Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Σχολή Θετικών Επιστημών
Τμήμα Πληροφορικής

Έκθεση αποτελεσμάτων στην εργασία με θέμα Support Vector Machines (SVMs)

Βασίλειος Ασημακόπουλος

Περίληψη

Στο παρόν report γίνεται μία προσπάθεια εξήγησης του κώδικα που γράφτηκε σε Python με σκοπό την υλοποίηση ενός **Support Vector Machine** για πρόβλημα διαχωρισμού κλάσεων. Επιλέχθηκαν δύο βάσεις δεδομένων με πολλαπλές κλάσεις η καθεμία. Στην αρχή πραγματοποιήθηκε προετοιμασία των δεδομένων της κάθε βάσης και έπειτα χωρίστηκε καθεμία σε train και test λίστες οι οποίες χρησιμοποιήθηκαν για την υλοποίηση **γραμμικού SVM, RBF SVM, KNN, NCC**, τα οποία θα συγκριθούν παρακάτω.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

MNIST DIGIT

Η βάση που επιλέχθηκε είναι η MNIST Digit και αποτελείται από 42000 σειρές και 785 στήλες.

1.1 Preprocessing

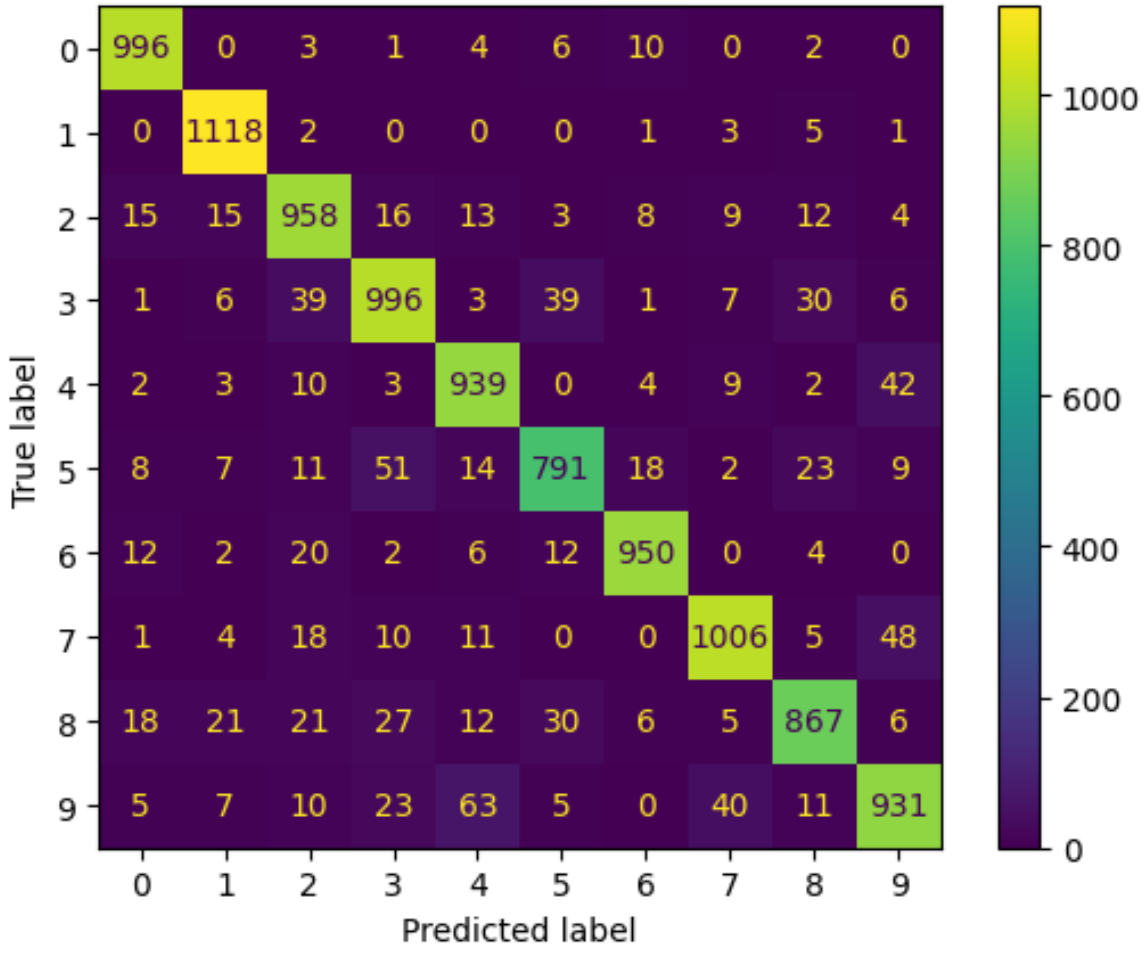
Ο κώδικας ξεκινάει με την εισαγωγή των κατάλληλων βιβλιοθηκών συγκεκριμένα την **sklearn, pandas, matplotlib, os, numpy** . Στην συνέχεια μέσω της pandas διαβάζεται το dataset το οποίο με την εντολή **describe** ελέγχθηκε το μέγεθος της βάσης, ο μέσος όρος κάθε στήλης και ο μέγιστος αριθμός σε κάθε στήλη. Με την εντολή **data.isnull().sum().head** ελέγχθηκε εάν λείπουν τιμές από τις στήλες. Εφόσον όλα ήταν καλά, χωρίστηκε η βάση σε δύο παραμέτρους/λίστες **X, y**. Η λίστα **X** περιέχει όλες τις λίστες πλήν της στήλης "label". Από την άλλη η λίστα **y** περιέχει μόνο την στήλη "label". Μετά τον διαχωρισμό της βάσης σε δύο λίστες , χρησιμοποιήθηκε η εντολή **MinMaxScaler(feature range=(**$0, 1$**))** στην λίστα **X**, για να κανονικοποιηθούν όλα τα δεδομένα ώστε να είναι πιο έυκολο να χρησιμοποιηθούν αργότερα από τα μοντέλα. Επίσης χρησιμοποίηθηκε η εντολή **scale(X)** η οποία τυποποιεί κάθε τιμή αφαιρώντας τον μέσο όρο της στήλης και διαιρώντας την με την τυπική απόκλιση.

1.2 Model Building

Μετα την επιτυχή ολοκλήρωση της επεξεργασίας της βάσης δεδομένων, ορίστηκαν τέσσερις παράμετροι με βάση τις δύο λίστες με την εντολή **X train, X test, y train, y test = train test split(X scaled, y, random state=**40**)**.

1.2.1 Linear SVM

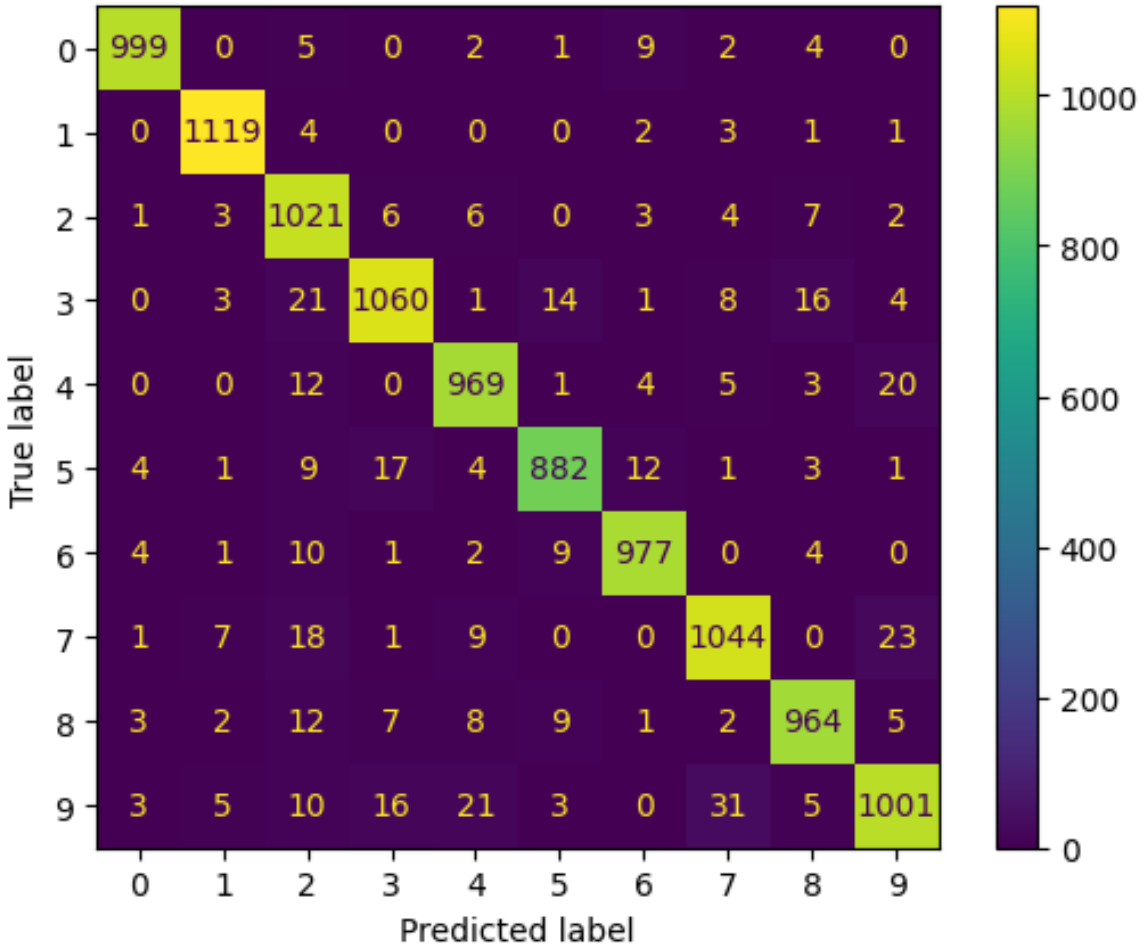
Στην συνέχεια αναπτύχθηκε το πρώτο μοντέλο, το οποίο είναι το γραμμικό SVM, και τυπώθηκε το ποσοστό ακρίβειας στο training και στο testing του μοντέλου. Επίσης τυπώθηκε το Confusion Matrix 1.1 του μοντέλου για το testing το οποίο δείχνει σε ένα διάγραμμα πόσα "labels" πέτυχε να κατηγοροποιήσει σωστά το μοντελο με τα δεδομένα που του δώθηκαν στο **X test**. Το **train accuracy** είναι 0.99428 ενώ το **test accuracy** είναι 0.90971. Ο χρόνος εκπαίδευσης του μοντέλου ήταν περίπου 2 minutes (1.47min)



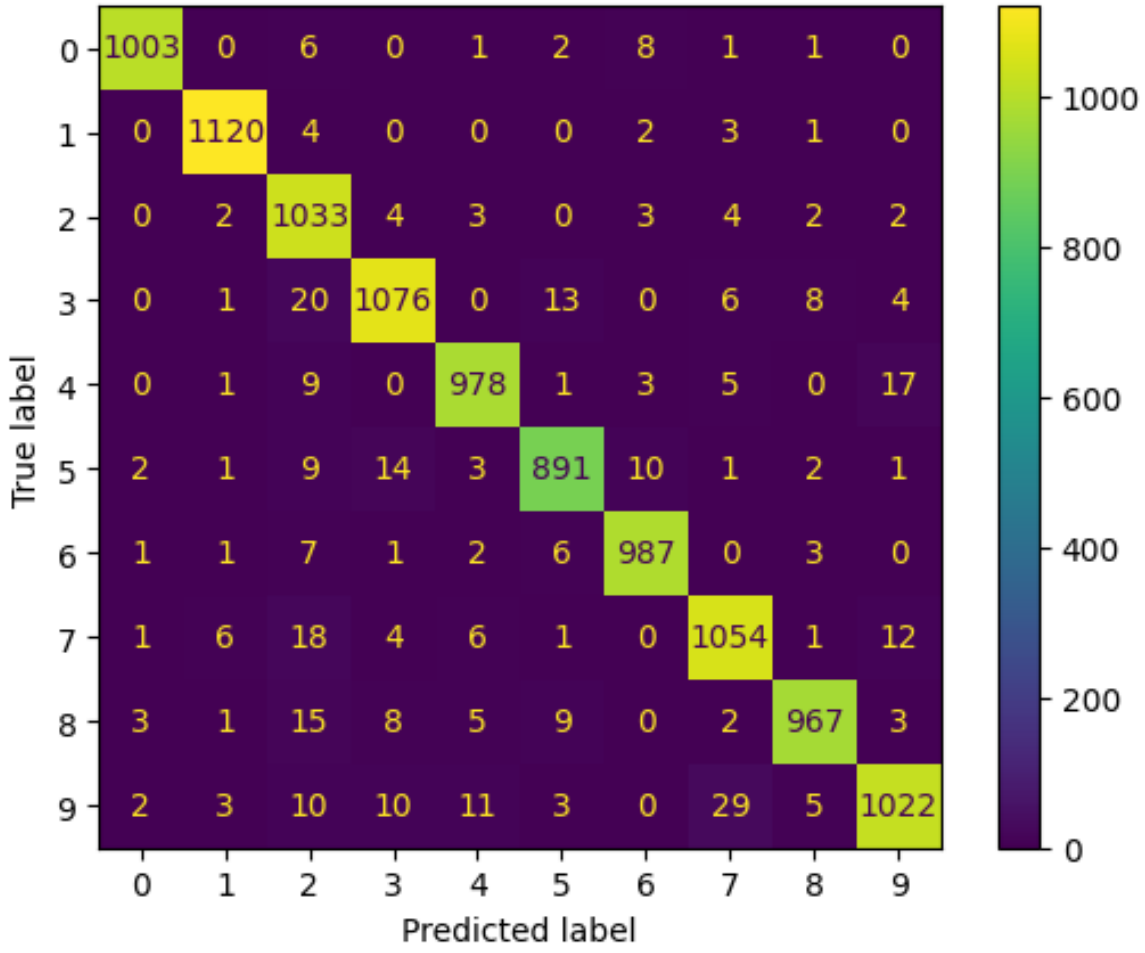
Σχήμα 1.1: Confusion Matrix linear SVM.

1.2.2 RBF SVM

Το δεύτερο μοντέλο που υλοποιήθηκε ήταν το **RBF SVM**. Και εδώ τυπώθηκαν τα ποσοστά ακρίβειας στο training και στο testing του μοντέλου, όπως και το Confusion Matrix 1.2 του. Το **train accuracy** είναι 0.98495 ενώ το **test accuracy** είναι 0.95580. Παρατηρείται μεγάλη αύξηση από το γραμμικό SVM(θα αναλυθεί περαιτέρω στα συμπεράσματα). Ο χρόνος εκπαίδευσης του μοντέλου ήταν περίπου 5 minutes (4.54min). Έπειτα χρησιμοποιήθηκε η εντολή **GridSearchCV** με την οποία έγινε **cross-validation** για συγκεκριμένες παραμέτρους **C** : $[0.5, 1, 5]$ και **gamma** : $['scale', 0.5, 0.001]$.Τυπώθηκαν τα αποτελέσματα του καλύτερου συνδυασμού παραμέτρων για τον πυρήνα rbf οι οποίες είναι **gamma** : $['scale']$ και **C** : $[5]$ με ποσοστό ακρίβειας στο train και test 0.99784, 0.96485 αντίστοιχα. Μετά απο σύγκριση των αποτελεσμάτων του **GridSearchCV** συμπεραίνεται ότι όσο μειώνεται η παράμετρος **gamma** τόσο μειώνεται το ποσοστό ακρίβειας του testing. Η διάρκεια εκτέλεσης του **cross-validation** ήταν πάνω απο 3 ώρες λόγω του όγκου των δεδομένων αλλά και επειδή επιλέχθηκε να γίνουν 5 folds για κάθε υποψήφιο συνδυασμό των δύο παραμέτρων (9 συνολικά υποψήφιες τιμές). Μετά το **cross-validation** υλοποιήθηκε το **RBF SVM** εκ νέου με τις νέες παραμέτρους για να ελεγχθούν τα ποσοστά ακρίβειας που τυπώθηκαν πιο πάνω. Τυπώθηκε το Confusion Matrix 1.3. Ο χρόνος υλοποιήσης του συγκεκριμένου μοντέλου ήταν περίπου 4 λεπτα (3.42min).



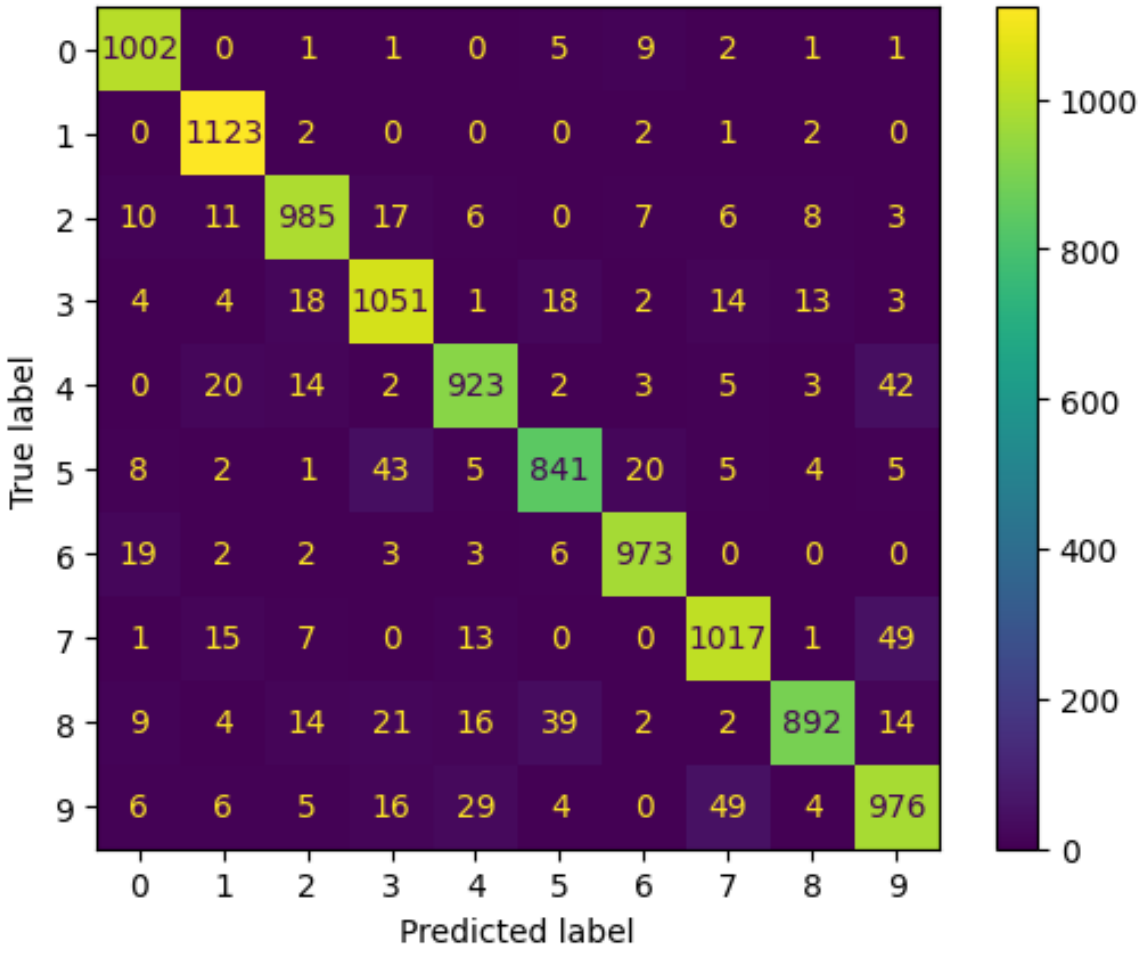
Σχήμα 1.2: Confusion Matrix RBF SVM



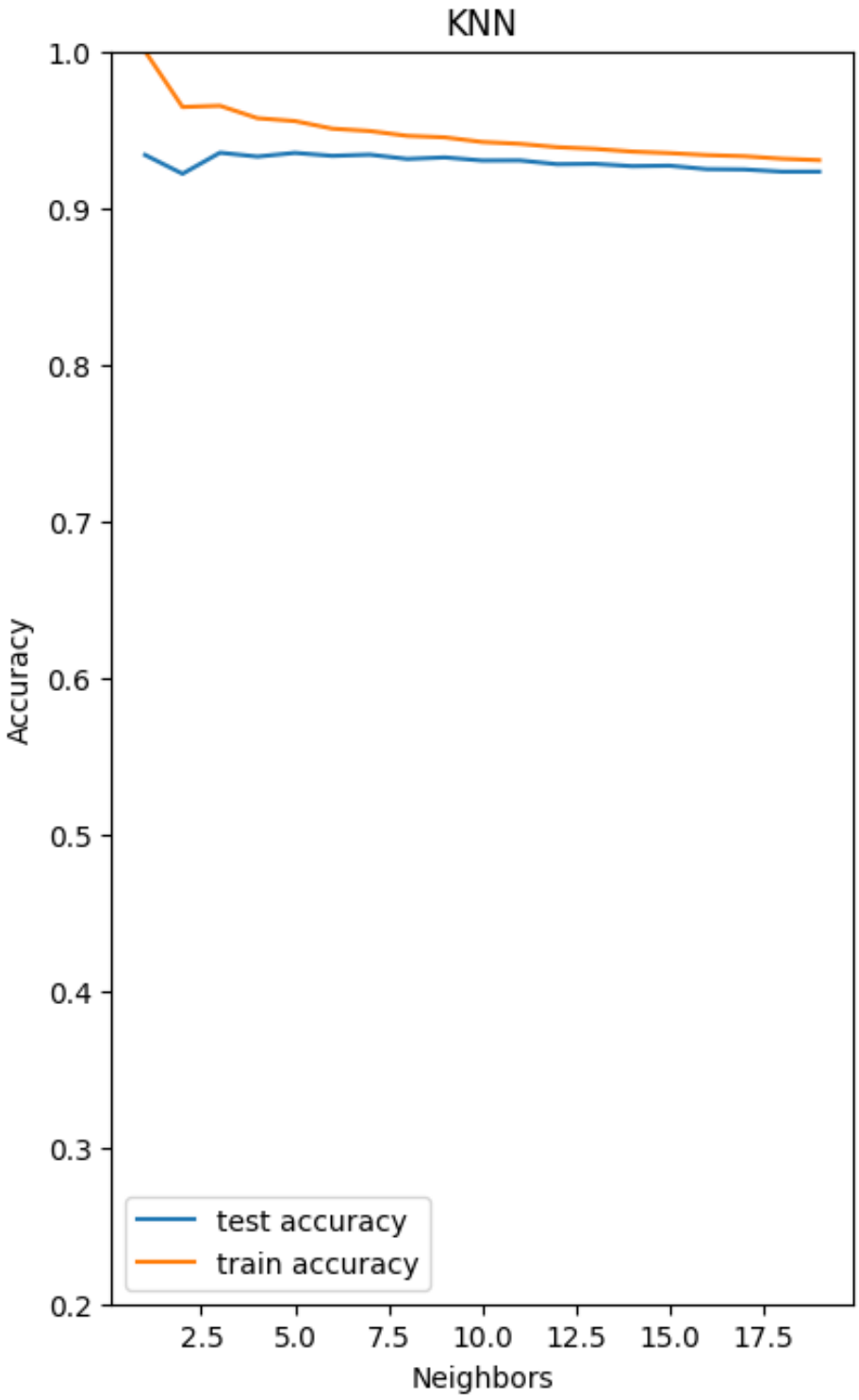
Σχήμα 1.3: Confusion Matrix RBF SVM with the best parameters

1.2.3 KNN

Το τρίτο μοντέλο που χρησιμοποιήθηκε ήταν το **KNN**. Επιλέχθηκε τυχαία το KNN να έχει 3 γείτονες μέσω της παραμέτρου **n neighbors**. Τυπώθηκαν τα ποσοστά ακρίβειας του μοντέλου για το training και testing τα οποία ήταν 0.96749, 0.9317 αντίστοιχα. Τυπώθηκε και σε αυτό το μοντέλο το Confusion Matrix 1.4. Ο χρόνος εκπαίδευσης του μοντέλου ήταν 12 second. Χρησιμοποιήθηκε το **cross-validation** μέσω της εντολής **GridSearchCV** για να βρεθεί η καλύτερη τιμή που θα πάρει η παράμετρος **n neighbors**, η οποία ήταν η αρχική(γείτονες : 3), οπότε δεν εκπαιδεύτηκε από την αρχή το μοντέλο. Ο χρόνος που "έτρεχε" το **cross-validation** ήταν περίπου 10 minutes (10.39min). Επίσης τυπώθηκε το διάγραμμα πλήθους γειτόνων-ακρίβειας 1.5, από το οποίο εξάγεται το συμπέρασαμα ότι στα συγκεκριμένη βάση δεδομένων όσο μεγαλώνει το πλήθος των γειτόνων τόσο μειώνεται η ακρίβεια της εκπαίδευσης και του testing.



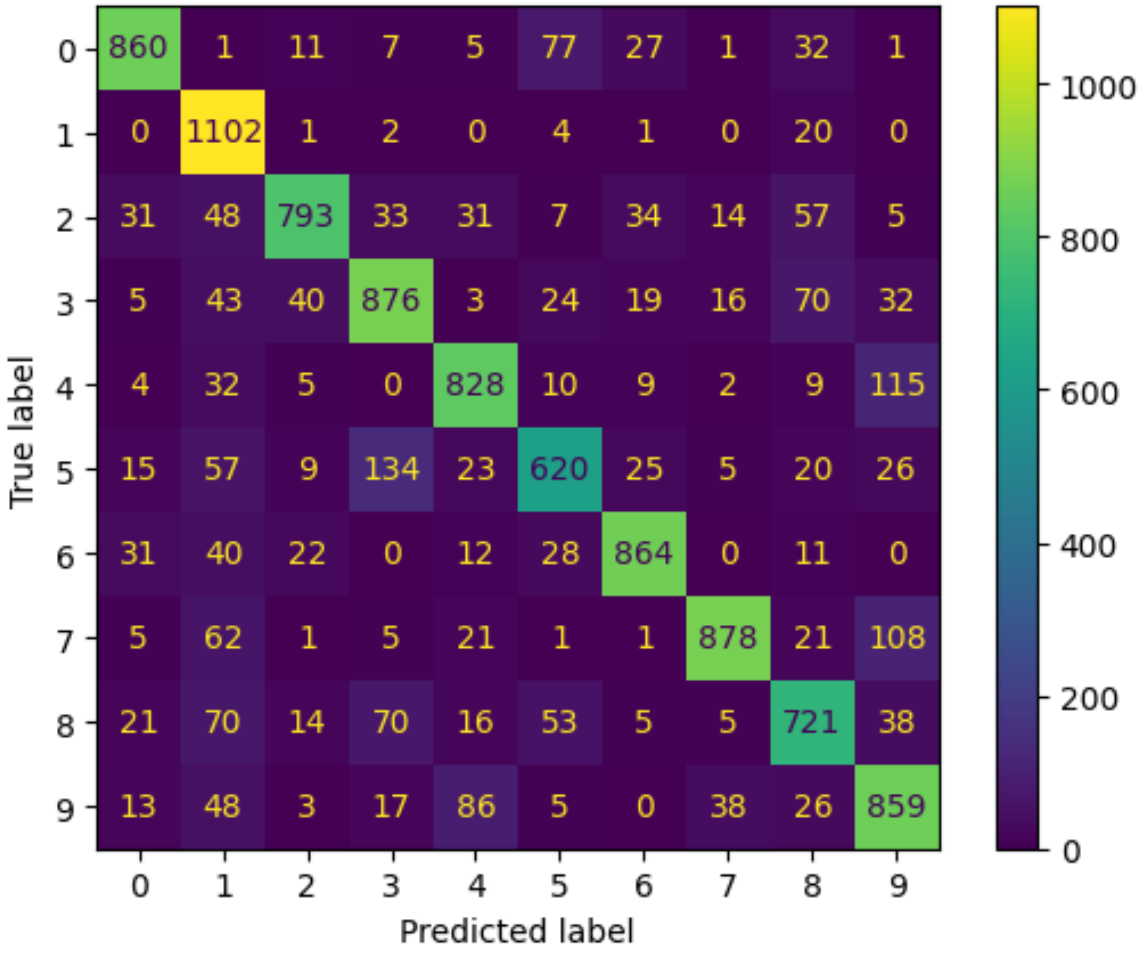
Σχήμα 1.4: Confusion Matrix KNN



Σχήμα 1.5: Διάγραμμα πλήθους γειτόνων-ακρίβειας

1.2.4 NCC

Το τελευταιο μοντέλο που χρησιμοιποιήθηκε ήταν το **NCC**. Τα ποσοστά ακρίβειας του μοντέλου για το training και testing ήταν 0.804, 0.8 αντίστοιχα. Τυπώθηκε και σε αυτό το μοντέλο το Confusion Matrix 1.4. Ο χρόνος εκπαίδευσης του μοντέλου ήταν 0.5 second.



Σχήμα 1.6: Confusion Matrix KNN

1.3 Σύγκριση Μοντέλων

Θα συγκριθούν τα δυο **SVM** μοντέλα μεταξύ τους και στην συνέχεια θα συγκριθούν τα δύο μοντέλα με το **RBF SVM**.

1.3.1 Linear SVM vs RBF SVM

Το γραμμικο μοντέλο του **SVM** εμφανίζει χαμηλότερη ακρίβεια διότι οι κλάσεις της βάσης πανω στις οποίες εκπαιδεύται είναι τυχαία διασκορπισμένες, άρα δεν μπορεί ευκολα να τις ταξινομήσει γραμμικά. Οπότε το μη γραμμικό μοντέλο παρόλο που εκπαιδεύεται στον διπλάσιο χρόνο (4min) έχει πολυ μεγαλύτερο ποσοστό ακρίβειας περίπου 97%. Αυτό συμβαίνει γιατί τα δεδομένα ανεβαίνουν σε διαστάσεις, έτσι το μοντέλο τα διαχωρίζει πιο εύκολα.

1.3.2 KNN vs RBF SVM

Μεταξύ των δύο μοντέλων το **RBF SVM** έχει μεγαλύτερο ποσστό ακρίβειας στο testing (97%) έναντι του **KNN** (93%), παρόλο που το **KNN** είναι πάρα πολύ γρήγορο.Χρόνος εκπαίδευσης **KNN** : 12sec. Χρόνος εκπαίδευσης **RBF SVM** : 4min.

1.3.3 NCC vs RBF SVM

Μεταξύ των δύο μοντέλων το **RBF SVM** έχει μεγαλύτερο ποσστό ακρίβειας στο testing (97%) απο το **NCC** (80%), παρόλο που το **NCC** είναι το πιο γρήγορο μοντέλο απο τα τέσσερα, με χρόνο εκπαίδευσης 0.5s.

Κεφάλαιο 2

Muscle Activity Dataset

Η παρακάτω βάση δεδομένων αποτελείται από 11678 σειρές και 65 στήλες (μετά την ένωση των αρχείων)

2.1 Preprocessing

Ο κώδικας ξεκινάει με την εισαγωγή των κατάλληλων βιβλιοθηκών συγκεκριμένα την **sklearn, pandas, matplotlib, os, numpy** . Στην συνέχεια μέσω της pandas διαβάζεται το dataset το οποίο είναι χωρισμένο σε 4 τύπου csv αρχεία. Μέσω της εντολής **pd.concat** δημιουργήθηκε ένα κοινό αρχείο, το οποίο με την εντολή **info** ελεγχθηκε εάν ήταν επιτυχής η ένωση των 4 τύπου csv , όπως και το μέγεθος του πίνακα. Με την εντολή **data.isnull().sum().head** ελέχθηκε εάν λείπουν δεδομένα, και με την εντολή **order= data[64].unique()** τυπώθηκε ο αριθμός των κλάσεων οι οποίες βρίσκονται στην στήλη 64, και με την εντολή **data[64].value counts().sort values(ascending=False)** τυπώθηκε το μέγεθος της κάθε κλάσης 2.1. Εφόσον όλα ήταν καλά, χωρίστηκε η βάση σε δύο

Πίνακας 2.1: Μέγεθος κλάσεων

Κλάσεις k	Μέγεθος
0	2910
1	2903
2	2943
3	2922

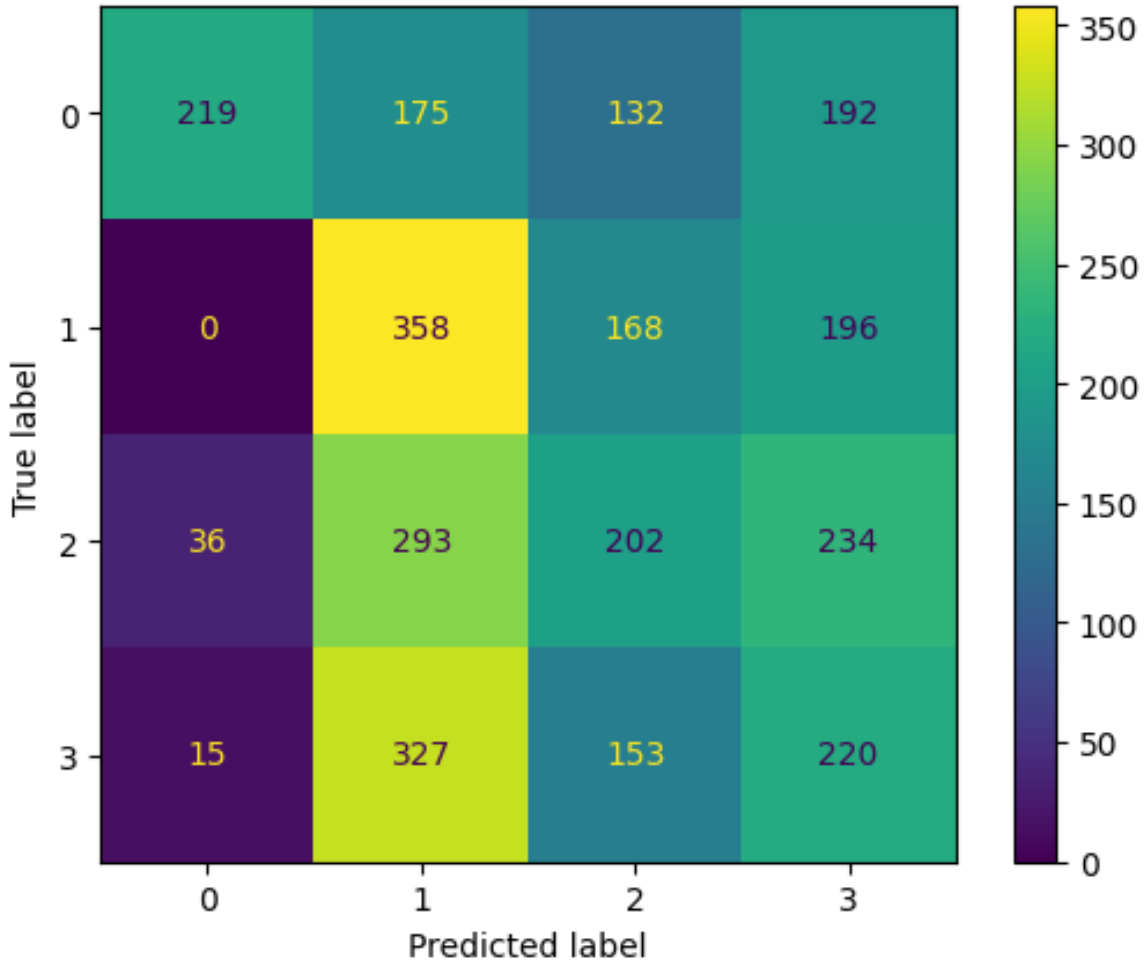
παραμέτρους/λίστες **X, y**. Η λίστα **X** περιέχει όλες τις λίστες πλήν της στήλης "64". Από την άλλη η λίστα **y** περιέχει μόνο την στήλη "64". Μετά τον διαχωρισμό της βάσης σε δύο λίστες , χρησιμοποιήθηκε η εντολή **MinMaxScaler(feature range=(**$0, 1$**))** στην λίστα **X**, για να κανονικοποιηθούν όλα τα δεδομένα ώστε να είναι πιο έυκολο να χρησιμοποιηθούν αργότερα από τα μοντέλα. Επίσης χρησιμοποίηθηκε η εντολή **scale(X)** η οποία τυποποιεί κάθε δεδομένο αφαιρώντας τον μέσο όρο και διαιρώντας κάθε τιμή με την τυπική απόκλιση.

2.2 Model Building

Μετα την επιτυχή ολοκλήρωση της επεξεργασίας της βάσης δεδομένων, ορίστηκαν τέσσερις παράμετροι με βάση τις δύο λίστες με την εντολή **X train, X test, y train, y test = train test split(X scaled, y, random state=**40**)**.

2.2.1 Linear SVM

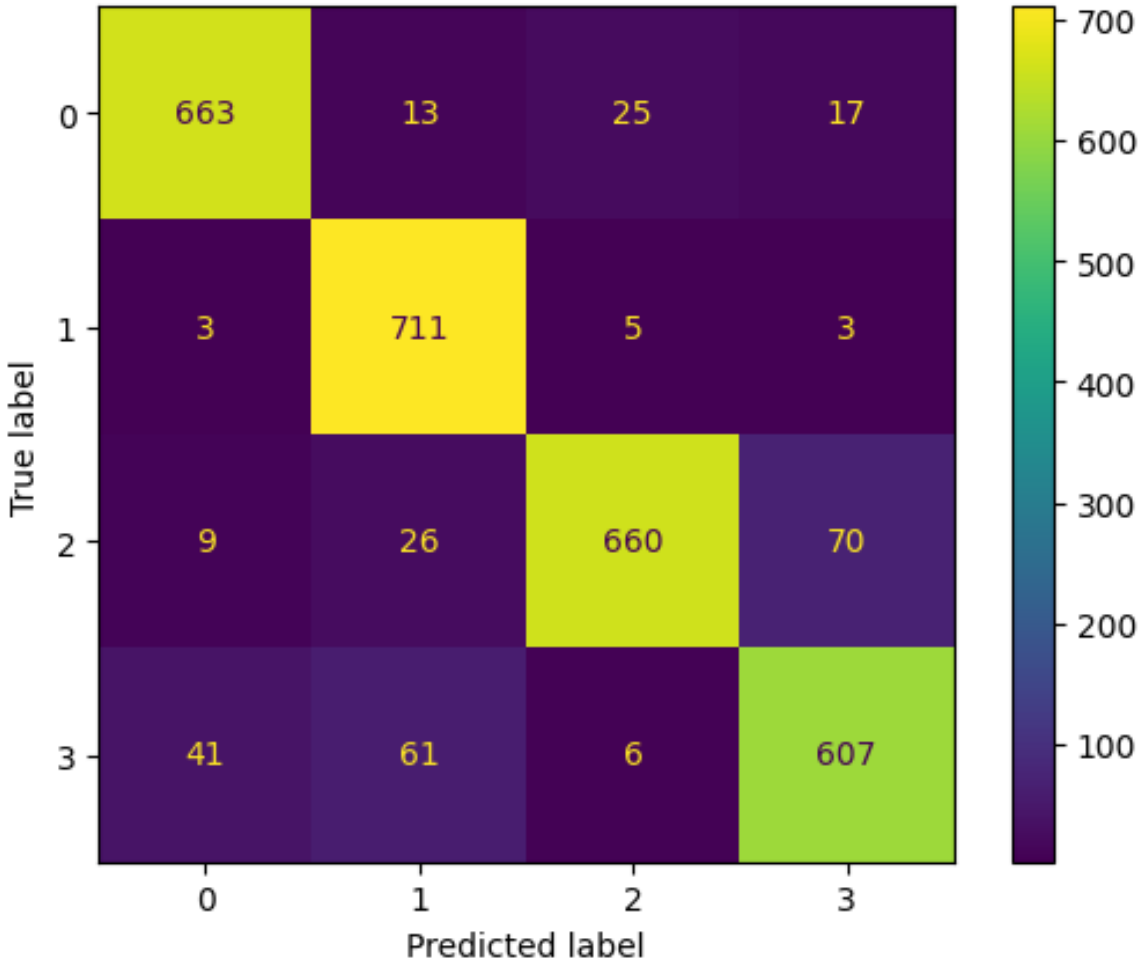
Στην συνέχεια αναπτύχθηκε το πρώτο μοντέλο, το οποίο είναι το γραμμικό SVM, και τυπώθηκε το ποσοστό ακρίβειας στο training και στο testing του μοντέλου. Επίσης τυπώθηκε το Confusion Matrix 2.1 του μοντέλου για το testing το οποίο δείχνει σε ένα διάγραμμα πόσα "labels" πέτυχε να κατηγοροποιήσει σωστά το μοντελο με τα δεδομένα που του δώθηκαν στο **X test**. Το **train accuracy** είναι 39% ενώ το **test accuracy** είναι 34%. Ο χρόνος εκπαίδευσης του μοντέλου ήταν 12.9 second.



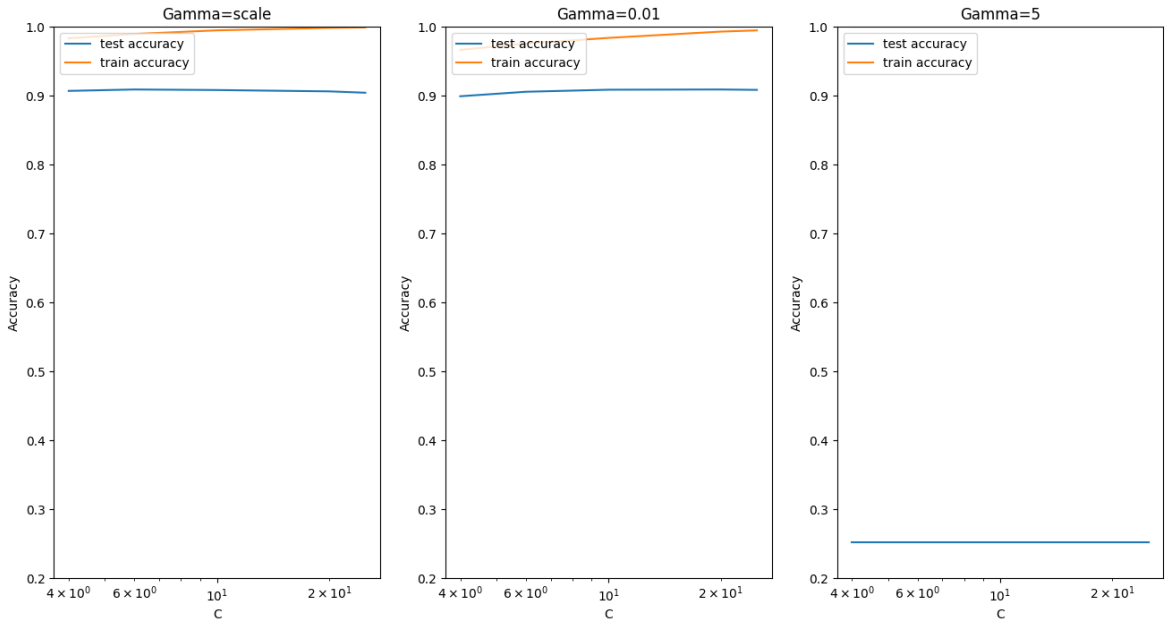
Σχήμα 2.1: Confusion Matrix linear SVM.

2.2.2 RBF SVM

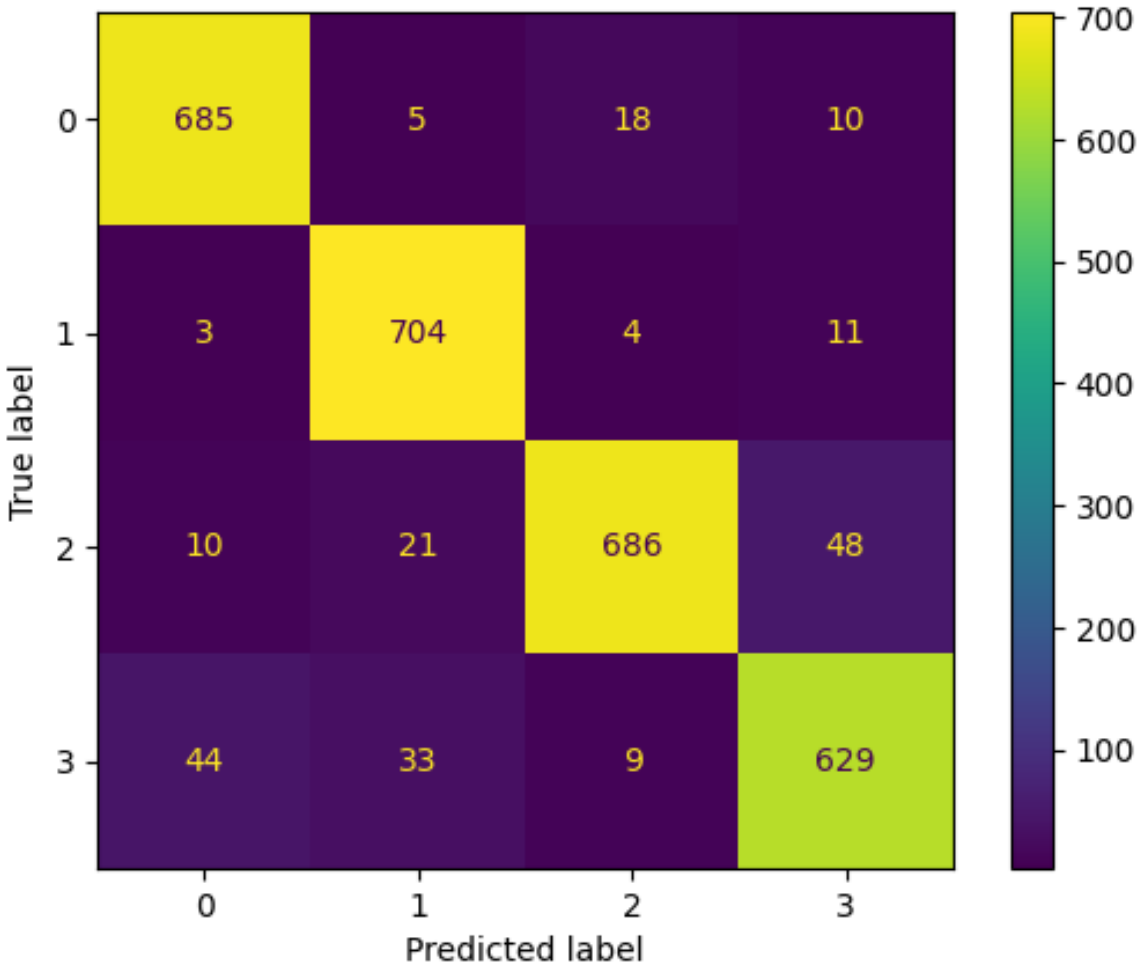
Το δεύτερο μοντέλο που υλοποιήθηκε ήταν το **RBF SVM**. Και εδώ τυπώθηκαν τα ποσοστά ακρίβειας στο training και στο testing του μοντέλου, όπως και το Confusion Matrix 2.2 του. Το **train accuracy** είναι 95.1% ενώ το **test accuracy** είναι 90.4%. Ο χρόνος εκπαίδευσης του μοντέλου ήταν περίπου 2.7 second. Έπειτα χρησιμοποιήθηκε η εντολή **GridSearchCV** με την οποία έγινε **cross-validation** για συγκεκριμένες παραμέτρους **C** : $[4, 6, 10, 20, 25]$ και **gamma** : $['scale', 0.01, 5]$.Τυπώθηκαν τα αποτελέσματα του καλύτερου συνδυασμού παραμέτρων για τον πυρήνα rbf οι οποίες είναι **gamma** : $['scale']$ και **C** : $[6]$ με ποσοστό ακρίβειας στο train και test 98.9%, 92.6% αντίστοιχα. Με την βοήθεια της βιβλιοθήκης matplotlib τυπώθηκαν τρία διαγράμματα για κάθε **gamma** που χρησιμοποιήθηκε στο **cross-validation** Έτσι μετά απο σύγκριση των διαγραμμάτων συμπεραίνεται ότι όσο αυξάνεται η παράμετρος **gamma** τόσο μειώνεται το ποσοστό ακρίβειας του testing 2.3. Η διάρκεια εκτέλεσης του **cross-validation** ήταν 1.39 minutes. Επιλέχθηκε να γίνουν 5 folds για κάθε υποψήφιο συνδυασμό των δύο παραμέτρων (15 συνολικά υποψήφιες τιμές). Μετά το **cross-validation** υλοποιήθηκε το **RBF SVM** εκ νέου με τις νέες παραμέτρους για να ελεγχθούν τα ποσοστά ακρίβειας που τυπώθηκαν πιο πάνω. Τυπώθηκε το Confusion Matrix 2.4. Ο χρόνος υλοποιήσης του συγκεκριμένου μοντέλου ήταν περίπου 4.6 δευτερόλεπτα.



Σχήμα 2.2: Confusion Matrix RBF SVM



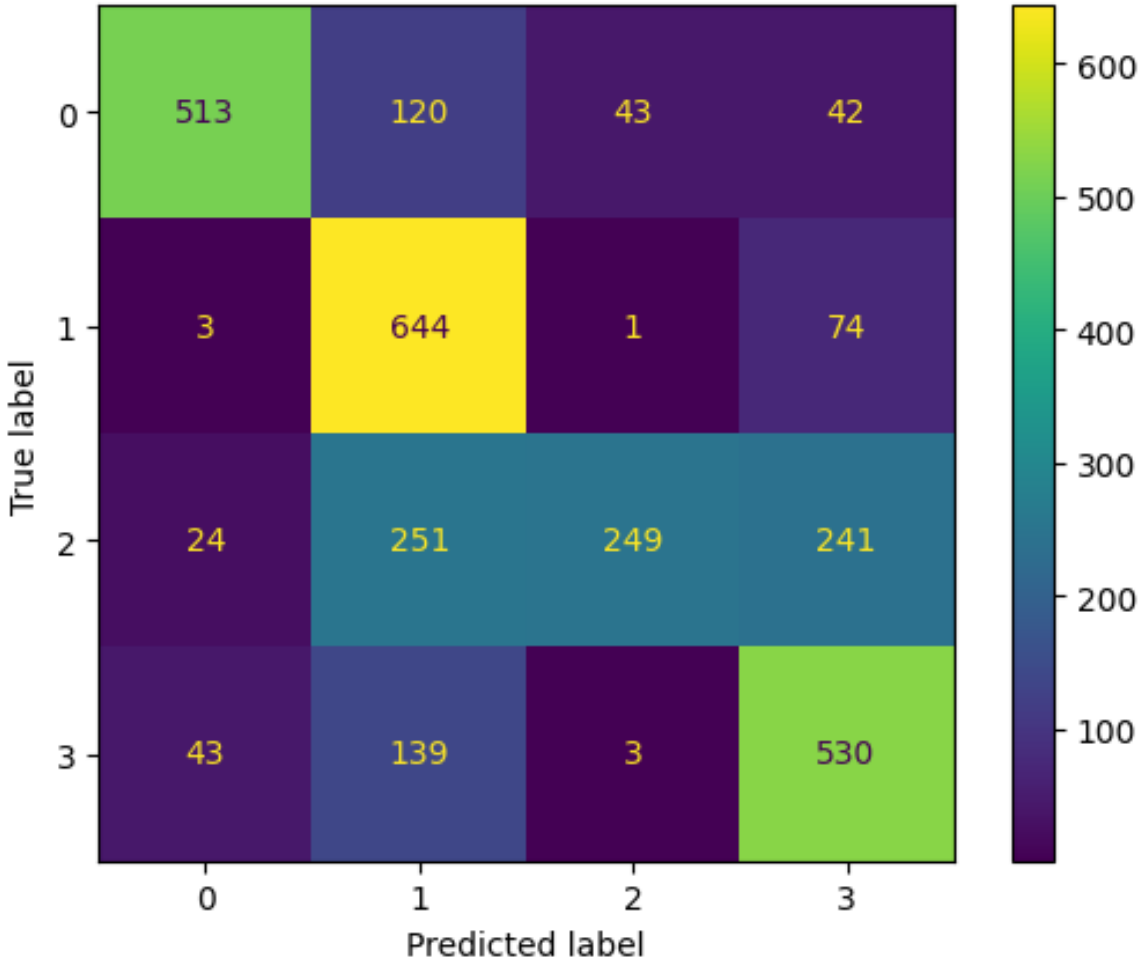
Σχήμα 2.3: Διαγράμματα **C**-ακρίβειας για σταθερό **gamma**



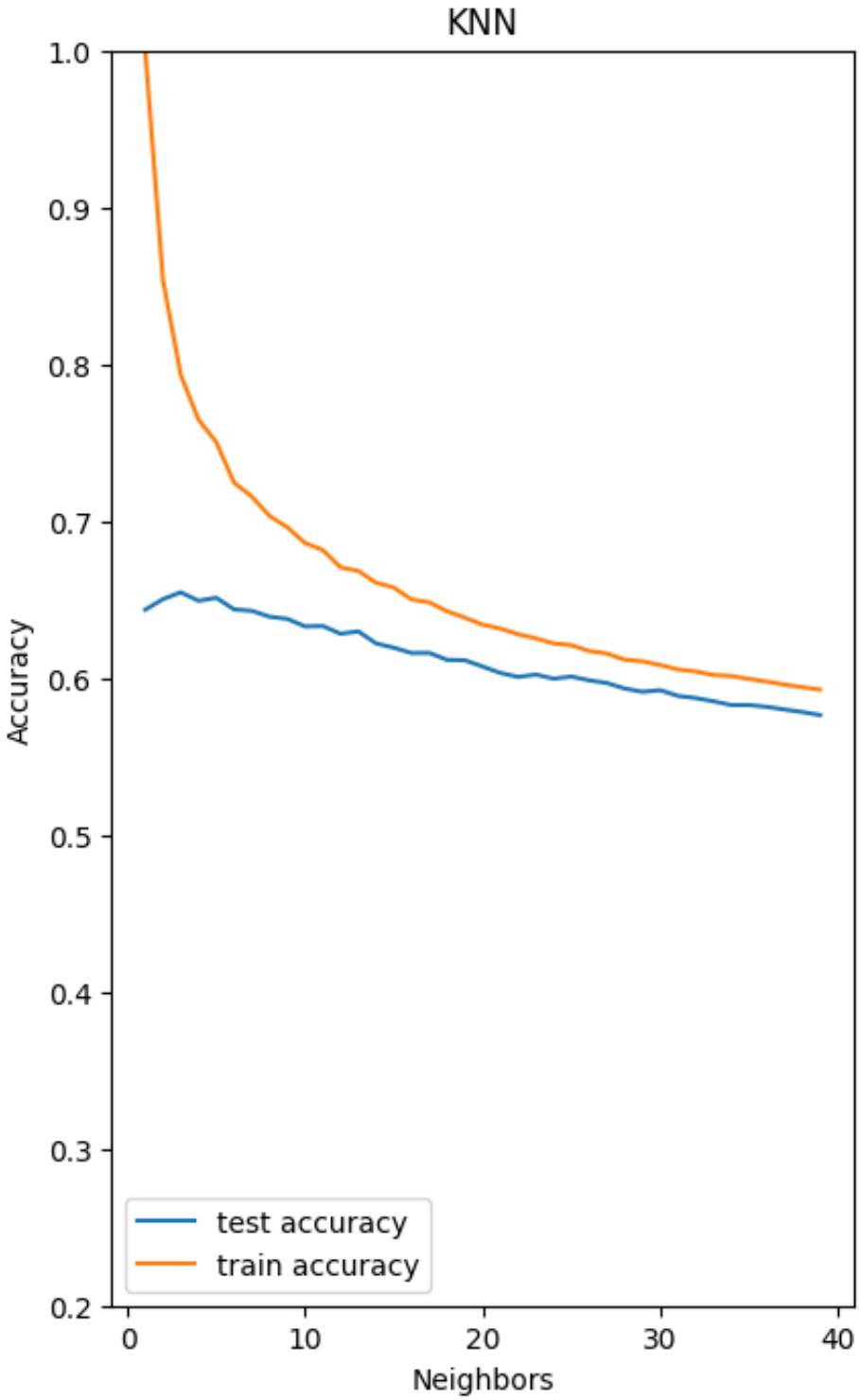
Σχήμα 2.4: Confusion Matrix RBF SVM with the best parameters

2.2.3 KNN

Το τρίτο μοντέλο που χρησιμοποιήθηκε ήταν το **KNN**. Επιλέχθηκε τυχαία το KNN να έχει 3 γείτονες μέσω της παραμέτρου **n neighbors**. Τυπώθηκαν τα ποσοστά ακρίβειας του μοντέλου για το training και testing τα οποία ήταν $79.9\%, 66\%$ αντίστοιχα. Τυπώθηκε και σε αυτό το μοντέλο το Confusion Matrix 2.5. Ο χρόνος εκπαίδευσης του μοντέλου ήταν 0.5 second. Χρησιμοποιήθηκε το **cross-validation** μέσω της εντολής **GridSearchCV** για να βρεθεί η καλύτερη τιμή που θα πάρει η παράμετρος **n neighbors**, η οποία ήταν η αρχική (γείτονες : 3), οπότε δεν εκπαιδεύτηκε από την αρχή το μοντέλο. Ο χρόνος που "έτρεχε" το **cross-validation** ήταν περίπου 14 second. Επίσης τυπώθηκε το διάγραμμα πλήθους γειτόνων-ακρίβειας 2.6, από το οποίο εξάγεται το συμπέρασαμα ότι στα συγκεκριμένη βάση δεδομένων όσο μεγαλώνει το πλήθος των γειτόνων τόσο μειώνεται η ακρίβεια της εκπαίδευσης (εκθετική μείωση) και του testing.



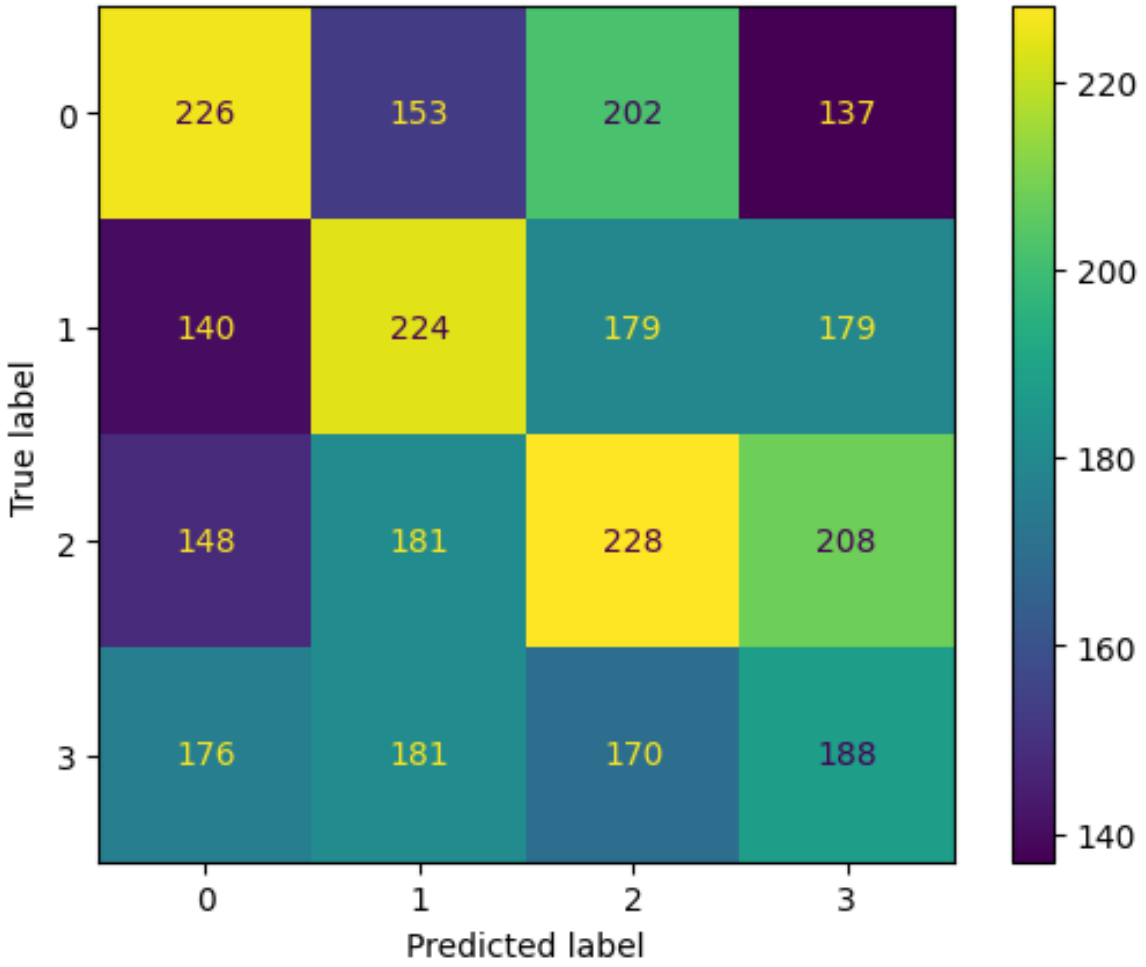
Σχήμα 2.5: Confusion Matrix KNN



Σχήμα 2.6: Διάγραμμα πλήθους γειτόνων-ακρίβειας

2.2.4 NCC

Το τελευταιο μοντέλο που χρησιμοιποιήθηκε ήταν το **NCC**. Τα ποσοστά ακρίβειας του μοντέλου για το training και testing ήταν 33.3%, 29.65% αντίστοιχα. Τυπώθηκε και σε αυτό το μοντέλο το Confusion Matrix 1.4. Ο χρόνος εκπαίδευσης του μοντέλου ήταν 0.1 second.



Σχήμα 2.7: Confusion Matrix NCC

2.3 Σύγκριση Μοντέλων

Θα συγκριθούν τα δυο **SVM** μοντέλα μεταξύ τους και στην συνέχεια θα συγκριθούν τα δύο μοντέλα με το **RBF SVM**.

2.3.1 Linear SVM vs RBF SVM

Το γραμμικο μοντέλο του **SVM** εμφανίζει χειρότερα αποτελέσματα διότι οι κλάσεις της βάσης πανω στις οποίες εκπαιδεύται είναι τυχαία διασκορπισμένες , άρα δεν μπορεί ευκολα να τις ταξινομήσει γραμμικά. Οπότε το μη γραμμικό μοντέλο βγάζει πολυ μεγαλύτερο ποσοστό ακρίβειας περίπου 92.6%, και στην συγκεκριμένη περίπτωση (λογικά λόγω μικρού πλήθους δεδομένων) ο χρόνος εκτέλεσης είανι πιο σύντομος (4.6sec). Αυτό συμβαίνει γιατί τα δεδομένα ανεβαίνουν σε διαστάσεις, οπότε το μοντέλο μπορεί να τα διαχωρίσει πιο εύκολα.

2.3.2 KNN vs RBF SVM

Μεταξύ των δύο μοντέλων το **RBF SVM** έχει μεγαλύτερο ποσστό ακρίβειας στο testing (92.6%) απο το **KNN** (66%), παρόλο που το **KNN** είναι πάρα πολύ γρήγορο.Χρόνος εκπαίδευσης **KNN** : 0.5sec. Χρόνος εκπαίδευσης **RBF SVM** : 4.6sec.

2.3.3 NCC vs RBF SVM

Μεταξύ των δύο μοντέλων το **RBF SVM** έχει μεγαλύτερο ποσστό ακρίβειας στο testing (92.6%) απο το **NCC** (29.65%), παρόλο που το **NCC** είναι το πιο γρήγορο μοντέλο απο τα 4, με χρόνο εκπαίδευσης 0.1sec.