The MNIST dataset of handwritten digits

SVMs – Classification (A)

KPCA+LDA (B)

Το Mnist dataset περιέχει 70.000 28 x 28 εικόνες από χειρόγραφα ψηφία από το 0 έως το 9.

A) SVMs Classification

Μέρος 1^{ο} : Εισαγωγή Δεδομένων & Χωρισμός σε Train, Validation & Test Sets.

Η εισαγωγή των δεδομένων έγινε μέσω των TensorFlow - Keras για ευκολία και αποτελεσματικότητα. Τα δεδομένα είναι ήδη χωρισμένα σε σύνολα εκπαίδευσης (train set) – δοκιμής (test set) με ποσοστό 90%-10%, επομένως για να τα χωρίσουμε σε 60%-40% τα ενώνουμε με την βοήθεια της συνάρτησης concatenate(). Επειδή το Mnist dataset είναι αρκετά μεγάλο επιλέγουμε ένα υποσύνολο 10.000 εικόνων προκείμενου να δουλέψουμε πάνω σε αυτό γρηγορότερα. Για τον διαχωρισμό των δεδομένων αρχικά τα μετατρέπουμε σε numpy arrays ώστε να χρησιμοποιήσουμε την συνάρτηση train_test_split() της scikit-learn. Δημιουργούμε επίσης από το σύνολο δοκιμής (test set) ένα έξτρα σύνολο επικύρωσης (validation set), το οποίο θα χρησιμοποιηθεί κατά τη διάρκεια της εκπαίδευσης.

Μέρος 2^{ο} : Προεπεξεργασία Δεδομένων για εκπαίδευση με SVM & Ταξινόμηση του Mnist Dataset.

Αρχικά με την βοήθεια της shape() εμφανίζουμε το σχήμα των δεδομένων εκπαίδευσης, επικύρωσης και δοκιμής και βλέπουμε τις διαστάσεις του πίνακα που περιέχει τις εικόνες. Κάθε εικόνα στο Mnist έχει διαστάσεις 28 x 28 pixels.

Κανονικοποίηση των εικόνων: Normalization of Data

Η «Κανονικοποίηση» εφαρμόζεται σε κάθε εικόνα ξεχωριστά, με βάση το σύνολο των τιμών pixel [0-255]. Έχει σκοπό να μειώσει την κλίμακα των τιμών σε [0, 1], διευκολύνοντας έτσι την εκπαίδευση του μοντέλου. Αυτή η μέθοδος διατηρεί τη δομή των εικόνων κατά τη διάρκεια της κανονικοποίησης.

Μετατροπή των εικόνων σε μονοδιάστατα διανύσματα: Reshaping of Images

Ο SVM δεν μπορεί να επεξεργαστεί δεδομένα εικόνας απευθείας σε μορφή 2D ή 3D. Επομένως, για να μπορέσει να δεχθεί τις εικόνες ως δεδομένα εισόδου και να ακολουθήσει η διαδικασία της εκπαίδευσης είναι απαραίτητη η μετατροπή των δεδομένων σε μονοδιάστατα διανύσματα (flat vectors) με τη βοήθεια της συνάρτησης reshape().

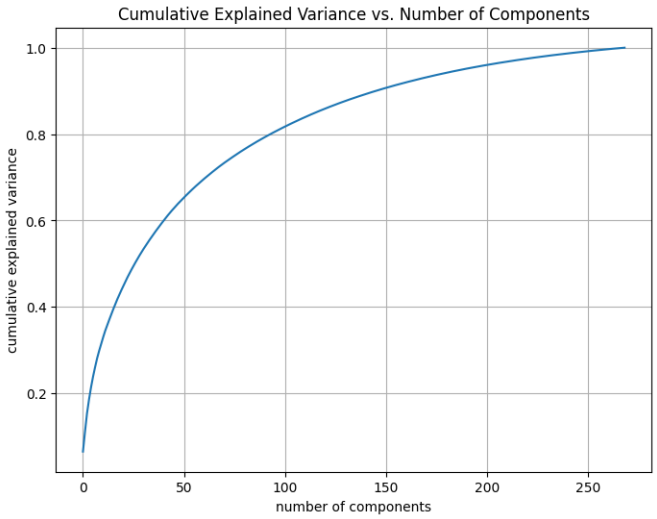
Τυποποίηση των εικόνων: Standardization of Images

Η «Τυποποίηση» χρησιμοποιείται προκειμένου να εξισώσει όλα τα χαρακτηριστικά των εικόνων , ώστε να έχουν μέσο όρο ίσο με 0 (mean=0) και τυπική απόκλιση ίση με 1 (mean=1). Αυτό μπορεί να αποδειχτεί ιδιαίτερα χρήσιμο κατά την χρήση της PCA (Principal

Component Analysis), καθώς εξασφαλίζει ότι όλα τα χαρακτηριστικά έχουν ίση βαρύτητα ανάλυσης.

Διάγραμμα Συσσωρευμένης Εξηγούμενης Διακύμανσης: Cumulative Explained Variance

Το συγκεκριμένο διάγραμμα μας δείχνει πόση από τη συνολική διακύμανση εξηγούν οι πρώτες συνιστώσες. Είναι πολύ χρήσιμο για να καταλάβουμε πόσες συνιστώσες χρειαζόμαστε για να διατηρήσουμε την πλειονότητα της πληροφορίας. Όπως βλέπουμε και στο παρακάτω διάγραμμα από τις 250 συνιστώσες και πάνω φαίνεται να διατηρείται το μεγαλύτερο ποσοστό της πληροφορίας.



Μείωση Διάστασης Δεδομένων: Principal Component Analysis

Η PCA επιτυγχάνει τη μείωση διάστασης των δεδομένων κρατώντας τις πιο σημαντικές συνιστώσες. Η τεχνική αυτή προσπαθεί να βρει τους πιο σημαντικούς άξονες (ή αλλιώς κύριες συνιστώσες), στους οποίους οι παρατηρήσεις των δεδομένων έχουν τη μεγαλύτερη διακύμανση (variance). Μετά την εφαρμογή της PCA βλέπουμε ότι για την διατήρηση του 95% της πληροφορίας χρειάζονται 269 συνιστώσες.

Μέρος 3^{ο}: Εκπαίδευση Μοντέλου

Η εκπαίδευση του SVM δοκιμάστηκε με διαφορετικούς πυρήνες (Linear, RBF, Polynomial Kernel). Οι Kernels είναι συναρτήσεις που επιτρέπουν στο SVM να μετασχηματίσει τα δεδομένα σε υψηλότερη διάσταση. Κάθε τύπος Kernel μπορεί να βοηθήσει το μοντέλο να μάθει διαφορετικούς τύπους διαχωριστικών υπέρ-επιπέδων.

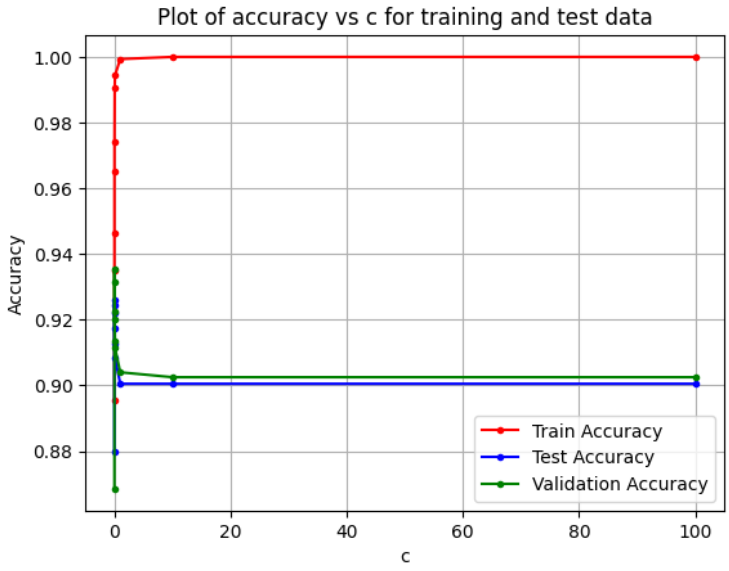
Αρχικά δοκιμάζουμε διαφορετικούς Kernels με τις default παραμέτρους τους και βλέπουμε ποιος δίνει την καλύτερη ακρίβεια για το σύνολο επικύρωσης (validation set). Από τους τρεις Kernels την καλύτερη ακρίβεια την δίνει ο RBF Kernel (accuracy=0.935). Στη συνέχεια αξιολογούμε το μοντέλο με τον καλύτερο Kernel, κάνοντας πρόβλεψη στο σύνολο δοκιμών

(test set). Για το test set το accuracy = 0.9385, δηλαδή και ελάχιστα υψηλότερο από αυτό του validation set.

SVM Linear Kernel

Δημιουργούμε μία μέθοδο «Linear Kernel Model» και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου C (regularization parameter). Όσο μικρότερη είναι η τιμή της παραμέτρου C τόσο πιο «ελαστικό» είναι το μοντέλο, άρα αφήνει το να γίνουν περισσότερα λάθη κατά την διάρκεια της εκπαίδευσης. Από την άλλη πλευρά όσο πιο πολύ μεγαλώνει το c, τόσο το μοντέλο γίνεται πιο «σφιχτό» και δεν αποτρέπει τα λάθη κατά την εκπαίδευση του.

Ορίζουμε ένα πίνακα με διαφορετικές τιμές της παραμέτρου C από 100 και κάτω. Ο πίνακας περιλαμβάνει τις τιμές C= [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.01, 1, 10, 100]. Η μεγαλύτερη ακρίβεια για το test set επιτυγχάνεται για C=0.005. Παρατηρούμε ότι για C από 0.01 και πάνω η ακρίβεια αρχίζει να έχει καθοδική πορεία.



	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	linear	0.0001	0.895333	0.8685	0.8800
1	linear	0.0005	0.935167	0.9135	0.9175
2	linear	0.0010	0.946333	0.9225	0.9220
3	linear	0.0050	0.965333	0.9355	0.9260
4	linear	0.0100	0.974167	0.9315	0.9245
5	linear	0.0500	0.990667	0.9200	0.9125
6	linear	0.1000	0.994333	0.9115	0.9085
7	linear	1.0000	0.999333	0.9040	0.9005
8	linear	10.0000	1.000000	0.9025	0.9005
9	linear	100.0000	1.000000	0.9025	0.9005

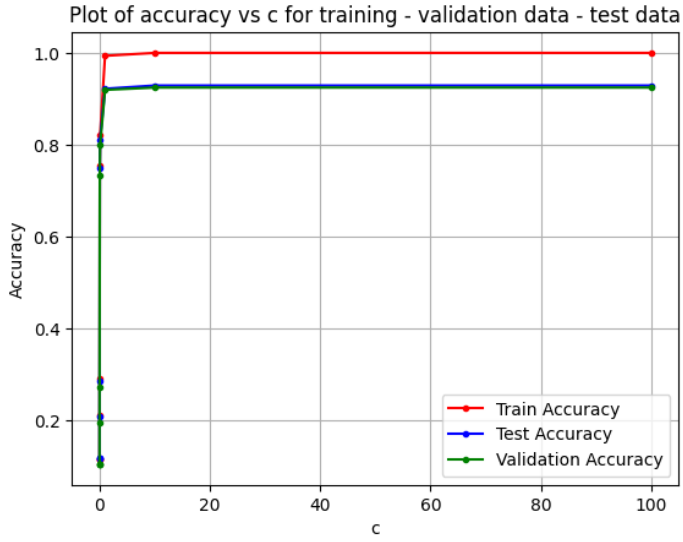
	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	linear	0.0001	0.8800	0.866492	0.4405	1.464411
1	linear	0.0005	0.9175	0.908248	0.2920	1.170043
2	linear	0.0010	0.9220	0.913258	0.2620	1.094075
3	linear	0.0050	0.9260	0.917709	0.2545	1.079583
4	linear	0.0100	0.9245	0.916042	0.2595	1.090183
5	linear	0.0500	0.9125	0.902692	0.2915	1.140833
6	linear	0.1000	0.9085	0.898247	0.3050	1.169188
7	linear	1.0000	0.9005	0.889348	0.3360	1.231666
8	linear	10.0000	0.9005	0.889348	0.3395	1.239960
9	linear	100.0000	0.9005	0.889348	0.3395	1.239960

	Kernel	C	Test Accuracy	Mean Kappa	Mean MAE	Mean RMSE
0	linear	Mean	0.90925	0.899073	0.312	1.19199

Radial Basis Kernel

Δημιουργούμε μία μέθοδο «RBF Model» και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου C, καθώς και διαφορετικά gamma (decision boundary). Αν η παράμετρος gamma είναι πολύ μεγάλη, τότε ο διαχωριστικός χώρος είναι πολύ στενός. Σε αυτή την περίπτωση για να θεωρήσει ο πυρήνας ότι δύο σημεία είναι όμοια, θα πρέπει τα σημεία αυτά να είναι πολύ κοντά μεταξύ τους, γεγονός που μπορεί να οδηγήσει σε υπέρ-εκπαίδευση του μοντέλου. Από την άλλη, στην περίπτωση που έχουμε πολύ μικρό gamma, έχουμε μεγαλύτερο πεδίο επιρροής για κάθε δείγμα, δηλαδή το μοντέλο θεωρεί ότι δύο σημεία είναι όμοια ακόμα και αν δεν βρίσκονται τόσο κοντά μεταξύ τους, κάτι που μπορεί να οδηγήσει και σε υπό-εκπαίδευση του μοντέλου.

Ορίζουμε ένα πίνακα με διαφορετικές τιμές της παραμέτρου C από 100 και κάτω, καθώς και g=auto. Ο πίνακας περιλαμβάνει τις τιμές C= [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.01, 1, 10, 100]. Η μεγαλύτερη ακρίβεια για το test set επιτυγχάνεται για C=10. Ο RBF Kernel σε σχέση με τον Linear Kernel πετυχαίνει καλύτερη ακρίβεια για μεγαλύτερη τιμή C.

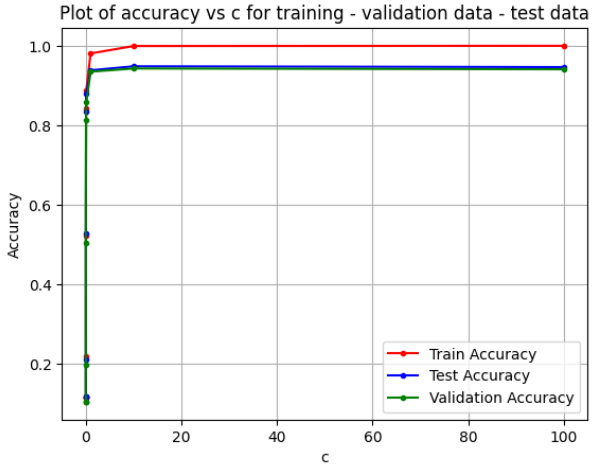


	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	RBF	0.0001	0.113667	0.1045	0.1180
1	RBF	0.0005	0.113667	0.1045	0.1180
2	RBF	0.0010	0.113667	0.1045	0.1180
3	RBF	0.0050	0.212000	0.1955	0.2085
4	RBF	0.0100	0.290167	0.2730	0.2845
5	RBF	0.0500	0.755667	0.7330	0.7495
6	RBF	0.1000	0.820500	0.7990	0.8115
7	RBF	1.0000	0.993667	0.9195	0.9220
8	RBF	10.0000	1.000000	0.9245	0.9290
9	RBF	100.0000	1.000000	0.9245	0.9290

	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	RBF	0.0001	0.1180	0.000000	3.6075	4.463687
1	RBF	0.0005	0.1180	0.000000	3.6075	4.463687
2	RBF	0.0010	0.1180	0.000000	3.6075	4.463687
3	RBF	0.0050	0.2085	0.117953	2.7975	3.660943
4	RBF	0.0100	0.2845	0.204615	2.2580	3.101129
5	RBF	0.0500	0.7495	0.721137	0.8765	2.016309
6	RBF	0.1000	0.8115	0.790273	0.6705	1.776936
7	RBF	1.0000	0.9220	0.913249	0.2775	1.133799
8	RBF	10.0000	0.9290	0.921035	0.2575	1.100227
9	RBF	100.0000	0.9290	0.921035	0.2575	1.100227

	Kernel	C	Test Accuracy	Mean Kappa	Mean MAE	Mean RMSE
0	rbf	Mean	0.5188	0.45893	1.82175	2.728063

Στη συνέχεια δοκιμάζουμε για g=scale και παρατηρούμε ότι δίνει λίγο καλύτερη ακρίβεια σε σχέση με το auto. Σε γενικές γραμμές ο RBF Kernel για το συγκεκριμένο dataset δίνει καλύτερη ακρίβεια για υψηλότερα C (1, 10, 100), σε σχέση με τον Linear Kernel.



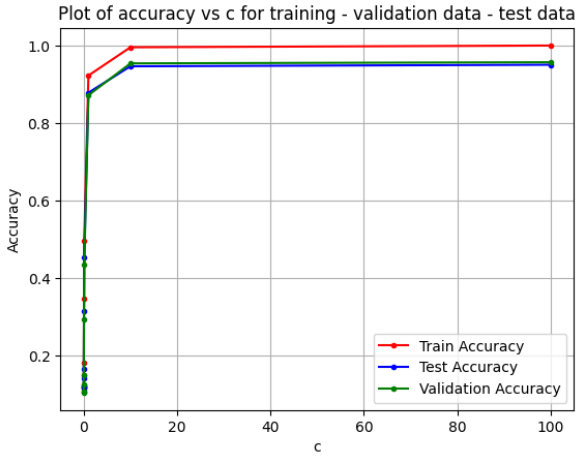
	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	RBF	0.0001	0.113667	0.1045	0.1180
1	RBF	0.0005	0.113667	0.1045	0.1180
2	RBF	0.0010	0.113667	0.1045	0.1180
3	RBF	0.0050	0.219500	0.1975	0.2110
4	RBF	0.0100	0.523167	0.5035	0.5280
5	RBF	0.0500	0.843000	0.8140	0.8355
6	RBF	0.1000	0.888667	0.8575	0.8800
7	RBF	1.0000	0.980833	0.9350	0.9385
8	RBF	10.0000	0.999500	0.9430	0.9485
9	RBF	100.0000	1.000000	0.9410	0.9465

	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	RBF	0.0001	0.1180	0.000000	3.6075	4.463687
1	RBF	0.0005	0.1180	0.000000	3.6075	4.463687
2	RBF	0.0010	0.1180	0.000000	3.6075	4.463687
3	RBF	0.0050	0.2110	0.116248	3.0375	3.943032
4	RBF	0.0100	0.5280	0.473912	1.4720	2.498399
5	RBF	0.0500	0.8355	0.816959	0.5820	1.645600
6	RBF	0.1000	0.8800	0.866523	0.4295	1.413683
7	RBF	1.0000	0.9385	0.931606	0.2060	0.970567
8	RBF	10.0000	0.9485	0.942728	0.1670	0.873499
9	RBF	100.0000	0.9465	0.940504	0.1755	0.895824

	Kernel	C	Test Accuracy	Mean Kappa	Mean MAE	Mean RMSE
0	rbf	Mean	0.5642	0.508848	1.6892	2.563166

Polynomial Kernel

Για τον «Polynomial Kernel» δημιουργούμε επίσης μία μέθοδο και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου C= [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.01, 1, 10, 100]. Για τον συγκεκριμένο πυρήνα παρατηρούμε ότι σε μεγαλύτερες τιμές C (10 & 100) πετυχαίνει αισθητά μεγαλύτερη ακρίβεια, καθιστώντας τον καλύτερο πυρήνα για classification από τους τρεις για το συγκεκριμένο dataset.



	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	Polynomial	0.0001	0.115000	0.1055	0.1190
1	Polynomial	0.0005	0.119167	0.1075	0.1215
2	Polynomial	0.0010	0.122333	0.1085	0.1230
3	Polynomial	0.0050	0.149333	0.1255	0.1415
4	Polynomial	0.0100	0.181833	0.1500	0.1660
5	Polynomial	0.0500	0.348167	0.2940	0.3155
6	Polynomial	0.1000	0.495667	0.4360	0.4550
7	Polynomial	1.0000	0.922667	0.8720	0.8785
8	Polynomial	10.0000	0.995500	0.9540	0.9465
9	Polynomial	100.0000	1.000000	0.9570	0.9505

	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	Polynomial	0.0001	0.1190	0.001182	3.6005	4.458195
1	Polynomial	0.0005	0.1215	0.004066	3.5935	4.454941
2	Polynomial	0.0010	0.1230	0.005788	3.5895	4.453257
3	Polynomial	0.0050	0.1415	0.027197	3.5375	4.425099
4	Polynomial	0.0100	0.1660	0.055554	3.4740	4.392721
5	Polynomial	0.0500	0.3155	0.227431	2.9420	4.063742
6	Polynomial	0.1000	0.4550	0.386868	2.3490	3.641703
7	Polynomial	1.0000	0.8785	0.864915	0.4670	1.513605
8	Polynomial	10.0000	0.9465	0.940511	0.1920	0.943928
9	Polynomial	100.0000	0.9505	0.944959	0.1750	0.901665

Μέθοδος Cross Validation – Grid Search

Δοκιμάζουμε 5-fold και 10-fold cross validation για να βρούμε τις καλύτερες παραμέτρους.

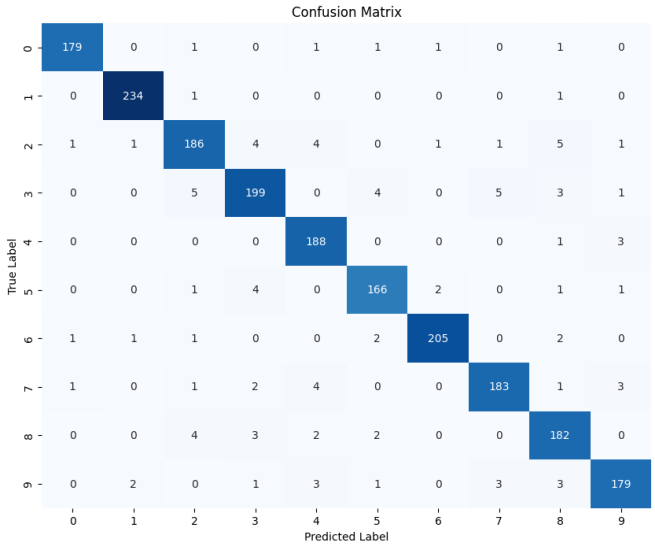
Για 5-fold cross validation: Ακρίβεια του test set, acc=0.9505

	Kernel	Best Parameters	Accuracy
0	linear	{'C': 0.005}	0.927167
1	poly	{'C': 10, 'gamma': 'auto'}	0.946500
2	rbf	{'C': 10, 'gamma': 'scale'}	0.936500

Για 10-fold cross validation: Ακρίβεια του test set, acc=0.9505

	Kernel	Best Parameters	Accuracy
0	linear	{'C': 0.005}	0.929000
1	poly	{'C': 100, 'gamma': 'scale'}	0.953167
2	rbf	{'C': 10, 'gamma': 'scale'}	0.938667

Από τα 5 & 10 Fold Cross Validation παρατηρούμε ότι η ακρίβεια στα test set εν τέλη είναι η ίδια. Ωστόσο, αν προσέξουμε τους πίνακες με τις καλύτερες παραμέτρους, που δίνουν τη μέγιστη ακρίβεια για το train set, θα διαπιστώσουμε ότι στην περίπτωση του Polynomial Kernel είναι διαφορετικές οι καλύτερες παράμετροι. Άρα, οι Polynomial και RBF δίνουν το ίδιο αποτέλεσμα για διαφορετικές παραμέτρους ο ένα για 5-fold Cross Validation και ο άλλος για 10-fold Cross Validation.




```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       184
           1       0.98      0.99      0.99       236
           2       0.93      0.91      0.92       204
           3       0.93      0.92      0.93       217
           4       0.93      0.98      0.95       192
           5       0.94      0.95      0.95       175
           6       0.98      0.97      0.97       212
           7       0.95      0.94      0.95       195
           8       0.91      0.94      0.93       193
           9       0.95      0.93      0.94       192

    accuracy                           0.95      2000
   macro avg       0.95      0.95      0.95      2000
weighted avg       0.95      0.95      0.95      2000
```

Μέρος 4^{o} : Εμφάνιση Σωστών και Λανθασμένων Προβλέψεων

Αρχικά εμφανίζουμε τα πρώτα 20 δείγματα του test set και τα πρώτα 20 που προβλέψαμε. Παρατηρούμε ότι το 3^{o} και το 14^{o} δείγμα έχουν προβλεφθεί λάθος.

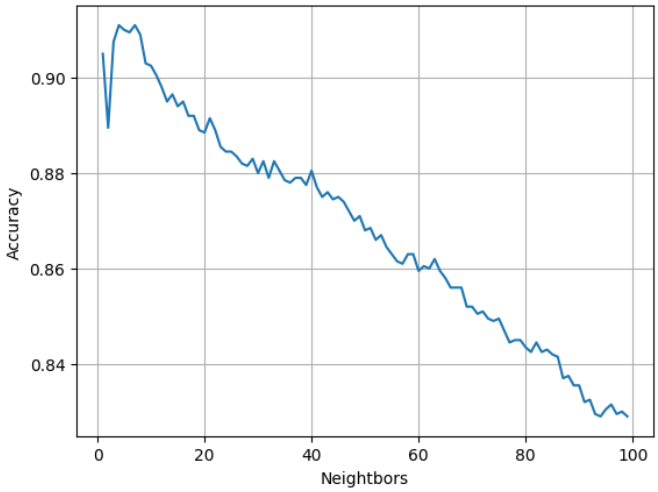
```
Correct Predictions: [0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19]
Incorrect Predictions: [3, 14]
```

	Prediction	Actual Value
0	9	9
1	8	8
2	0	0
3	2	6
4	0	0
5	9	9
6	6	6
7	4	4
8	5	5
9	8	8
10	0	0
11	0	0
12	2	2
13	8	8
14	2	0
15	4	4
16	6	6
17	9	9
18	1	1
19	6	6

K-NN: Με Μείωση Διαστασιμότηρας (PCA)

Εκπαιδεύουμε τον KNN με μείωση διαστασιμότητας (PCA) και βρίσκουμε για ποιον γείτονα το μοντέλο μας δίνει την καλύτερη απόδοση. Στην περίπτωση αυτή ο καλύτερος αριθμός γειτόνων ισούται με 4.

Από διάγραμμα Accuracy – Neighbors βλέπουμε πώς μεταβάλλεται η ακρίβεια σε συνάρτηση με τον αριθμό των γειτόνων σε εύρος από 1 έως 100 (Best Neighbor = 4 με Best Accuracy acc= 0.911). Από το διάγραμμα παρατηρούμε ότι καθώς αυξάνονται οι γείτονες η ακρίβεια πέφτει.



Εκτελούμε 5-fold και 10-fold Cross Validation – Grid Search για διαφορετικές τιμές παραμέτρων weights = [‘uniform’, ‘distance’] και metric = [‘euclidean’, ‘manhattan’], καθώς και για διαφορετικούς αριθμούς γειτόνων σε ένα εύρος 1 έως 10 γύρω από τον καλύτερο γείτονα που βρήκαμε πριν.

Για 5-fold Cross Validation: Validation Set Accuracy = 0.92, Test Set Accuracy = 0.917

Best Hyper parameters: metric = euclidean, n_neighbors=4, weights = distance

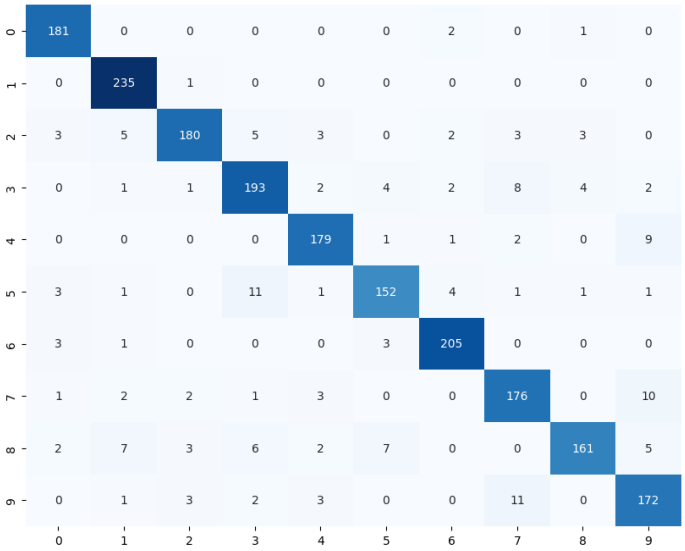
```
Validation Set Accuracy: 0.92
Test Set Accuracy: 0.917
```

Για 10-fold Cross Validation: Validation Set Accuracy = 0.92, Test Set Accuracy = 0.917

Best Hyper parameters: metric = euclidean, n_neighbors=4, weights = distance

```
Validation Set Accuracy: 0.92
Test Set Accuracy: 0.917
```

Συμπέρασμα: Η ακρίβεια παραμένει η ίδια είτε για 5-fold cross validation είτε για 10-fold cross validation.




```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       184
           1       0.93      1.00      0.96       236
           2       0.95      0.88      0.91       204
           3       0.89      0.89      0.89       217
           4       0.93      0.93      0.93       192
           5       0.91      0.87      0.89       175
           6       0.95      0.97      0.96       212
           7       0.88      0.90      0.89       195
           8       0.95      0.83      0.89       193
           9       0.86      0.90      0.88       192

    accuracy                           0.92      2000
   macro avg       0.92      0.92      0.92      2000
weighted avg       0.92      0.92      0.92      2000
```

Σωστές και Λανθασμένες Προβλέψεις

Παρατηρούμε ότι τα δείγματα 27 και 29 έχουν προβλεφθεί λάθος.

```
Correct Predictions: [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, 28]
Incorrect Predictions: [27, 29]
```

	Prediction	Actual Value
0	9	9
1	8	8
2	0	0
3	6	6
4	0	0
5	9	9
6	6	6
7	4	4
8	5	5
9	8	8
10	0	0
11	0	0
12	2	2
13	8	8
14	0	0
15	4	4
16	6	6
17	9	9
18	1	1
19	6	6
20	1	1
21	4	4
22	2	2
23	3	3
24	7	7
25	0	0
26	0	0
27	7	9
28	8	8
29	9	3

Συμπέρασμα: Τόσο ο SVM όσο και ο KNN δίνουν αρκετά υψηλή ακρίβεια στο test set για το συγκεκριμένο Dataset. Ο SVM έχει ελαφρώς υψηλότερη απόδοση.

NCC: Με Μείωση Διαστασιμότηρας (PCA)

Κάνουμε μείωση διαστασιμότητας και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου metric = [‘euclidean’, ‘manhattan’] και εκτελούμε 5-fold και 10-fold Cross Validation.

Για 5-fold Cross Validation: Test Set Accuracy = 0.81

Για 10-fold Cross Validation: Test Set Accuracy = 0.81

```
Best Model Parameters: {'metric': 'manhattan'}
Best Model Accuracy (CV): 0.8096666666666668
```

```
Validation Set Accuracy: 0.793
Test Set Accuracy: 0.81
```

Συμπέρασμα: Η ακρίβεια για 5 & 10 fold Cross Validation είναι η ίδια.

Σωστές και Λανθασμένες Προβλέψεις

```
Correct Predictions: [1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15, 16, 17, 18, 19]
Incorrect Predictions: [0, 2, 3, 10, 14]
```

```
NCC Test set Accuracy: 0.81
    Prediction  Actual Value
0            4             9
1            8             8
2            8             0
3            2             6
4            0             0
5            9             9
6            6             6
7            4             4
8            5             5
9            8             8
10           8             0
11           0             0
12           2             2
13           8             8
14           2             0
15           4             4
16           6             6
17           9             9
18           1             1
19           6             6
```

Συμπέρασμα: Ο NCC δεν έχει την ίδια ικανότητα να «μάθει» τα χαρακτηριστικά από τα δεδομένα, επομένως έχει και τη χειρότερη απόδοση σε σχέση μετ τους KNN και SVM.

B) KPCA + LDA

Μέρος 1^{ο} : Εισαγωγή

Η Kernel Principal Component Analysis (KPCA) είναι μία μέθοδος που βασίζεται στην κλασική Principal Component Analysis (PCA) , αλλά επιτρέπει τη μείωση διάστασης σε μη-γραμμικά δεδομένα μέσω της χρήσης ενός πυρήνα (Kernel).

Η PCA που εφαρμόστηκε στην προηγούμενη εργασία έχει σκοπό να βρει τις κύριες συνιστώσες (n_components), που μεγιστοποιούν τη διακύμανση των δεδομένων σε έναν νέο μειωμένο χώρο, διατηρώντας ταυτόχρονα τη μέγιστη πληροφορία. Ωστόσο, η PCA προϋποθέτει τα δεδομένα να είναι γραμμικά, δηλαδή οι σχέσεις μεταξύ των χαρακτηριστικών να μπορούν να περιγραφούν με ευθείες γραμμές.

Η Kernel PCA χρησιμοποιεί μια μη-γραμμική συνάρτηση μετασχηματισμού kernel, η οποία μεταφέρει τα δεδομένα σε έναν υψηλότερης διάστασης χώρο. Στο νέο αυτό χώρο, η γραμμική ανάλυση μπορεί να εφαρμοστεί και να ανιχνεύσει σύνθετες, μη γραμμικές σχέσεις.

Η Linear Discriminant Analysis (LDA) είναι μία μέθοδος μείωσης διάστασης που προσπαθεί να μεγιστοποιήσει τη διακριτικότητα των κλάσεων. Σε αντίθεση με το PCA που εστιάζει στο να μεγιστοποιήσει την διακύμανση των χαρακτηριστικών, η LDA εστιάζει στο να αυξήσει τη διαφορά μεταξύ των κατηγοριών και να μειώσει τη διασπορά μεταξύ εντός των κατηγοριών (το πόσο διασκορπισμένα είναι τα δεδομένα γύρω από τη μέση τιμή τους). Η LDA προσπαθεί να βρει ένα σύνολο γραμμικών συνδυασμών των χαρακτηριστικών προκειμένου να μεγιστοποιήσει τη διακριτικότητα μεταξύ των κατηγοριών. Επιλέγει τον συνδυασμό των καλύτερων χαρακτηριστικών, που καθιστούν τη διάκριση μεταξύ των κλάσεων πιο εύκολη.

Μέρος 2^{ο} : Εφαρμογή KPCA + LDA

Δοκιμάζουμε για διαφορετικούς Kernels (linear, polynomial & rbf) να εφαρμόσουμε KPCA & LDA και στη συνέχεια εκπαιδεύουμε τον SVM με linear για τα διαφορετικά KPCA, με διαφορετικές τιμές C (regularization parameter).

Ορίζουμε τους τύπους πυρήνων (Kernels = linear, polynomial, rbf) και τις τιμές της παραμέτρου C = [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 1] για τις οποίες θα εκπαιδεύσουμε τον SVM. Τα αποτελέσματα θα αποθηκευτούν σε έναν πίνακα results = [].

Δημιουργούμε μία for loop, η οποία θα προσπελάσει και τους 3 τύπους kernel του πίνακα kernels = [‘linear’, ‘poly’, ‘rbf’] και θα δημιουργήσει ένα αντικείμενο KernelPCA, το οποίο θα το εφαρμόζουμε στο σύνολο εκπαίδευσης (train set) των δεδομένων. Με την fit_transform() μειώνουμε τη διάσταση των δεδομένων χρησιμοποιώντας τον καθορισμένο πυρήνα κάθε φορά.

Υπολογίζουμε την συνολική εξηγούμενη διακύμανση. Η παράμετρος kpca.eigenvalues_ περιέχει τις ιδιοτιμές του μετασχηματισμένου χώρου. Η συνάρτηση cumsum() υπολογίζει την συνολική (αθροιστική) εξηγούμενη διακύμανση των συνιστωσών, την οποία την διαιρούμε το άθροισμα όλων των ιδιοτιμών. Η explained_variance_ratio μας δείχνει πόσο ποσοστό της συνολικής διακύμανσης εξηγείται από κάθε συνιστώσα.

Για την επιλογή του αριθμού των συνιστωσών που εξηγούν το 95% της συνολικής διακύμανσης χρησιμοποιούμε τη συνάρτηση argmax(), η οποία επιστρέφει το πρώτο σημείο στο οποίο η συνολικής διακύμανση ξεπερνάει το 95%.

Εφαρμόζουμε kpca με τον επιλεγμένο αριθμό συνιστωσών που υπολογίσαμε (n_components_95) και κάνουμε fit_transform() στα δεδομένα εκπαίδευση και transform() στα δεδομένα επικύρωσης και ελέγχου.

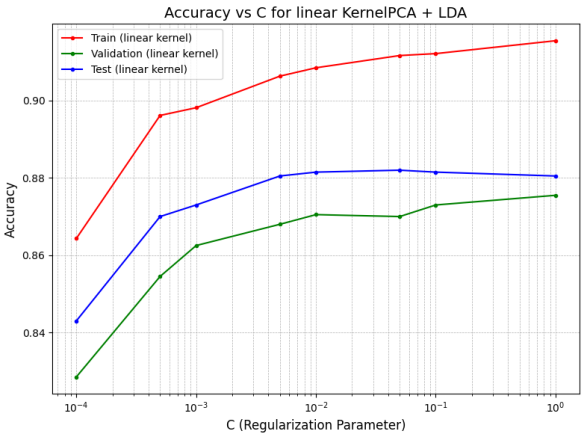
Στη συνέχεια χρησιμοποιούμε τα δεδομένα που προκύπτουν από το Kernel PCA και εφαρμόζουμε Linear Discriminant Analysis (LDA), για να βελτιώσουμε τη διάκριση μεταξύ των κατηγοριών.

Ο SVM εκπαιδεύεται ξεχωριστά με linear kernel για κάθε παράμετρο C του πίνακα που έχουμε ορίσει. Για κάθε εκπαίδευση πραγματοποιούνται προβλέψεις για το σύνολα επικύρωσης και ελέγχου και υπολογίζεται η ακρίβεια του μοντέλου. Τα αποτελέσματα αυτά αποθηκεύονται στον πίνακα results=[] που έχουμε δημιουργήσει.

Η μετρική '**MAE'** μας δείχνει την μέση απόλυτη διαφορά των προβλεπόμενων τιμών με τις πραγματικές τιμές. Το **'RMSE'** η ρίζα τού μέσου τετραγωνικού σφάλματος. Το **'Kappa'** αξιολογεί τη συμφωνία μεταξύ των δύο συνόλων λαμβάνοντας υπόψη και τον παράγοντα της τύχης. Για προβλήματα ταξινόμησης λαμβάνουμε υπόψη το kappa.

SVC (Linear Kernel) & Kernel PCA– Αποτελέσματα Ακρίβειας

1. **Linear KPCA**

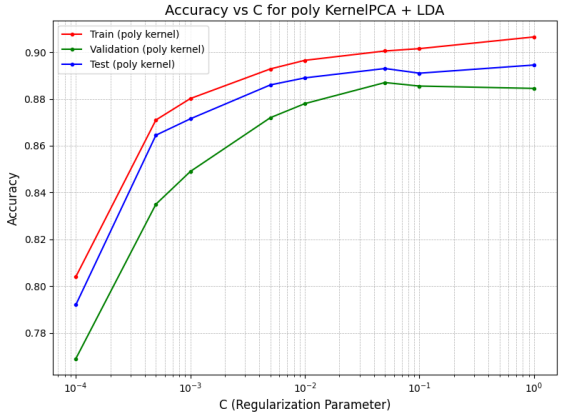


	Kernel (KPCA)	n_components	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	linear	269	0.0001	0.864333	0.8285	0.8430
1	linear	269	0.0005	0.896167	0.8545	0.8700
2	linear	269	0.0010	0.898167	0.8625	0.8730
3	linear	269	0.0050	0.906333	0.8680	0.8805
4	linear	269	0.0100	0.908500	0.8705	0.8815
5	linear	269	0.0500	0.911667	0.8700	0.8820
6	linear	269	0.1000	0.912167	0.8730	0.8815
7	linear	269	1.0000	0.915500	0.8755	0.8805

Μετρικές (Kappa, MAE, RMSE)

	C	Test Accuracy	Kernel (KPCA)	Kappa	MAE	RMSE
0	0.0001	0.8430	linear	0.825260	23.1900	1.598750
1	0.0005	0.8700	linear	0.855419	16.8885	1.453444
2	0.0010	0.8730	linear	0.858764	17.0100	1.427585
3	0.0050	0.8805	linear	0.867134	16.1025	1.380398
4	0.0100	0.8815	linear	0.868247	16.4820	1.355360
5	0.0500	0.8820	linear	0.868806	15.8510	1.362351
6	0.1000	0.8815	linear	0.868247	16.0940	1.367114
7	1.0000	0.8805	linear	0.867126	16.6095	1.379674

2. Polynomial Kernel PCA

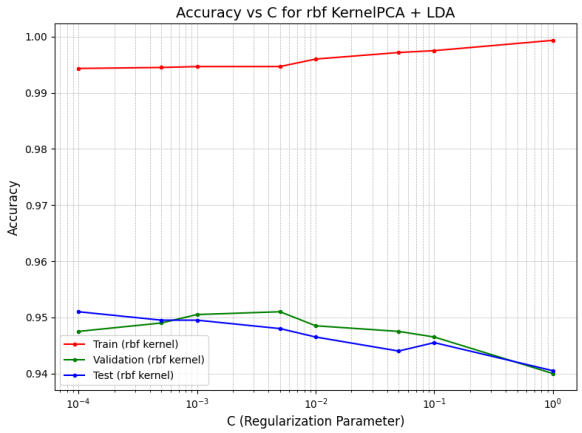


	Kernel (KPCA)	n_components	C	Train Accuracy	Validation Accuracy	Test Accuracy
8	poly	172	0.0001	0.804000	0.7690	0.7920
9	poly	172	0.0005	0.871000	0.8350	0.8645
10	poly	172	0.0010	0.880167	0.8490	0.8715
11	poly	172	0.0050	0.892833	0.8720	0.8860
12	poly	172	0.0100	0.896500	0.8780	0.8890
13	poly	172	0.0500	0.900500	0.8870	0.8930
14	poly	172	0.1000	0.901500	0.8855	0.8910
15	poly	172	1.0000	0.906500	0.8845	0.8945

Μετρικές (Kappa, MAE, RMSE)

	C	Test Accuracy	Kernel (KPCA)	Kappa	MAE	RMSE
8	0.0001	0.7920	poly	0.768252	36.6740	1.910236
9	0.0005	0.8645	poly	0.849265	17.9145	1.623730
10	0.0010	0.8715	poly	0.857074	17.3865	1.543211
11	0.0050	0.8860	poly	0.873226	16.3280	1.399643
12	0.0100	0.8890	poly	0.876566	15.8170	1.394274
13	0.0500	0.8930	poly	0.881020	15.4165	1.327968
14	0.1000	0.8910	poly	0.878797	16.0530	1.337161
15	1.0000	0.8945	poly	0.882689	15.6620	1.329286

3. RBF Kernel PCA



	Kernel (KPCA)	n_components	C	Train Accuracy	Validation Accuracy	Test Accuracy
16	rbf	2748	0.0001	0.994333	0.9475	0.9510
17	rbf	2748	0.0005	0.994500	0.9490	0.9495
18	rbf	2748	0.0010	0.994667	0.9505	0.9495
19	rbf	2748	0.0050	0.994667	0.9510	0.9480
20	rbf	2748	0.0100	0.996000	0.9485	0.9465
21	rbf	2748	0.0500	0.997167	0.9475	0.9440
22	rbf	2748	0.1000	0.997500	0.9465	0.9455
23	rbf	2748	1.0000	0.999333	0.9400	0.9405

Μετρικές (Kappa, MAE, RMSE)

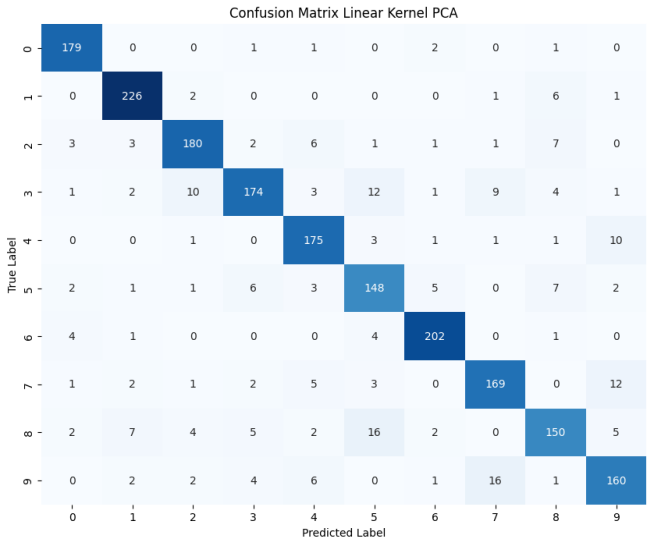
	C	Test Accuracy	Kernel (KPCA)	Kappa	MAE	RMSE
16	0.0001	0.9510	rbf	0.945511	5.2810	0.959687
17	0.0005	0.9495	rbf	0.943844	6.5340	0.918695
18	0.0010	0.9495	rbf	0.943842	7.2740	0.918150
19	0.0050	0.9480	rbf	0.942168	7.7605	0.931933
20	0.0100	0.9465	rbf	0.940501	7.8880	0.949210
21	0.0500	0.9440	rbf	0.937724	8.5180	0.993479
22	0.1000	0.9455	rbf	0.939392	8.5145	0.953153
23	1.0000	0.9405	rbf	0.933826	10.0300	0.992472

Kernel PCA + LDA Grid Search – Cross Validation

Χρησιμοποιούμε για την εκτέλεση της διαδικασίας, grid search (αναζήτηση υπερπαραμέτρων) με 5-fold cross validation. Για τη μείωση διάστασης εφαρμόζουμε Kernel PCA και LDA για περαιτέρω μείωση διάστασης και γραμμική διάκριση μεταξύ των κλάσεων. Το pipeline χρησιμοποιείται για την εκτέλεση των σταδίων kpca, lda και svc σε ένα ενιαίο βήμα, τηρώντας τη σειρά προτεραιότητας.

1. **Linear Kpca** (Validation Accuracy, acc= 0.8730 & Test Accuracy, acc=0.8815)

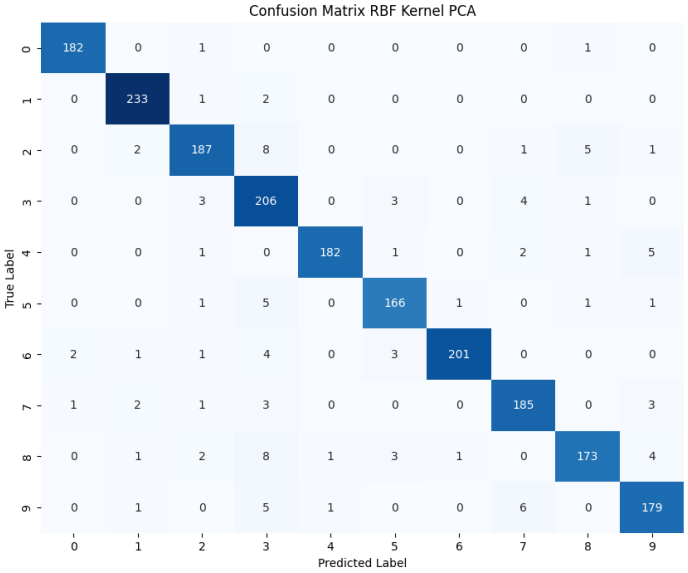
	Best Parameters	Accuracy	Number of Components
0	{'svc__C': 0.1}	0.887333	269



Classification Report:	precision	recall	f1-score	support
0	0.93	0.97	0.95	184
1	0.93	0.96	0.94	236
2	0.90	0.88	0.89	204
3	0.90	0.80	0.85	217
4	0.87	0.91	0.89	192
5	0.79	0.85	0.82	175
6	0.94	0.95	0.95	212
7	0.86	0.87	0.86	195
8	0.84	0.78	0.81	193
9	0.84	0.83	0.84	192
accuracy			0.88	2000
macro avg	0.88	0.88	0.88	2000
weighted avg	0.88	0.88	0.88	2000

2. **RBF Kernel PCA** (Validation Accuracy, acc= 0.949 & Test Accuracy, acc=0.947)

	Best Parameters	Accuracy	Number of Components
0	{'svc__C': 1}	0.941833	2748

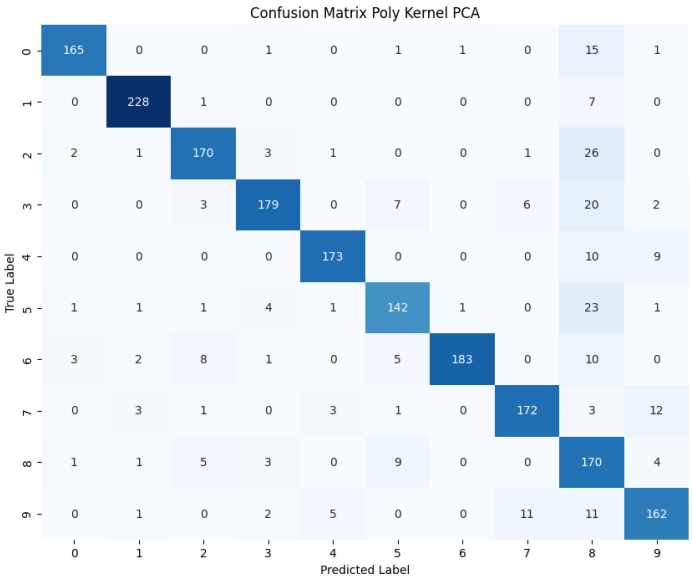


Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	184
1	0.97	0.99	0.98	236
2	0.94	0.92	0.93	204
3	0.85	0.95	0.90	217
4	0.99	0.95	0.97	192
5	0.94	0.95	0.95	175
6	0.99	0.95	0.97	212
7	0.93	0.95	0.94	195
8	0.95	0.90	0.92	193
9	0.93	0.93	0.93	192
accuracy			0.95	2000
macro avg	0.95	0.95	0.95	2000
weighted avg	0.95	0.95	0.95	2000

3. **Poly Kernel PCA** (Validation Accuracy, acc= 0.862 & Test Accuracy, acc=0.872)

	Best Parameters	Accuracy	Number of Components
0	{'svc__C': 1}	0.862167	172



Classification Report:

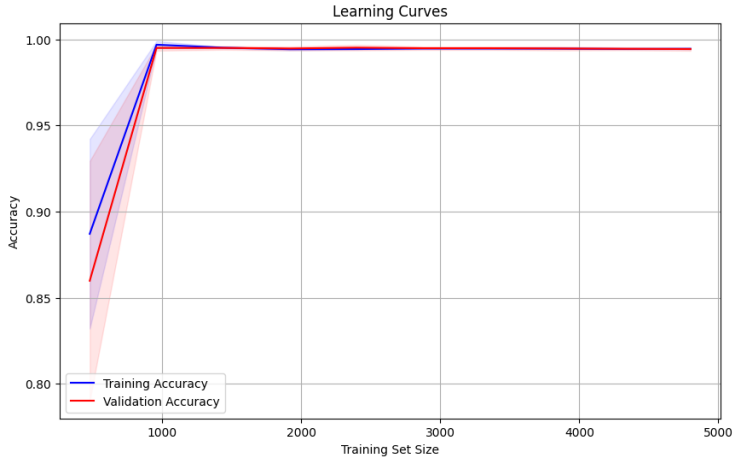
	precision	recall	f1-score	support
0	0.96	0.90	0.93	184
1	0.96	0.97	0.96	236
2	0.90	0.83	0.87	204
3	0.93	0.82	0.87	217
4	0.95	0.90	0.92	192
5	0.86	0.81	0.84	175
6	0.99	0.86	0.92	212
7	0.91	0.88	0.89	195
8	0.58	0.88	0.70	193
9	0.85	0.84	0.85	192
accuracy			0.87	2000
macro avg	0.89	0.87	0.87	2000
weighted avg	0.89	0.87	0.88	2000

Best Model (RBF Kernel PCA & Linear SVC with C=0.0001)

Χρησιμοποιούμε το καλύτερο μοντέλο με test accuracy = 0,951 και κάνουμε plot τα learning curves.

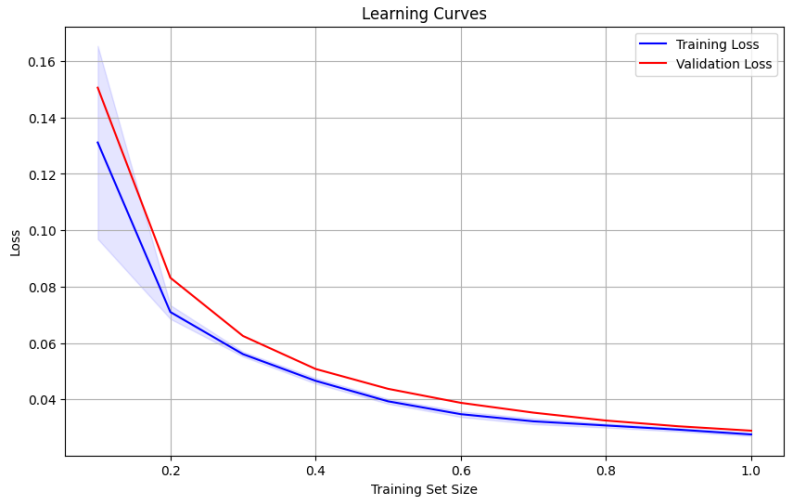
Με τα learning curves μπορούμε να δούμε την απόδοση του μοντέλου καθώς αυξάνεται το μέγεθος εκπαίδευσης. Ο άξονας x αντιπροσωπεύει τον αριθμό των δειγμάτων από τα δεδομένα εκπαίδευσης και ο άξονας y την ακρίβεια καθώς αυξάνονται τα δεδομένα εκπαίδευσης. Στο παρακάτω διάγραμμα βλέπουμε δύο καμπύλες, μία για το train accuracy και μια για το validation accuracy. Οι δύο καμπύλες είναι πολύ κοντά μεταξύ τους επομένως, το μοντέλο φαίνεται να εκπαιδεύεται σωστά και να γενικεύει καλά σε νέα δεδομένα.

Learning Curve Accuracy



Για το Learning Curve loss ο άξονας x αντιπροσωπεύει τον αριθμό των δειγμάτων από τα δεδομένα εκπαίδευσης και ο άξονας y το σφάλμα πρόβλεψης. Για χαμηλές τιμές loss έχουμε καλή απόδοση. Στο παρακάτω διάγραμμα παρατηρούμε ότι η καμπύλη loss πέφτει καθώς αυξάνονται τα δείγματα.

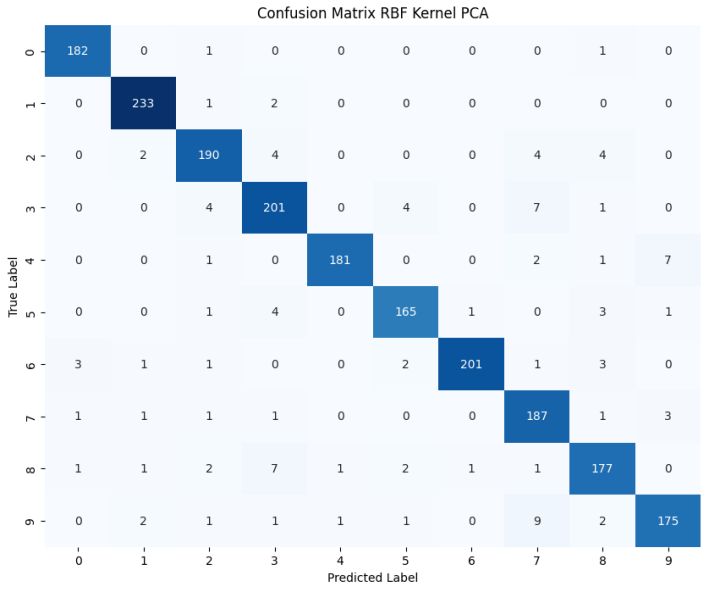
Learning Curve Loss



KNN + Kernel Pca + Lda

5 – Fold Cross Validation: Best Accuracy rbf Kernel pca (Test Accuracy = 0.946)

	linear	poly	rbf
Best n_components	269	172	2748
Best Parameters for KNN	{'metric': 'manhattan', 'n_neighbors': 5, 'wei...	{'metric': 'euclidean', 'n_neighbors': 9, 'wei...	{'metric': 'euclidean', 'n_neighbors': 10, 'we...
Best Score for KNN	0.913333	0.913833	0.994833



Classification Report:	precision	recall	f1-score	support
0	0.97	0.99	0.98	184
1	0.97	0.99	0.98	236
2	0.94	0.93	0.93	204
3	0.91	0.93	0.92	217
4	0.99	0.94	0.97	192
5	0.95	0.94	0.95	175
6	0.99	0.95	0.97	212
7	0.89	0.96	0.92	195
8	0.92	0.92	0.92	193
9	0.94	0.91	0.93	192
accuracy			0.95	2000
macro avg	0.95	0.95	0.95	2000
weighted avg	0.95	0.95	0.95	2000

Παίρνω τις καλύτερες παραμέτρους του KNN για polynomial kpca και δοκιμάζω διαφορετικά coef0 = [0.01, 0.05, 0.1, 0.5, 1, 10].

Coef0	Accuracy Test set
0.001	0.7705
0.005	0.7745
0.1	0.8015
0.5	0.8825
1	0.9055
10	0.9055

Παρατηρώ ότι ενώ αυξάνεται η ακρίβεια καθώς αυξάνουμε το coef0 μετά το 1 φαίνεται να παραμένει σταθερή.

Για coef0=1 δοκιμάζω διαφορετικά degree = [2, 3, 4, 5]

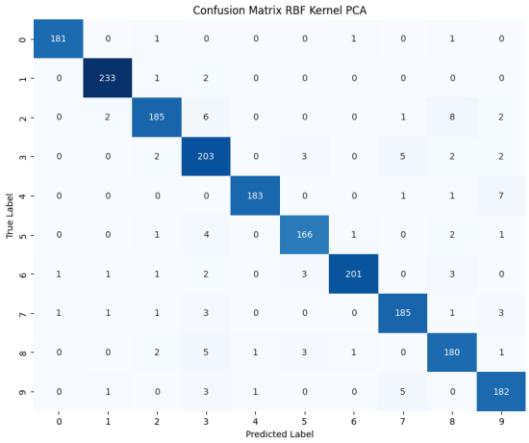
Degree	Accuracy
2	0.911
3	0.9055
4	0.8805
5	0.809

Συμπέρασμα: Καθώς αυξάνεται ο βαθμός του πολυωνύμου η ακρίβεια πέφτει.

NCC + Kernel Pca + Lda

5-Fold Cross Validation (Test Accuracy = 0.9495)

	Best n_components	Best Parameters for NCC	Best Score for NCC
linear	269	{'metric': 'euclidean'}	0.886667
poly	172	{'metric': 'euclidean'}	0.861833
rbf	2748	{'metric': 'euclidean'}	0.994833



Classification Report:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	184
1	0.98	0.99	0.98	236
2	0.95	0.91	0.93	204
3	0.89	0.94	0.91	217
4	0.99	0.95	0.97	192
5	0.95	0.95	0.95	175
6	0.99	0.95	0.97	212
7	0.94	0.95	0.94	195
8	0.91	0.93	0.92	193
9	0.92	0.95	0.93	192
accuracy			0.95	2000
macro avg	0.95	0.95	0.95	2000
weighted avg	0.95	0.95	0.95	2000

Παίρνουμε τις καλύτερες παραμέτρους του NCC για polynomial kpca και δοκιμάζουμε διαφορετικά coef0 = [0.01, 0.05, 0.1, 0.5, 1, 10].

Coef0	Accuracy Test set
0.001	0.306
0.005	0.306
0.1	0.3835
0.5	0.7985
1	0.855
10	0.866

Για coef0=10 δοκιμάζουμε διαφορετικά degree = [2, 3, 4, 5].

Degree	Accuracy
1	0.863
2	0.8655
3	0.866
4	0.8695
5	0.8705
6	0.868