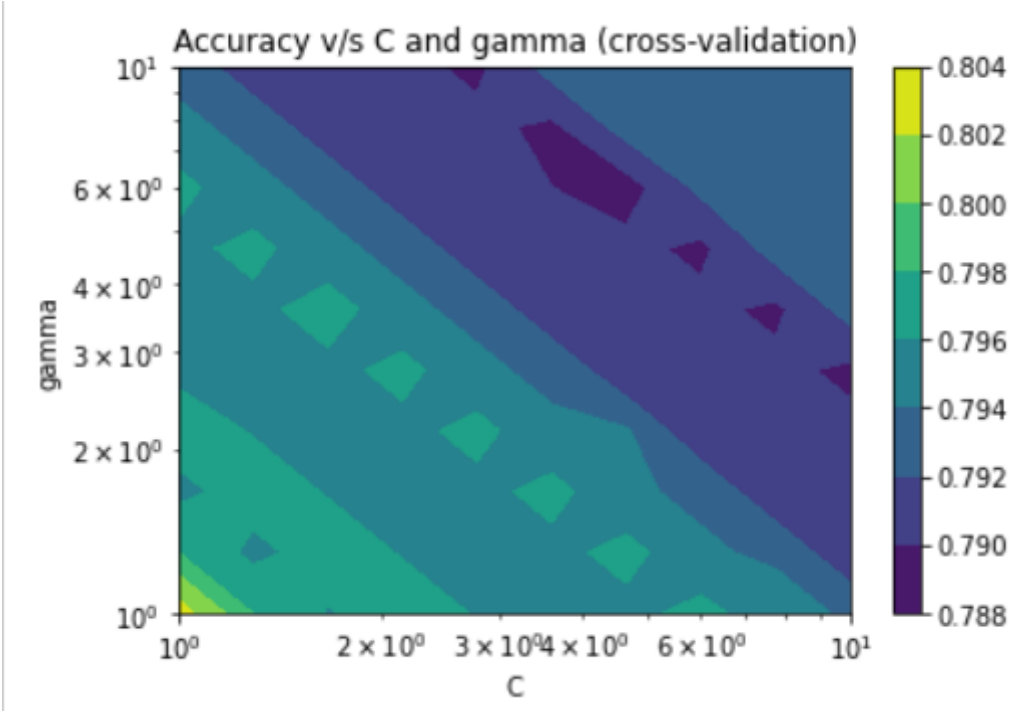

Figure 53: C and gamma vs Accuracy (Score) plot for poly kernel for cross validation

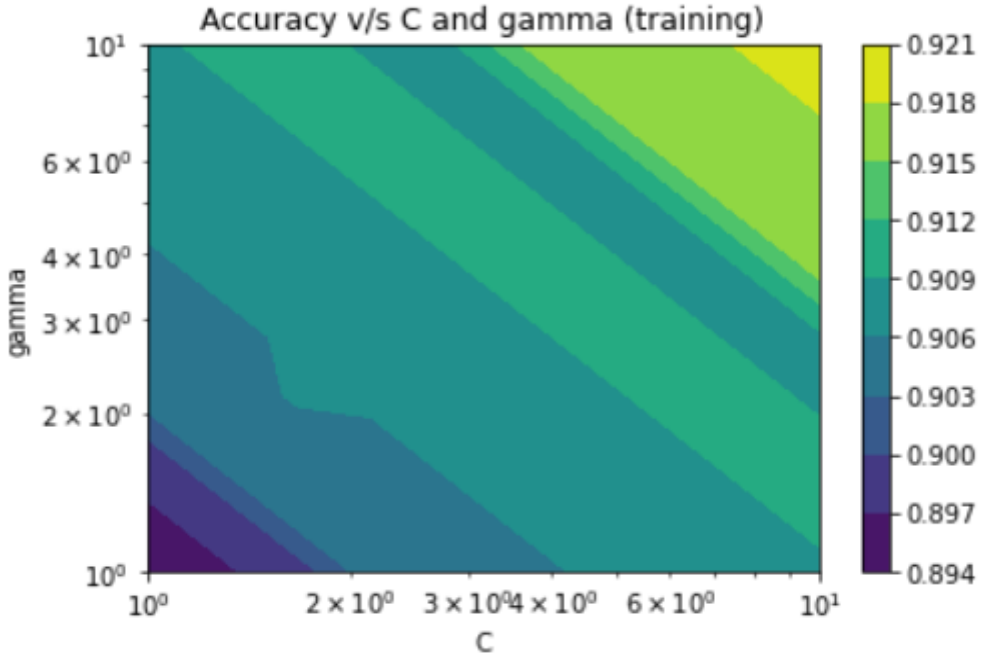

Figure 54: C and gamma vs Accuracy (Score) plot for poly kernel for Training

Using 25 Features

Linear Kernel

Training score: 0.98
Cross validation score: 0.85
Best Hyperparameters: {'SVM__C': 1.0}

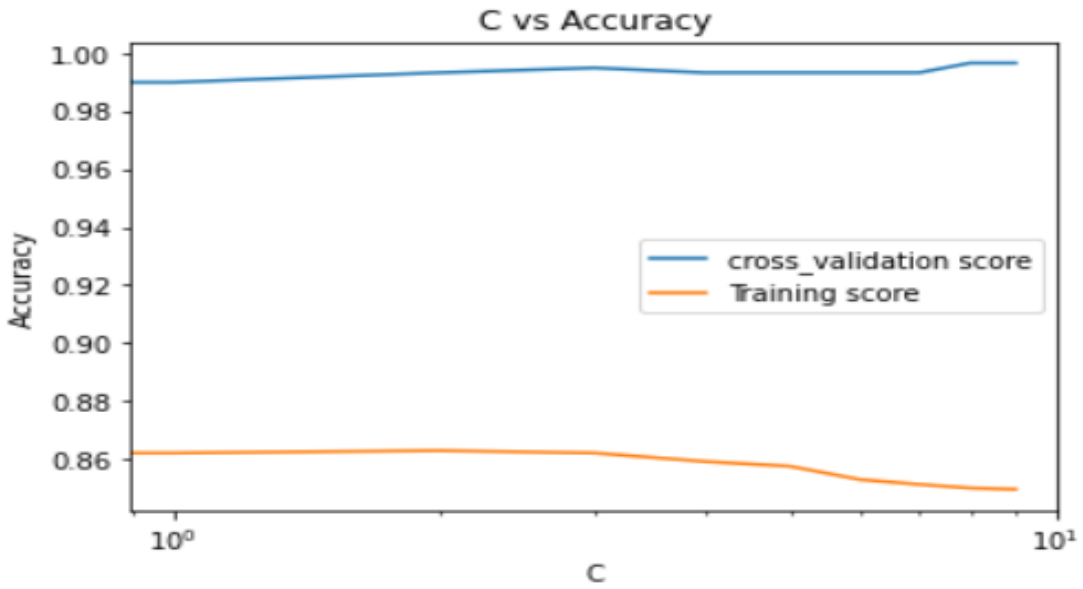

Figure 55: C vs Accuracy (Score) plot for linear kernel

RBF Kernel

Training score: 1.0
Cross validation score: 0.18125
Best Hyperparameters: {'SVM__C': 1.29154966, 'SVM__gamma': 1.0}

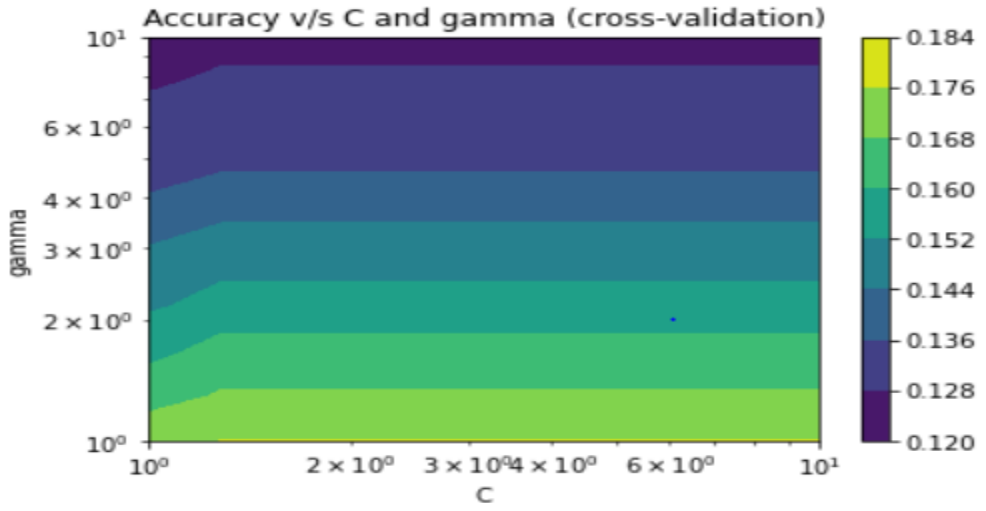

Figure 56: C and gamma vs Accuracy (Score) plot for rbf kernel for cross validation

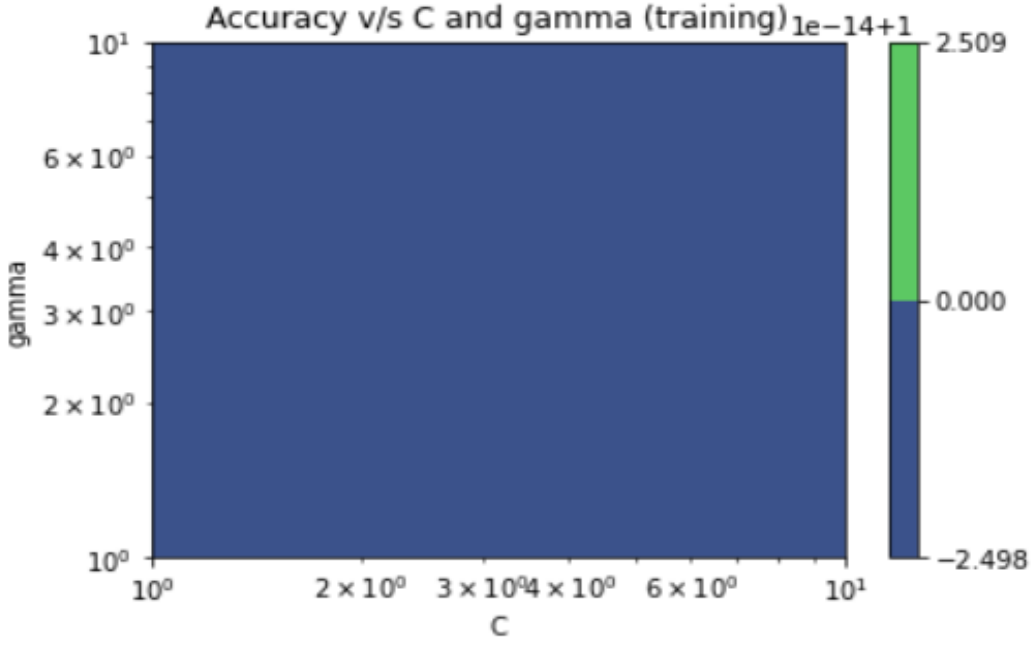

Figure 57: C and gamma vs Accuracy (Score) plot for rbf kernel for Training

Poly Kernel

Training score: 0.98
Cross validation score: 0.85
Best Hyperparameters: {'SVM__C': 1.0, 'SVM__degree': 1, 'SVM__gamma': 1.0}

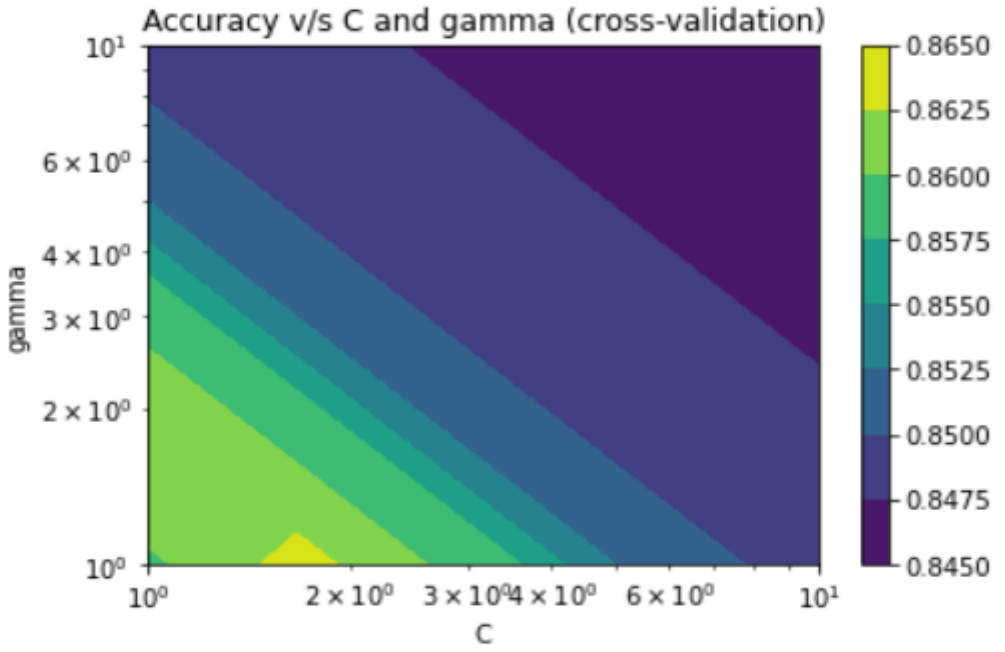

Figure 58: C and gamma vs Accuracy (Score) plot for poly kernel for cross validation

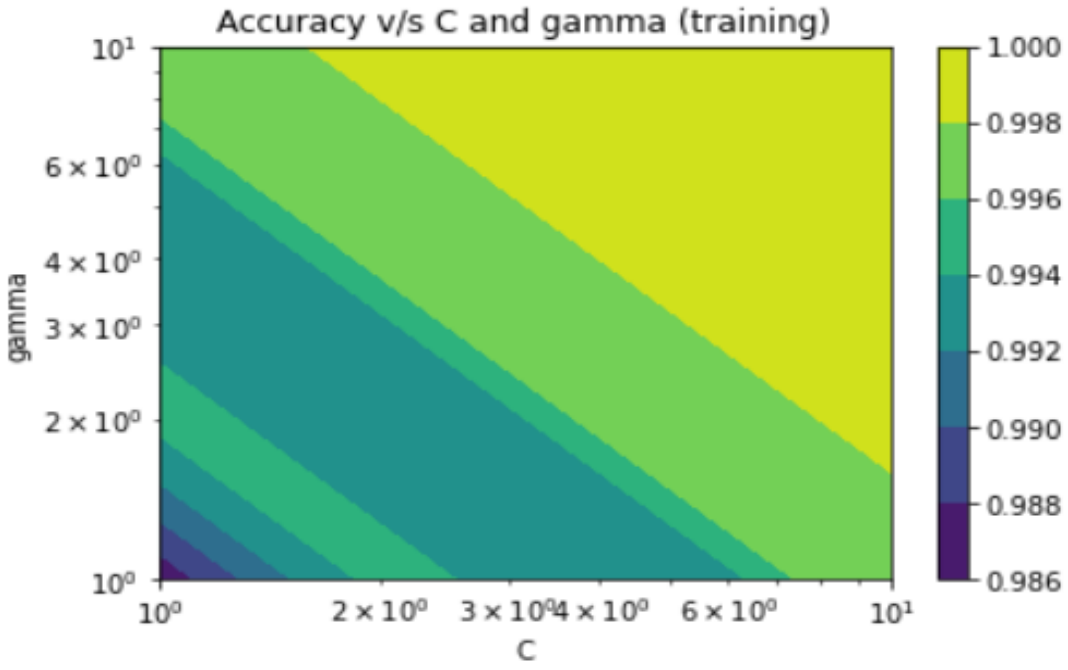

Figure 59: C and gamma vs Accuracy (Score) plot for poly kernel for Training

COMPARISON FOR VARIOUS KERNEL FOR 25 FEATURES

KERNEL	TRAINING SCORE	CROSS VALIDATION SCORE	BEST C	BEST GAMMA
Linear	0.98	0.85	1	-
RBF	1	0.18	1.29	1
Poly	0.98	0.85	1	1

Figure 60: Comparison for various kernels

COMPARISON FOR VARIOUS KERNEL FOR 10 FEATURES

KERNEL	TRAINING SCORE	CROSS VALIDATION SCORE	BEST C	BEST GAMMA
Linear	0.89	0.82	1.29	-
RBF	1	0.76	1.29	1
Poly	0.89	0.82	1	1

Figure 61: Comparison for various kernels

CONCLUSION

The above tables show the values of training error, cross validation error and values of best hyperparameters obtained using LIBSVM for different kernel functions

We can observe that values of best hyperparameters obtained are different form binary classification case because in binary classification hyperparameters also depends on which classes are chosen whereas in multiple classification we are taking all the classes so hyperparameters are independent of classes and are different.

Case1: Linear kernel

We can observe that training and cross validation scores are higher when we are using 25 features than 10 features. As number of features increases it provides a greater number of dimensions, so data of different classes can be used for better classification of data – points.

Case2: RBF kernel

We can observe that training and cross validation scores are higher when we are using 25 features than 10 features. As number of features increases it provides a greater number of dimensions, so data of different classes can be used for better classification of data – points.

Case3: Poly kernel

We can observe that training and cross validation scores are higher when we are using 25 features than 10 features. As number of features increases it provides a greater number of dimensions, so data of different classes can be used for better classification of data – points.

PART 1 B)

The Following code is used for Analysis

```
def simplified_SMO(C,tol,max_iter,X,y):
    alpha = np.zeros(X.shape[0])
    b = 0
    iter = 0
    while(iter<max_iter):
        num_changed_alphas = 0
        for i in range(X.shape[0]):
            Ei = computing_F(i,b,alpha,X,y) - y[i]
            if(((y[i]*Ei)<(-1*tol) and alpha[i]<C) or ((y[i]*Ei)>tol and alpha[i]>0)):
                j = randomy(X.shape[0]-1,i)
                Ej = computing_F(j,b,alpha,X,y) - y[j]
                alphaolds_j = alpha[j]
                alphaolds_i = alpha[i]
                L =  computing_L(alpha[i],alpha[j],C,y[i],y[j])
                H =  computing_H(alpha[i],alpha[j],C,y[i],y[j])
                if(L==H):
                    continue
                neta = computing_neta(X[i],X[j])
                if(neta>=0):
                    continue
                alpha[j] = cliping_alpha_j(L,H, computing_alpha_j(neta,y[j],Ei,Ej,alpha[j]))
                if(np.abs(alpha[j]-alphaolds_j)<tol):
                    continue
                alpha[i] = alpha[i] + (y[i] * y[j]) * (alphaolds_j - alpha[j])
                b1 = computing_b1(b,X[i],X[j],y[i],y[j],alpha[j],alpha[i],alphaolds_i,alphaolds_j,Ei,Ej)
                b2 = computing_b2(b,X[i],X[j],y[i],y[j],alpha[j],alpha[i],alphaolds_i,alphaolds_j,Ei,Ej)
                b = computing_b(C,alpha[j],alpha[i],b1,b2)
                num_changed_alphas += 1
        if (num_changed_alphas == 0):
            iter += 1
        else:
            iter = 0
    return alpha ,b
```

Figure 62: Wrapper code

OTHER HELPER FUNCTIONS

```
def computing_F(i,b,alpha,X,Y):
     sum=0
     for j in range(X.shape[0]):
          sum += (alpha[j]*Y[j])*(np.inner(X[j],X[i]))
     return sum+b


def randomy(m,i):
     random.seed(time.time())
     j = random.randint(0, m)
     while (i == j):
          j = random.randint(0, m)
     return j

def computing_L(alpha_i,alpha_j,C,Yi,Yj):
     if(Yi!=Yj):
          return max(0,alpha_j-alpha_i)
     else:
          return max(0,alpha_i+alpha_j-C)

def computing_H(alpha_i,alpha_j,C,Yi,Yj):
     if(Yi!=Yj):
          return min(C,C+alpha_j-alpha_i)
     else:
          return min(C,alpha_j+alpha_i)

def computing_neta(Xi,Xj):
     return 2*(np.inner(Xi,Xj))-(np.inner(Xi,Xi))-(np.inner(Xj,Xj))

def computing_alpha_j(neta,yj,Ei,Ej,alpha_j):
     return alpha_j-((yj)*(Ei-Ej))/neta
```

```
def cliping_alpha_j(L,H,alpha_j):
    if(alpha_j>H):
        return H
    elif(alpha_j<L):
        return L
    return alpha_j


def computing_b2(b,Xi,Xj,yi,yj,alpha_j,alpha_i,alphaold_i,alphaold_j,Ei,Ej):
    return (b - Ej - yi) * (alpha_i - alphaold_i) * np.inner(Xi, Xj) - (yj * (alpha_j - alphaold_j)) * np.inner(Xj, Xj)

def computing_b1(b,Xi,Xj,yi,yj,alpha_j,alpha_i,alphaold_i,alphaold_j,Ei,Ej):
    return (b - Ei - yi) * (alpha_i - alphaold_i) * np.inner(Xi, Xi) - (yj * (alpha_j - alphaold_j)) * np.inner(Xi, Xj)

def computing_b(C,alpha_j,alpha_i,b1,b2):
    if(alpha_i<C and alpha_i>0):
        return b1
    elif(alpha_j<C and alpha_j>0):
        return b2
    else:
        return (b1+b2)/2
```

Figure 63: Wrapper code

```
def weights(alpha,X,y):
     m,n = X.shape
     w = np.zeros(n)
     for i in range(X.shape[0]):
          w += alpha[i]*y[i]*X[i,:]
     return w

def get_scores(X,y,w,b):
     p = np.dot(X,w.T)+b
     m = y.shape[0]
     score = 0
     for j in range(m):
          if (p[j] >= 0):
               p[j] = 1
          else :
               p[j] = -1
     for i in range(m):
          if (p[i]*y[i]) > 0 :
               score=score+1
     return score/m
```

Figure 64: Wrapper code

COMPARISON FOR SPEED, ACCURACY FOR DIFFERENT METHODS

Linear Kernel

25 features and class label = {0,1}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.99	1.94
CVX	1	0.998	1.84
Simplified_SMO	1	1	236.44

Figure 65: Comparison Table (max no of iteration for smo are 1)

Linear Kernel

25 features and class label = {2,3}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	0.97	0.92	2.73
CVX	0.90	0.92	0.87
Simplified_SMO	0.974	0.928	1407.14

Figure 66: Comparison Table (max no of iteration for smo are 1)

Linear Kernel

25 features and class label = {4,5}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.97	2.15
CVX	1	0.98	0.685
Simplified_SMO	0.998 ~ 1	0.96	164.8

Figure 67: Comparison Table (max no of iteration for smo are 1)

In the above calculations Best hyperparameters obtained by Libsvm are used and tolerance = 0.0005 is used.

COMPARISON FOR SPEED, ACCURACY FOR DIFFERENT METHODS

Poly Kernel

25 features and class label = {0,1}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.99	13.12
CVX	1	0.998	2.00
Simplified_SMO	1	1	88.47

Figure 68: Comparison Table (max no of iteration for smo are 1)

Poly Kernel

25 features and class label = {2,3}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.95	21.37
CVX	0.516	0.517	0.72
Simplified_SMO	0.953	0.906	975.41

Figure 69: Comparison Table (max no of iteration for smo are 500)

Poly Kernel

25 features and class label = {4,5}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.97	14.78
CVX	1	0.97	0.80
Simplified_SMO	0.998	0.966	673.41

Figure 70: Comparison Table (max no of iteration for smo are 2)

In the above calculations Best hyperparameters obtained by Libsvm are used and tolerance = 0.0005 is used.

PART 1 C)

The Following code is used for Analysis

```
def TS(i1,i2,alpha,x,y,C,eps,b,m):
    if(i1 == i2):
        return 0
    alph1 = alpha[i1]
    alph2 = alpha[i2]
    E1 =  Error_function(alpha,y,x,b)[i1]
    E2 =  Error_function(alpha,y,x,b)[i2]
    s = y[i1]*y[i2]
    L =  computing_L(alph1,alph2,C,y[i1],y[i2])
    H =  computing_H(alph1,alph2,C,y[i1],y[i2])
    if(L==H):
        return 0
    neta = computing_neta(x[i1],x[i2])
    if(neta>0):
        a2 = find_a2(alph2,y[i2],E1,E2,neta,L,H)
    else:
        A = alpha.copy()
        A[i2] = L
        Lobj = OF(A, y,x)
        A[i2] = H
        Hobj = OF(A, y,x)
        a2=find_aa2(L,H,Lobj,Hobj,eps,alph2)
    if (a2 < 1e-4):
        a2 = 0.0
    elif (1e-4 > (C - a2)):
        a2 = C

    if (np.abs(a2-alph2) < eps*(a2+alph2+eps)):
        return 0
    a1 = alph1+s*(alph2-a2)
    b1 = computing_b1(b,x[i1],x[i2],y[i1],y[i2],a2,a1,alph1,alph2,E1)
    b2 = computing_b2(b,x[i1],x[i2],y[i1],y[i2],a2,a1,alph1,alph2,E2)
    b_new = computing_b(C,a2,a1,b1,b2)
    alpha[i1] = a1
    alpha[i2] = a2
for i in range(i1,i2):
    for alph in range(int(a1),int(a2)):
        if(0.0<int(alph)<C):
            Error_function(alpha,y,x,b)[i] = 0.0

for p in range(m):
    if(p!=i1 & p!=i2):
        Error_function(alpha,y,x,b)[p] +=y[i1]*(a1 - alph1)*np.inner(x[i1], x[p]) + y[i2]*(a2 - alph2)*np.inner(x[i2],x[
b = b_new
return 1
```

Figure 71: Wrapper code

```
def ExExa(alpha,x,y,C,eps,b,i2,m,tol):
    y2 = y[i2]
    a2 = alpha[i2]
    E2 = Error_function(alpha,y,x,b)[i2]
    r2 = E2*y2
    if ((r2 < -tol and a2 < C) or (r2 > tol and a2 > 0)):
        if (len(alpha[(alpha != 0) & (alpha != C)])) > 1:
            if Error_function(alpha,y,x,b)[i2] > 0:
                i1 = np.argmin(Error_function(alpha,y,x,b))
            elif Error_function(alpha,y,x,b)[i2]  <= 0:
                i1 = np.argmax(Error_function(alpha,y,x,b))
            if (TS(i1, i2, alpha,x,y,C,eps,b,m)):
                return 1
        t1 = list(range(m))
        random.shuffle(t1)
        for i1 in t1:
            if((alpha[i1] != 0) & (alpha[i1] != C)):
                if TS(i1, i2, alpha,x,y,C,eps,b,m):
                    return 1
        t = list(range(m))
        random.shuffle(t)
        for i1 in t:
            if TS(i1, i2, alpha,x,y,C,eps,b,m):
                return 1
    return 0
def Full_SMO(x,y,C,eps,m,tol):
    b=0.0
    alpha = np.zeros(x.shape[0])
    NC = 0
    EA = 1
    while (NC > 0 or EA):
        NC = 0
        if (EA):
            for i in range(alpha.shape[0]):
                NC += ExExa(alpha,x,y,C,eps,b,i,m,tol)
        else:
             for i in np.where((alpha != 0) & (alpha != C))[0]:
                NC += ExExa(alpha,x,y,C,eps,b,i,m,tol)

        if (EA == 1):
            EA = 0
        elif(NC == 0):
            EA = 1
    return alpha,b
```

Figure 72: Wrapper code

OTHER HELPER FUNCTIONS

```
def computing_L(alpha_i,alpha_j,C,Yi,Yj):
    if(Yi!=Yj):
        return max(0,alpha_j-alpha_i)
    else:
        return max(0,alpha_i+alpha_j-C)

def computing_H(alpha_i,alpha_j,C,Yi,Yj):
    if(Yi!=Yj):
        return min(C,C+alpha_j-alpha_i)
    else:
        return min(C,alpha_j+alpha_i)

def computing_neta(Xi,Xj):
    return 2*(np.inner(Xi,Xj))-(np.inner(Xi,Xi))-(np.inner(Xj,Xj))

def computing_b2(b,Xi,Xj,yi,yj,alpha_j,alpha_i,alphaold_i,alphaold_j,Ej):
    return b + Ej - yi * (alpha_i - alphaold_i) * np.inner(Xi, Xj) + yj * (alpha_j - alphaold_j) * np.inner(Xj, Xj)

def computing_b1(b,Xi,Xj,yi,yj,alpha_j,alpha_i,alphaold_i,alphaold_j,Ei):
    return b + Ei - yi * (alpha_i - alphaold_i) * np.inner(Xi, Xi) + yj * (alpha_j - alphaold_j) * np.inner(Xi, Xj)

def computing_b(C,alpha_j,alpha_i,b1,b2):
    if(alpha_i<C and alpha_i>0):
        return b1
    elif(alpha_j<C and alpha_j>0):
        return b2
    else:
        return (b1+b2)/2
```

```
def OF(alpha,y,x):
    s1 = np.sum(alpha)
    p = ((np.matmul(y.reshape((-1,1)) , y.reshape((1,-1)))) * (x.dot(x.T)) * (np.matmul(alpha.reshape((-1,1)),alpha.reshape(
    s2 = np.sum(p)
    return s1 - (s2/2)

def Error_function(alpha,y,x,b):
    return np.dot((alpha * y), x.dot(x.T)) - b -y

def find_a2(alpha,y,Ei,Ej,neta,L,H):
    c = alpha - y*(Ei-Ej)/neta
    if(c<L):
        return L
    elif(c>H):
        return H
    else:
        return c

def find_aa2(L,H,Lobj,Hobj,eps,alph2):
    if (Lobj < Hobj-eps):
        return L
    elif (Lobj > Hobj+eps):
        return H
    else:
        return alph2
```

```
def weights(alpha,X,y):
    m,n = X.shape
    w = np.zeros(n)
    for i in range(X.shape[0]):
        w += alpha[i]*y[i]*X[i,:]
    return w

def get_scores(X,y,w,b):
    p = np.dot(X,w.T)+b
    m = y.shape[0]
    score = 0
    for j in range(m):
        if (p[j] >= 0):
            p[j] = 1
        else :
            p[j] = -1
    for i in range(m):
        if (p[i]*y[i]) > 0 :
            score=score+1
    return score/m
```

Figure 73: Wrapper code

Ques) Explain in detail how you chose which Lagrange multipliers to optimise.

Ans) The outer loop in the code is based on the choice of first langrage multiplier. It iterates over the complete training set in order to determine whether it violates KKT conditions or not. If it violates the KKT condition's then it used is for optimization. In this way first Lagrange multiplier is chosen which has to be optimized.

The second multiplier is chosen in such a way that maximizes the step size. We can do this by evaluating kernel function again and again but this process takes a lot of time. So instead of this we approximate step size by absolute value of E_1-E_2 in the below equation.

$$\alpha_2^{new} = \alpha_2 + (y_2*(E_1-E_2))/neta$$

SMO maintains the cache error values for all the non-bound example of training set. By the above equation we can say that if $E_1 > 0$ then to maximize $|E_1 - E_2|$ we have to chose minimum value of E_2. And if $E_1 < 0$ then to maximize $|E_1 - E_2|$ we have to choose maximum value of E_2.

COMPARISON FOR SPEED, ACCURACY FOR DIFFERENT METHODS

Linear Kernel

25 features and class label = {0,1}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.99	1.94
CVX	1	0.998	1.84
Simplified_SMO	1	1	236.44
Full_SMO	1	0.999	41.22

Figure 74: Comparison Table

Linear Kernel

25 features and class label = {2,3}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	0.97	0.92	2.73
CVX	0.90	0.92	0.87
Simplified_SMO	0.974	0.928	1407.14
Full_SMO			

Figure 75: Comparison Table

Linear Kernel

25 features and class label = {4,5}

METHOD	**TRAINING SCORE**	**CROSS VALIDATION SCORE**	TIME
LIBSVM	1	0.97	2.15
CVX	1	0.98	0.685
Simplified_SMO	0.998 ~ 1	0.96	164.8
Full_SMO	0.966	0.941	3546.23

Figure 76: Comparison Table

In the above calculations Best hyperparameters obtained by Libsvm are used and tolerance = 0.0005 and eps = 0.0005 is used.

PART 2)

Attempt 1: using Linear Kernel Using 25 features

Linear kernel is used and best hyperparameters are calculated by tuning the hyperparameters. Cross validation is also used to get better results.

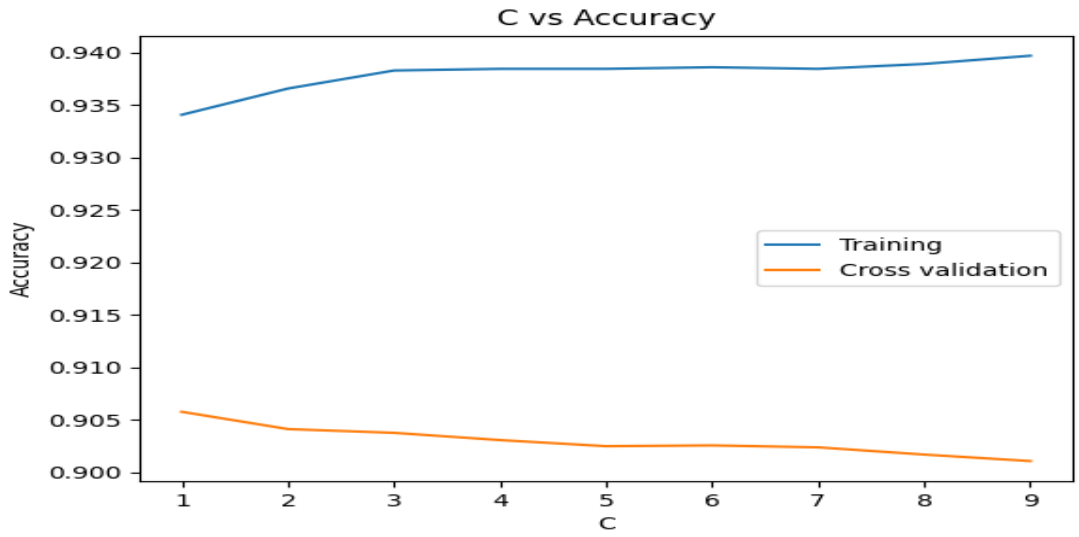

Figure 77: C vs accuracy

Attempt 2: using Linear Kernel Using 10 features

Linear kernel is used and best hyperparameters are calculated by tuning the hyperparameters. Cross validation is also used to get better results.

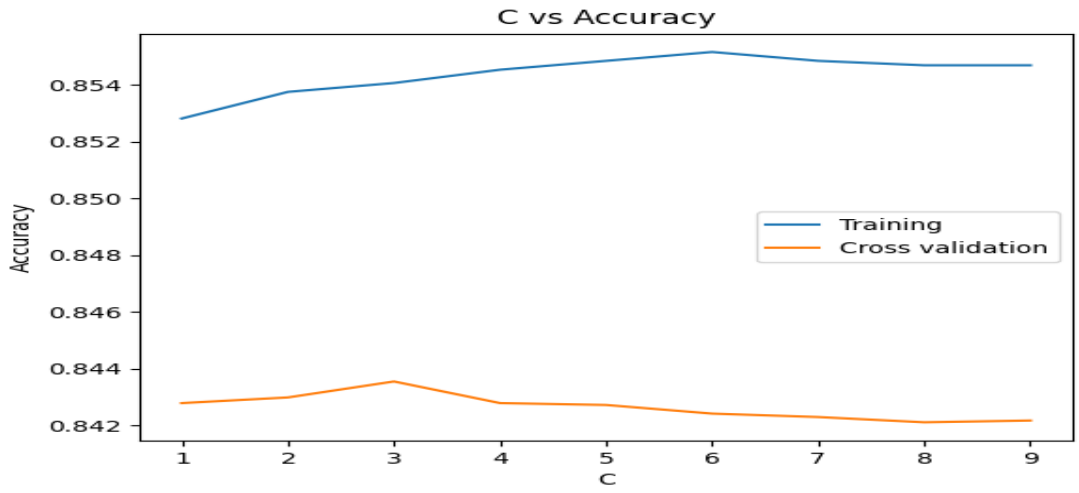

Figure 78: C vs accuracy

Attempt 3: using Poly Kernel using 25 features

Polynomial kernel is used and best hyperparameters are calculated by tuning the hyperparame ters. Cross validation is also used to get better results.

Attempt 3.1:

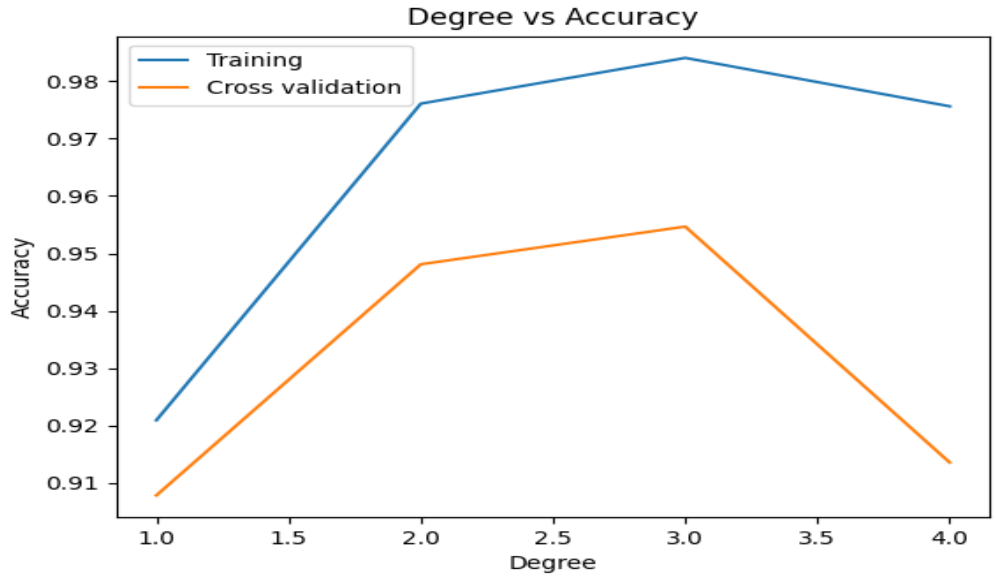

Figure 79: Degree vs accuracy

Attempt 3.2:

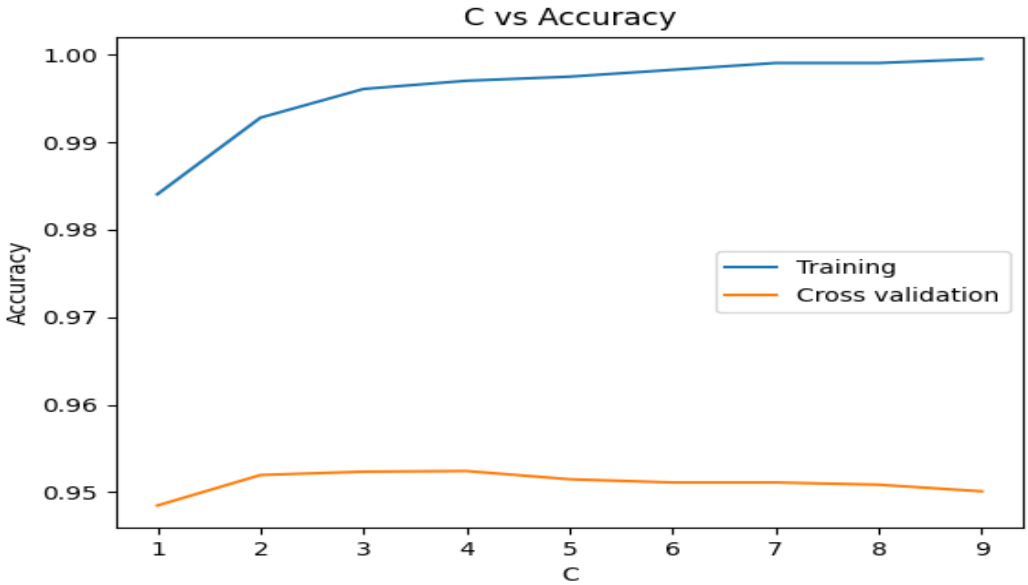

Figure 80: C vs accuracy

Attempt 3.3:

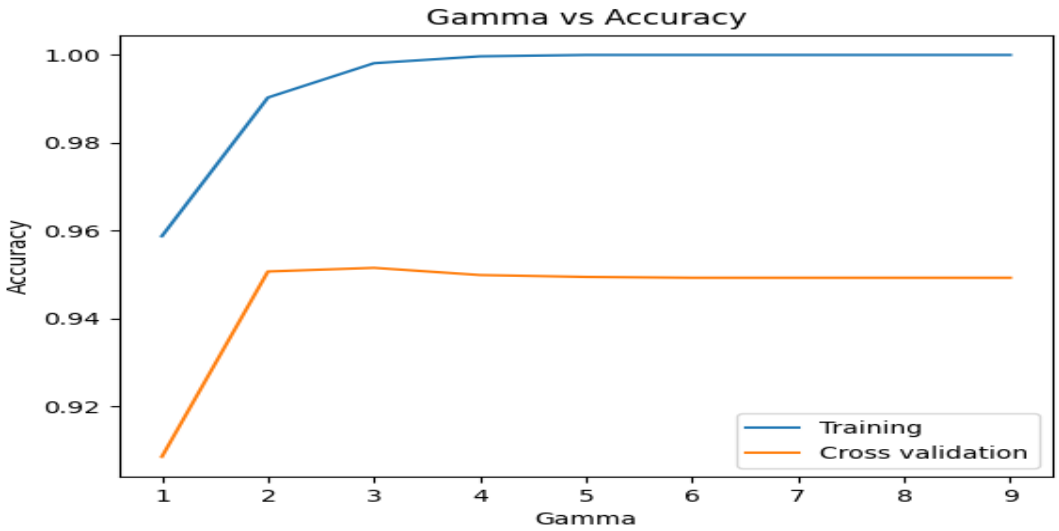

Figure 81: Gamma vs accuracy

Attempt 3.4:

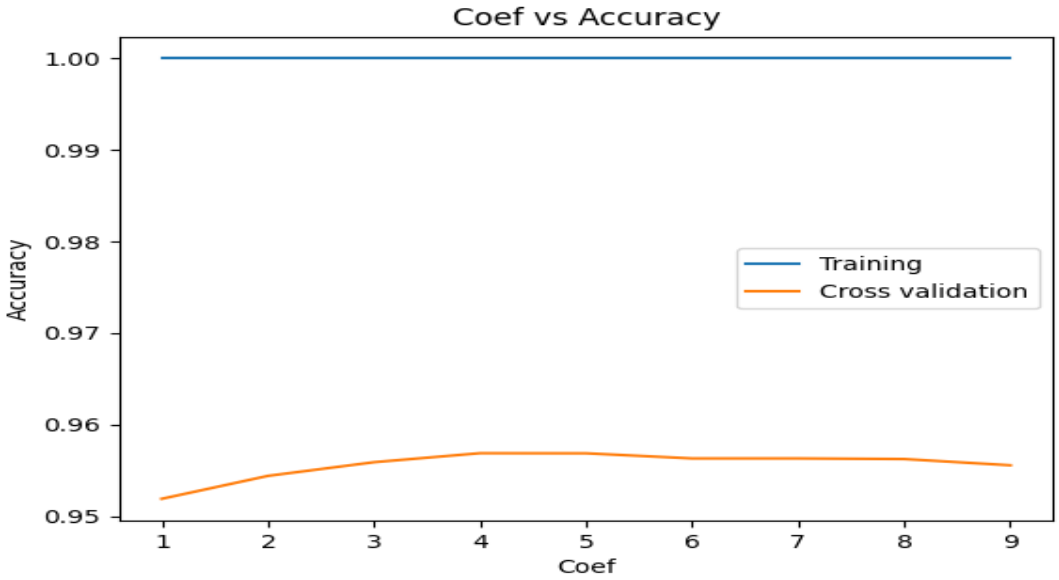

Figure 82: Coef vs accuracy

Attempt 4: using Poly Kernel using 10 features

Polynomial kernel is used and best hyperparameters are calculated by tuning the hyperparame ters. Cross validation is also used to get better results.

Attempt 4.1:

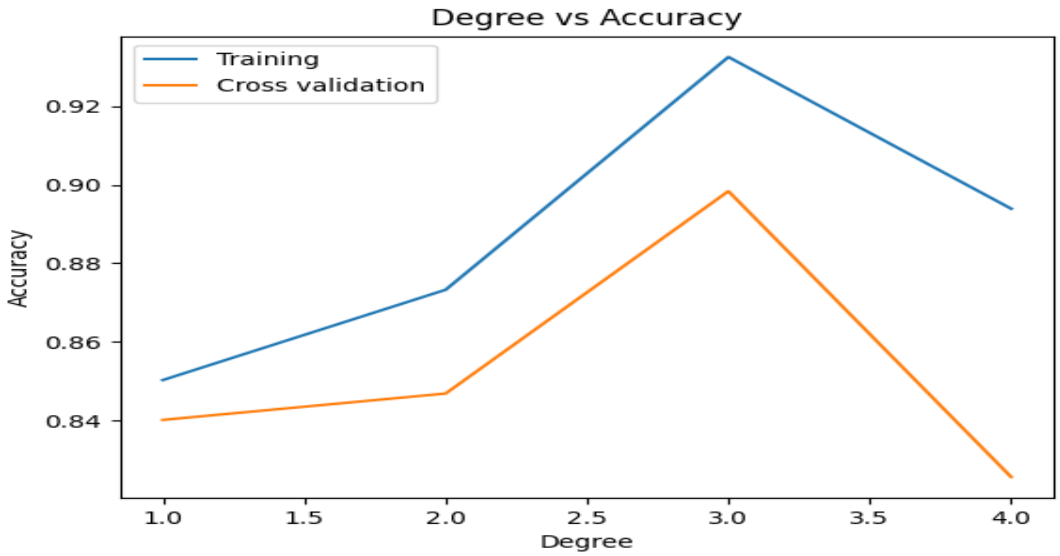

Figure 83: Degree vs accuracy

Attempt 4.2:

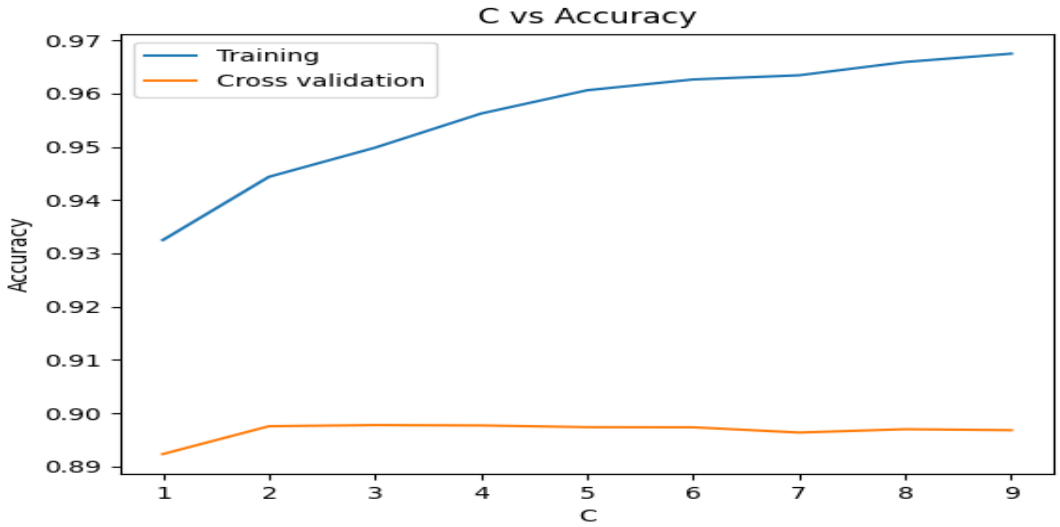

Figure 84: C vs accuracy

Attempt 4.3:

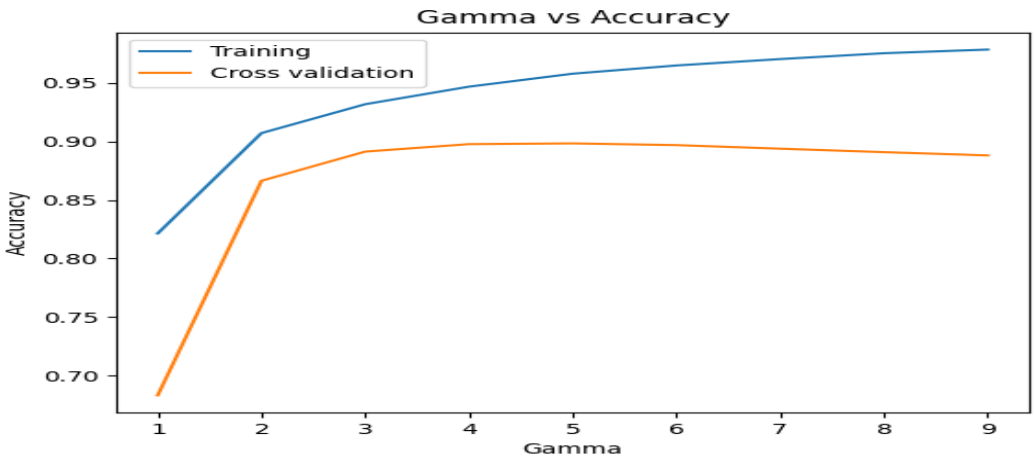

Figure 85: Gamma vs accuracy

Attempt 4.4:

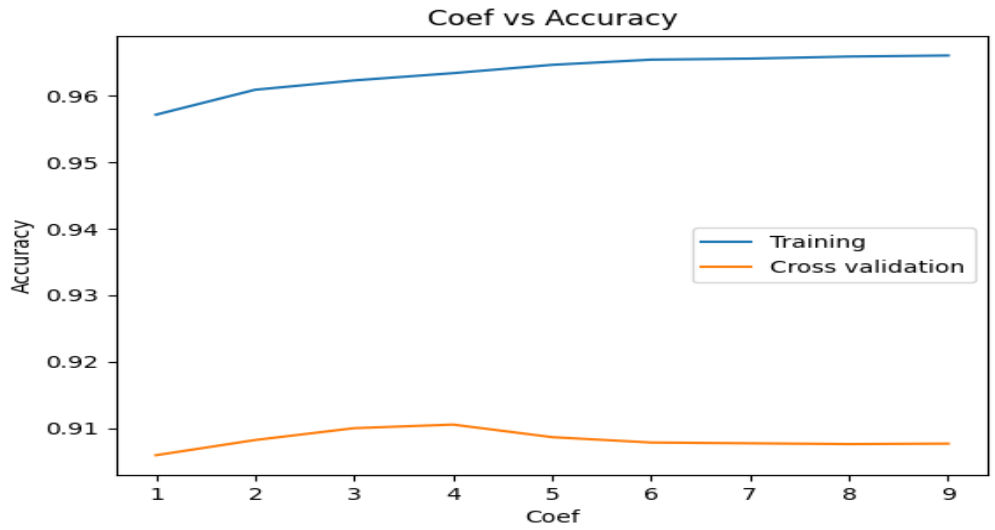

Figure 86: Coef0 vs accuracy

Attempt 5: using Rbf Kernel using 25 features

Rbf kernel is used and best hyperparameters are calculated by tuning the hyperparameters. Cr oss validation is also used to get better results.

Attempt 5.1:

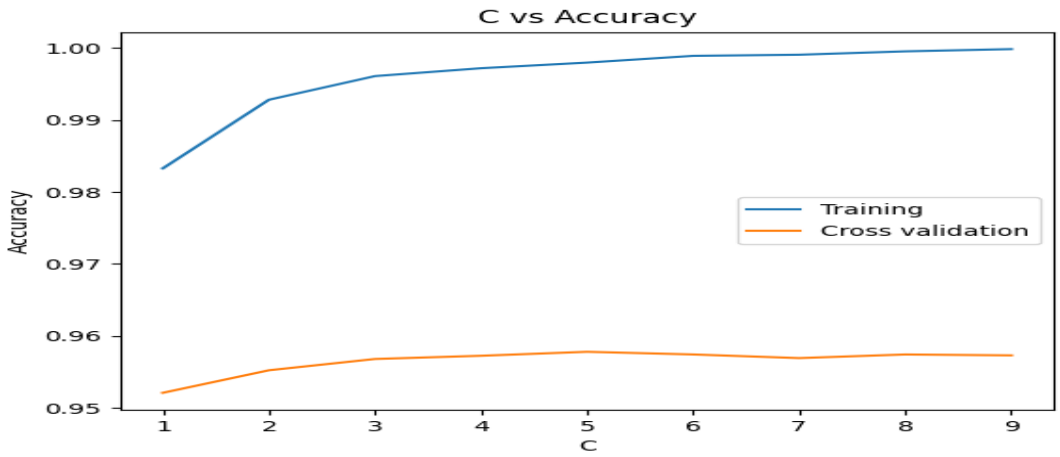

Figure 87: C vs accuracy

Attempt 5.2:

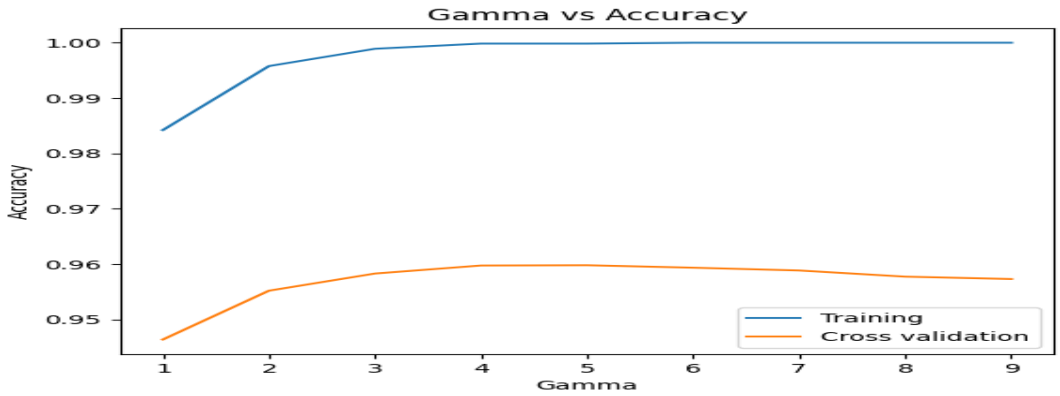

Figure 88: Gamma vs accuracy

Attempt 6: using Rbf Kernel using 10 features

Rbf kernel is used and best hyperparameters are calculated by tuning the hyperparameters. Cr oss validation is also used to get better results.

Attempt 6.1:

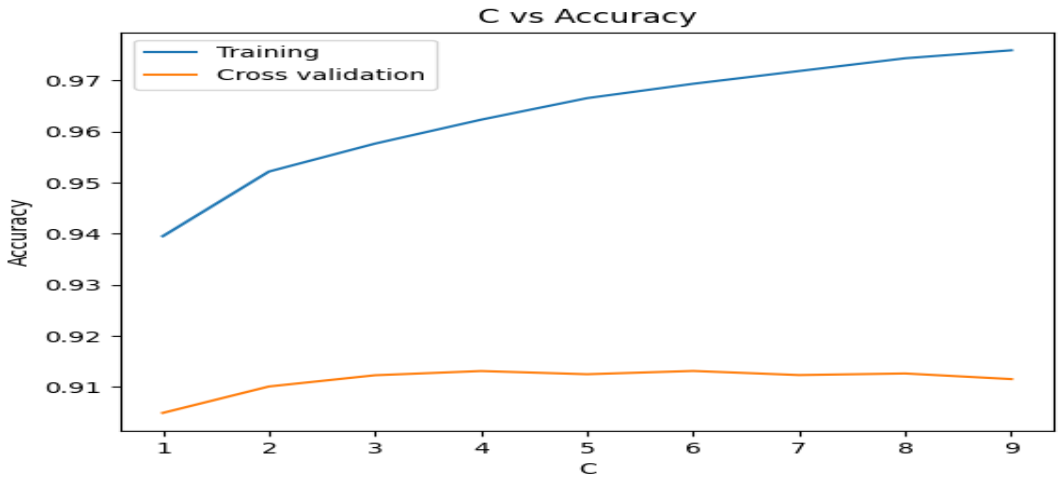

Figure 89: C vs accuracy

Attempt 6.2:

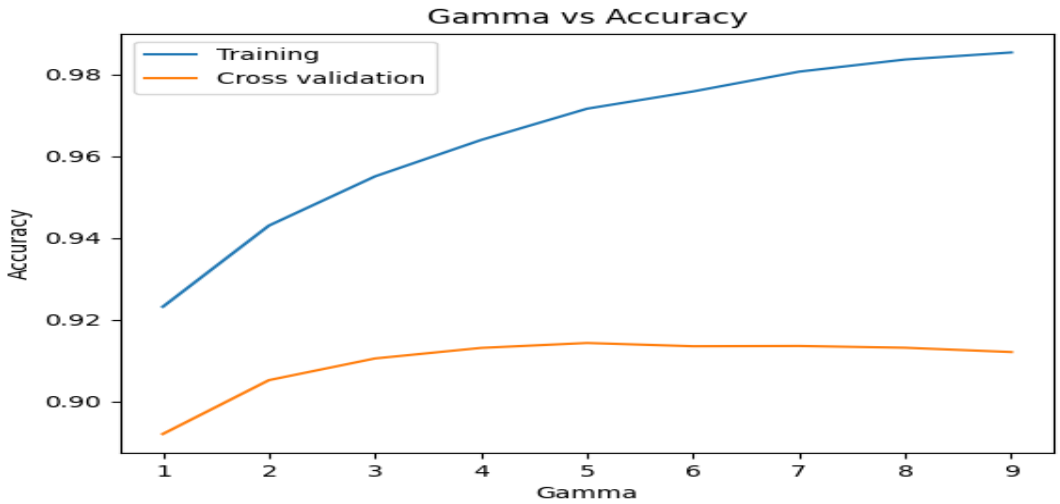

Figure 90: Gamma vs accuracy

For this part I have used various kernels and did hyperparameter tuning. I also did cross validation to get better results. I also used feature selections to get better results.

On Kaggle I have did total of 22 submission and get score of 96.625.