2η ΥΠΟΧΡΕΩΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΑΘΗΜΑ: ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ: 2024-25

Έκθεση Αποτελεσμάτων :

Σε συνέχεια της Εργασίας 1 μας ζητήθηκε να γραφεί πρόγραμμα σε οποιαδήποτε γλώσσα προγραμματισμού το οποίο να υλοποιεί ένα **Support Vector Machine** το οποίο θα εκπαιδευτεί. Επειδή έχουμε ήδη φορτώσει το dataset και έχουμε τροποποιήσει κάποια χαρακτηριστικά του τώρα θα δημιουργήσουμε και θα εκπαιδεύσουμε το SVM model χρησιμοποιώντας διάφορες παραμέτρους και συγκρίνοντας τις έτσι ώστε να έχουμε μια μέγιστη ακρίβεια και σωστά αποτελέσματα.

SVM CLASSIFIER S:

Are machine learning algorithms that are used for classification and regression purposes. SVMs are one of the powerful machine learning algorithms for classification, regression and outlier detection purposes. An SVM classifier builds a model that assigns new data points to one of the given categories. It can be viewed as a non-probabilistic binary linear classifier.

SVM TERMINOLOGY:

1.Hyperplane ένα όριο απόφασης που διαχωρίζει ένα δεδομένο σύνολο σημείων δεδομένων που έχουν διαφορετικές ετικέτες κλάσεων. Ο ταξινομητής SVM διαχωρίζει τα σημεία δεδομένων χρησιμοποιώντας ένα υπερεπίπεδο με το μέγιστο ποσό περιθωρίου. Αυτό το υπερεπίπεδο είναι γνωστό ως υπερεπίπεδο μέγιστου περιθωρίου και ο γραμμικός ταξινομητής που ορίζει είναι γνωστός ως ταξινομητής μέγιστου περιθωρίου.

2.Support Vectors: sample points close to hyperplane. Καθορίζουν την διαχωριστική γραμμή ή υπερεπίπεδο υπολογίζοντας τα περιθώρια

3.Margin: Το περιθώριο είναι ένα κενό διαχωρισμού μεταξύ των δύο γραμμών στα πλησιέστερα σημεία δεδομένων. Υπολογίζεται ως η κάθετη απόσταση από τη γραμμή προς υποστήριξη διανυσμάτων ή πλησιέστερων σημείων δεδομένων. Στα SVM, προσπαθούμε να μεγιστοποιήσουμε αυτό το χάσμα διαχωρισμού, ώστε να έχουμε μέγιστο περιθώριο.

SVM HYPERPARAMETERS:

Kernel: transforms the input data into a higher dimensional space where a linear separator can be applied(different options below)

C: balances the trade off between maximizing the margin and minimizing classification errors

Small C: results in a larger margin (high bias low variance)

Large C: tries to classify all training points correctly→overfitting

Gamma: controls how far the influence of a single training point reaches

Υπάρχουν διάφορα είδη Kernel τα οποία θα μπορούσαμε να χρησιμοποιήσουμε

Αρχικά υπάρχει το

1.Linear Kernel

2.Polynomial Kernel

3.Radial Basis Function Kernel

4.Sigmoid Kernel

Θα υλοποιήσουμε και τα 3 και θα συγκρίνουμε την ακρίβεια καθώς και τη χρονική διάρκεια που χρειάζονται. Αν τα δεδομένα μας είναι γραμμικά(σημαίνει 'ότι μπορούμε να διαχωρίσουμε τα δεδομένα με μια απλή γραμμή) και η ακρίβεια είναι παρόμοια τότε θα προτιμήσουμε να χρησιμοποιήσουμε την Linear Kernel γιατί είναι πιο γρήγορη γιατί πρεπεί απλά να κάνουμε optimize την C ενώ στα υπόλοιπα πρέπει να βρούμε και μια σωστή τιμή για την παράμετρο γ.

<u>Αρχικά έχω δοκιμάσει να υλοποιήσω SVM χρησιμοποιώντας default hyperparameters:</u>

Επιτεύχθηκε ακρίβεια 98,25% test dataset και 100% training dataset. Default hyperparameters : RBF kernel σε συνδυασμό με C=1.0 και τον συντελεστή πυρήνα (gamma='scale'), κατέγραψε αποτελεσματικά τις σχέσεις στα δεδομένα. Αυτό το αποτέλεσμα δείχνει ότι το SVM είναι κατάλληλο για αυτήν την εργασία ταξινόμησης και έπειτα θα γίνει μια σύγκριση με άλλα μοντέλα. Αν και αυτή η απόδοση είναι ικανοποιητική μπορεί να υπάρχουν βελτιώσεις μέσω διαφόρων υπερπαραμέτρων.

<u>SUCCESS RATES IN TRAINING AND TESTING WITH LINEAR AND NON LINEAR KERNEL</u>

Ο πιο κάτω πίνακας αφορά διαφορετικές υλοποιήσεις ενός SVM implementation χρησιμοποιώντας διαφορετικές υπερπαραμέτρους και διατηρώντας το gamma='scale' το οποίο είναι το default.

KERNEL	C VALUE	TRAINING ACCURACY	TESTING ACCURACY
LINEAR	1	100%	100%
LINEAR	100	100%	100%
LINEAR	1000	100%	100%
POLYNOMIAL	1	95.5%	94.75%
POLYNOMIAL	100	100%	100%
POLYNOMIAL	1000	100%	100%
SIGMOID	1	39.67%	37.75%
SIGMOID	100	32.33%	31.75%
SIGMOID	1000	32.33%	31.75%
RBF	1	97.83%	98.25%
RBF	100	100%	100%
RBF	1000	100%	100%

TRAINING TIME SVM WITH DIFFERENT KERNEL AND PARAMETER VALUES:

KERNEL	C VALUE	TRAINING TIME
LINEAR	1	0.0119
LINEAR	100	0.0133
LINEAR	1000	0.0128
POLYNOMIAL	1	0.0188
POLYNOMIAL	100	0.0281
POLYNOMIAL	1000	0.0250
SIGMOID	1	0.0467
SIGMOID	100	0.0406
SIGMOID	1000	0.0468
RBF	1	0.0236
RBF	100	0.0158
RBF	1000	0.0145

1. LINEAR KERNEL WITH PARAMETER C HAS VALUES 1,100,1000:

Επιτεύχθηκε ακρίβεια 100% test dataset και 100% training dataset ε όλες τις τιμές παραμέτρων (C=1, C=100, C=1000). Συμπεραίνουμε ότι αυτή η απόδοση είναι σταθερή και υποδεικνύει ότι το σύνολο δεδομένων είναι γραμμικά διαχωρισμό(linear separable) επιτρέποντας στο μοντέλο να ταξινομεί σωστά τα δεδομένα.

Σε αυτή τη περίπτωση η υπερπαράμετρος C δεν επηρεάζει την ακρίβεια στο test and training set υποδεικνύοντας έτσι ότι ο πυρήνας kernel είναι κατάλληλος για τα δεδομένα. Αυτό έχει σαν συμπέρασμα ότι το σύνολο δεδομένων όπως είπαμε

φαίνεται να είναι γραμμικά διαχωρίσιμο και τα δεδομένα δεν είναι τόσο ευαίσθητα σε μικρές διακυμάνσεις στο όριο απόφασης που προκαλούνται από αλλαγές στο C.

Όσο αφορά το χρόνο εκπαίδευσης παραμένει σχετικά χαμηλός και στις 3 τιμές του C δείχνοντας έτσι ότι ο linear kernel είναι αποδοτικός σε κάθε περίπτωση ανεξαρτήτως της υπερπαραμέτρου C.

Άρα ο γραμμικός πυρήνας είναι επαρκής για την βέλτιστη απόδοση και δεν χρειάζεται πρόσθετη πολυπλοκότητα.

2. POLYNOMIAL KERNEL WITH PARAMETER C HAS VALUES 1,100,1000:

C=1: Πέτυχε ακρίβεια 94,75%, υποδηλώνοντας ότι το μοντέλο αρχικά επέτρεπε μεγαλύτερο περιθώριο, το οποίο οδήγησε σε ορισμένες εσφαλμένες ταξινομήσεις.

C=100 & C=1000: Επιτεύχθηκε τέλεια ακρίβεια 100%, υποδεικνύοντας ότι η χρήση μεγαλύτερου C βοήθησε το μοντέλο να ταξινομήσει σωστά τα δεδομένα προσαρμόζοντας το όριο απόφασης πιο κοντά στα δεδομένα εκπαίδευσης.

Όσο αφορά το χρόνο εκπαίδευσης υπάρχει μια αύξηση του χρόνου όταν το C έχει τιμές 100 και 1000 σε σχέση με το C=1 υποδεικνύοντας έτσι ότι εάν προσθέσουμε μια μικρή πολυπλοκότητα στο κώδικα μας (100&1000) τότε υπάρχει και αύξηση του χρόνου.

3. SIGMOID KERNEL WITH PARAMETER C HAS VALUES 1,100,1000:

Ο πυρήνας Sigmoid δεν πέτυχε υψηλή ακρίβεια όπως τα υπόλοιπα είναι αρκετά χαμηλή στο test dataset έτσι σίγουρα δεν θα το χρησιμοποιήσουμε στα δεδομένα μας.

Όσο αφορά το χρόνο εκπαίδευσης φαίνεται ότι ο Sigmoid kernel παρουσιάζει το μεγαλύτερο χρόνο εκπαίδευσης σε σύγκριση και με τους υπόλοιπους πυρήνες. Αυτό μας δείχνει για ακόμη μια φορά ότι απαιτεί περισσότερο χρόνο και πολυπλοκότητα και δεν είναι ο κατάλληλος για το συγκεκριμένο dataset.

4. RADIAL BASIS FUNCTION(RBF) KERNEL WITH PARAMETER C HAS VALUES 1,100,1000:

C=1: Πέτυχε ακρίβεια 98.25%, υποδηλώνοντας ότι το μοντέλο διατηρεί μια ισορροπία μεταξύ ενός ομαλού ορίου απόφασης και της ελαχιστοποίησης των λαθών εκπαίδευσης

C=100 & C=1000: Επιτεύχθηκε τέλεια ακρίβεια 100%, υποδεικνύοντας ότι η χρήση μεγαλύτερου C βοήθησε το μοντέλο να ταξινομήσει σωστά τα δεδομένα προσαρμόζοντας το όριο απόφασης πιο κοντά στα δεδομένα εκπαίδευσης.

Όσο αφορά το χρόνο εκπαίδευσης φαίνεται ότι ο RBF έχει χαμηλότερο χρόνο στις υψηλότερες τιμές του C σε σύγκριση με το C=1. Συμπερασματικά λοιπόν ο πυρήνας χειρίζεται αποτελεσματικότερα τις μεγαλύτερες τιμές του C.

Τα αποτελέσματα δείχνουν ότι ο πυρήνας RBF είναι αποτελεσματικός για αυτό το σύνολο δεδομένων. Ενώ το μοντέλο είχε καλή απόδοση στο C=1, η αύξηση σε C=100 ή C=1000 είχε ως αποτέλεσμα 100% ακρίβεια ταξινόμησης.

Συμπερασματικά για τους διαφορετικούς πυρήνες του SVM:

Τα αποτελέσματα δείχνουν ότι το σύνολο δεδομένων είναι γραμμικά διαχωρισμό, καθώς ο γραμμικός πυρήνας πέτυχε με συνέπεια τέλεια ακρίβεια σε όλες τις τιμές του C. Ο πυρήνας RBF, αν και αποτελεσματικός, δεν παρείχε κανένα πρόσθετο πλεονέκτημα όσον αφορά την ακρίβεια, υποδηλώνοντας ότι ο γραμμικός πυρήνας είναι απλούστερος και υπολογιστικά πιο αποτελεσματικός για αυτό το πρόβλημα. Αυτά τα αποτελέσματα τονίζουν ότι για γραμμικά διαχωρισμένα δεδομένα, ο γραμμικός πυρήνας είναι η βέλτιστη επιλογή, ενώ ο πυρήνας RBF μπορεί να προσφέρει ευελιξία αλλά με κόστος αυξημένης πολυπλοκότητας και υπολογιστικού κόστους.

LINEAR KERNEL:

- είναι ιδανικός για σύνολα δεδομένων που είναι γραμμικά διαχωρισμένα, όπου μια απλή ευθεία γραμμή (ή υπερεπίπεδο σε υψηλότερες διαστάσεις) μπορεί να διαιρέσει τις κλάσεις.
- πέτυχε ακρίβεια 100% σε όλες τις τιμές του C, αυτό υποδηλώνει ότι το σύνολο δεδομένων δεν απαιτεί πολύπλοκα, μη γραμμικά όρια απόφασης για την επίτευξη τέλειας ταξινόμησης.
- Απαιτεί λιγότερο χρόνο εκπαίδευσης σε σύγκριση με πιο σύνθετους πυρήνες.

WHAT I WIN USING LINEAR KERNEL:

- Exploratory analysis (e.g., PCA, visualization) suggests linear separability.
- A linear model like logistic regression performs well.
- Simplicity and computational efficiency are priorities.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΟΡΘΗΣ ΚΑΙ ΕΣΦΑΛΜΕΝΗΣ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗΣ:

1.LINEAR KERNEL:

There are not incorrect examples using all C values and the DataFrame is empty

2.POLYNOMIAL KERNEL

There are not incorrect examples using C=100 AND C=1000 values and the DataFrame is empty

But in C=1.0 there are 3(we only provided 3 of them)

3.SIGMOID KERNEL:

There are in all 3 values of C incorrect examples

4.RBF kernel

There aren't any incorrect examples using C=100 & C=1000 but if C=1 there are incorrect examples I showed only 3 of them

Αρχικά δημιούργησα ξεχωριστούς κώδικες για να δοκιμάσω τη κάθε παράμετρό ξεχωριστά και ο βασικός κώδικας που δεν άλλαζε ήταν ο kernel και άλλαζα χειροκίνητα την υπερπαράμετρο C & gamma στους μη γραμμικούς πυρήνες και στον γραμμικό πυρήνα άλλαζα χειροκίνητα μόνο τη υπερπαράμετρο C και κατέγραφα τις τιμές training time,training accuracy & testing accuracy. Έπειτα ρώτησα το γλωσσικό μοντέλο chat gpt πως θα μπορούσα σε ένα κώδικα να τα συμπτύξω όλα μαζί και στην έξοδο να μου εμφανίζει όλα τα αποτελέσματα που θέλω με βάση το αρχικό κώδικα που είχα φτιάξει να το συνεχίσει να υλοποιηθεί αυτό που του έχω ζητήσει και η βάση του κώδικα ήταν η δική μου υλοποιήση.

RESULT: (το μετέφερα στο notepad για να είναι σε μορφή εικόνας)

```
import time
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Define typical gamma values to test , different kernel and c values
gamma_values = [0.001, 0.01, 0.1, 1, 10]
kernels = ['sigmoid', 'rbf', 'poly']
C_values = [1, 100, 1000]

# Loop through kernels, C values, and gamma values
for kernel in kernels:
    print(f"\nTesting Kernel: {kernel}")
    for C in C_values:
        for gamma in gamma_values:
            print(f"\nKernel: {kernel}, C: {C}, Gamma: {gamma}")

            # svc
            model = SVC(kernel=kernel, C=C, gamma=gamma, random_state=42)

            # Training time
            start_time = time.time()
            model.fit(X_train, y_train)
            training_time = time.time() - start_time

            # Make predictions
            y_pred_train = model.predict(X_train)
            y_pred_test = model.predict(X_test)

            # Calculate accuracy
            train_accuracy = accuracy_score(y_train, y_pred_train)
            test_accuracy = accuracy_score(y_test, y_pred_test)

            # Print results
            print(f"Training Time: {training_time:.4f} seconds")
            print(f"Training Accuracy: {train_accuracy:.4f}")
            print(f"Testing Accuracy: {test_accuracy:.4f}")
```

Με τη παραγωγή αυτού του κώδικα κερδίζεις χρόνο και είναι πιο αυτοματοποιημένα έτσι ώστε να διαλέξεις την πιο σωστή επιλογη για αυτό το λόγο έκανα μια μικρή επέκταση αυτού του κώδικα για να εμφανίσει μόνο τις 3 πιο καλές επιλογές και έπειτα θα τα συγκρίνω με τον linear kernel.

RESULTS:

1.LINEAR KERNEL:

C	Gamma	Training accuracy	Testing Accuracy	Training Time
1	**0.001**	**100%**	**100%**	**0.0158**
1	**0.01**	**100%**	**100%**	**0.0103**
1	**0.1**	**100%**	**100%**	**0.0097**
1	**1**	**100%**	**100%**	**0.0116**
1	**10**	**100%**	**100%**	**0.0111**
100	**0.001**	**100%**	**100%**	**0.0124**
100	**0.01**	**100%**	**100%**	**0.0123**
100	**0.1**	**100%**	**100%**	**0.0130**
100	**1**	**100%**	**100%**	**0.0112**
100	**10**	**100%**	**100%**	**0.0130**
1000	**0.001**	**100%**	**100%**	**0.0110**
1000	**0.01**	**100%**	**100%**	**0.0129**
1000	**0.1**	**100%**	**100%**	**0.0115**
1000	**1**	**100%**	**100%**	**0.0110**
1000	**10**	**100%**	**100%**	**0.0117**

Αυτά τα αποτελέσματα μας δείχνουν ότι τα δεδομένα μας είναι γραμμικά διαχωρίσιμα και ότι ο linear kernel είναι η βέλτιστη επιλογή για αυτό το dataset διότι επιτυγχάνει ακρίβεια 100% για όλες τις τιμές C & gamma . Απαιτεί ελάχιστους υπολογιστικούς πόρους όπως αποδεικνύεται και από τον χαμηλό χρόνο εκπαίδευσής του. Συνήθως η υπερπαράμετρος gamma δεν χρησιμοποιείται στο γραμμικό πυρήνα επειδή δεν είναι απαραίτητη άρα δεν χρειάζεται περαιτέρω ανάλυση.

2.SIGMOID KERNEL:

C	Gamma	Training accuracy	Testing Accuracy	Training Time
1	0.001	36.5%	35%	0.0429
1	0.01	36.5%	36.5%	0.0605
1	0.1	36.5%	36.5%	0.0331
1	1	36.5%	36.5%	0.0332
1	10	36.5%	36.5%	0.0340
100	0.001	32.17%	32%	0.0337
100	0.01	36.5%	36.5%	0.0533
100	0.1	36.5%	36.5%	0.0319
100	1	36.5%	36.5%	0.0345
100	10	36.5%	36.5%	0.0306
1000	0.001	32.17%	32%	0.0353
1000	0.01	36.5%	36.5%	0.0622
1000	0.1	36.5%	36.5%	0.0359
1000	1	36.5%	36.5%	0.0306
1000	10	36.5%	36.5%	0.0381

Sigmoid πυρήνας έχει κακή απόδοση σε όλες τις παραμέτρους και η ακρίβεια εκπαίδευσης και ελέγχου είναι πολύ χαμηλή συμπεραίνοντας έτσι ότι ο σιγμοειδής πυρήνας είναι ακατάλληλος για αυτό το σύνολο δεδομένων και δεν μπορεί να διαχωρίσει σωστά τις κλάσεις. Πιθανότατα είναι ακατάλληλος λόγω του ότι τα δεδομένα μας είναι γραμμικά διαχωρίσιμα και ο sigmoid kernel είναι πιο περίπλοκος και ενδέχεται να μην λειτουργεί καλά σε αυτές τις προϋποθέσεις.

3.RBF KERNEL:

C	Gamma	Training accuracy	Testing Accuracy	Training Time
1	0.001	97.83%	98.25%	0.0241
1	0.01	100%	100%	0.0159
1	0.1	100%	98.75%	0.0147
1	1	100%	98.25%	0.0197
1	10	100%	95.25%	0.0200
100	0.001	100%	100%	0.0156
100	0.01	100%	100%	0.0128
100	0.1	100%	98.75%	0.0147
100	1	100%	98.25%	0.0166
100	10	100%	95.25%	0.0169
1000	0.001	100%	100%	0.0118
1000	0.01	100%	100%	0.0185
1000	0.1	100%	98.75%	0.0142
1000	1	100%	98.25%	0.0164
1000	10	100%	95.25%	0.0171

Τα αποτελέσματα αυτά δείχνουν ότι τα καλύτερα αποτελέσματα είναι gamma=0.01 or gamma=1 and C=100/C=1000. Θέλουμε να αποφύγουμε υψηλές τιμές gamma για να αποφύγουμε το overfitting. Ο πυρήνας RBF επιτυγχάνει τέλεια ακρίβεια στα περισσότερα αποτελέσματα αλλά είναι πιο πολύπλοκος υπολογιστικά από τον linear kernel και δεν προσφέρει extra οφέλη για αυτό το σύνολο δεδομένων.

4.POLY KERNEL:

C	Gamma	Training accuracy	Testing Accuracy	Training Time
1	**0.001**	**97.33%**	**96.75%**	**0.0164**
1	**0.01**	**100%**	**100%**	**0.0208**
1	**0.1**	**100%**	**100%**	**0.0213**
1	**1**	**100%**	**100%**	**0.0231**
1	**10**	**100%**	**100%**	**0.0173**
100	**0.001**	**100%**	**100%**	**0.0168**
100	**0.01**	**100%**	**100%**	**0.0169**
100	**0.1**	**100%**	**100%**	**0.0166**
100	**1**	**100%**	**100%**	**0.0175**
100	**10**	**100%**	**100%**	**0.0171**
1000	**0.001**	**100%**	**100%**	**0.0171**
1000	**0.01**	**100%**	**100%**	**0.0200**
1000	**0.1**	**100%**	**100%**	**0.0190**
1000	**1**	**100%**	**100%**	**0.0256**
1000	**10**	**100%**	**100%**	**0.0192**

Για τη χρήση ενός πολυωνυμικού πυρήνα χρησιμοποιούμε gamma>=0.01 για επαρκή πολυπλοκότητα . Φαίνεται ότι οι τιμές του C δεν επηρεάζουν την απόδοση για αυτό το λόγο δεν χρειάζεται να χρησιμοποιήσουμε τις υψηλές τιμές. Επειδή τα δεδομένα μας είναι γραμμικά διαχωρίσιμα το πολυωνυμικό μοντέλο προσθέτει

μόνο υπολογιστικό κόστος και δεν προσφέρει άλλα πρόσθετα οφέλη για αυτό το σύνολο δεδομένων.

TOP-3 OVERALL:

KERNEL	C	GAMMA	Training Accuracy	Testing Accuracy	Training Time
linear	1	0.001	100%	100%	0.0157
linear	1	0.01	100%	100%	0.0103
linear	1	0.1	100%	100%	0.0097

Hinge loss: BINARY CLASSSIFICATION

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

Είναι μια συνάρτηση απώλειας που είναι χρήσιμη κυρίως για δυαδική ταξινόμηση. Χρησιμοποιείται για να μεγιστοποιεί το margin μεταξύ των κλάσεων καθιστώντας το ιδιαίτερα αποτελεσματικό στα SVM. Η βασική του ιδέα είναι να τιμωρεί το μοντέλο όταν ταξινομεί εσφαλμένα ένα δείγμα το οποίο είναι κοντά στο όριο απόφασης.

MULTI-CLASS CLASSIFICATION:

$$\text{Hinge Loss} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j\neq y_i}\max(0, 1 - \text{score}_{y_i} + \text{score}_j)$$

Το μοντέλο εισάγει μια βαθμολογία σε κάθε τάξη για μια δεδομένη είσοδο. Με την χρήση του hinge loss βοηθά τη βαθμολογία της σωστά ταξινομημένης τάξης να έχει υψηλότερη βαθμολογία από την βαθμολογία των εσφαλμένων κλάσεων κατά ένα τουλάχιστον margin. Εάν όμως η βαθμολογία της σωστής τάξης δεν είναι μεγαλύτερη από τη λανθασμένη τάξη τότε επιβάλλεται ποινή 1−scoreyi+ scorej . Το μοντέλο μαθαίνει να ελαχιστοποιεί αυτή την απώλεια σε όλα τα παραδείγματα εκπαίδευσης διασφαλίζοντας ότι η σωστή τάξη έχει σταθερά την υψηλότερη βαθμολογία.

Άρα θα υλοποιήσουμε ένα mlp with one hidden layer and hinge loss optimization η μόνη διαφορά από τις προηγούμενες υλοποιήσεις είναι το ένα κρυφό επίπεδο και στη θέση του activation=relu or softmax χρησιμοποιούμε identity

Identity διασφαλίζει ότι τα raw scores προωθούνται στο hinge loss

COMPARISON KNN-1,KNN-3,MLP,SVM:

CLASSIFIER	TRAINING ACCURACY	TESTING ACCURACY	TRAINING TIME(s)
KNN-1	**100%**	**99.5%**	**0.0026**
KNN-3	**100%**	**99.5%**	**0.0027**
NEAREST CENTROID	**72%**	**73.5%**	**0.0080**
MLP	**100%**	**100%**	**3.7583**
SVM	**100%**	**100%**	**0.0103**

TRAINING & TESTING ACCURACY:

SVM&MLP: πετυχαίνουν άριστη ακρίβεια στα δεδομένα ελέγχου και εκπαίδευσης το οποίο μας δείχνει την ικανότητα τους για αυτό το σύνολο δεδομένων να μοντελοποιήσουν τις σχέσεις αυτές

KNN-1 & KNN-3: πετυχαίνουν σχεδόν άριστη ακρίβεια(99.5%) στα δεδομένα ελέγχου και άριστη ακρίβεια στα δεδομένα εκπαίδευσης.

Nearest Centroid: αυτός είναι ο χειρότερος από όλους τους classifiers έχει χαμηλή ακρίβεια τόσο στα δεδομένα ελέγχου όσο και στα δεδομένα εκπαίδευσης διότι ίσως κάνει τόσο απλά τα decision boundaries χρησιμοποιώντας της κεντροειδής κλάσης έτσι χάνεται η ακρίβεια των δεδομένων.

TRAINING TIME:

KNN-1 & KNN-3: έχουν το μικρότερο χρόνο εκπαίδευσης διότι δεν κάνουν πραγματική εκπαίδευση και κάνουν απλά αποθήκευση των δεδομένων.

Nearest Centroid: είναι πιο γρήγορο από το MLP και το SVM αλλά πιο αργό από το KNN αφού υπολογίζει τα κεντροειδής για κάθε κλάση**.**

SVM: έχει καλύτερο χρόνο από το MLP και μας δείχνει ξανά ότι η καλή του απόδοση οφείλεται στο ότι το σύνολο δεδομένων είναι γραμμικά διαχωρίσιμο.

MLP: έχει το χειρότερο χρόνο εκπαίδευσης διότι έχει πολλές επαναλήψεις έτσι ώστε να πάρει τη βέλτιστη του απόδοση.

OVERALL:

Ο καλύτερος classifier είναι ο SVM λόγω της τέλειας του ακρίβειας και του καλού χρόνου εκπαίδευσης. KNN-1 & KNN-3 είναι επίσης καλοί αλλά είναι ακριβοί για να υλοποιηθούν υπολογιστικά έτσι δεν τους προτιμάμε για να τους χρησιμοποιούμε .