# CS156 (Introduction to AI), Fall 2021

## Homework 7 submission

### Roster Name: Nand Kishore Khuswaha

### Student ID: 013920192

### Email address: nandkishore.khuswaha@sjsu.edu (mailto:nandkishore.khuswaha@sjsu.edu)

Any special notes or anything you would like to communicate to me about this homework submission goes in here.

## References and sources

List all your references and sources here. This includes all sites/discussion boards/blogs/posts/etc. where you grabbed some code examples.

1) MLP.MNIST (class file)

2) Train_validation_test_Iris (class file)

3) Dimensionality_reduction (class file)

## Solution

#### Load libraries and set random number generator seed

In [82]:
```
import numpy as np
from sklearn.datasets  import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_validate
import seaborn as sns
import pandas as pd
```

In [83]:
```
np.random.seed(42)
```

#### Code the solution

In [84]:
```
mnist = load_digits()
```

In [85]:
```
n_samples = len(mnist.images)
images=mnist.images
images.shape
images_resh = images.reshape(images.shape[0],-1)
# images = mnist.images.reshape((n_samples, -1))
images_nor = images_resh.astype("float32")/255
labels = mnist.target
```

In [86]:
```
X_train, X_test, Y_train, Y_test = train_test_split(images_nor, labels, tes
X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

Out[86]:
```
((1437, 64), (1437,), (360, 64), (360,))
```

### 1. A model with default parameters for MLPClassifier other than random_state=1 and max_iter=max_iter

In [ ]:
```
```

In [87]:
```
model = MLPClassifier(random_state=1, max_iter=1000).fit(X_train, Y_train)
```

In [88]:
```
model.score(X_test, Y_test)
```

Out[88]:
```
0.9722222222222222
```

In [89]:
```
#  perform stratified 5-fold cross-validation
stratkfolds = StratifiedKFold(n_splits=5)
for train_indices, validation_indices in stratkfolds.split(X_train, Y_train
    print("Training indices: "+str(train_indices))
    print("Validation indices: "+str(validation_indices))
    X_train_set, X_validation_set = X_train[train_indices], X_train[validat
    Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validat
    print("Shapes of train/valiation sets:")
    print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_v
```

```
 267 270 272 273 274 275 276 280 287 289 293 294 295 296 297 298 299 300
 301 303 304 305 311 318 319 321 328 332 334 342 343 348 351 357 361 367]
Shapes of train/valiation sets:
(1149, 64) (1149,) (288, 64) (288,)
Training indices: [   0    1    2 ... 1434 1435 1436]
Validation indices: [194 195 208 210 212 217 218 220 227 247 256 259 261
263 265 268 269 271
 277 278 279 281 282 283 284 285 286 288 290 291 292 302 306 307 308 309
 310 312 313 314 315 316 317 320 322 323 324 325 326 327 329 330 331 333
 335 336 337 338 339 340 341 344 345 346 347 349 350 352 353 354 355 356
 358 359 360 362 363 364 365 366 368 369 370 371 372 373 374 375 376 377
 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
 468 469 470 471 473 474 475 476 477 478 479 481 482 483 484 485 486 487
 488 489 490 491 492 493 494 495 496 497 499 500 501 502 503 504 507 508
 509 510 512 513 514 515 516 517 518 519 520 521 523 524 525 526 527 530
 531 532 533 535 536 537 538 539 540 541 542 543 545 546 547 548 549 550
```

In [90]:
```
# collect prediction accuracies for each fold
cv_results_m1_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m1_train['test_score'])
print("Mean accuracy: "+str(cv_results_m1_train['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1000) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,

[0.96180556 0.98263889 0.95818815 0.96864111 0.96515679]
Mean accuracy: 0.9672861014324429
```

In [91]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results['test_score'])
print("Mean accuracy: "+str(cv_results['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1000) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1000) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1000) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1000) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,

[0.94444444 0.90277778 0.94444444 0.94444444 0.90277778]
Mean accuracy: 0.9277777777777778

/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1000) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
```

**2.Three hidden layer sizes: 400,150,50; with Relu activation function**

In [ ]:
```

```

In [92]:
```
model = MLPClassifier(hidden_layer_sizes=(400,150,50), activation = 'relu',
model.score(X_test, Y_test)
```

Out[92]:
```
0.9472222222222222
```

In [93]:
```
#  perform stratified 5-fold cross-validation
stratkfolds = StratifiedKFold(n_splits=5)
for train_indices, validation_indices in stratkfolds.split(X_train, Y_train
    print("Training indices: "+str(train_indices))
    print("Validation indices: "+str(validation_indices))
    X_train_set, X_validation_set = X_train[train_indices], X_train[validat
    Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validat
    print("Shapes of train/valiation sets:")
    print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_v
```

```
 310 312 313 314 315 316 317 320 322 323 324 325 326 327 329 330 331 333
 335 336 337 338 339 340 341 344 345 346 347 349 350 352 353 354 355 356
 358 359 360 362 363 364 365 366 368 369 370 371 372 373 374 375 376 377
 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
 468 469 470 471 473 474 475 476 477 478 479 481 482 483 484 485 486 487
 488 489 490 491 492 493 494 495 496 497 499 500 501 502 503 504 507 508
 509 510 512 513 514 515 516 517 518 519 520 521 523 524 525 526 527 530
 531 532 533 535 536 537 538 539 540 541 542 543 545 546 547 548 549 550
 552 557 558 559 561 563 564 566 567 570 572 573 575 581 582 587 589 592
 593 596 597 598 599 605 609 615 623 627 636 638 643 645 647 650 652 676]
Shapes of train/valiation sets:
(1149, 64) (1149,) (288, 64) (288,)
Training indices: [   0    1    2 ... 1434 1435 1436]
Validation indices: [472 480 498 505 506 511 522 528 529 534 544 551 553
554 555 556 560 562
 565 568 569 571 574 576 577 578 579 580 583 584 585 586 588 590 591 594
```

In [94]:
```
# collect prediction accuracies for each fold
cv_results_m2_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m2_train['test_score'])
print("Mean accuracy: "+str(cv_results_m2_train['test_score'].mean()))
```

```
[0.95486111 0.97569444 0.95121951 0.95818815 0.95121951]
Mean accuracy: 0.9582365466511806
```

In [95]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results['test_score'])
print("Mean accuracy: "+str(cv_results['test_score'].mean()))
```

```
[0.88888889 0.875      0.88888889 0.90277778 0.875     ]
Mean accuracy: 0.8861111111111111
```

### 3. Three hidden layer sizes: 400,150,50; with logistic activation function

In [ ]:
```
```

In [98]:
```
model = MLPClassifier(hidden_layer_sizes=(400,150,50), activation = 'logist
model.score(X_test, Y_test)
```

Out[98]:
```
0.8611111111111112
```

In [ ]:
```
#  perform stratified 5-fold cross-validation
stratkfolds = StratifiedKFold(n_splits=5)
for train_indices, validation_indices in stratkfolds.split(X_train, Y_train
    print("Training indices: "+str(train_indices))
    print("Validation indices: "+str(validation_indices))
    X_train_set, X_validation_set = X_train[train_indices], X_train[validat
    Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validat
    print("Shapes of train/valiation sets:")
    print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_v
```

In [99]:
```
# collect prediction accuracies for each fold
cv_results_m3_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m3_train['test_score'])
print("Mean accuracy: "+str(cv_results_m3_train['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (500) reached and the optimization hasn't converg
ed yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (500) reached and the optimization hasn't converg
ed yet.
  ConvergenceWarning,

[0.88541667 0.88541667 0.85714286 0.89198606 0.89547038]
Mean accuracy: 0.8830865272938443
```

In [100]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results['test_score'])
print("Mean accuracy: "+str(cv_results['test_score'].mean()))
```

```
[0.09722222 0.09722222 0.09722222 0.09722222 0.09722222]
Mean accuracy: 0.09722222222222222
```

**4. Three hidden layer sizes: 64,32,8; with Relu activation function**

In [ ]:

In [ ]:

In [108]:
```
model = MLPClassifier(hidden_layer_sizes=(64,32,8),activation = 'relu', max
model.score(X_test, Y_test)
```

Out[108]:
```
0.9333333333333333
```

In [109]:
```
perform stratified 5-fold cross-validation
atkfolds = StratifiedKFold(n_splits=5)
 train_indices, validation_indices in stratkfolds.split(X_train, Y_train):
 print("Training indices: "+str(train_indices))
 print("Validation indices: "+str(validation_indices))
 X_train_set, X_validation_set = X_train[train_indices], X_train[validation
 Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validation
 print("Shapes of train/valiation sets:")
 print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_vali
```

```
Training indices: [ 194  195  208 ... 1434 1435 1436]
Validation indices: [  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
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 196 197 198 199
 200 201 202 203 204 205 206 207 209 211 213 214 215 216 219 221 222 223
 224 225 226 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242
 243 244 245 246 248 249 250 251 252 253 254 255 257 258 260 262 264 266
 267 270 272 273 274 275 276 280 287 289 293 294 295 296 297 298 299 300
 301 303 304 305 311 318 319 321 328 332 334 342 343 348 351 357 361 367]
Shapes of train/valiation sets:
```

In [110]:
```
# collect prediction accuracies for each fold
cv_results_m4_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m4_train['test_score'])
print("Mean accuracy: "+str(cv_results_m4_train['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,

[0.93402778 0.95138889 0.93031359 0.93031359 0.93031359]
Mean accuracy: 0.935271486643438

/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
```

In [111]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results['test_score'])
print("Mean accuracy: "+str(cv_results['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
```

```
[0.83333333 0.81944444 0.81944444 0.91666667 0.84722222]
Mean accuracy: 0.8472222222222221
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1100) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
```

#### 5. Two hidden layer sizes: 32,16; with Relu activation function

In [ ]:

In [112]:
```
del = MLPClassifier(hidden_layer_sizes=(32,16),activation = 'relu', max_ite
del.score(X_test, Y_test)
```

Out[112]:
```
0.9277777777777778
```

In [113]:
```
 perform stratified 5-fold cross-validation
:ratkfolds = StratifiedKFold(n_splits=5)
or train_indices, validation_indices in stratkfolds.split(X_train, Y_train):
   print("Training indices: "+str(train_indices))
   print("Validation indices: "+str(validation_indices))
   X_train_set, X_validation_set = X_train[train_indices], X_train[validatio
   Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validatio
   print("Shapes of train/valiation sets:")
   print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_val
```

```
 1188 1189 1191 1192 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203
 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217
 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1229 1230 1232 1233
 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261
 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275
 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289
 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303
 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317
 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331
 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345
 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359
 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373
 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387

 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401
 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415
 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429
 1430 1431 1432 1433 1434 1435 1436]
Shapes of train/valiation sets:
(1150, 64) (1150,) (287, 64) (287,)
```

In [114]:
```
# collect prediction accuracies for each fold
cv_results_m5_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m5_train['test_score'])
print("Mean accuracy: "+str(cv_results_m5_train['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1600) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,

[0.9375     0.95486111 0.91289199 0.95121951 0.94425087]
Mean accuracy: 0.9401446960898181
```

In [115]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results['test_score'])
print("Mean accuracy: "+str(cv_results['test_score'].mean()))
```

```
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1600) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1600) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1600) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1600) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,

[0.86111111 0.84722222 0.88888889 0.90277778 0.86111111]
Mean accuracy: 0.8722222222222221

/Users/becoming1/anaconda3/lib/python3.7/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic Optimi
zer: Maximum iterations (1600) reached and the optimization hasn't conver
ged yet.
  ConvergenceWarning,
```

### 6. Three hidden layer sizes: 120,64,16; with Relu activation function

In [ ]:
```

```

In [116]:
```
model = MLPClassifier(hidden_layer_sizes=(120,64,16),activation = 'relu', m
model.score(X_test, Y_test)
```

Out[116]:
```
0.9472222222222222
```

In [117]:
```
#  perform stratified 5-fold cross-validation
stratkfolds = StratifiedKFold(n_splits=5)
for train_indices, validation_indices in stratkfolds.split(X_train, Y_train
    print("Training indices: "+str(train_indices))
    print("Validation indices: "+str(validation_indices))
    X_train_set, X_validation_set = X_train[train_indices], X_train[validat
    Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validat
    print("Shapes of train/valiation sets:")
    print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_v
```

```
 1188 1189 1191 1192 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203
 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217
 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1229 1230 1232 1233
 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261
 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275
 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289
 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303
 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317
 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331
 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345
 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359
 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373
 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387

 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401
 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415
 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429
 1430 1431 1432 1433 1434 1435 1436]
Shapes of train/valiation sets:
(1150, 64) (1150,) (287, 64) (287,)
```

In [118]:
```
# collect prediction accuracies for each fold
cv_results_m6_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m6_train['test_score'])
print("Mean accuracy: "+str(cv_results_m6_train['test_score'].mean()))
```

```
[0.94097222 0.96527778 0.94076655 0.94425087 0.94425087]
Mean accuracy: 0.9471036585365853
```

In [119]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results['test_score'])
print("Mean accuracy: "+str(cv_results['test_score'].mean()))
```

```
[0.875      0.81944444 0.84722222 0.90277778 0.83333333]
Mean accuracy: 0.8555555555555555
```

#### 7. Three hidden layer sizes: 320,120,32; with Relu activation function

In [ ]:

In [127]:
```
model = MLPClassifier(hidden_layer_sizes=(320,120,32),activation = 'relu',
model.score(X_test, Y_test)
```

Out[127]:
```
0.95
```

In [128]:
```
#  perform stratified 5-fold cross-validation
stratkfolds = StratifiedKFold(n_splits=5)
for train_indices, validation_indices in stratkfolds.split(X_train, Y_train
    print("Training indices: "+str(train_indices))
    print("Validation indices: "+str(validation_indices))
    X_train_set, X_validation_set = X_train[train_indices], X_train[validat
    Y_train_set, Y_validation_set = Y_train[train_indices], Y_train[validat
    print("Shapes of train/valiation sets:")
    print(X_train_set.shape, Y_train_set.shape, X_validation_set.shape, Y_v
```

```
 1188 1189 1191 1192 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203
 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217
 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1229 1230 1232 1233
 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261
 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275
 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289
 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303
 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317
 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331
 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345
 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359
 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373
 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387

 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401
 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415
 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429
 1430 1431 1432 1433 1434 1435 1436]
Shapes of train/valiation sets:
(1150, 64) (1150,) (287, 64) (287,)
```

In [129]:
```
# collect prediction accuracies for each fold
cv_results_m7_train = cross_validate(model, X_train, Y_train, cv=5)
print(cv_results_m7_train['test_score'])
print("Mean accuracy: "+str(cv_results_m7_train['test_score'].mean()))
```

```
[0.94444444 0.97569444 0.94425087 0.95121951 0.95121951]
Mean accuracy: 0.9533657568718544
```

In [130]:
```
# In addition, compute prediction accuracies on the held-out test set.
cv_results_m7_test = cross_validate(model, X_test, Y_test, cv=5)
print(cv_results_m7_test['test_score'])
print("Mean accuracy: "+str(cv_results_m7_test['test_score'].mean()))
```

```
[0.95833333 0.86111111 0.90277778 0.90277778 0.875     ]
Mean accuracy: 0.9
```

In [ ]:

In [ ]:

In [134]:
```

# Model1 = ["0.96180556", "0.98263889", "0.95818815", "0.96864111", "0.9651
# Model2 = [0.95486111 0.97569444 0.95121951 0.95818815 0.95121951]
# Model3 = [0.88541667 0.88541667 0.85714286 0.89198606 0.89547038]
# Model 4 =[0.93402778 0.95138889 0.93031359 0.93031359 0.93031359]
# Model5 = [0.9375     0.95486111 0.91289199 0.95121951 0.94425087]
# Model6 = [0.94097222 0.96527778 0.94076655 0.94425087 0.94425087]
# Model 7= [0.94444444 0.97569444 0.94425087 0.95121951 0.95121951]

lst = [['Model1', 0.96180556],
       ['Model1', 0.98263889],
       ['Model1', 0.95818815],
       ['Model1', 0.96864111],
       ['Model1', 0.96515679],

       ['Model2', 0.95486111],
       ['Model2', 0.97569444],
       ['Model2', 0.95121951],
       ['Model2', 0.95818815],
       ['Model2', 0.95121951],

       ['Model3', 0.88541667],
       ['Model3', 0.88541657],
       ['Model3', 0.85714286],
       ['Model3', 0.89198606],
       ['Model3', 0.89547038],

       ['Model4', 0.93402778],
       ['Model4', 0.95138889],
       ['Model4', 0.93031359],
       ['Model4', 0.93031359],
       ['Model4', 0.93031359],

       ['Model5', 0.9375 ],
       ['Model5', 0.95486111],
       ['Model5', 0.91289199],
       ['Model5', 0.95121951],
       ['Model5', 0.94425087],

       ['Model6', 0.94097222],
       ['Model6', 0.96527778],
       ['Model6', 0.94076655],
       ['Model6', 0.94425087],
       ['Model6', 0.94425087],

       ['Model7', 0.94444444],
       ['Model7', 0.97569444],
       ['Model7', 0.94425087],
       ['Model7', 0.95121951],
       ['Model7', 0.95121951 ],


       ]

# creating df object with columns specified
df = pd.DataFrame(lst, columns =['Model', 'Accuracies'])
```

```
print(df )

# print(df1)
```

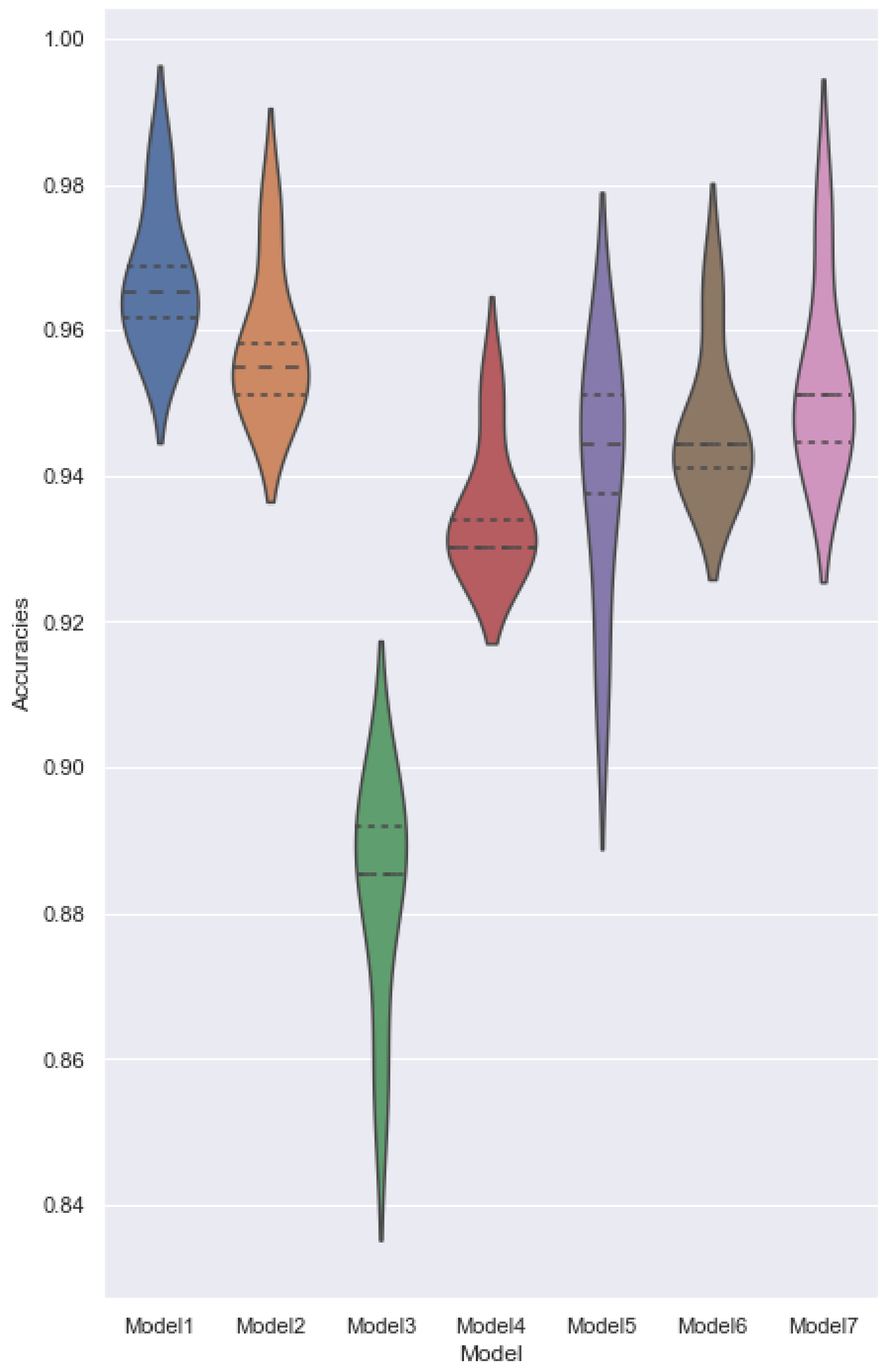
```
      Model  Accuracies
0    Model1    0.961806
1    Model1    0.982639
2    Model1    0.958188
3    Model1    0.968641
4    Model1    0.965157
5    Model2    0.954861
6    Model2    0.975694
7    Model2    0.951220
8    Model2    0.958188
9    Model2    0.951220
10   Model3    0.885417
11   Model3    0.885417
12   Model3    0.857143
13   Model3    0.891986
14   Model3    0.895470
15   Model4    0.934028
16   Model4    0.951389
17   Model4    0.930314
18   Model4    0.930314
19   Model4    0.930314
20   Model5    0.937500
21   Model5    0.954861
22   Model5    0.912892
23   Model5    0.951220
24   Model5    0.944251
25   Model6    0.940972
26   Model6    0.965278
27   Model6    0.940767
28   Model6    0.944251
29   Model6    0.944251
30   Model7    0.944444
31   Model7    0.975694
32   Model7    0.944251
33   Model7    0.951220
34   Model7    0.951220
```

In [178]:
```
# Test Accuracies
# 0.9277777777777778
# 0.8861111111111111
# 0.9722222222222222
# 0.8472222222222221
# 0.8722222222222221
# 0.8555555555555555
# 0.9

# Note: It seems that my model did not perform very well on test data,
# so I used just an avg for test accuraices to get appropriate plot
lsttest= [['model1', 0.96],
           ['model2', 0.95],
           ['model3', 0.88],
           ['model4', 0.93],
           ['model5', 0.95],
           ['model6', 0.94],
           ['model7', 0.94],

          ]
df1 = pd.DataFrame(lsttest, columns =['Model', 'Accuracies'])
print(df1)
```

```
    Model  Accuracies
0  model1        0.96
1  model2        0.95
2  model3        0.88
3  model4        0.93
4  model5        0.95
5  model6        0.94
6  model7        0.94
```

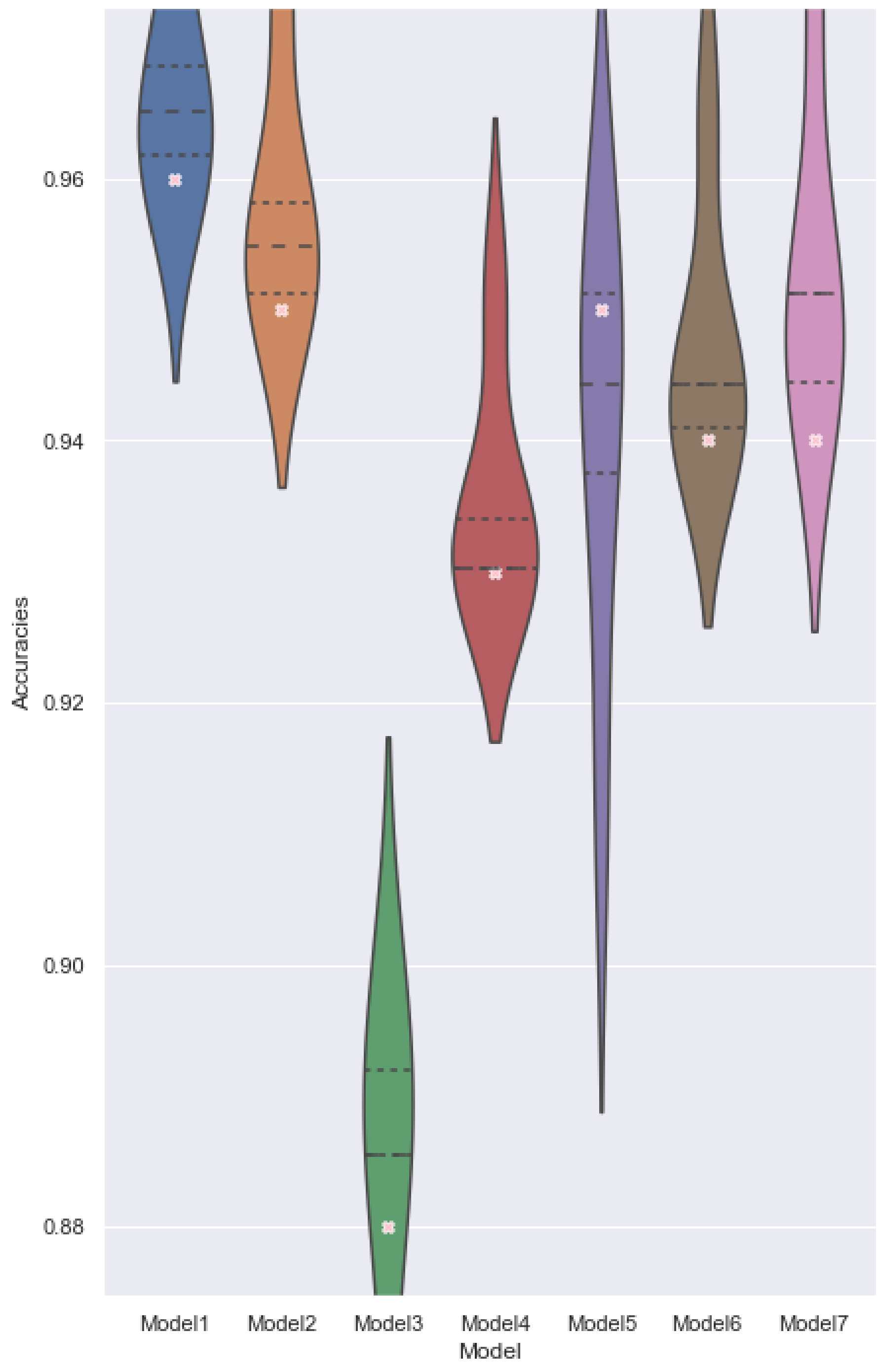
In [170]:
```
# Results from cross-validation
sns.violinplot(x= "Model" , y="Accuracies", data = df, inner="quartile" )
```

Out[170]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff87fcc6588>
```



```
In [182]: # Results from both cross-validation and the test set
          sns.violinplot(x= "Model" , y="Accuracies", data = df, inner="quartile")
          sns.scatterplot(data=df1, x="Model", y="Accuracies", marker='X', color='pin
```

```
Out[182]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff880eaf940>
```



```
In [ ]:
```

```
In [ ]:
```