The Cifar10 dataset of handwritten digits

SVMs – Classification (A)

KPCA+LDA (B)

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 Cifar10 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 Cifar10 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 Cifar10 έχει διαστάσεις 32 x 32 x 3 (32 x 32 είναι το ύψος και το πλάτος των εικόνων σε pixels και 3 είναι ο αριθμός των καναλιών χρώματος, RGB).

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().

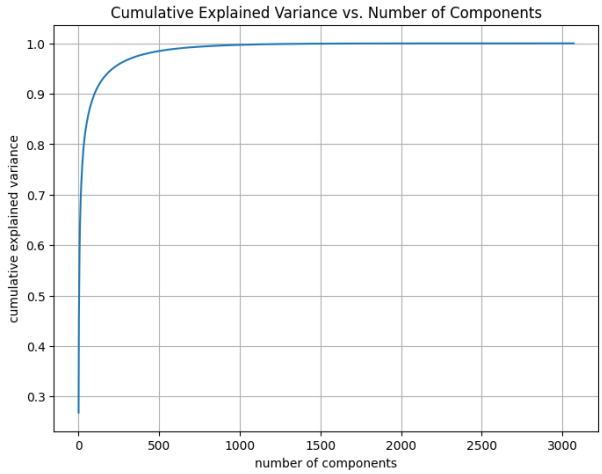
Τυποποίηση των εικόνων: Standardization of Images

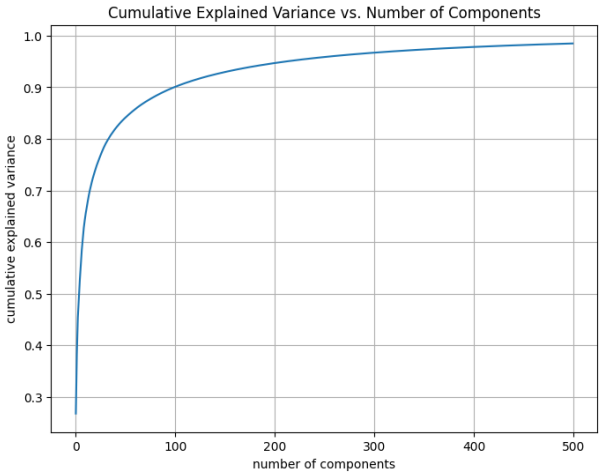
Η «Τυποποίηση» χρησιμοποιείται προκειμένου να εξισώσει όλα τα χαρακτηριστικά των εικόνων , ώστε να έχουν μέσο όρο ίσο με 0 (mean=0) και τυπική απόκλιση ίση με 1 (mean=1). Αυτό μπορεί να αποδειχτεί ιδιαίτερα χρήσιμο κατά την χρήση της PCA (Principal

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 200 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 211 συνιστώσες.

Μέρος 3^{ο}: Εκπαίδευση Μοντέλου

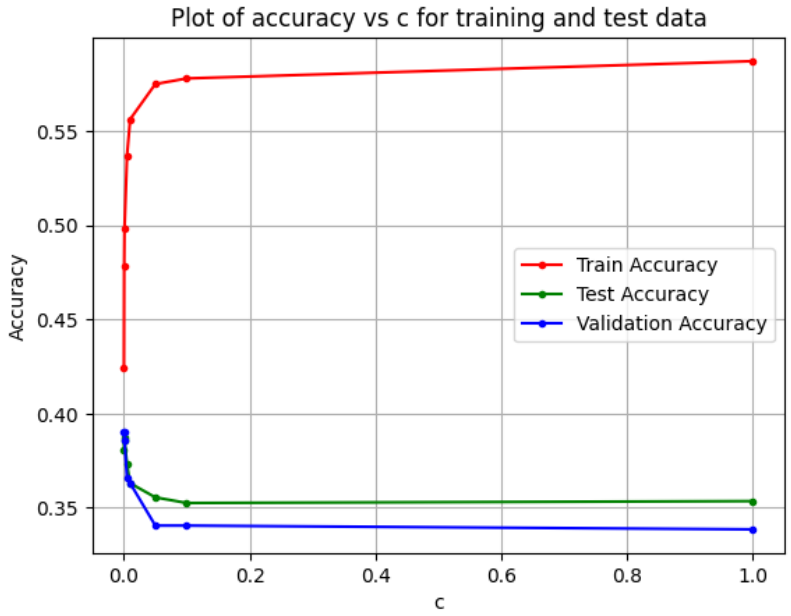
Η εκπαίδευση του SVM δοκιμάστηκε με διαφορετικούς πυρήνες (Linear, RBF, Polynomial Kernel). Οι Kernels είναι συναρτήσεις που επιτρέπουν στο SVM να μετασχηματίσει τα δεδομένα σε υψηλότερη διάσταση. Κάθε τύπος Kernel μπορεί να βοηθήσει το μοντέλο να μάθει διαφορετικούς τύπους διαχωριστικών υπέρ-επιπέδων.

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=0.4615). Στη συνέχεια αξιολογούμε το μοντέλο με τον καλύτερο Kernel, κάνοντας πρόβλεψη στο σύνολο δοκιμών (test set). Για το test set το accuracy = 0.4575, δηλαδή και ελάχιστα χαμηλότερο από αυτό του validation set.

SVM Linear Kernel

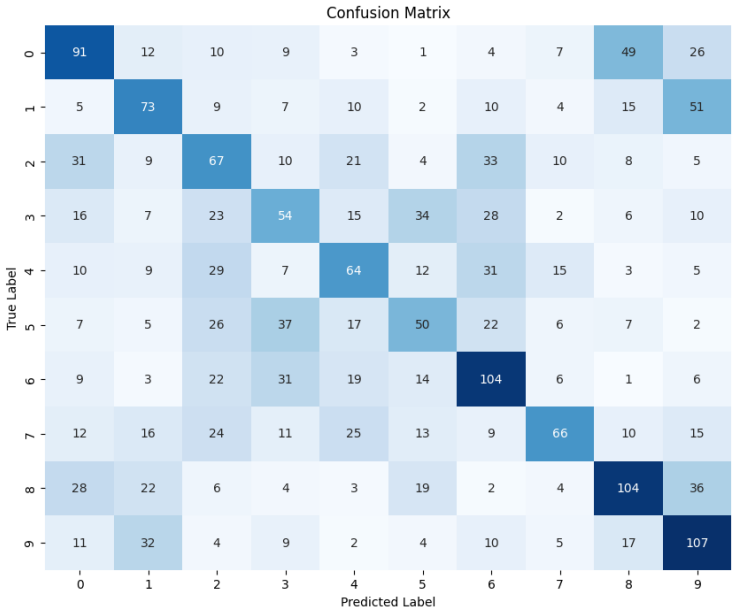
Δημιουργούμε μία μέθοδο «Linear Kernel Model» και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου C (regularization parameter). Όσο μικρότερη είναι η τιμή της παραμέτρου C τόσο πιο «ελαστικό» είναι το μοντέλο, άρα αφήνει το να γίνουν περισσότερα λάθη κατά την διάρκεια της εκπαίδευσης. Από την άλλη πλευρά 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 1 και κάτω. Ο πίνακας περιλαμβάνει τις τιμές C= [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.01, 1]. Η μεγαλύτερη ακρίβεια για το test set επιτυγχάνεται για C=0.0001 & 0.0005 (acc=0.39). Παρατηρούμε ότι για C από 0.001 και πάνω η ακρίβεια αρχίζει να έχει καθοδική πορεία.



	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy	Run time
0	linear	0.0001	0.424333	0.3805	0.3900	10.814700
1	linear	0.0005	0.478333	0.3875	0.3900	10.139913
2	linear	0.0010	0.498000	0.3860	0.3855	10.513339
3	linear	0.0050	0.537000	0.3730	0.3655	12.870203
4	linear	0.0100	0.556333	0.3630	0.3630	16.814012
5	linear	0.0500	0.575000	0.3555	0.3405	63.313946
6	linear	0.1000	0.578000	0.3525	0.3405	143.388564
7	linear	1.0000	0.587167	0.3535	0.3385	1968.112263

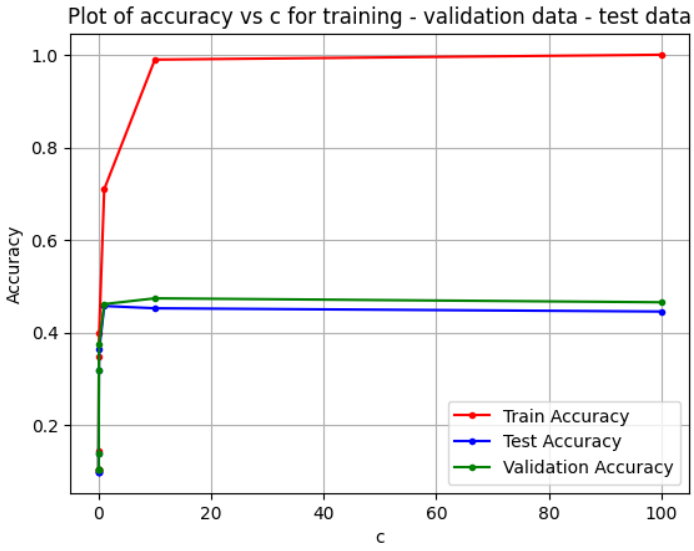
	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	linear	0.0001	0.3900	0.321635	2.3210	3.566651
1	linear	0.0005	0.3900	0.321852	2.2800	3.499143
2	linear	0.0010	0.3855	0.316846	2.2960	3.516959
3	linear	0.0050	0.3655	0.294794	2.3955	3.589777
4	linear	0.0100	0.3630	0.292031	2.3910	3.570014
5	linear	0.0500	0.3405	0.267191	2.5065	3.676479
6	linear	0.1000	0.3405	0.267138	2.4865	3.653286
7	linear	1.0000	0.3385	0.264998	2.4770	3.635107



Radial Basis Kernel

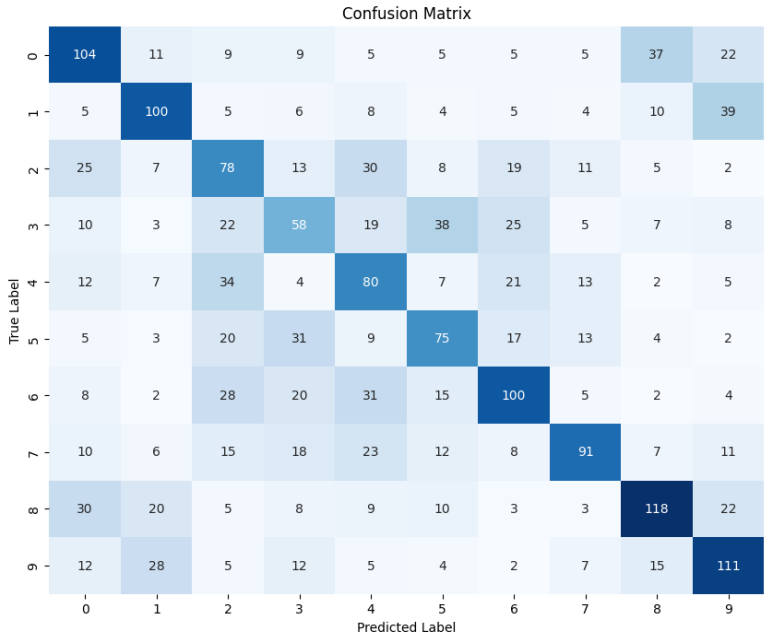
Δημιουργούμε μία μέθοδο «RBF Model» και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου C, καθώς και διαφορετικά gamma (decision boundary). Αν η παράμετρος gamma είναι πολύ μεγάλη, τότε ο διαχωριστικός χώρος είναι πολύ στενός. Σε αυτή την περίπτωση για να θεωρήσει ο πυρήνας ότι δύο σημεία είναι όμοια, 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=scale. 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=1 (acc=0.4575). Ο RBF Kernel σε σχέση με τον Linear Kernel πετυχαίνει καλύτερη ακρίβεια .



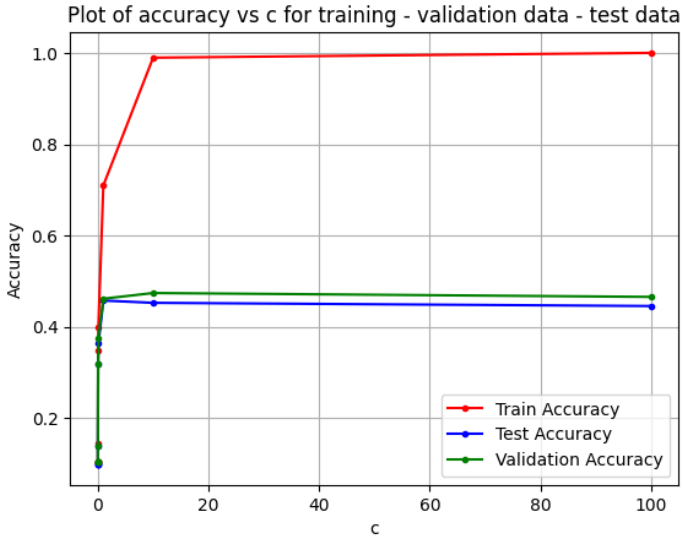
	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy	Run time:
0	RBF	0.0001	0.104167	0.1045	0.0990	26.858293
1	RBF	0.0005	0.104167	0.1045	0.0990	26.323399
2	RBF	0.0010	0.104167	0.1045	0.0990	28.037527
3	RBF	0.0050	0.104167	0.1045	0.0990	27.937462
4	RBF	0.0100	0.143500	0.1380	0.1400	26.655180
5	RBF	0.0500	0.347500	0.3200	0.3195	24.458428
6	RBF	0.1000	0.399333	0.3755	0.3630	23.971993
7	RBF	1.0000	0.709667	0.4615	0.4575	23.100727
8	RBF	10.0000	0.989500	0.4740	0.4525	24.503078
9	RBF	100.0000	1.000000	0.4655	0.4455	23.893211

	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	RBF	0.0001	0.0990	0.000000	3.1760	3.879562
1	RBF	0.0005	0.0990	0.000000	3.1760	3.879562
2	RBF	0.0010	0.0990	0.000000	3.1760	3.879562
3	RBF	0.0050	0.0990	0.000000	3.1760	3.879562
4	RBF	0.0100	0.1400	0.043785	3.3530	4.241462
5	RBF	0.0500	0.3195	0.243177	2.5205	3.658210
6	RBF	0.1000	0.3630	0.291773	2.3870	3.582457
7	RBF	1.0000	0.4575	0.397067	2.0395	3.318659
8	RBF	10.0000	0.4525	0.391710	2.0460	3.316172
9	RBF	100.0000	0.4455	0.383997	2.0900	3.361250



Polynomial Kernel

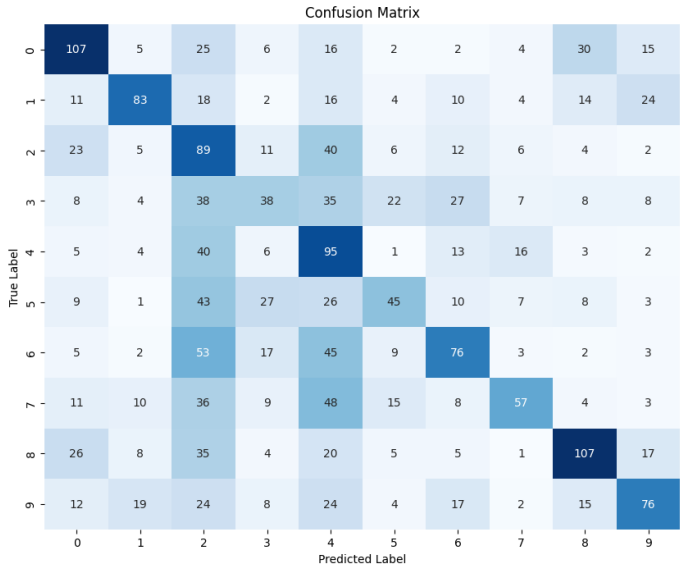
Για τον «Polynomial Kernel» δημιουργούμε επίσης μία μέθοδο και δοκιμάζουμε διαφορετικές τιμές της παραμέτρου C= [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.01, 1, 10, 100]. Για τον συγκεκριμένο πυρήνα παρατηρούμε ότι πετυχαίνει καλύτερη ακρίβεια (acc=0.3865) για C=10.



	Kernel	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	Polynomial	0.0001	0.104167	0.1045	0.0990
1	Polynomial	0.0005	0.104333	0.1040	0.0990
2	Polynomial	0.0010	0.105833	0.1050	0.1000
3	Polynomial	0.0050	0.115167	0.1120	0.1110
4	Polynomial	0.0100	0.155000	0.1390	0.1425
5	Polynomial	0.0500	0.296167	0.2225	0.2260
6	Polynomial	0.1000	0.368333	0.2560	0.2660
7	Polynomial	1.0000	0.709333	0.3505	0.3525
8	Polynomial	10.0000	0.952333	0.3735	0.3865
9	Polynomial	100.0000	0.997333	0.3755	0.3785

	Kernel	C	Test Accuracy	Kappa	MAE	RMSE
0	Polynomial	0.0001	0.0990	0.000000	3.1760	3.879562
1	Polynomial	0.0005	0.0990	0.000000	3.1760	3.879562
2	Polynomial	0.0010	0.1000	0.001075	3.1760	3.882396
3	Polynomial	0.0050	0.1110	0.012996	3.1700	3.901410
4	Polynomial	0.0100	0.1425	0.047907	3.0455	3.827075
5	Polynomial	0.0500	0.2260	0.140704	2.7165	3.632699
6	Polynomial	0.1000	0.2660	0.185412	2.5650	3.559635
7	Polynomial	1.0000	0.3525	0.281121	2.3045	3.387108
8	Polynomial	10.0000	0.3865	0.318379	2.2200	3.350522
9	Polynomial	100.0000	0.3785	0.309176	2.2675	3.422353

	Kernel	C	Test Accuracy	Mean Kappa	Mean MAE	Mean RMSE
0	poly	Mean	0.2161	0.129677	2.7817	3.672232

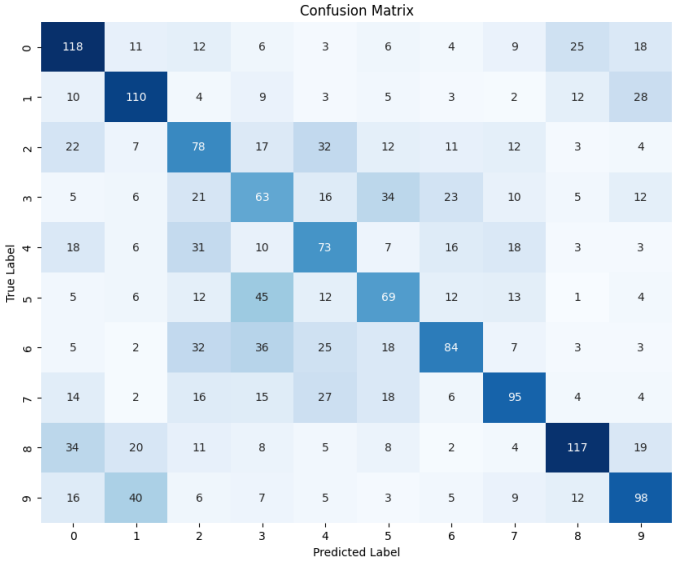


Μέθοδος Cross Validation – Grid Search

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

Για 5-fold cross validation: Καλύτερη ακρίβεια για το train set δίνεται από τον RBF Kernel (acc=0.447167), ωστόσο εξακολουθεί και πάλι να είναι πολύ χαμηλή. Για το validation set η ακρίβεια είναι 0.474 και για το test set 0.4525.

	Kernel	Best Parameters	Accuracy
0	linear	{'C': 0.0005}	0.383000
1	poly	{'C': 10, 'gamma': 'scale'}	0.364167
2	rbf	{'C': 10, 'gamma': 'scale'}	0.447167



Classification Report:

	precision	recall	f1-score	support
0	0.48	0.56	0.51	212
1	0.52	0.59	0.56	186
2	0.35	0.39	0.37	198
3	0.29	0.32	0.31	195
4	0.36	0.39	0.38	185
5	0.38	0.39	0.38	179
6	0.51	0.39	0.44	215
7	0.53	0.47	0.50	201
8	0.63	0.51	0.57	228
9	0.51	0.49	0.50	201
accuracy			0.45	2000
macro avg	0.46	0.45	0.45	2000
weighted avg	0.46	0.45	0.45	2000

Η μετρική Precision μας δείχνει την αναλογία των σωστών θετικών προβλέψεων προς το σύνολο των θετικών προβλέψεων. Η μετρική Recall δείχνει την αναλογία των σωστών θετικών προβλέψεων προς το σύνολο των πραγματικών θετικών προβλέψεων . Η μετρική F1-Score αποτελεί έναν συνδυασμό των Precision και Recall και μας βοηθάει να καταλάβουμε την απόδοση του μοντέλου.

Σε αυτή την περίπτωση με βάση την F1-Score , η κατηγορία με την καλύτερη απόδοση είναι η 8 και μετά ακολουθεί και η 1. Ενώ την χειρότερη απόδοση έχει η κατηγορία 3. Σαν γενικό συμπέρασμα παρατηρούμε χαμηλή απόδοση και κακή ικανότητα το μοντέλου να ταξινομεί τα δεδομένα.

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

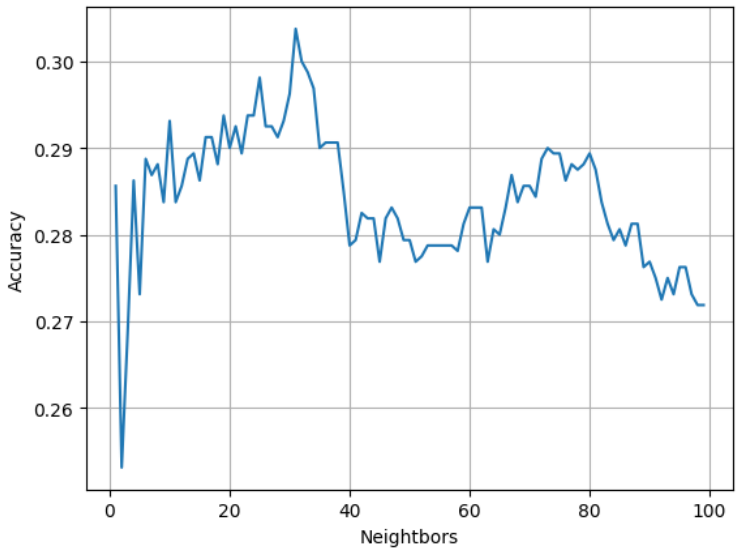
Αρχικά εμφανίζουμε τα πρώτα 20 δείγματα του test set και τα πρώτα 20 που προβλέψαμε. Παρατηρούμε ότι το 4^{ο} και το 7^{ο} δείγμα έχουν προβλεφθεί λάθος.

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

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

KNN Classifier

Για τον KNN ακολουθούμε την ίδια προεπεξεργασία των δεδομένων του cifar10 και εκτελούμε pca ορίζοντας τα n_components έτσι ώστε να κρατήσουμε το 95% της πληροφορίας. Στη συνέχεια βρίσκουμε σε ένα εύρος γειτόνων από 1 έως 100 τον καλύτερο γείτονα (neighbor=31, acc=0.30375), δηλαδή τον γείτονα που δίνει την καλύτερη ακρίβεια για το test set. Κάνουμε plot την ακρίβεια με τους γείτονες για αυτό το εύρος (1-100) και βλέπουμε πώς μεταβάλλεται η ακρίβεια. Παρατηρούμε ότι ακρίβεια του μοντέλου πέφτει καθώς αυξάνονται οι γείτονες.



Εκτελούμε Grid Search 5 & 10 Fold CV για ένα εύρος γειτόνων από 1 έως 31 καθώς και για διαφορετικές τιμές ‘weights’ και ‘metric’. Αφού βρούμε τις καλύτερες παραμέτρους

(metric=euclidean & weights=distance), ελέγχουμε την ακρίβεια για τα train, validation & test set.

5-Fold Cross Validation

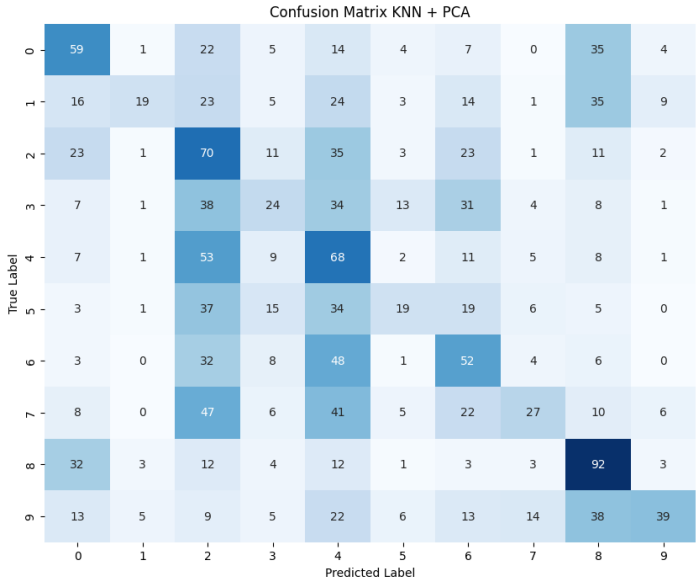
```
Best Hyperparameters: KNeighborsClassifier(metric='euclidean', weights='distance')
Best Model Accuracy (CV): 0.29541666666666666
```

10-Fold Cross Validation

```
Best Hyperparameters: KNeighborsClassifier(metric='euclidean', weights='distance')
Best Model Accuracy (CV): 0.29666666666666675
```

Results: Και στις δύο περιπτώσεις τα αποτελέσματα για την ακρίβεια είναι τα ίδια

```
Validation Set Accuracy: 0.285625
Test Set Accuracy: 0.293125
```




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

           0       0.35      0.39      0.37       151
           1       0.59      0.13      0.21       149
           2       0.20      0.39      0.27       180
           3       0.26      0.15      0.19       161
           4       0.20      0.41      0.27       165
           5       0.33      0.14      0.19       139
           6       0.27      0.34      0.30       154
           7       0.42      0.16      0.23       172
           8       0.37      0.56      0.45       165
           9       0.60      0.24      0.34       164

    accuracy                           0.29      1600
   macro avg       0.36      0.29      0.28      1600
weighted avg       0.36      0.29      0.28      1600
```

Σωστές και λάθος προβλέψεις:

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

```
Correct Predictions: [3, 11, 14, 16, 17, 22, 29]
Incorrect Predictions: [0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28]
```

NCC Classifier

Αρχικά κάνουμε ένα grid search 5 & 10 -Fold CV για τον NCC με metric = euclidean & manhattan.

5 – Fold Cross Validation

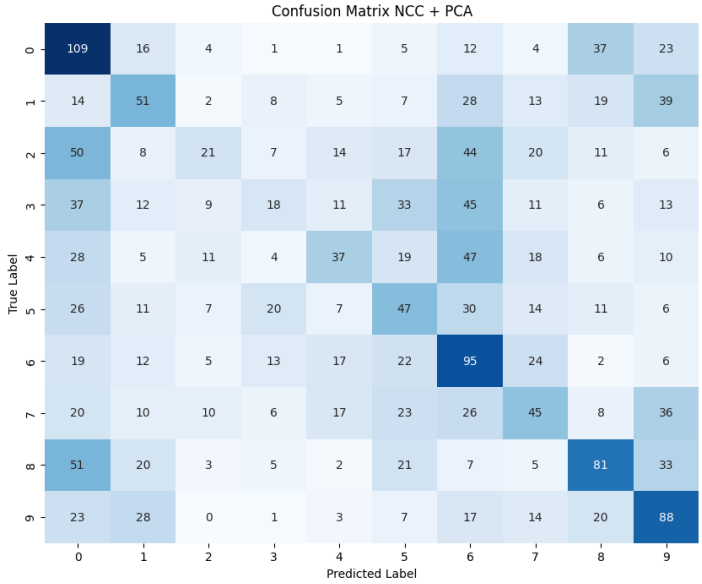
```
Best Model Parameters: {'metric': 'manhattan'}
Best Model Accuracy (CV): 0.2840000000000003
```

10 – Fold Cross Validation

```
Best Model Parameters: {'metric': 'manhattan'}
Best Model Accuracy (CV): 0.28483333333333327
```

Results: Και στις δύο περιπτώσεις η ακρίβεια είναι η ίδια

```
Validation Set Accuracy: 0.282
Test Set Accuracy: 0.296
```



Classification Report:	precision	recall	f1-score	support
0	0.29	0.51	0.37	212
1	0.29	0.27	0.28	186
2	0.29	0.11	0.16	198
3	0.22	0.09	0.13	195
4	0.32	0.20	0.25	185
5	0.23	0.26	0.25	179
6	0.27	0.44	0.34	215
7	0.27	0.22	0.24	201
8	0.40	0.36	0.38	228
9	0.34	0.44	0.38	201
accuracy			0.30	2000
macro avg	0.29	0.29	0.28	2000
weighted avg	0.30	0.30	0.28	2000

Σωστά και λάθος αποτελέσματα:

NCC Test set Accuracy: 0.296

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

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

Β) <u>KPCA + LDA</u>

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%.

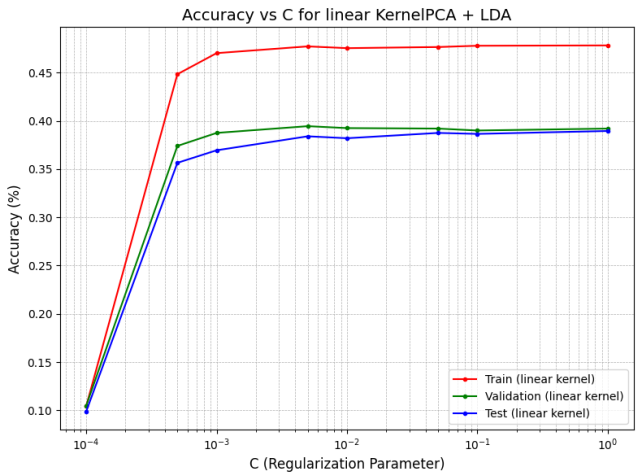
Εφαρμόζουμε kpca με τον επιλεγμένο αριθμό συνιστωσών που υπολογίσαμε (n_components_95) και κάνουμε fit_transform() στα δεδομένα εκπαίδευση και transform() στα δεδομένα επικύρωσης και ελέγχου.

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. Για το LDA τα n_components ορίστηκαν ίσα με 9, δηλαδή ο αριθμός των κλάσεων πλην 1.

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.

SVC (Linear Kernel) & Kernel PCA– Αποτελέσματα Ακρίβειας

1. **Linear KPCA**

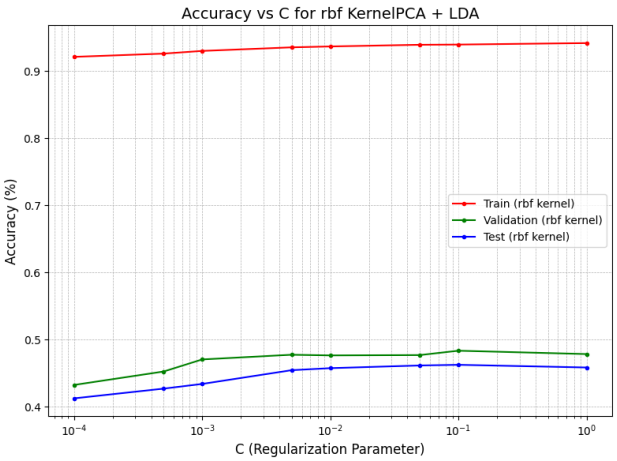


	Kernel (KPCA)	n_components	C	Train Accuracy	Validation Accuracy	Test Accuracy
0	linear	211	0.0001	0.104500	0.1050	0.0990
1	linear	211	0.0005	0.448333	0.3740	0.3565
2	linear	211	0.0010	0.470167	0.3875	0.3695
3	linear	211	0.0050	0.477167	0.3945	0.3840
4	linear	211	0.0100	0.475333	0.3925	0.3820
5	linear	211	0.0500	0.476500	0.3920	0.3875
6	linear	211	0.1000	0.477833	0.3900	0.3865
7	linear	211	1.0000	0.478167	0.3920	0.3895

Μετρικές (Kappa, MAE, RMSE)

	C	Test Accuracy	Kernel (KPCA)	Kappa	MAE	RMSE
0	0.0001	0.0990	linear	0.000000	3.1760	3.879562
1	0.0005	0.3565	linear	0.284336	2.3950	3.562864
2	0.0010	0.3695	linear	0.299029	2.3345	3.517314
3	0.0050	0.3840	linear	0.315167	2.3070	3.510556
4	0.0100	0.3820	linear	0.313028	2.3080	3.506280
5	0.0500	0.3875	linear	0.319124	2.2765	3.475558
6	0.1000	0.3865	linear	0.318031	2.2900	3.490559
7	1.0000	0.3895	linear	0.321368	2.2745	3.476996

1. RBF KPCA

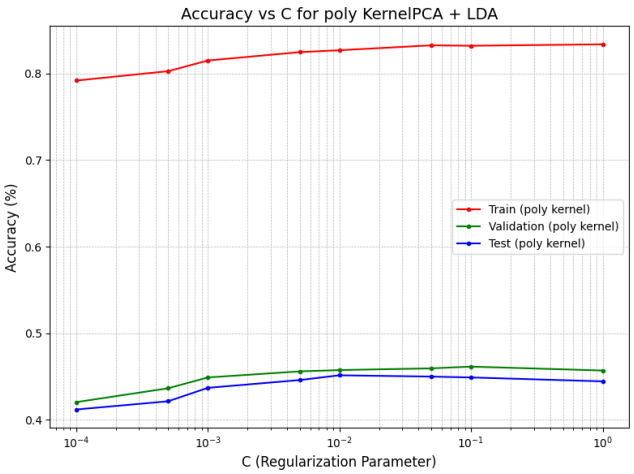


	Kernel (KPCA)	n_components	C	Train Accuracy	Validation Accuracy	Test Accuracy
16	rbf	4042	0.0001	0.921333	0.4325	0.4125
17	rbf	4042	0.0005	0.926167	0.4525	0.4270
18	rbf	4042	0.0010	0.930167	0.4705	0.4340
19	rbf	4042	0.0050	0.935500	0.4775	0.4545
20	rbf	4042	0.0100	0.936833	0.4765	0.4575
21	rbf	4042	0.0500	0.939333	0.4770	0.4615
22	rbf	4042	0.1000	0.939667	0.4835	0.4625
23	rbf	4042	1.0000	0.941833	0.4785	0.4585

Μετρικές (Kappa, MAE, RMSE)

	C	Test Accuracy	Kernel (KPCA)	Kappa	MAE	RMSE
16	0.0001	0.4125	rbf	0.347170	2.1175	3.317152
17	0.0005	0.4270	rbf	0.363113	2.0795	3.316399
18	0.0010	0.4340	rbf	0.370907	2.0600	3.297272
19	0.0050	0.4545	rbf	0.393696	1.9910	3.259908
20	0.0100	0.4575	rbf	0.397070	2.0120	3.295300
21	0.0500	0.4615	rbf	0.401682	1.9645	3.226686
22	0.1000	0.4625	rbf	0.402836	1.9670	3.228467
23	1.0000	0.4585	rbf	0.398506	1.9800	3.231254

1. Polynomial KPCA



	Kernel (KPCA)	n_components	C	Train Accuracy	Validation Accuracy	Test Accuracy
8	poly	2470	0.0001	0.791667	0.4205	0.4120
9	poly	2470	0.0005	0.802500	0.4365	0.4215
10	poly	2470	0.0010	0.814833	0.4490	0.4370
11	poly	2470	0.0050	0.824500	0.4560	0.4460
12	poly	2470	0.0100	0.826667	0.4575	0.4515
13	poly	2470	0.0500	0.832333	0.4595	0.4500
14	poly	2470	0.1000	0.831833	0.4615	0.4490
15	poly	2470	1.0000	0.833500	0.4570	0.4445

Μετρικές (Kappa, MAE, RMSE)

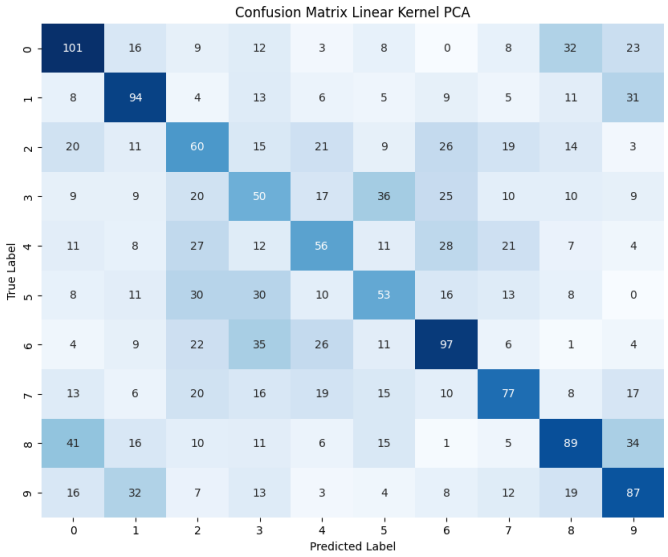
	C	Test Accuracy	Kernel (KPCA)	Kappa	MAE	RMSE
8	0.0001	0.4120	poly	0.346677	2.0195	3.140303
9	0.0005	0.4215	poly	0.357153	2.0270	3.209829
10	0.0010	0.4370	poly	0.374194	2.0165	3.236588
11	0.0050	0.4460	poly	0.384188	2.0290	3.283291
12	0.0100	0.4515	poly	0.390375	2.0220	3.283443
13	0.0500	0.4500	poly	0.388794	2.0375	3.299773
14	0.1000	0.4490	poly	0.387704	2.0265	3.279253
15	1.0000	0.4445	poly	0.382715	2.0495	3.312024

Grid Search 5- Fold Cross Validation (Linear SVC)

A. Linear Kpca + lda (Results)

Καλύτερη ακρίβεια για το validation set acc = 0.3925 & test set acc = 0.3820

	Best Parameters	Accuracy	Number of Components
0	{'svc__C': 0.01}	0.375	211



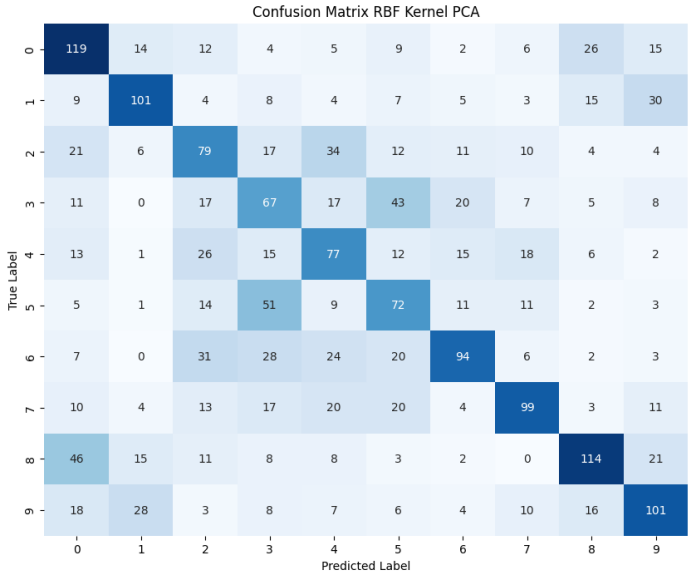
Classification Report:

	precision	recall	f1-score	support
0	0.44	0.48	0.46	212
1	0.44	0.51	0.47	186
2	0.29	0.30	0.29	198
3	0.24	0.26	0.25	195
4	0.34	0.30	0.32	185
5	0.32	0.30	0.31	179
6	0.44	0.45	0.45	215
7	0.44	0.38	0.41	201
8	0.45	0.39	0.42	228
9	0.41	0.43	0.42	201
accuracy			0.38	2000
macro avg	0.38	0.38	0.38	2000
weighted avg	0.38	0.38	0.38	2000

B. RBF Kpca + lda (Results)

Καλύτερη ακρίβεια για το validation set acc = 0.4755 & test set acc = 0.4615

	Best Parameters	Accuracy	Number of Components
0	{'svc__C': 1}	0.445333	4042



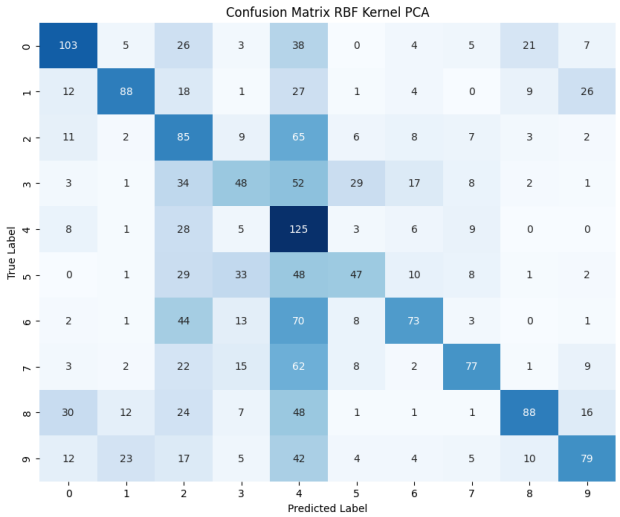
Classification Report:

	precision	recall	f1-score	support
0	0.46	0.56	0.51	212
1	0.59	0.54	0.57	186
2	0.38	0.40	0.39	198
3	0.30	0.34	0.32	195
4	0.38	0.42	0.39	185
5	0.35	0.40	0.38	179
6	0.56	0.44	0.49	215
7	0.58	0.49	0.53	201
8	0.59	0.50	0.54	228
9	0.51	0.50	0.51	201
accuracy			0.46	2000
macro avg	0.47	0.46	0.46	2000
weighted avg	0.47	0.46	0.47	2000

C. Poly Kpca + lda (Results)

Καλύτερη Ακρίβεια για validation set acc=0.3970 & για test set acc = 0.4065

	Best Parameters	Accuracy	Number of Components
0	{'svc__C': 1}	0.370667	2470



Classification Report:

	precision	recall	f1-score	support
0	0.56	0.49	0.52	212
1	0.65	0.47	0.55	186
2	0.26	0.43	0.32	198
3	0.35	0.25	0.29	195
4	0.22	0.68	0.33	185
5	0.44	0.26	0.33	179
6	0.57	0.34	0.42	215
7	0.63	0.38	0.48	201
8	0.65	0.39	0.48	228
9	0.55	0.39	0.46	201
accuracy			0.41	2000
macro avg	0.49	0.41	0.42	2000
weighted avg	0.49	0.41	0.42	2000

Linear SVC with RBF Kernel PCA (C=0.1)

Gamma	Accuracy
0.0001	0.4235
0.0005	0.4575
0.001	0.4405
0.01	0.1095
0.1	0.093

Συμπέρασμα: Για gamma από 0.01 και μετά η ακρίβεια πέφτει σε μεγάλο βαθμό. Για μεγάλο gamma κάθε σημείο δεδομένων επηρεάζει μόνο τα πολύ κοντινά του και αυτό μπορεί να οδηγήσει σε υπερπροσαρμογή των δεδομένων εκπαίδευσης χωρίς να γενικεύει καλά στα δεδομένα του test set.

Learning Curves Accuracy – Loss

Δημιουργούμε ένα pipeline με τα 3 στάδια: a) kpca (rbf kernel), b) lda για καλύτερη διάκριση των κατηγοριών με n_components ίσα με 9 και g) εκπαίδευση με svm ταξινομητή με linear kernel.

Υπολογίζουμε τα learning curves για accuracy και loss. Αρχικά δημιουργούμε 10 ομοιόμορφα κατανεμημένα ποσοστά του train set (ξεκινάμε από το 10% έως το 100%), για να εξετάσουμε πώς μεταβάλλεται η απόδοση του μοντέλου καθώς αυξάνεται το πλήθος των δεδομένων. Υπολογίζουμε την απόδοση του pipeline σε υποσύνολα του train set και του validation set, με κριτήριο το accuracy και το loss. Χρησιμοποιούμε 5 –fold cross validation. Υπολογίζουμε το μέσο όρο και την τυπική απόκλιση των αποτελεσμάτων για τα το 5 - fold cross validation για κάθε σύνολο. Τέλος κάνουμε plot την ακρίβεια και το loss. Το fill_between προσθέτει ένα εύρος τυπικής απόκλισης γύρω από τις γραμμές της ακρίβειας και του loss.

KNN Classifier (kpca + lda)

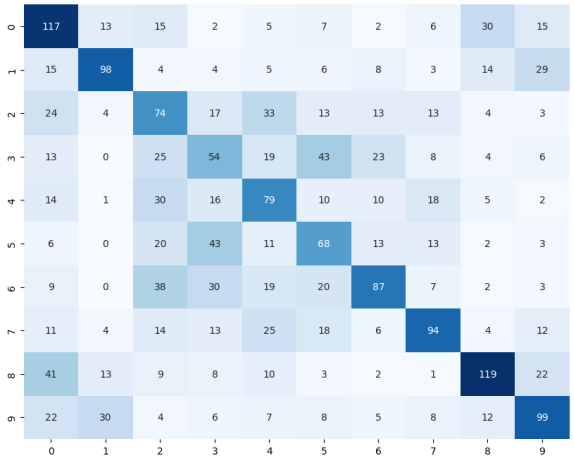
Για τον KNN ακολουθούμε την ίδια προεπεξεργασία με τον svm για τα δεδομένα του cifar10. Στη συνέχεια βρίσκουμε σε ένα εύρος γειτόνων από 1 έως 100 τον καλύτερο γείτονα (best_neighbor=11 & acc=0.275), δηλαδή τον γείτονα που δίνει την καλύτερη ακρίβεια για το test set. Κάνουμε plot την ακρίβεια με τους γείτονες για αυτό το εύρος (1 έως 100) και βλέπουμε πώς μεταβάλλεται η ακρίβεια. Παρατηρούμε ότι η ακρίβεια του μοντέλου πέφτει καθώς αυξάνονται οι γείτονες.

Grid Search 5 Fold Cross Validation

Για διαφορετικούς KernelPCA (linear, rbf, poly) εκτελούμε Grid Search για να δοκιμάσουμε διάφορες τιμές παραμέτρων για τον KNN ταξινομητή. Επιλέγω ένα εύρος γειτόνων από 1 έως 20, weights=[’uniform’, ‘distance’], metric=[‘euclidean’, ‘manhattan’].

Αποτελέσματα 5-Fold Cross Validation: Best Test Set Accuracy, acc = 0.4445 (Best Model rbf)

	Best Parameters	Best Score
linear	{'metric': 'euclidean', 'n_neighbors': 20, 'we...	0.444167
poly	{'metric': 'euclidean', 'n_neighbors': 17, 'we...	0.817167
rbf	{'metric': 'euclidean', 'n_neighbors': 17, 'we...	0.928500



Classification Report:

	precision	recall	f1-score	support
0	0.43	0.55	0.48	212
1	0.60	0.53	0.56	186
2	0.32	0.37	0.34	198
3	0.28	0.28	0.28	195
4	0.37	0.43	0.40	185
5	0.35	0.38	0.36	179
6	0.51	0.40	0.45	215
7	0.55	0.47	0.51	201
8	0.61	0.52	0.56	228
9	0.51	0.49	0.50	201
accuracy			0.44	2000
macro avg	0.45	0.44	0.44	2000
weighted avg	0.46	0.44	0.45	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	0.403
0.005	0.419
0.1	0.4315
0.5	0.4425
1	0.441
10	0.398

Συμπέρασμα: Παρατηρούμε ότι για coef0 έως 0.5 το accuracy αυξάνεται ενώ μετά το 0.5 αρχίζει και πέφτει.

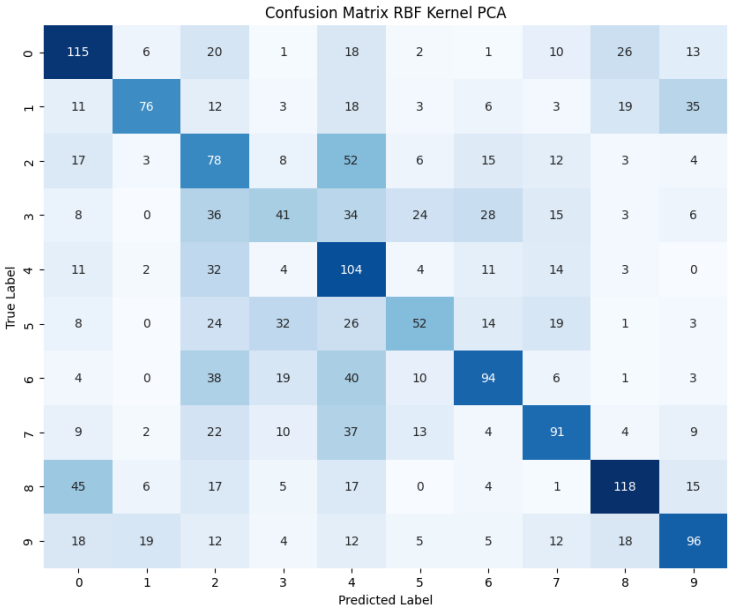
Για coef0=0.5 δοκιμάζω διαφορετικά degree = [2, 3, 4, 5]

Degree	Accuracy
2	0.4085
3	0.4425
4	0.447
5	0.439

NCC Classifier (kpca + lda)

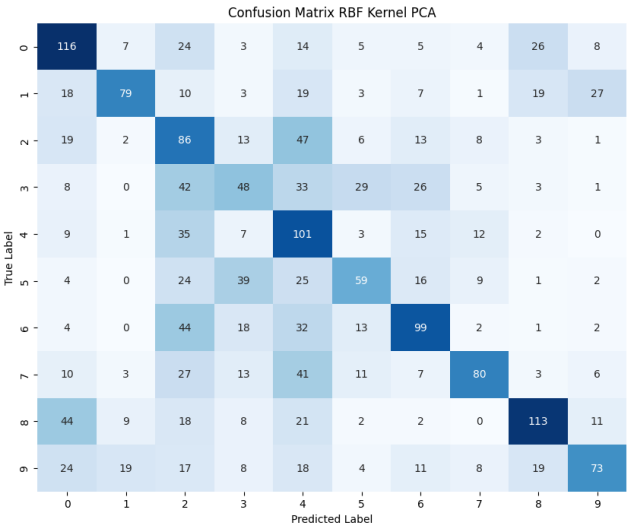
Δοκιμάζουμε kpca (rbf kernel) + lda με Nearest Centroid Classifier με metric = [‘manhattan’, ‘euclidean’]. Η ακρίβεια με metric = ‘manhattan’ για το test set ισούται με acc=0.4325 ενώ για metric = ‘euclidean’ είναι acc=0.427.

Για metric = ‘manhattan’:



Classification Report:	precision	recall	f1-score	support
0	0.47	0.54	0.50	212
1	0.67	0.41	0.51	186
2	0.27	0.39	0.32	198
3	0.32	0.21	0.25	195
4	0.29	0.56	0.38	185
5	0.44	0.29	0.35	179
6	0.52	0.44	0.47	215
7	0.50	0.45	0.47	201
8	0.60	0.52	0.56	228
9	0.52	0.48	0.50	201
accuracy			0.43	2000
macro avg	0.46	0.43	0.43	2000
weighted avg	0.46	0.43	0.44	2000

Για metric=’euclidean’:



Classification Report:	precision	recall	f1-score	support
0	0.45	0.55	0.50	212
1	0.66	0.42	0.52	186
2	0.26	0.43	0.33	198
3	0.30	0.25	0.27	195
4	0.29	0.55	0.38	185
5	0.44	0.33	0.38	179
6	0.49	0.46	0.48	215
7	0.62	0.40	0.48	201
8	0.59	0.50	0.54	228
9	0.56	0.36	0.44	201
accuracy			0.43	2000
macro avg	0.47	0.42	0.43	2000
weighted avg	0.47	0.43	0.43	2000

NCC Grid Search – 5 & 10 -Fold Cross Validation

Εκτελέσαμε grid search (metric=[euclidean, manhattan]) 5-fold και 10-fold cross validation με kpca (linear, rbf & polynomial) και βρήκαμε το καλύτερο score για το 5 & 10-fold cross validation για κάθε kpca.

5-fold cross validation results

	linear	poly	rbf
Best n_components	211	2470	4042
Best Parameters for NCC	{'metric': 'euclidean'}	{'metric': 'euclidean'}	{'metric': 'euclidean'}
Best Score for NCC	0.4715	0.815667	0.928667

10-fold cross validation results

	linear	poly	rbf
Best n_components	211	2470	4042
Best Parameters for NCC	{'metric': 'euclidean'}	{'metric': 'euclidean'}	{'metric': 'euclidean'}
Best Score for NCC	0.472	0.8155	0.928333

Ο καλύτερος kernel είναι ο RBF και η ακρίβεια για το test set ισούται με acc=0.427 και στις δύο περιπτώσεις. Σε σχέση με τον SVM ο KNN με rbf kernel pca δίνει χαμηλότερη ακρίβεια.

Παρατηρώ επίσης ότι με linear kernel pca ακόμα και στο train set η ακρίβεια του CV είναι αρκετά χαμηλή.

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

Coef0	Accuracy
0.01	0.338
0.05	0.381
0.1	0.4015
0.5	0.4345
1	0.4375
10	0.399

Συμπέρασμα: Παρατηρούμε ότι για coef0 έως 1 το accuracy αυξάνεται ενώ μετά το 1 αρχίζει και πέφτει.

Για coef0=1 δοκιμάζουμε διαφορετικά Degree = [2, 3, 4, 5]

Degree	Accuracy
2	0.4
3	0.4375
4	0.4345
5	0.4345
6	0.4165