Νευρωνικά Δίκτυα
Δεύτερη Εργασία

Αποστολίδου Αθηνά, ΗΜΜΥ, ΑΕΜ:10400

Περίληψη—Αντικείμενο της εργασίας είναι οι μηχανές διανυσμάτων υποστήριξης καθώς και η απόδοσή τους σε σύγκριση με άλλους κατηγοριοποιητές

I . Εισαγωγή

ΣΤΟ πλαίσιο εργασίας του μαθήματος Νευρωνικά δίκτυα και βαθία μάθηση υλοποιήθηκαν δύο μηχανές διανυσμάτων υποστήριξης. Η πρώτη αφορά την κατηγοριοποίηση των δύο πρώτων κλάσεων της Cifar-10 και η δεύτερη την κατηγοριοποίση όλων των κλάσεων της Cifar-10. Στην συνέχεια τα ποσοστά επιτυχούς κατηγοριοποίησης συγκρίθηκαν με τα αντιστοίχα ποσοστά που έδωσαν οι κατηγοριοποιητές πλησιέστερου γείτονα 1ος και 3ων γειτόνων, ο κατηγοριοποιήτης πλησιέστερου κέντρου και ενός MLP με ένα κρυφό επίπεδο που χρησιμοποιεί Hinge loss για βελτιστοποίηση. Για την εργασία χρησιμοποιήθηκε γλώσσα προγραμματισμού Python και βιβλιοθήκες numpy, scipy, pickle, time, sklearn, matplotlib, copy, pandas, cvxopt.

II . Κατηγοριοποίηση των 2 πρώτων κλάσεων της CIFAR-10

Σε αυτή την ενότητα παρουσιάζονται ο αλγόριθμος, οι δοκιμές και οι συγκρίσεις που έγιναν στο πλαίσιο της υλοποίησης του SVM που κατηγοριοποιεί τις δύο πρώτες κλάσεις της Cifar-10. Για λόγους καλύτερης κατανόησης της θεωρίας των SVM σε αυτό το κομάτι της εργασίας δεν χρησιμοποιήθηκαν καθόλου έτοιμες συναρτήσεις της βιβλιοθήκης sklearn αντίθετα διατυπώθηκαν οι μαθηματικές εξισώσεις και επιλύθηκαν ως ένα πρόβλημα τετραγωνικού προγραμματισμού. Το πρόβλημα λύθηκε με την συνάρτηση solvers της βιβλιοθήκης cvxopt.

Στην περιπτώση μας επειδή τα δεδομένα δεν είναι απόλυτα γραμμικά διαχωρίσιμα, χρησιμοποιούμε τον Soft Margin SVM, όπου επιτρέπονται κάποιες παραβιάσεις (σφάλματα). Ακόμα, χρησιμοποιήθηκε γραμμικός πηρύνας. Το πρόβλημα διατυπώνεται ως εξής:

Πρόβλημα βελτιστοποίησης

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

υπό τους περιορισμούς:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

Όπου:

- C: Υπερπαράμετρος που καθορίζει το trade-off μεταξύ του περιθωρίου και του σφάλματος,
- ξ_i: Μεταβλητές slack (μετρούν την παραβίαση του περιθωρίου).

Δυϊκή Μορφή

Για την επίλυση, μετατρέπουμε το πρόβλημα στην δυϊκή μορφή με χρήση των Lagrange multipliers (α):

$$\max_{\alpha} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

υπό τους περιορισμούς:

$$0 \leq \alpha_i \leq C, \quad \forall i$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0$$

Όπου:

- α_i: Lagrange multipliers,
- $K(x_i, x_j)$: Ο πίνακας Gram ή πυρήνας ($K(x_i, x_j) = x_i^T x_j$ για γραμμικό SVM).

Υπολογισμός SUPPORT VECTORS, Διανύσματος Βαρών w και BIAS b

Εντοπισμός των *Support Vectors*

Κατά την εκπαίδευση του SVM, οι παράγοντες Lagrange α_i που είναι μη μηδενικοί ($\alpha_i > 0$) αντιστοιχούν στα Support Vectors. Ο εντοπισμός γίνεται ως εξής:

$$\text{Support Vectors: } X_{\text{support}} = \{x_i \mid \alpha_i > \epsilon\}, \quad \text{με } \epsilon \text{ μικρό θετικό όριο (π.χ., } \epsilon = 10^{-5}). \tag{1}$$

Οι ετικέτες των Support Vectors είναι:

$$y_{\text{support}} = \{y_i \mid \alpha_i > \epsilon\}. \tag{2}$$

Υπολογισμός του Διανύσματος Βαρών w

Το διάνυσμα βαρών w υπολογίζεται ως γραμμικός συνδυασμός των Support Vectors, των ετικετών τους y_i, και των αντίστοιχων παραγόντων α_i:

$$w = \sum_{i \in \text{support}} \alpha_i y_i x_i. \tag{3}$$

Υπολογισμός του Βιας b

Το bias b υπολογίζεται χρησιμοποιώντας τους Support Vectors. Έστω ένας Support Vector x_k με την αντίστοιχη ετικέτα y_k. Το bias δίνεται από τη μέση τιμή:

$$b = \frac{1}{N_{\text{support}}} \sum_{i \in \text{support}} \left(y_i - w^T x_i \right), \tag{4}$$

όπου $N_{\text{συππορτ}}$ είναι ο αριθμός των Support Vectors.

Πρόβλεψη Ετικετών

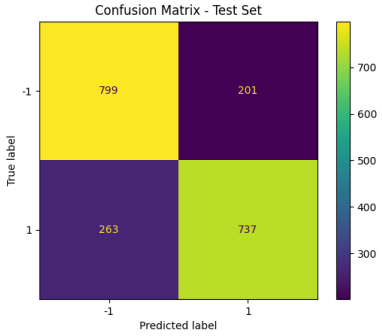
Για να προβλέψουμε την ετικέτα ενός νέου δείγματος x, χρησιμοποιούμε την εξίσωση απόφασης:

$$\hat{y} = \text{sign}(w^T x + b), \tag{5}$$

όπου $\text{sign}(\cdot)$ είναι η συνάρτηση πρόσημου που επιστρέφει $+1$ ή -1.

Από το σύνολο της Cifar-10 απομονώθηκαν 2 δύο πρώτες κλάσεις και έπειτα απο μια κανονικοποίση που δέχτηκαν πέρασαν στο SVM. Τα αποτελέσματα για μεταβλητή $C = 1$ φαίνονται στις παρακάτω εικόνες. Ο χρόνος εκτέλεσης του αλγορίθμου είναι 2225,92 δευτερόλεπτα, και η ακρίβεια κατηγοριοποίησης στο

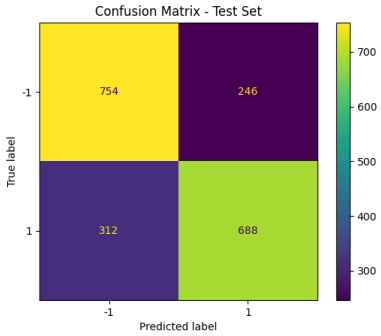
τραινινγ σετ και test set είναι 92,11% και 76,8% αντίστοιχα.



Σχήμα 1: Confusion Diagram

Σχήμα 2: 2 παραδείγματα σωστής και δυο παραδείγματα λάθος κατηγοριοποίησης

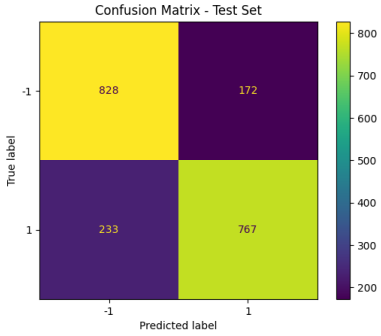
Για $C = 10$ ο χρόνος εκτέλεσης του αλγορίθμου είναι 2109,05 δευτερόλεπτα, και η ακρίβεια κατηγοριοποίησης στο τραινινγ σετ και test set είναι 93,54% και 72,1% αντίστοιχα. Τα αποτελέσματα φαίνονται στις παρακάτω εικόνες.



Σχήμα 3: Confusion Diagram

Σχήμα 4: 2 παραδείγματα σωστής και δυο παραδείγματα λάθος κατηγοριοποίησης

Για $C = 10$ ο χρόνος εκτέλεσης του αλγορίθμου είναι 1388,83 δευτερόλεπτα, και η ακρίβεια κατηγοριοποίησης στο τραινινγ σετ και test set είναι 89,50% και 79,75% αντίστοιχα. Τα αποτελέσματα φαίνονται στις παρακάτω εικόνες.



Σχήμα 5: Confusion Diagram

Σχήμα 6: 2 παραδείγματα σωστής και δυο παραδείγματα λάθος κατηγοριοποίησης

Παρατηρούμε ότι τα καλύτερα αποτελέσματα λαμβάνονται για $C = 0.10$. Στα παραδείγματα που εξετάσαμε όσο μεγαλώνει το C τόσο το train set accuracy μεγαλώνει και το test set accuracy φθίνει αυτό μας δείχνει πιθανό overfitting του μοντέλου.

Παρακάτω παρατίθεται ο κώδικας του σμ:

```
# Η δική σου κλάση SVM
class SVM:
    def __init__(self, C=1.0):
        self.C = C
        self.alpha = None
        self.w = None
        self.b = None
        self.support_vectors = None
        self.support_labels = None

    def fit(self, X, y):
        print("Starting training of the SVM model...")
        n_samples, n_features = X.shape

        # Υπολογισμός του πίνακα Gram
        print("Computing the Gram matrix...")
        K = np.dot(X, X.T)
        P = matrix(np.outer(y, y) * K)  # P = Q_kl
        q = matrix(-np.ones(n_samples))  # q_i = -1 for all i
        A = matrix(y.reshape(1, -1), (1, n_samples), tc='d')  # A = y^T
        b = matrix(0.0)  # b = 0

        # Περιορισμοί: 0 <= alpha_k <= C
        print("Setting up constraints...")
        G = matrix(np.vstack((-np.eye(n_samples), np.eye(n_samples))))
        h = matrix(np.hstack((np.zeros(n_samples), np.ones(n_samples) * self.C)))
```

Σχήμα 7: Το μοντέλο

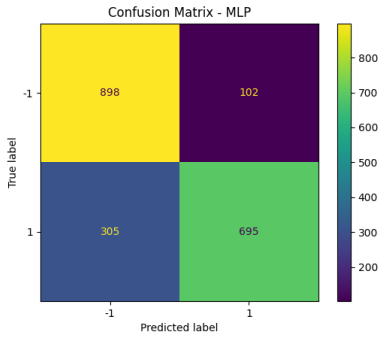
```
# Επίλυση του προβλήματος QP
print("Solving the quadratic programming problem...")
solution = solvers.qp(P, q, G, h, A, b)
self.alpha = np.ravel(solution['x'])  # Lagrange multipliers
print("QP problem solved.")

# Εντοπισμός support vectors
print("Identifying support vectors...")
support_indices = self.alpha > 1e-5
self.support_vectors = X[support_indices]
self.support_labels = y[support_indices]
self.alpha = self.alpha[support_indices]

# Υπολογισμός του διανύσματος βαρών w
print("Computing weight vector w...")
self.w = np.sum(self.alpha[:, None] * self.support_labels[:, None] * self.support_vectors, axis=0)

# Υπολογισμός του bias b
print("Computing bias b...")
K_index = np.argmax(self.alpha)  # Find the largest alpha
x_k = self.support_vectors[K_index]
y_k = self.support_labels[K_index]
epsilon_k = 1e-5
self.b = np.mean(self.support_labels - np.dot(self.support_vectors, self.w))
print("Training completed.")
```

Σχήμα 8: Επίλυση τετραγωνικού προγραμματισμού

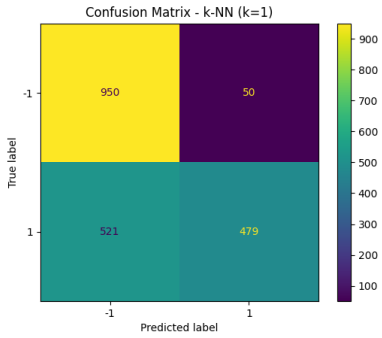
```
def predict(self, X):
    print("Predicting labels...")
    decision = np.dot(X, self.w) + self.b
    return np.sign(decision)
```

Σχήμα 9: Η συνάρτηση predict

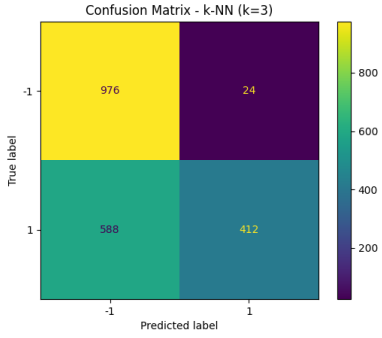
Για να συγκρίνουμε τα αποτελέσματα του svm μας χρησιμοποιήσαμε και άλλους κατηγοριοποιήτες για το ίδιο πρόβλημα κατηγοριοποίησης. Με τη βοήθεια του Chat GPT εκτελέστηκαν ένα MLP με ένα κρυφό επίπεδο 32 νευρώνων και Hinge loss που ετρέξε για 30 εποχές και πέτυχε test accuracy 79,65%, ένας κατηγοριοποιήτης πλησιέστερου γείτονα με 1 γείτονα που πέτυχε test accuracy 71,45%, ένας κατηγοριοποιήτης πλησιέστερου γείτονα με 3 γείτονες που πέτυχε test accuracy 69,4% και ένας κατηγοριοποιήτης πλησιέστερου κέντρου που πέτυχε test accuracy 72,65%. Τα αποτελέσματα φαίνονται και στα παρακάτω confusion matrices



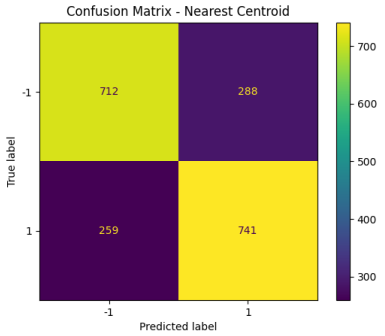
Σχήμα 10: Confusion Diagram for MLP



Σχήμα 11: Confusion Diagram for knn1



Σχήμα 12: Confusion Diagram for knn3



Σχήμα 13: Confusion Diagram for nc

Τα συνολικά αποτελέσματα για την κατηγοριοποιήση 2 κλάσεων συνοψίζονται στον παρακάτω πίνακα

Είδος μοντέλου	Ακρίβεια κατηγοριοποίσης
SVM με $C = 0.1$	79,75%
SVM με $C = 1$	76,8%
SVM με $C = 10$	72,1%
MLP	79,65%
KNN1	71,45%
KNN3	69,40%
NC	72,65%

Πίνακας I : Συνολικά Αποτελέσματα

Παρατηρούμε πως οι κατηγοριοποιήτες πλησιέστερου γείτονα και πλησιέστερου κέντρου υστερούν σε σχέση με τον MLP και το SVM

III . Κατηγοριοποιήτης ολόκληρης της CIFAR-10

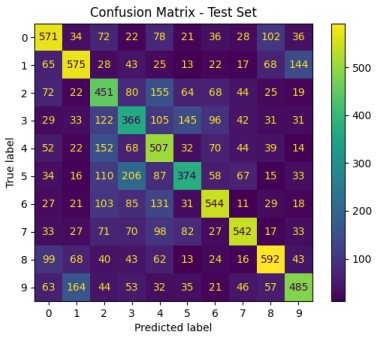
Παρόλο που ο ρόλος μια μηχανης διανυσματός υποστήριξης είναι ο διαχωρισμός του χώρου σε δυο μέρη με ένα υπερεπίπεδο τα διανύσματα υποστηριζείς μπορούν να χρησιμοποιήθουν και σε προβλήματα κατηγοριοποίησης πολλαπλών κλάσεων. Υπάρχουν διαφορετικές μέθοδοι για να γίνει κάτι τέτοιο. Στην παρούσα εργασία χρησιμοποιήθηκε η συνάρτηση SVC της sklearn η οποία υλοποιείται με "one-versus-one" approach. Η στρατηγική "one-versus-one" δημιουργεί έναν ΣΜ για κάθε ζευγάρι κλάσεων. Για K κλάσεις, δημιουργούνται $K(K-1)/2$ ταξινομητές και η τελική απόφαση λαμβάνεται με βάση την ψήφο πλειοψηφίας. Για να δω ποιοι τύποι πύρηνα και για ποιές παράμετροι δίνουν τα βέλτιστα αποτελέσματα εκτέλεσα μια σειρά απο δοκιμές. Αρχικά, εκτέλεσα τον αλγόριθμο για ένα υποσύνολου training και του test set. Συγκεκριμένα απο κάθε κλάση πήρα 2000 δείγματα εκπαίδευσης και 200 δείγματα τεστ. Χρησιμοποιήσα επίσης τοσο γραμμικούς πυρήνες όσο και πολυωνιμικούς και rbf. Οι παράμετροι που δοκίμασα καθώς και οι χρόνοι εκτέλεσης και τα ποσοστά επιτυχίας στα στάδια του training και του testing φαίνονται στον παρακάτω πίνακα:

Results Summary:

	Kernel	C	Degree/Gamma	Train Accuracy	Test Accuracy	Time (s)
0	linear	0.1	N/A	0.76565	0.345333	2020.418964
1	linear	1.0	N/A	0.93120	0.304333	4146.026132
2	linear	10.0	N/A	0.99755	0.295667	6412.865683
3	linear	100.0	N/A	1.00000	0.294333	6395.906539
4	poly	0.1	2	0.35770	0.320333	1607.017318
5	poly	0.1	3	0.42610	0.338000	1612.391472
6	poly	0.1	5	0.35265	0.224667	1704.092856
7	poly	1.0	2	0.62345	0.423000	1388.193960
8	poly	1.0	3	0.72885	0.418333	1457.117380
9	poly	1.0	5	0.62205	0.289333	1599.450350
10	poly	10.0	2	0.94660	0.446000	1374.449560
11	poly	10.0	3	0.95835	0.437000	1628.281971
12	poly	10.0	5	0.85115	0.330000	1685.151818
13	poly	100.0	2	0.99910	0.434333	1488.680047
14	poly	100.0	3	0.99770	0.421667	1725.316512
15	poly	100.0	5	0.96890	0.351000	1914.271544
16	rbf	0.1	0.1	1.00000	0.100333	2180.727407
17	rbf	0.1	0.5	1.00000	0.258000	2271.089256
18	rbf	0.1	1	1.00000	0.151000	2213.036261
19	rbf	1.0	0.1	1.00000	0.101333	2165.969958
20	rbf	1.0	0.5	1.00000	0.258667	2243.201144
21	rbf	1.0	1	1.00000	0.151333	2226.469870
22	rbf	10.0	0.1	1.00000	0.104000	2165.262261
23	rbf	10.0	0.5	1.00000	0.241333	2241.475881
24	rbf	10.0	1	1.00000	0.151333	2229.720806
25	rbf	100.0	0.1	1.00000	0.104000	2157.599687
26	rbf	100.0	0.5	1.00000	0.241333	2208.777648
27	rbf	100.0	1	1.00000	0.151333	2206.426101

Σχήμα 14: Αποτελέσματα γαι διαφορετικές παραμέτρους

Παρατηρούμε ότι για τις παραμέτρους που εξετάστηκαν οι πολυωνύμικοί πυρήνες δίνουν το κάλυτερο αποτέλεσμα. Επίσης, βλέπουμε ότι οι μηχανές που χρησιμοποιούν rbf kernel υπερεκπαιδεύονται πολυ εύκολα αφού φαίνεται ότι απλά μαθαίνουν το training set και για τις παραμέτρους που δοκιμάσαμε έχουν μικρή δυνατότητα γενίκευσης. Τα καλύτερα αποτελέσματα τα λάβαμε για πολυωνυμικό kernel δύο πυρήνων και για $C = 10$. Για αυτές τις παραμέτρους έτρεξα την μηχανή για όλα τα δεδομένα της Cifar-10 και αναφέρω παρακάτω τα αποτελέσματα. Ο αλγόριθμος χρειάστηκε 5951,32 δευτερόλεπτα για να εκτελεστεί και πέτυχε ποσοστά επιτυχίας 91,64% στο σύνολο της εκπαίδευσης και 50% στο σύνολο του τεστ.

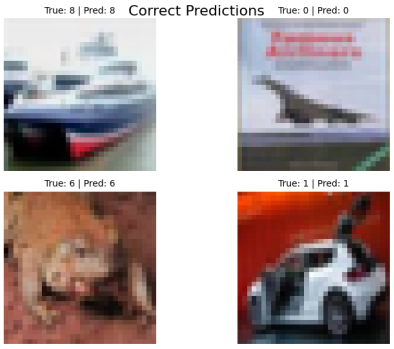


Σχήμα 15: Confusion Diagram

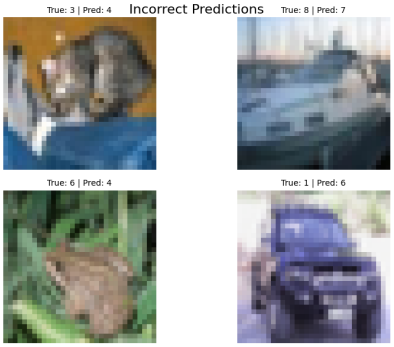
Classification Report:

	precision	recall	f1-score	support
0	0.55	0.57	0.56	1000
1	0.59	0.57	0.58	1000
2	0.38	0.45	0.41	1000
3	0.35	0.37	0.36	1000
4	0.40	0.51	0.44	1000
5	0.46	0.37	0.41	1000
6	0.56	0.54	0.55	1000
7	0.63	0.54	0.58	1000
8	0.61	0.59	0.60	1000
9	0.57	0.48	0.52	1000
accuracy			0.50	10000
macro avg	0.51	0.50	0.50	10000
weighted avg	0.51	0.50	0.50	10000

Σχήμα 16: Precision and recall



Σχήμα 17: 4 παραδείγματα σωστής κατηγοριοποίησης



Σχήμα 18: 4 παραδείγματα εσφαλμένης κατηγοριοποίησης

Έπειτα εκτέλεσα κάποιες δοκιμές για να δω πόσο επηρεάζει το αποτέλεσμα η μείωση των δεδομένων χρησιμοποιώντας PCA και κρατώντας το 90% της πληροφορίας όπως φαίνεται στο παρακάτω κομμάτι κώδικα.

```
# Εφαρμογή PCA για διατήρηση του 90% της πληροφορίας
pca = PCA(n_components=0.90)  # Διατήρηση 90% της εξηγημένης διακύμανσης

print("Applying PCA to reduce dimensions...")
x_train_flattened = x_train.reshape(x_train.shape[0], -1) / 255.0
x_test_flattened = x_test.reshape(x_test.shape[0], -1) / 255.0

x_train_pca = pca.fit_transform(x_train_flattened)
x_test_pca = pca.transform(x_test_flattened)
```

Σχήμα 19: PCA

Οι δοκιμές που εκτέλεσα ήταν για πολυωνυμικούς πυρήνες και φαίνονται στον παρακάτω πίνακα.

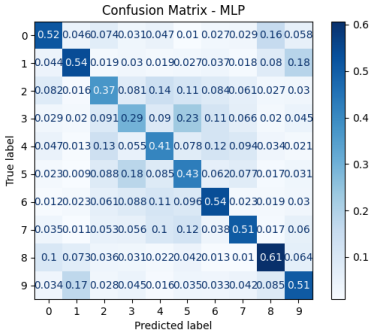
Results Summary:

	Kernel	C	Degree/Gamma	Train Accuracy	Test Accuracy
0	poly	0.1	2	0.49226	0.4191
1	poly	0.1	3	0.41310	0.2678
2	poly	0.1	5	0.29028	0.1121
3	poly	1.0	2	0.71748	0.5013
4	poly	1.0	3	0.81862	0.4474
5	poly	1.0	5	0.60826	0.1531
6	poly	10.0	2	0.89418	0.4634
7	poly	10.0	3	0.97126	0.4781
8	poly	10.0	5	0.85074	0.2349

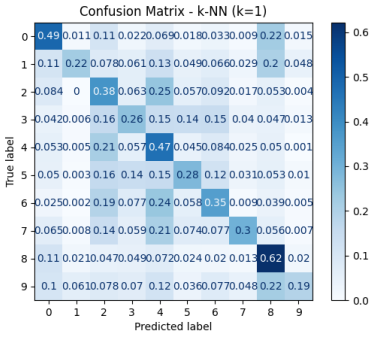
Σχήμα 20: Αποτελέσματα με PCA

Τα καλύτερα αποτελέσματα τα λαμβάνουμε για kernel 2 και 3ών πυρήνων και $C = 1$ ή $C = 10$.

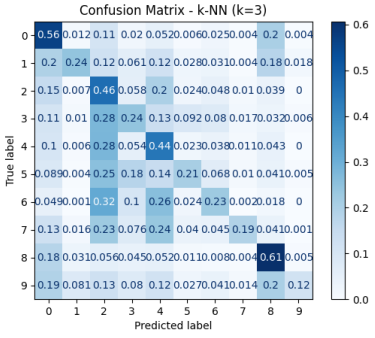
Τέλος, όπως και στον κατηγοριοποιητή των 2 κλάσεων θα παραθέσουμε τα αποτελέσματα της σύγκρισης του SVM με άλλους κατηγοριοποιήτες. Με τη βοήθεια του Chat GPT εκτελέστηκαν ένα MLP με ένα κρυφό επίπεδο 64 νευρώνων και Hinge loss που ετρέξε για 150 εποχές και πέτυχε test accuracy 47,26%, ένας κατηγοριοποιήτης πλησιέστερου γείτονα με 1 γείτονα που πέτυχε test accuracy 35,67%, ένας κατηγοριοποιήτης πλησιέστερου γείτονα με 3 γείτονες που πέτυχε test accuracy 33,08% και ένας κατηγοριοποιήτης πλησιέστερου κέντρου που πέτυχε test accuracy 28,12%. Τα αποτελέσματα φαίνονται και στα παρακάτω confusion matrices.



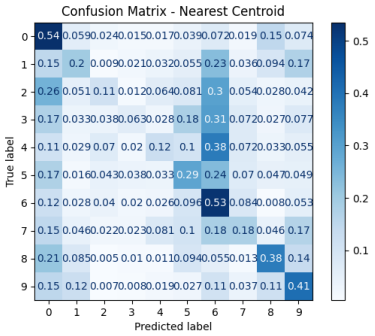
Σχήμα 21: Confusion Diagram for MLP



Σχήμα 22: Confusion Diagram for knn1



Σχήμα 23: Confusion Diagram for knn3



Σχήμα 24: Confusion Diagram for nc

Τα συνολικά αποτελέσματα για την κατηγοριοποιήση 2 κλάσεων συνοψίζονται στον παρακάτω πίνακα

Είδος μοντέλου	Ακρίβεια κατηγοριοποίσης
SVM, poly με $C = 10$	50%
MLP	47,26%
KNN1	35,67%
KNN3	33,08%
NC	28,12%

Πίνακας II : Συνολικά Αποτελέσματα

IV . Συμπεράσματα

Η μηχανές διανυσμάτων υποστήριξης αν και είναι υπολογιστικά βαρύτερες δίνουν στο σύνολο τους καλύτερα αποτελέσματα απο τους άλλους κατηγοριοποιήτες. Υπαρχεί, όμως, αυξημένος κίνδυνος υπό-εκπαίδευσης και υπερεκπαίδευσης αν δεν επιλεχθούν με προσοχή οι παράμετροι.