ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

Ονοματεπώνυμο: Βασιλική Παντελοπούλου  
Email: vaspntl@gmail.com

Περιεχόμενα

[Γενική Περιγραφή 2](#_Toc190283904)

[Περιγραφή Σημασίας/Τιμών Χαρακτηριστικών 3](#_Toc190283905)

[Ανάλυση και Επεξεργασία Δεδομένων 3](#_Toc190283906)

[Μέθοδοι Κωδικοποίησης 4](#_Toc190283907)

[Κλιμάκωση 5](#_Toc190283908)

[Επιλογή Χαρακτηριστικών 6](#_Toc190283909)

[Διαχωρισμός Train και Test 6](#_Toc190283910)

[Εκπαίδευση και Αξιολόγηση Κατηγοριοποιητών 7](#_Toc190283911)

[Κ-Πλησιέστεροι Γείτονες/ Πλησιέστερο Κέντρο 7](#_Toc190283912)

[Συμπεράσματα 9](#_Toc190283913)

[Καμπύλες Εκμάθησης 10](#_Toc190283914)

[Δημιουργία και Εκπαίδευση Νευρωνικού Δικτύου 11](#_Toc190283915)

[Αρχιτεκτονική με δύο κρυφά επίπεδα 12](#_Toc190283916)

[Αρχιτεκτονική με 10 κρυφά επίπεδα 20](#_Toc190283917)

[Παρατηρήσεις 23](#_Toc190283918)

[Σύγκριση δύο Αρχιτεκτονικών 24](#_Toc190283919)

[Παραδείγματα Ορθής και Εσφαλμένης Κατηγοριοποίησης 24](#_Toc190283920)

[Αντιστοιχίες κλάσεων 25](#_Toc190283921)

[Σύγκριση με ΚΝΝ και NC 25](#_Toc190283922)

[Δημιουργία και Εκπαίδευση SVM 25](#_Toc190283923)

[SVC μοντέλο 25](#_Toc190283924)

[LinearSVC μοντέλο 34](#_Toc190283925)

[Συμπεράσματα 35](#_Toc190283926)

[Παραδείγματα Ορθής και Εσφαλμένης Κατηγοριοποίησης 36](#_Toc190283927)

[Σύγκριση με ΚΝΝ και NC 36](#_Toc190283928)

[Σύγκριση με MLP 36](#_Toc190283929)

[Δημιουργία RBF Νευρωνικού Δικτύου 37](#_Toc190283930)

[Συμπεράσματα 42](#_Toc190283931)

[Παραδείγματα Ορθής και Εσφαλμένης Κατηγοριοποίησης 44](#_Toc190283932)

[Σύγκριση με ΚΝΝ και NC 44](#_Toc190283933)

[Συνολικά Αποτελέσματα 45](#_Toc190283934)

[Αναφορές 47](#_Toc190283935)

# Γενική Περιγραφή

Η βάση δεδομένων που χρησιμοποιήθηκε, προέρχεται από το Kaggle και περιλαμβάνει δεδομένα για την εκτίμηση των επιπέδων παχυσαρκίας σε άτομα από το Μεξικό, το Περού και την Κολομβία, με βάση τις διατροφικές τους συνήθειες και τη φυσική τους κατάσταση. Τα δεδομένα περιέχουν αρχικά **17 χαρακτηριστικά (διαστάσεις)** και **2.111 εγγραφές**, ενώ η κύρια μεταβλητή ταξινόμησης είναι η "NObesity" (Επίπεδο Παχυσαρκίας), η οποία διακρίνει τα άτομα σε **επτά κλάσεις** (Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III). Περιλαμβάνει τα εξής χαρακτηριστικά (εκτός target) : Gender, Age, Height, Weight, family\_history\_with\_overweight, FAVC, FCVC, NCP, CAEC, SMOKE, CH2O, SCC, FAF, TUE, CALC, MTRANS. Η βάση βρίσκετε στον παρακάτω σύνδεσμο αλλά επισυνάπτεται και στο παραδοτέο εφόσον είναι 267kB.

<https://www.kaggle.com/datasets/fatemehmehrparvar/obesity-levels>

Για την εκπόνηση της εργασίας χρησιμοποιήθηκε Python σε περιβάλλον **Google Colab**. Στο πλαίσιο της υλοποίησης, εκπαίδευσης και αξιολόγησης των αλγορίθμων ταξινόμησης, χρησιμοποιήθηκαν οι κατηγοριοποιητές K-πλησιέστερων γειτόνων (**KNN**) με 1 και 3 γείτονες, καθώς και ο κατηγοριοποιητής πλησιέστερου κέντρου (**NC**). Η υλοποίηση έγινε με τα εργαλεία που παρέχει η βιβλιοθήκη **sklearn**.

Για την εκκαθάριση, την επεξεργασία και τον διαχωρισμό των δεδομένων σε test και train set χρησιμοποιούνται οι εξής βιβλιοθήκες:

* sklearn.model\_selection.train\_test\_split
* sklearn.preprocessing.LabelEncoder
* sklearn.preprocessing.StandardScaler
* sklearn.preprocessing.MinMaxScaler
* numpy
* pandas

## Περιγραφή Σημασίας/Τιμών Χαρακτηριστικών

**Gender**: Male/ Female  
**Age**: Συνεχείς τιμές  
**Height**: Συνεχείς τιμές  
**Weight**: Συνεχείς τιμές  
**family\_history\_with\_overweight**: Yes/No

**FAVC** (Καταναλώνετε συχνά τροφές με υψηλή θερμιδική αξία;): Yes/No

**FCVC** (Συνηθίζετε να τρώτε λαχανικά στα γεύματά σας;): Yes/No

**NCP** (Πόσα κύρια γεύματα καταναλώνετε καθημερινά;) : Συνεχείς τιμές από 1 έως 4

**CAEC** (Καταναλώνετε κάποιο σνακ μεταξύ των γευμάτων;): No/Sometimes/Frequently/Always

**SMOKE**: Yes/No  
**CH2O** (Πόσο νερό πίνετε καθημερινά;): Συνεχείς τιμές από 1 έως 3

**SCC** (Παρακολουθείτε τις θερμίδες που καταναλώνετε καθημερινά;): Yes/No  
**FAF** (Πόσο συχνά γυμνάζεστε;): Συνεχείς τιμές από 0 έως 3  
**TUE** (Πόσο χρόνο χρησιμοποιείτε συσκευές τεχνολογίας όπως κινητό, βιντεοπαιχνίδια, τηλεόραση, υπολογιστή κ.λπ.;): Συνεχείς τιμές από 0 έως 2

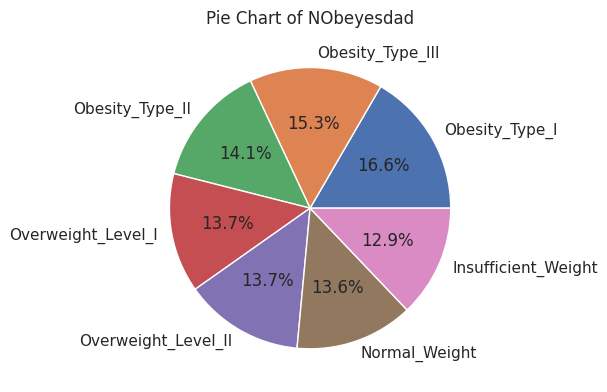
**CALC** (Πόσο συχνά καταναλώνετε αλκοόλ;): No/Sometimes/Frequently/Always  
**MTRANS** (Ποιο μέσο μεταφοράς χρησιμοποιείτε συνήθως;):

Public\_Transportation/Automobile/Walking/Motorbike/Bike

# Ανάλυση και Επεξεργασία Δεδομένων

Αρχικά, εξετάστηκε αν στο dataset υπάρχουν κελιά με NULL τιμή ώστε να χρησιμοποιηθούν μόνο τα δείγματα που έχουν συμπληρωμένες όλες τις τιμές των χαρακτηριστικών. Δεν εντοπίστηκαν κενά, επομένως **διατηρήθηκαν όλα τα δείγματα για ανάλυση**.

Έπειτα, αναλύθηκαν τα ποσοστά των χαρακτηριστικών του dataset, συνοδευόμενα από τα αντίστοιχα γραφήματα. Διαπιστώθηκε ότι οι διάφορες κλάσεις κατηγοριοποίησης διαθέτουν επαρκή δείγματα, γεγονός που υποδηλώνει την καταλληλόλητα του dataset για ανάλυση. Η παρουσία **ικανοποιητικού αριθμού παραδειγμάτων από κάθε κλάση** ενισχύει την **αξιοπιστία** των μοντέλων που θα εκπαιδευτούν και επιτρέπει την **καλύτερη** **γενίκευση** των αποτελεσμάτων.



Στην συνέχεια, παρατηρήθηκε ότι ο **τύπος των χαρακτηριστικών δεν είναι ο ίδιος** εφόσον στο dataset υπάρχουν τόσο αριθμητικά χαρακτηριστικά όσο και μη αριθμητικά. Συνεπώς, εντοπίστηκαν τα **μη αριθμητικά χαρακτηριστικά** ώστε να **μετατραπούν** σε **αριθμητικά** προκειμένου να υπάρχει συμβατότητα στους αλγορίθμους που θα χρησιμοποιηθούν βελτιώνοντας ταυτόχρονα και την απόδοσή τους. Συγκεκριμένα, τα μη αριθμητικά χαρακτηριστικά είναι : ['Gender', 'CALC', 'FAVC', 'SCC', 'SMOKE', 'family\_history\_with\_overweight', 'CAEC', 'MTRANS', 'NObeyesdad']. Αυτά τα χαρακτηριστικά, κωδικοποιήθηκαν σε αριθμητική μορφή με δύο μεθόδους για σύγκριση αποτελεσμάτων.

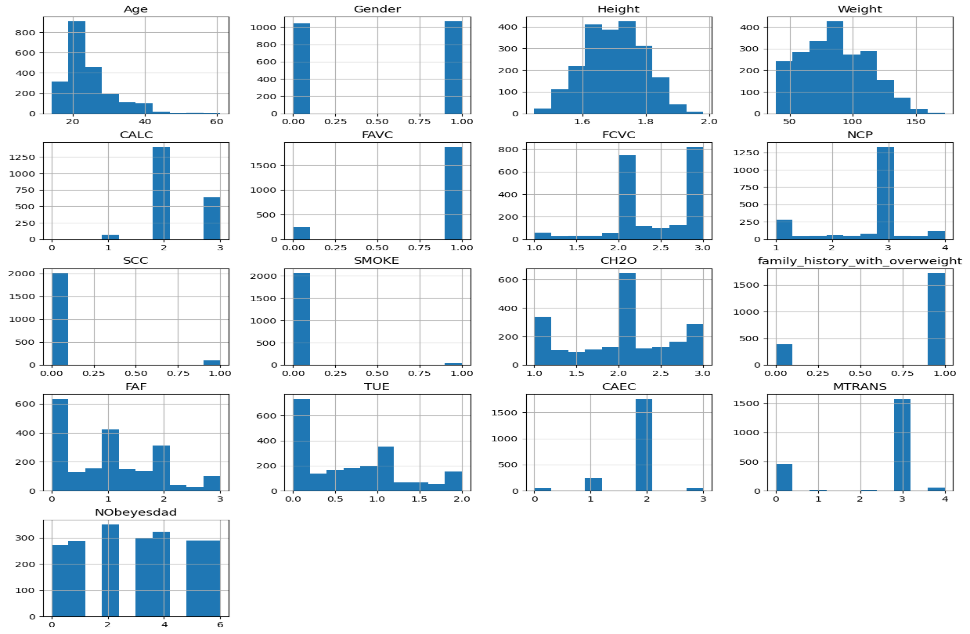
### Μέθοδοι Κωδικοποίησης

1. Χρησιμοποιήθηκε **LabelEncoder** για τη μετατροπή κατηγορικών χαρακτηριστικών σε αριθμητικά με μοναδικούς ακεραίους. Δεν δημιουργεί επιπλέον διαστάσεις (στήλες) αλλά εισάγει μια ψευδή αλληλουχία μεταξύ των κατηγοριών, γεγονός που ενδέχεται να παραπλανήσει τα μοντέλα. Για παράδειγμα, για το χαρακτηριστικό Gender οι τιμές τους μετατράπηκαν ως εξής : "Male" σε 0 και "Female" σε 1. Για το χαρακτηριστικό MTRANS το οποίο δεν έχει καθορισμένη σειρά, εφόσον αφορά μέσα μετακίνησης, χρησιμοποιώντας αυτή τη μέθοδο θα προκύψουν οι τιμές 0 έως 4 για το κάθε τρόπο μετακίνησης. Το αντίστοιχο έγινε και για τη κλάση στόχο ώστε να διευκολυνθεί η διαδικασία κατηγοριοποίησης . **Τελικά έχουμε 17 διαστάσεις**. Για λόγους ευκολίας θα ονομάζομαι αυτή τη μέθοδο **simple encoding**.
2. Χρησιμοποιήθηκε simple encoding για τα κατηγορικά χαρακτηριστικά εκτός από το χαρακτηριστικό **MTRANS** για το οποίο εφαρμόστηκε η μέθοδος **One-Hot Encoding**. Συγκεκριμένα, δημιουργείται μια νέα διάσταση(στήλη) για κάθε μοναδική τιμή του χαρακτηριστικού, αντιπροσωπεύοντας την παρουσία ή την απουσία κάθε μεταβλητής ως 0 ή 1. Με αυτό τον τρόπο, αποφεύγεται η εισαγωγή ψευδών σχέσεων μεταξύ των κατηγοριών. **Τελικά έχουμε 21 διαστάσεις.**

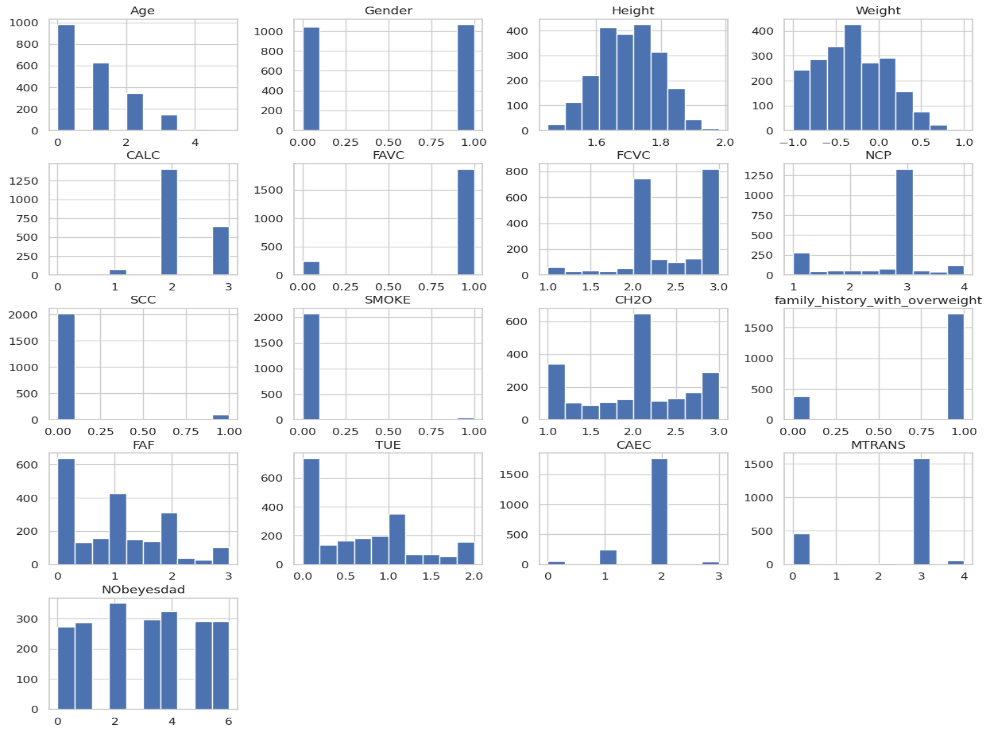
## Κλιμάκωση

Για να πετύχουν οι κατηγοριοποιητές καλύτερες αποδόσεις έγινε scale των χαρακτηριστικών **Weight** και **Age** χρησιμοποιώντας **MinMaxScaler** με εύρος από -1 έως 1 για το βάρος και 0 έως 5 με βάση προκαθορισμένα όρια για την ηλικία. Αυτό γίνεται γιατί υψηλές τιμές μπορούν να επηρεάσουν το μοντέλο οδηγώντας σε λανθασμένα αποτελέσματα. Τώρα όλα τα χαρακτηριστικά συμβάλλουν με την ίδια βαρύτητα στην εκπαίδευση.

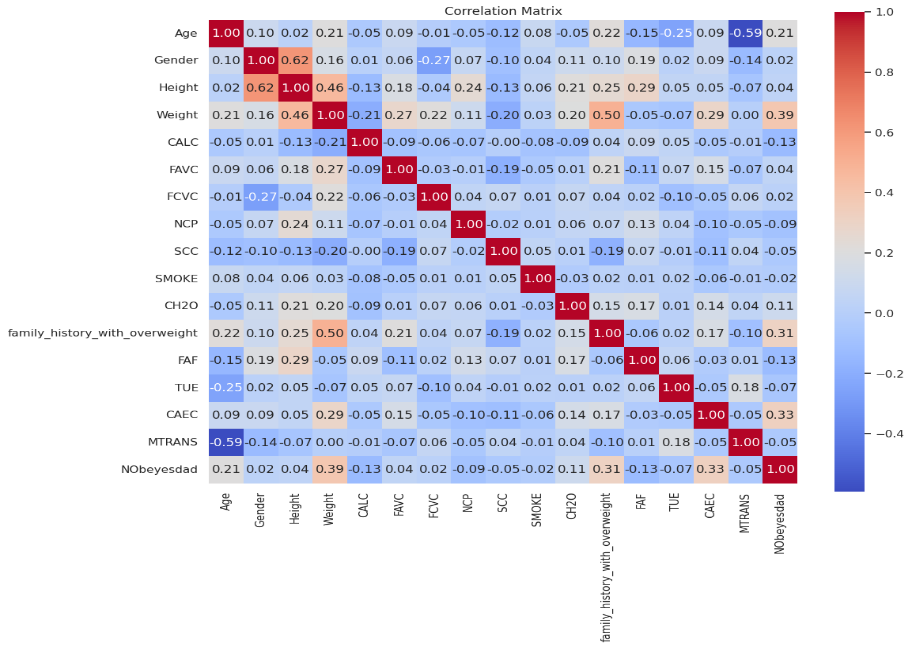
* Εύρος στήλης πριν την κλιμάκωση :



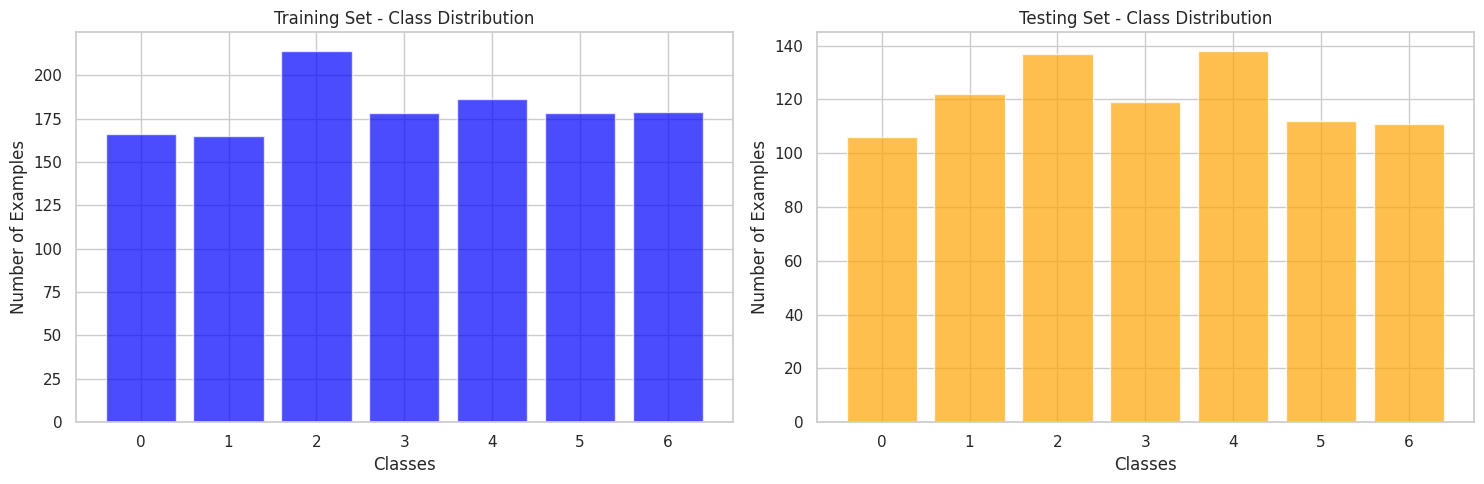
* Εύρος στήλης μετά την κλιμάκωση :



## Επιλογή Χαρακτηριστικών

Επιπλέον εξετάστηκε αν μπορεί να γίνει μείωση των διαστάσεων αφαιρώντας τα χαρακτηριστικά που δεν είναι σχετικά με τη μεταβλητή στόχο. Για να πραγματοποιηθεί αυτό χρησιμοποιήθηκε ο **πίνακας συσχέτισης** (correlation matrix) που αναπαριστά το πόσο καλά σχετίζονται τα χαρακτηριστικά μεταξύ τους. Επομένως, τα χαρακτηριστικά που φαίνονται να είναι πιο χρήσιμα για τη πρόβλεψη είναι τα εξής : Age, Weight, NCP, CALC, CH2O, family\_history\_with\_overweight, FAF, TUE, CAEC, NObeyesdad. Για λόγους σύγκρισης εξετάστηκε τόσο το αποτέλεσμα των μοντέλων με χρήση και των 17 διαστάσεων όσο και με τη χρήση λιγότερων διαστάσεων δηλαδή **10**. Παρακάτω φαίνεται ο πίνακας συσχέτισης :

## Διαχωρισμός Train και Test

Για να διαχωρίσω τα δείγματα σε train και test sets, απομονώνω τη στήλη NObeyesdad, διασφαλίζοντας ότι θα περιλαμβάνεται μόνο στο train set. Χρησιμοποιώ τη συνάρτηση train\_test\_split από τη βιβλιοθήκη sklearn για να χωρίσω τα δεδομένα, δίνοντας το **60% των δειγμάτων στο train set (1266 δείγματα)** και το **40% στο test set (845 δείγματα)**. Αυτή η συνάρτηση εξασφαλίζει ομοιόμορφη κατανομή των δειγμάτων από κάθε κλάση στα train και test sets, επιτρέποντας έτσι αξιόπιστη εκτίμηση της απόδοσης του μοντέλου. Ελέγχω την κατανομή των κλάσεων στο train και test set με χρήση του collections.Counter. Η κατανομή φαίνεται παρακάτω :

# Εκπαίδευση και Αξιολόγηση Κατηγοριοποιητών

Δεδομένου ότι η κατανομή των παραδειγμάτων στις κλάσεις είναι **ισορροπημένη**, δηλαδή υπάρχουν επαρκή και συγκρίσιμα δεδομένα για κάθε κλάση, η **ακρίβεια** των μοντέλων θεωρείται **αξιόπιστη**. Αυτό συμβαίνει επειδή, σε μια τέτοια περίπτωση, το μοντέλο δεν έχει την τάση να «προτιμά» μία συγκεκριμένη κλάση λόγω υπεραφθονίας παραδειγμάτων σε αυτή. Αντιθέτως, μαθαίνει να διακρίνει σωστά μεταξύ των διαφορετικών κλάσεων, και έτσι το ποσοστό ακρίβειας που προκύπτει δεν είναι παραπλανητικό, καθώς το μοντέλο δεν βασίζεται μόνο στην πολυπληθέστερη κλάση για να επιτύχει υψηλή ακρίβεια. Παρόλαυτα, **εξετάζονται τόσο το precision όσο και το recall όπως και το f1-score για την αξιολόγηση των μοντέλων**.

## Κ-Πλησιέστεροι Γείτονες/ Πλησιέστερο Κέντρο

Για την εκπαίδευση των κατηγοριοποιητών έγιναν οι εξής δοκιμές με βάση την κατάσταση των δεδομένων :

* **16 διαστάσεις χωρίς κανονικοποίηση δεδομένων**

Κρατώντας όλα τα χαρακτηριστικά του dataset χωρίς κανονικοποίηση των Age και Weight τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | **0.8887** | **0.8639** | 0.4792 |

* **16 διαστάσεις με κανονικοποίηση δεδομένων**

Κρατώντας όλα τα χαρακτηριστικά του dataset με κανονικοποίηση των Age και Weight τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | 0.8071 | 0.7893 | 0.4627 |

* **20 διαστάσεις one-hot encoding χωρίς κανονικοποίηση δεδομένων**

Κρατώντας όλα τα χαρακτηριστικά του dataset χωρίς κανονικοποίηση των Age και Weight αλλά προσθέτοντας επιπλέον διαστάσεις με χρήση της one-hot encoding για το MTRANS τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | **0.8899** | **0.8674** | 0.4792 |

* **20 διαστάσεις one-hot encoding με κανονικοποίηση δεδομένων**

Κρατώντας όλα τα χαρακτηριστικά του dataset με κανονικοποίηση των Age και Weight αλλά προσθέτοντας επιπλέον διαστάσεις με χρήση της one-hot encoding για το MTRANS τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | 0.8130 | 0.7964 | **0.5443** |

* **9 διαστάσεις χωρίς κανονικοποίηση δεδομένων**

Κρατώντας τα χαρακτηριστικά του dataset που κρίθηκαν πιο σημαντικά από το πίνακα συσχέτισης χωρίς κανονικοποίηση των Age και Weight τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | 0.8591 | 0.8449 | 0.4816 |

* **9 διαστάσεις με κανονικοποίηση δεδομένων:**

Κρατώντας τα χαρακτηριστικά του dataset που κρίθηκαν πιο σημαντικά από το πίνακα συσχέτισης με κανονικοποίηση των Age και Weight τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | 0.7467 | 0.7479 | 0.4816 |

Πραγματοποιήθηκε επίσης ανάλυση **PCA** για την αξιολόγηση των μοντέλων, με στόχο να εντοπιστούν τα χαρακτηριστικά του dataset που συνεισφέρουν περισσότερο στη διασπορά των δεδομένων και να εξεταστεί η ακρίβεια του μοντέλου μόνο με αυτά. Όπως αναμενόταν, βάσει και του πίνακα συσχέτισης, τα δύο χαρακτηριστικά που ξεχωρίζουν είναι η ηλικία (Age) και το βάρος (Weight). Αυτό επιβεβαιώνεται από τους συντελεστές φορτίου της PCA. Συγκεκριμένα, το χαρακτηριστικό **Weight** παρουσιάζει ισχυρή θετική συνεισφορά στην πρώτη κύρια συνιστώσα (PC1) με τιμή **0.4635**, ενώ το χαρακτηριστικό **Age** συνεισφέρει θετικά στη δεύτερη κύρια συνιστώσα (PC2) με τιμή **0.5615**.

PC1 PC2

Age 0.198183 0.561537

Gender 0.326641 -0.211850

Height 0.449886 -0.304217

Weight 0.463551 0.008473

CALC -0.142483 0.028360

FAVC 0.260167 0.074571

FCVC -0.040565 0.016771

NCP 0.172806 -0.179578

SCC -0.219595 -0.082026

SMOKE 0.042085 0.059645

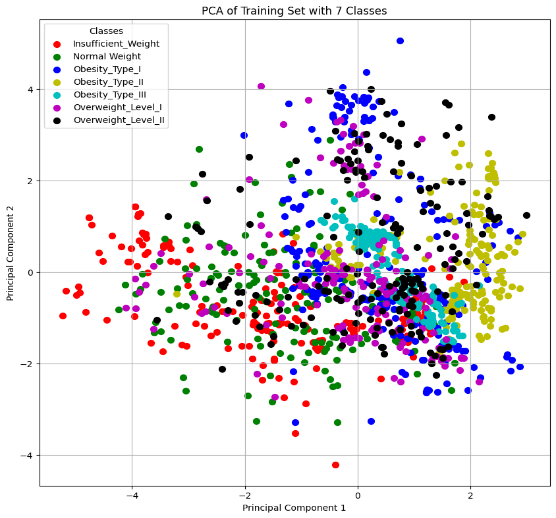
CH2O 0.191930 -0.202815

family\_history\_with\_overweight 0.379399 0.100470

FAF 0.057472 -0.356864

TUE -0.046922 -0.332220

CAEC 0.232709 0.080301

MTRANS -0.163781 -0.450109

Παρατηρείται σημαντική επικάλυψη μεταξύ των κλάσεων όπως τα Normal Weight, Overweight Level I, και Overweight Level II. Αυτό υποδηλώνει ότι οι **δύο κύριες συνιστώσες δεν παρέχουν αρκετή πληροφορία για να διαχωρίσουν αποτελεσματικά αυτές τις κλάσεις** εφόσον εμφανίζεται υψηλή πολυπλοκότητα δεδομένων.

* **Χρήση PCA για μείωση διαστάσεων από 16 σε 2**

Κρατώντας μόνο τα χαρακτηριστικά Age και Weight τα αποτελέσματα είναι τα εξής:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **KNN, k=1** | **KNN, k=3** | **NC** |
| **Accuracy** | 0.7763 | 0.7810 | 0.4792 |

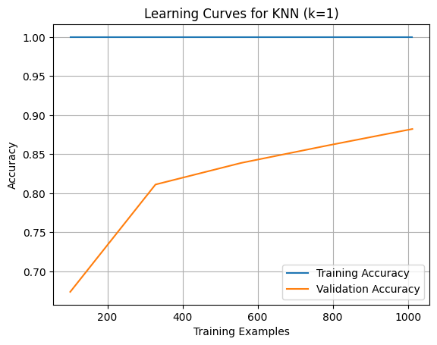
### Συμπεράσματα

Η **κανονικοποίηση** των χαρακτηριστικών **Age** και **Weight** **δεν βελτιώνει την ακρίβεια** των μοντέλων **KNN** και, αντίθετα, οδηγεί σε μείωση της απόδοσης. Αυτά τα χαρακτηριστικά είναι κρίσιμα για την κατηγοριοποίηση, όπως προκύπτει από τον πίνακα συσχέτισης, και η απουσία κανονικοποίησης διατηρεί την πληροφορία που περιέχουν.

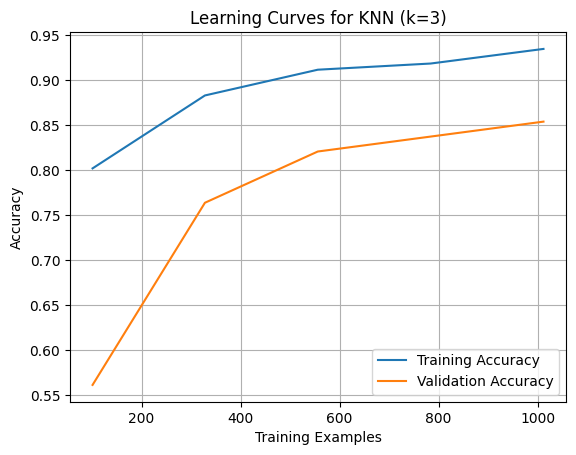
* Το **KNN (k=1)** και το **KNN (k=3)** επιτυγχάνουν υψηλή ακρίβεια σε 16 διαστάσεις. με την κανονικοποίηση να μειώνει αισθητά την απόδοση. Αυτό δείχνει ότι η απουσία κανονικοποίησης επιτρέπει στο μοντέλο να αξιοποιήσει πλήρως τη σημασία των χαρακτηριστικών βάρος και ηλικία. Η **προσθήκη one-hot encoding στο MTRANS**, βελτιώνει ελαφρώς την ακρίβεια του KNN, με το **KNN (k=1)** να φτάνει την υψηλότερη τιμή **0.8899**. Αυτό δείχνει ότι ο τρόπος μετακίνησης (MTRANS) περιέχει σημαντική πληροφορία για την κατηγοριοποίηση. Ταυτόχρονα, η επιλογή μόνο των πιο σημαντικών χαρακτηριστικών (**9 διαστάσεις χωρίς κανονικοποίηση)** διατηρεί υψηλή απόδοση στο KNN, αλλά υπάρχει μια μικρή απώλεια πληροφοριών.
* Η **κανονικοποίηση των χαρακτηριστικών** βοηθά το **Nearest Centroid** να λειτουργεί καλύτερα, καθώς διασφαλίζει ότι όλα τα χαρακτηριστικά συνεισφέρουν ισότιμα στον υπολογισμό των αποστάσεων από τους κεντροειδείς των κλάσεων. Χωρίς κανονικοποίηση, χαρακτηριστικά με μεγαλύτερη κλίμακα επηρεάζουν υπερβολικά την απόφαση του μοντέλου, οδηγώντας σε χαμηλότερη ακρίβεια. Η υψηλότερη ακρίβεια του μοντέλου **0.5443** επιτυγχάνεται με τη προσθήκη διαστάσεων μέσω one-hot encoding της MTRANS.
* Αναφορικά με τη **PCA μείωση σε 2 διαστάσεις,** αν και η ακρίβεια μειώνεται, η PCA επιβεβαιώνει ότι τα χαρακτηριστικά **Age** και **Weight** είναι καθοριστικά για την κατηγοριοποίηση, διατηρώντας ικανοποιητική ακρίβεια 0.7810.

Η καλύτερη στρατηγική είναι η χρήση όλων των χαρακτηριστικών **χωρίς κανονικοποίηση** για **KNN** και με προσθήκη **one-hot encoding στο MTRANS**, καθώς αυτό οδηγεί στη βέλτιστη απόδοση. Παρακάτω αναλύονται οι καμπύλες εκμάθησης για τη συγκεκριμένη προεπεξεργασία.

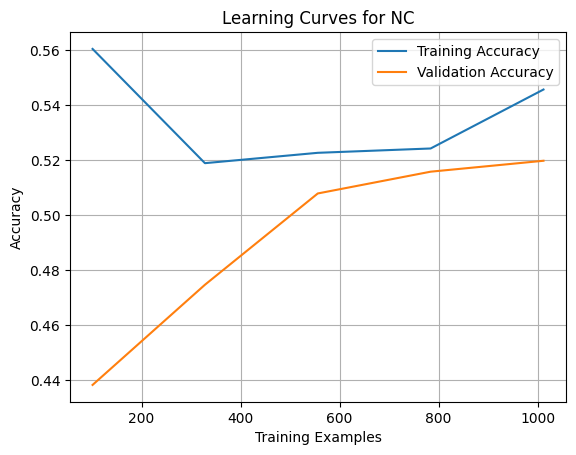
### Καμπύλες Εκμάθησης

**Η καμπύλη εκμάθησης για το KNN με k=1 είναι η εξής** :

Το training accuracy παραμένει σταθερά στο 100% υποδηλώνοντας ότι το μοντέλο ταιριάζει τέλεια σε κάθε δείγμα του συνόλου εκπαίδευσης. Αυτό είναι αναμενόμενο για το KNN k=1 όπου κάθε δείγμα είναι ο πλησιέστερος γείτονας του ίδιου. Ωστόσο, συχνά αυτό σημαίνει ότι μπορεί να μην γενικεύσει καλά σε άγνωστα δείγματα. Το validation accuracy αυξάνεται σταθερά με τη προσθήκη περισσότερων δειγμάτων όμως δεν προσεγγίζει τη καμπύλη εκπαίδευσης υποδηλώνοντας **υπερπροσαρμογή**.

**Η καμπύλη εκμάθησης για το KNN με k=3 είναι η εξής** :

Το training accuracy ξεκινάει από 0.85 και ανεβαίνει σταδιακά καθώς αυξάνεται το πλήθος των δειγμάτων, σταθεροποιούμενο περίπου στο 0.90. Αυτό υποδηλώνει ότι το μοντέλο μαθαίνει καλύτερα καθώς προστίθενται περισσότερα δείγματα κα προσαρμόζεται στις τάσεις του συνόλου χωρίς έντονο φαινόμενο υπερπροσαρμογής. Το validation accuracy ξεκινά χαμηλά και βελτιώνεται σημαντικά καθώς αυξάνονται τα δείγματα, προσεγγίζοντας τη καμπύλη εκπαίδευσης. Αυτή η συνεχής βελτίωση και η προσέγγιση του training accuracy υποδηλώνει ότι το μοντέλο **γενικεύει ικανοποιητικά**.

**Η καμπύλη εκμάθησης για το NC είναι η εξής**:

Το validation accuracy έχει μια σταθερή αύξηση καθώς προστίθενται περισσότερα δεδομένα, πράγμα που υποδηλώνει βελτίωση της ικανότητας του μοντέλου να γενικεύει. Ταυτόχρονα, το training accuracy παρουσιάζει κάποιες διακυμάνσεις αλλά τελικά σταθεροποιείται, υποδηλώνοντας ότι το μοντέλο δεν υπερπροσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης. Aν και η κανονικοποίηση βοηθάει, το μοντέλο δεν είναι κατάλληλο για την επίλυση του προβλήματος εφόσον δυσκολεύεται αισθητά.

# Δημιουργία και Εκπαίδευση Νευρωνικού Δικτύου

Για τη δημιουργία του νευρωνικού δικτύου εξετάστηκαν δύο διαφορετικές αρχιτεκτονικές, μία με **δύο κρυφά επίπεδα** (ρηχή) και μία με **δέκα κρυφά επίπεδα** (βαθιά). Τα δίκτυα αυτά υλοποιήθηκαν ως **Dense** layers, που σημαίνει ότι κάθε νευρώνας σε κάθε επίπεδο είναι πλήρως συνδεδεμένος με όλους τους νευρώνες του επόμενου επιπέδου με βάση το **Sequential** μοντέλο της βιβλιοθήκης Keras. Χρησιμοποιήθηκαν οι μετρικές αξιολόγησης **categorial crossentropy loss** και **Accuracy.** Για κάθε μία από αυτές, εξετάστηκαν τα αποτελέσματα με βάση τις εξής παραμέτρους κατά την εκπαίδευση : **αριθμός νευρώνων ανά κρυφό επίπεδο**, **συναρτήσεις ενεργοποίησης**, **βελτιστοποιητές**, **αριθμό εποχών**, **batch size**, **ρυθμό μάθησης**. Πιο συγκεκριμένα, οι διαφορετικές τιμές των παραμέτρων που εξετάστηκαν είναι οι εξής:

* Αριθμός νευρώνων ανά κρυφό επίπεδο : 10,50,100,200
* Συναρτήσεις Ενεργοποίησης : Sigmoid, ReLU, συνδυασμός, Tanh, Swish
* Συναρτήσεις Βελτιστοποίησης : Adam, SGD, default(RMSprop)
* Αριθμός Εποχών : 10,50,100
* Batch size: 12,32,64
* Ρυθμό μάθησης : 0.0001, 0.01, 0.1

Προκειμένου να συγκριθούν με τους κατηγοριοποιητές KNN και NC που αναλύθηκαν παραπάνω, οι δοκιμές γίνονται με **20 χαρακτηριστικά** (one-hot-encoding) **χωρίς κανονικοποίηση**. Για την αξιολόγηση της απόδοσης του νευρωνικού δικτύου χρησιμοποιήθηκε η τεχνική **cross-validation**. Συγκεκριμένα, η τεχνική αυτή επιτρέπει την κατανομή του συνόλου δεδομένων σε *k* υποσύνολα, έτσι ώστε η εκπαίδευση και η αξιολόγηση να επαναλαμβάνονται *k* φορές, με κάθε φορά ένα διαφορετικό υποσύνολο να χρησιμοποιείται ως σύνολο δοκιμής και τα υπόλοιπα *k-1* υποσύνολα να χρησιμοποιούνται για την εκπαίδευση του μοντέλου. **Η διαδικασία αυτή μειώνει τον κίνδυνο overfitting και παρέχει μία πιο αξιόπιστη εκτίμηση της γενίκευσης του μοντέλου**. Υπολογίζονται **μέση όροι accuracy και loss** των folds.

\*Προκειμένου να συγκρίνονται ορθά τα πειράματα , κάθε φορά εξετάζονται στο ίδιο σύνολο δεδομένων διασφαλίζοντας ότι η τυχαία αναδιάταξη των δεδομένων είναι σταθερή και ξεκινάνε με την ίδια αρχικοποίηση βαρών.

Πιο συγκεκριμένα, χρησιμοποιούνται **5 folds** και το σύνολο δεδομένων έχει **2.111** δείγματα, οπότε τα δεδομένα χωρίζονται σε πέντε ίσα μέρη των 422 ή 423 δειγμάτων. Σε κάθε επανάληψη, το μοντέλο εκπαιδεύεται σε τέσσερα από αυτά τα μέρη (περίπου 1.688 δείγματα) και δοκιμάζεται στο υπόλοιπο ένα μέρος (περίπου 423 δείγματα). Η διαδικασία αυτή επαναλαμβάνεται πέντε φορές, έτσι ώστε κάθε fold να χρησιμοποιείται μία φορά ως σύνολο δοκιμής.

Πριν την εκπαίδευση, και λόγω χρήσης της συνάρτησης απώλειας **cross-entropy**, είναι απαραίτητη η μετατροπή της μεταβλητής **στόχου** σε **one-hot encoding**. Αυτό επιτρέπει στο μοντέλο να αναγνωρίζει τις κατηγορίες ως διακριτές τιμές (διανύσματα) και όχι ως συνεχείς αριθμητικές τιμές. Η **cross-entropy loss** υπολογίζει το σφάλμα συγκρίνοντας τις πιθανότητες που προβλέπει το μοντέλο για κάθε κατηγορία με το πραγματικό one-hot encoded διάνυσμα του στόχου, επιτρέποντας έτσι τη σωστή εκπαίδευση και κατηγοριοποίηση των δεδομένων.

## Αρχιτεκτονική με δύο κρυφά επίπεδα

Αρχικά, διερευνήθηκαν οι **επιδόσεις ως προς τον αριθμό των νευρώνων ανά κρυφό επίπεδο και της συνάρτησης ενεργοποίησής τους**. Διατηρήθηκε, ο default optimizer (RMSprop) με default ρυθμό μάθησης (0.0001), batch size 32 και **αριθμό εποχών 50**.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 10 | Sigmoid | Default | Default | 32 | 7.86s | 0.6034 | 1.0430 | 0.5931 | 1.0558 |
| 50 | Sigmoid | Default | Default | 32 | 8.35s | 0.7681 | 0.5858 | 0.7532 | 0.6092 |
| **100** | **Sigmoid** | **Default** | **Default** | **32** | **9.14s** | **0.7970** | **0.5035** | **0.7831** | **0.5337** |
| 200 | Sigmoid | Default | Default | 32 | 9.16s | 0.7859 | 0.5162 | 0.7703 | 0.5507 |
| 10 | ReLU | Default | Default | 32 | 8.18s | 0.6839 | 0.7577 | 0.6717 | 0.7692 |
| 50 | ReLU | Default | Default | 32 | 10.58s | 0.7637 | 0.5772 | 0.7347 | 0.6390 |
| **100** | **ReLU** | **Default** | **Default** | **32** | **10.91s** | **0.7875** | **0.5284** | **0.7622** | **0.6196** |
| 200 | ReLU | Default | Default | 32 | 13.27s | 0.8349 | 0.3996 | **0.8030** | 0.5124 |
| 10 | ReLU & Sigmoid | Default | Default | 32 | 10.37s | 0.6101 | 1.0089 | 0.6040 | 1.0139 |
| 50 | ReLU & Sigmoid | Default | Default | 32 | 9.80s | 0.7839 | 0.5637 | 0.7636 | 0.5945 |
| 100 | ReLU & Sigmoid | Default | Default | 32 | 10.35s | 0.7987 | 0.5056 | 0.7679 | 0.5547 |
| **200** | **ReLU & Sigmoid** | **Default** | **Default** | **32** | **12.22s** | **0.8167** | **0.4634** | **0.7850** | **0.5289** |
| 10 | Tanh | Default | Default | 32 | 11.08s | 0.6724 | 0.7920 | 0.6698 | 0.8237 |
| **50** | **Tanh** | **Default** | **Default** | **32** | **12.14s** | **0.8004** | **0.4875** | **0.7755** | **0.5448** |
| 100 | Tanh | Default | Default | 32 | 12.63s | 0.7797 | 0.5519 | 0.7518 | 0.6186 |
| 200 | Tanh | Default | Default | 32 | 14.70s | 0.8110 | 0.4763 | 0.7717 | 0.5731 |
| 10 | Swish | Default | Default | 32 | 12.39s | 0.7099 | 0.6922 | 0.6997 | 0.7146 |
| 50 | Swish | Default | Default | 32 | 10.91s | 0.7866 | 0.5198 | 0.7565 | 0.5798 |
| **100** | **Swish** | **Default** | **Default** | **32** | **11.86s** | **0.8173** | **0.4564** | **0.7864** | **0.5404** |
| 200 | Swish | Default | Default | 32 | 14.72s | 0.8516 | 0.3616 | **0.8139** | 0.4806 |

\*\*Με κόκκινο εμφανίζονται τα μοντέλα με τάση overfitting εφόσον η διαφορά μεταξύ **train accuracy** και **test accuracy** και το γεγονός ότι το **train loss** είναι σημαντικά χαμηλότερο από το **test loss** είναι σαφείς ενδείξεις αυτής της τάσης.

Μία πρώτη παρατήρηση είναι ότι **κανένα μοντέλο δεν υπερπροσαρμόζεται έντονα** στα δεδομένα γεγονός που μπορεί να αποδοθεί στη χρήση **cross-validation** η οποία συμβάλλει στο να διατηρείται η ισορροπία μεταξύ εκπαίδευσης και γενίκευσης του μοντέλου, με αποτέλεσμα να μειώνεται ο κίνδυνος έντονης υπερπροσαρμογής.

Αρχικά, χρησιμοποιώντας 10 νευρώνες ανά κρυφό επίπεδο, σε όλες τις συναρτήσεις ενεργοποίησης, η ακρίβεια τόσο στο train όσο και στο test set είναι χαμηλότερη συγκριτικά με τους περισσότερους νευρώνες, κάτι που υποδηλώνει ότι το δίκτυο δεν έχει αρκετή χωρητικότητα για να μάθει αποτελεσματικά τα μοτίβα των δεδομένων. Επομένως, η χρήση περισσότερων νευρώνων είναι απαραίτητη για να βελτιωθεί η απόδοση. Ωστόσο, παρατηρείται ότι η απόδοση δεν βελτιώνεται αισθητά μετά από 100 νευρώνες χωρίς να υπάρχουν τάσης υπερπροσαρμογής. Αυτό μπορεί να δείχνει ότι αύξηση των νευρώνων δεν βελτιώνει και την απόδοση.

Αναφορικά με τις συναρτήσεις ενεργοποίησης, η **Sigmoid** πετυχαίνει το χαμηλότερο χρόνο εκπαίδευσης αλλά και τη χαμηλότερη ακρίβεια, με μέγιστη την 78.31 με χρήση 100 νευρώνων. Δεν ωφελεί όμως η χρήση της εφόσον η Tanh (όπως η Sigmoid αλλά στο κέντρο της καμπύλης έχει 0) πετυχαίνει 0.7755 μέγιστη ακρίβεια με μειωμένη πολυπλοκότητα (χρήση 50 νευρώνων). Η χρήση της ReLU, προσφέρει σταθερή βελτίωση στην απόδοση, ειδικά για μεγαλύτερους αριθμούς νευρώνων. Πετυχαίνει ακρίβεια 0.8030 με χρήση 200 νευρώνων αλλά παρουσιάζει τάση για overfitting. Παράλληλα, ο συνδυασμός της με τη Sigmoid, δεν φαίνεται να προσφέρει σημαντική βελτίωση και σε ορισμένες περιπτώσεις παρουσιάζει μεγαλύτερο training και test loss πετυχαίνοντας μέγιστη ακρίβεια 0.7850 με χρήση του ίδιου πλήθους νευρώνων. Τέλος, η **προτεινόμενη από τη Google Swish**, φαίνεται να έχει την καλύτερη απόδοση με οποιονδήποτε αριθμό νευρώνων. Ειδικότερα, όταν αυξάνεται ο αριθμός των νευρώνων (200), το **training loss** και το **test loss** είναι τα **χαμηλότερα**, με **ακρίβεια 0.8139**, κάτι που δείχνει ότι η Swish μπορεί να βοηθήσει το δίκτυο να μάθει πιο αποτελεσματικά. Η αύξηση του αριθμού των νευρώνων αυξάνει τον **training time**, όπως αναμένεται, αλλά τα κέρδη απόδοσης δεν είναι πάντα γραμμικά. Η χρήση 200 νευρώνων και της συνάρτησης **Swish** οδηγεί σε το καλύτερο αποτέλεσμα, αλλά με σημαντικό κόστος στο χρόνο εκπαίδευσης. Παρόλα αυτά, λόγω τάσης overfitting με χρήση 200 νευρώνων, **η καλύτερη επιλογή είναι η χρήση 100 νευρώνων όπου πετυχαίνει ικανοποιητική απόδοση 0.7864** παραμένοντας η υψηλότερη από τα υπόλοιπα μοντέλα και με αλλαγές στις υπόλοιπες παραμέτρους ενδέχεται να επιφέρει ακόμα καλύτερα αποτελέσματα.

Συμπερασματικά, για όλες τις συναρτήσεις φαίνεται πως **μέγιστο πλήθος νευρώνων ανά κρυφό επίπεδο που πρέπει να χρησιμοποιηθούν είναι 100** για να μειωθεί το φαινόμενο overfitting. Το μόνο μοντέλο που δεν παρουσιάζει έντονη τάση για overfitting με χρήση 200 νευρώνων είναι ο συνδυασμός ReLU & Sigmoid. Τελικά για το επόμενο στάδιο πειραματισμού θα κρατηθούν τα μοντέλα που επέφεραν τις καλύτερες επιδόσεις :

* Tanh 50 νευρώνες 0.7755
* Sigmoid 100 νευρώνες 0.7831
* ReLU & Sigmoid 200 νευρώνες 0.7850
* Swish 100 νευρώνες 0.7864

Στην επόμενη φάση, εξετάστηκαν οι επιδόσεις των παραπάνω μοντέλων ως προς δύο διαφορετικούς optimizer, τον SGD και τον Adam με το default τους learning rate 0.01 και 0.001 αντίστοιχα.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 50 | Tanh | Default | Default | 32 | 12.14s | 0.8004 | 0.4875 | 0.7755 | 0.5448 |
| **50** | **Tanh** | **Adam** | **Default** | **32** | **11.99s** | **0.8701** | **0.3361** | **0.8314** | **0.4166** |
| 50 | Tanh | SGD | Default | 32 | 9.40s | 0.3445 | 1.5259 | 0.3534 | 1.5069 |
| 100 | Sigmoid | Default | Default | 32 | 9.14s | 0.7970 | 0.5035 | 0.7831 | 0.5337 |
| **100** | **Sigmoid** | **Adam** | **Default** | **32** | **10.63s** | **0.8592** | **0.3831** | **0.8337** | **0.4293** |
| 100 | Sigmoid | SGD | Default | 32 | 11.56s | 0.3476 | 1.5655 | 0.3487 | 1.5703 |
| 200 | ReLU & Sigmoid | Default | Default | 32 | 12.22s | 0.8167 | 0.4634 | 0.7850 | 0.5289 |
| **200** | **ReLU & Sigmoid** | **Adam** | **Default** | **32** | **13.47s** | **0.8919** | **0.2950** | **0.8541** | **0.3782** |
| 200 | ReLU & Sigmoid | SGD | Default | 32 | 11.85s | 0.4851 | 1.2173 | 0.4775 | 1.2241 |
| 100 | Swish | Default | Default | 32 | 11.86s | 0.8173 | 0.4564 | 0.7864 | 0.5404 |
| **100** | **Swish** | **Adam** | **Default** | **32** | **12.74s** | **0.8646** | **0.3531** | **0.8247** | **0.4446** |
| 100 | Swish | SGD | Default | 32 | 10.38s | 0.4574 | 1.2238 | 0.4571 | 1.2309 |

Από τον παραπάνω πίνακα αποτελεσμάτων, προκύπτει ότι **Adam** είναι ο **καλύτερος βελτιστοποιητής** για όλα τα σενάρια, επιτυγχάνοντας υψηλότερη ακρίβεια και χαμηλότερη απώλεια τόσο στο training όσο και στο test set, ενώ αυξάνει ανεπαίσθητα το χρόνο εκπαίδευσης. Αυτό υποδηλώνει ότι ο Adam μπορεί να χειριστεί καλύτερα τη σύγκλιση των παραμέτρων. Ωστόσο, υπάρχει μία τάση **υπερπροσαρμογής ειδικά με τη χρήση της συνάρτησης ενεργοποίησης Tanh**. O RMSprop αποδίδει ικανοποιητικά αλλά παραμένει χαμηλότερος από τον Adam. Τέλος, ο **SGD** έχει τα χειρότερα αποτελέσματα, με πολύ χαμηλή ακρίβεια και υψηλή απώλεια, επιδρώντας αρνητικά δείχνοντας προβλήματα στη μάθηση και την αποτελεσματικότητα.

Πιο συγκεκριμένα, o **SGD** ενημερώνει τα βάρη του νευρωνικού δικτύου χρησιμοποιώντας ένα μόνο δείγμα τη φορά (ή μικρά batches), κάτι που μπορεί να οδηγήσει σε ακανόνιστες ενημερώσεις. Αυτό σημαίνει ότι ο αλγόριθμος μπορεί να "πηδάει" γύρω από το ελάχιστο σημείο της συνάρτησης κόστους, χωρίς να συγκλίνει σταθερά. Αυτό μπορεί να οφείλεται στο learning rate σε συνδυασμό με τις υπόλοιπες παραμέτρους που πρέπει να προσαρμοστεί κατάλληλα για να ταιριάζει στο πρόβλημα. Για παράδειγμα οι καμπύλες απώλειας του μοντέλου που χρησιμοποιεί **Tanh και 50** **νευρώνες** που φαίνονται παρακάτω, υποδηλώνουν ότι πρέπει να μειωθεί το learning rate ώστε να μειωθούν οι διακυμάνσεις.

Στη συνέχεια, εξετάστηκαν οι επιδόσεις των μοντέλων σε διαφορετικά learning rates ώστε να προκύψουν πιο ορθά συμπεράσματα.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| **50** | **Tanh** | **Default** | **Default** | **32** | **12.14s** | **0.8004** | **0.4875** | **0.7755** | **0.5448** |
| 50 | Tanh | Default | 0.1 | 32 | 12.67s | 0.1466 | 5.3496 | 0.1402 | 5.3630 |
| 50 | Tanh | Default | 0.01 | 32 | 11.94s | 0.7446 | 0.6415 | 0.7134 | 0.7193 |
| 50 | Tanh | Default | 0.001 | 32 | 10.11s | 0.6178 | 1.0349 | 0.6035 | 1.0499 |
| **50** | **Tanh** | **Adam** | **Default** | **32** | **11.99s** | **0.8701** | **0.3361** | **0.8314** | **0.4166** |
| 50 | Tanh | Adam | 0.1 | 32 | 12.33s | 0.1373 | 2.2426 | 0.1412 | 2.2297 |
| 50 | Tanh | Adam | 0.01 | 32 | 11.85s | 0.8035 | 0.4851 | 0.7579 | 0.5874 |
| 50 | Tanh | Adam | 0.0001 | 32 | 11.27s | 0.6729 | 0.9146 | 0.6585 | 0.9306 |
| 50 | Tanh | SGD | Default | 32 | 9.40s | 0.3445 | 1.5259 | 0.3534 | 1.5069 |
| 50 | Tanh | SGD | 0.001 | 32 | 10.71s | 0.3723 | 1.5547 | 0.3789 | 1.5585 |
| 50 | Tanh | SGD | 0.0001 | 32 | 11.14s | 0.2842 | 1.8403 | 0.2818 | 1.8426 |
| **100** | **Sigmoid** | **Default** | **Default** | **32** | **9.14s** | **0.7970** | **0.5035** | **0.7831** | **0.5337** |
| 100 | Sigmoid | Default | 0.1 | 32 | 10.12s | 0.1500 | 2.1328 | 0.1554 | 2.1299 |
| 100 | Sigmoid | Default | 0.01 | 32 | 11.11s | 0.8063 | 0.4983 | 0.7622 | 0.6308 |
| 100 | Sigmoid | Default | 0.0001 | 32 | 12.42s | 0.5092 | 1.2780 | 0.5007 | 1.2863 |
| **100** | **Sigmoid** | **Adam** | **Default** | **32** | **10.63s** | **0.8592** | **0.3831** | **0.8337** | **0.4293** |
| 100 | Sigmoid | Adam | 0.1 | 32 | 12.63s | 0.1551 | 1.9956 | 0.1307 | 1.9929 |
| 100 | Sigmoid | Adam | 0.01 | 32 | 11.66s | 0.8939 | 0.2983 | 0.8318 | 0.4852 |
| 100 | Sigmoid | Adam | 0.0001 | 32 | 11.46s | 0.6028 | 1.0741 | 0.5903 | 1.0849 |
| 100 | Sigmoid | SGD | Default | 32 | 11.56s | 0.3476 | 1.5655 | 0.3487 | 1.5703 |
| 100 | Sigmoid | SGD | 0.001 | 32 | 9.86s | 0.2138 | 1.9057 | 0.1961 | 1.9111 |
| 100 | Sigmoid | SGD | 0.0001 | 32 | 10.58s | 0.1682 | 1.9576 | 0.1473 | 1.9667 |
| **200** | **ReLU & Sigmoid** | **Default** | **Default** | **32** | **12.22s** | **0.8167** | **0.4634** | **0.7850** | **0.5289** |
| 200 | ReLU & Sigmoid | Default | 0.1 | 32 | 13.18s | 0.1493 | 10.6596 | 0.1464 | 10.6742 |
| 200 | ReLU & Sigmoid | Default | 0.01 | 32 | 12.99s | 0.1470 | 2.1018 | 0.1559 | 2.1069 |
| 200 | ReLU & Sigmoid | Default | 0.0001 | 32 | 14.09s | 0.6280 | 0.9583 | 0.6154 | 0.9736 |
| **200** | **ReLU & Sigmoid** | **Adam** | **Default** | **32** | **13.47s** | **0.8919** | **0.2950** | **0.8541** | **0.3782** |
| 200 | ReLU & Sigmoid | Adam | 0.1 | 32 | 15.33s | 0.1373 | 2.7969 | 0.1388 | 2.7903 |
| 200 | ReLU & Sigmoid | Adam | 0.01 | 32 | 14.27s | 0.7262 | 0.6760 | 0.6925 | 0.8036 |
| 200 | ReLU & Sigmoid | Adam | 0.0001 | 32 | 14.73s | 0.6832 | 0.8819 | 0.6746 | 0.9005 |
| 200 | ReLU & Sigmoid | SGD | Default | 32 | 11.85s | 0.4851 | 1.2173 | 0.4775 | 1.2241 |
| 200 | ReLU & Sigmoid | SGD | 0.001 | 32 | 12.45s | 0.3787 | 1.5847 | 0.3709 | 1.5896 |
| **100** | **Swish** | **Default** | **Default** | **32** | **11.86s** | **0.8173** | **0.4564** | **0.7864** | **0.5404** |
| 100 | Swish | Default | 0.1 | 32 | 12.83s | 0.1509 | 2.0403 | 0.1511 | 2.0427 |
| 100 | Swish | Default | 0.01 | 32 | 13.14s | 0.7668 | 0.6204 | 0.7324 | 0.7506 |
| 100 | Swish | Default | 0.0001 | 32 | 12.56s | 0.6788 | 0.8309 | 0.6585 | 0.8538 |
| **100** | **Swish** | **Adam** | **Default** | **32** | **12.74s** | **0.8646** | **0.3531** | **0.8247** | **0.4446** |
| 100 | Swish | Adam | 0.1 | 32 | 14.47s | 0.1509 | 1.9735 | 0.1511 | 1.9795 |
| 100 | Swish | Adam | 0.01 | 32 | 12s | 0.8286 | 0.4665 | 0.7797 | 0.6541 |
| 100 | Swish | Adam | 0.0001 | 32 | 12.48s | 0.7247 | 0.7663 | 0.7120 | 0.7883 |
| 100 | Swish | SGD | Default | 32 | 10.38s | 0.4574 | 1.2238 | 0.4571 | 1.2309 |
| 100 | Swish | SGD | 0.001 | 32 | 11.15s | 0.4826 | 1.2489 | 0.4723 | 1.2506 |
| 100 | Swish | SGD | 0.0001 | 32 | 11.27s | 0.3524 | 1.5540 | 0.3486 | 1.5565 |

Από τον παραπάνω πίνακα φαίνεται πόσο σημαντικό είναι το learning rate στην επίδοση των μοντέλων. Σε κάθε περίπτωση οι δύο ακραίες τιμές τους learning rate (0.1 και 0.0001) επιδρούν πολύ αρνητικά στην επίδοση των μοντέλων μειώνοντας πολύ την ακρίβεια και αυξάνοντας την απώλεια. Πιο συγκεκριμένα, ενώ η μείωση του learning rate σε 0.001 βοηθάει τις επιδόσεις του SGD όπως αναμενόταν ενώ η υπερβολική μείωση του τα κάνει χειρότερα. Συνεπώς, τα αποτελέσματα παραμένουν χαμηλά οπότε η χρήση του στο πρόβλημα κρίνεται ακατάλληλη. Αναφορικά με τον default optimizer αυτός συνεχίζει να έχει καλές επιδόσεις με το default learning rate του αλλά παραμένει λιγότερο ικανοποιητικός από τον Adam. O **Adam** συνεχίζει να είναι ο καλύτερος optimizer για όλα τα μοντέλα με το **default του learning rate 0.001**. **Συμπερασματικά, το default learning rate είναι και το ιδανικό**.

Για την επόμενη φάση θα κρατηθούν τα μοντέλα με optimizer τον RMSprop και Adam με default learning rate.

Προκειμένου να εξεταστεί αν μπορεί να αυξηθεί το accuracy των μοντέλων θα παρατηρηθεί πως αυτό επηρεάζεται από την μεταβολή του αριθμού εποχών σε συνδυασμό με το batch size.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Epochs** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 50 | Tanh | Default | Default | 10 | - | 3.05s | 0.6562 | 0.8349 | 0.6390 | 0.8487 |
| 50 | Tanh | Default | Default | 10 | 16 | 5.46s | 0.6913 | 0.7538 | 0.6860 | 0.7702 |
| 50 | Tanh | Default | Default | 50 | - | 12.24s | 0.8004 | 0.4875 | 0.7755 | 0.5448 |
| 50 | Tanh | Default | Default | 50 | 16 | 19.50s | 0.8118 | 0.4655 | 0.7841 | 0.5348 |
| 50 | Tanh | Default | Default | 50 | 32 | 12.14s | 0.8004 | 0.4875 | 0.7755 | 0.5448 |
| 50 | Tanh | Default | Default | 50 | 64 | 10.93s | 0.7795 | 0.5499 | 0.7584 | 0.5963 |
| 50 | Tanh | Default | Default | 100 | - | 18.01s | 0.8444 | 0.3829 | 0.8115 | 0.4761 |
| 50 | Tanh | Default | Default | 100 | 16 | 34.65s | 0.8747 | 0.3124 | 0.8475 | 0.4211 |
| 50 | Tanh | Default | Default | 100 | 32 | 19.35s | 0.8444 | 0.3829 | 0.8115 | 0.4761 |
| 50 | Tanh | Default | Default | 100 | 64 | 15.44s | 0.8322 | 0.4269 | 0.7963 | 0.5081 |
| 50 | Tanh | Adam | Default | 10 | - | 3.73s | 0.7070 | 0.7534 | 0.6916 | 0.7764 |
| 50 | Tanh | Adam | Default | 10 | 16 | 5.78s | 0.7477 | 0.6204 | 0.7300 | 0.6543 |
| 50 | Tanh | Adam | Default | 50 | - | 11.99s | 0.8701 | 0.3361 | 0.8314 | 0.4166 |
| **50** | **Tanh** | **Adam** | **Default** | **50** | **16** | **25.90s** | **0.8666** | **0.3273** | **0.8366** | **0.4174** |
| 50 | Tanh | Adam | Default | 50 | 32 | 11.99s | 0.8701 | 0.3361 | 0.8314 | 0.4166 |
| 50 | Tanh | Adam | Default | 50 | 64 | 10.20s | 0.8460 | 0.4177 | 0.8157 | 0.4745 |
| 50 | Tanh | Adam | Default | 100 | - | 20.86s | 0.9133 | 0.2205 | 0.8750 | 0.3350 |
| 50 | Tanh | Adam | Default | 100 | 16 | 35.70s | 0.9224 | 0.1965 | 0.8669 | 0.3571 |
| 50 | Tanh | Adam | Default | 100 | 32 | 19.59s | 0.9133 | 0.2205 | 0.8750 | 0.3350 |
| 50 | Tanh | Adam | Default | 100 | 64 | 22.84s | 0.9118 | 0.2464 | 0.8745 | 9.3374 |
| 100 | Sigmoid | Default | Default | 10 | - | 2.91s | 0.5467 | 1.0419 | 0.5448 | 1.0473 |
| 100 | Sigmoid | Default | Default | 10 | 16 | 4.19s | 0.6252 | 0.8876 | 0.6101 | 0.8964 |
| 100 | Sigmoid | Default | Default | 50 | **-** | 9.93s | 0.7970 | 0.5035 | 0.7831 | 0.5337 |
| 100 | Sigmoid | Default | Default | 50 | 16 | 17.87s | 0.7978 | 0.5040 | 0.7850 | 0.5417 |
| 100 | Sigmoid | Default | Default | 50 | 32 | 9.14s | 0.7970 | 0.5035 | 0.7831 | 0.5337 |
| 100 | Sigmoid | Default | Default | 50 | 64 | 8.45s | 0.7363 | 0.6327 | 0.7338 | 0.6528 |
| 100 | Sigmoid | Default | Default | 100 | - | 18.51s | 0.8431 | 0.3919 | 0.8143 | 0.4376 |
| 100 | Sigmoid | Default | Default | 100 | 16 | 33.35s | 0.8906 | 0.2874 | 0.8660 | 0.3497 |
| 100 | Sigmoid | Default | Default | 100 | 32 | 18.68s | 0.8431 | 0.3919 | 0.8143 | 0.4376 |
| 100 | Sigmoid | Adam | Default | 10 | - | 3.86s | 0.6631 | 0.8743 | 0.6608 | 0.8555 |
| 100 | Sigmoid | Adam | Default | 10 | 16 | 4.77s | 0.7178 | 0.7338 | 0.7115 | 0.7434 |
| 100 | Sigmoid | Adam | Default | 50 | - | 11.80s | 0.8592 | 0.3831 | 0.8337 | 0.4293 |
| 100 | Sigmoid | Adam | Default | 50 | 16 | 17.77s | 0.8630 | 0.3441 | 0.8404 | 0.3988 |
| 100 | Sigmoid | Adam | Default | 50 | 32 | 10.63s | 0.8592 | 0.3831 | 0.8337 | 0.4293 |
| 100 | Sigmoid | Adam | Default | 50 | 64 | 9.04s | 0.8186 | 0.4836 | 0.8025 | 0.5176 |
| **100** | **Sigmoid** | **Adam** | **Default** | **100** | **-** | **19.30s** | **0.9057** | **0.2561** | **0.8768** | **0.3314** |
| 100 | Sigmoid | Adam | Default | 100 | 16 | 34.32s | 0.9090 | 0.2274 | 0.8659 | 0.3295 |
| 200 | ReLU & Sigmoid | Default | Default | 10 | - | 4.49s | 0.5680 | 0.9985 | 0.5590 | 1.0216 |
| 200 | ReLU & Sigmoid | Default | Default | 10 | 16 | 4.59s | 0.6574 | 0.8222 | 0.6471 | 0.8464 |
| 200 | ReLU & Sigmoid | Default | Default | 50 | - | 13.25s | 0.8167 | 0.4634 | 0.7850 | 0.5289 |
| 200 | ReLU & Sigmoid | Default | Default | 50 | 16 | 17.91s | 0.8526 | 0.3821 | 0.8209 | 0.4903 |
| 200 | ReLU & Sigmoid | Default | Default | 50 | 32 | 12.22s | 0.8167 | 0.4634 | 0.7850 | 0.5289 |
| 200 | ReLU & Sigmoid | Default | Default | 50 | 64 | 9.37s | 0.7414 | 0.6350 | 0.7120 | 0.6795 |
| 200 | ReLU & Sigmoid | Default | Default | 100 | - | 22.29s | 0.8999 | 0.2525 | 0.8569 | 0.4048 |
| 200 | ReLU & Sigmoid | Default | Default | 100 | 16 | 34.81s | 0.9135 | 0.2292 | 0.8631 | 0.4139 |
| 200 | ReLU & Sigmoid | Adam | Default | 10 | - | 4.21s | 0.7000 | 0.7628 | 0.6821 | 0.7928 |
| 200 | ReLU & Sigmoid | Adam | Default | 10 | 16 | 5.69s | 0.7203 | 0.6618 | 0.7025 | 0.6965 |
| **200** | **ReLU & Sigmoid** | **Adam** | **Default** | **50** | **-** | **13.10s** | **0.8919** | **0.2950** | **0.8541** | **0.3782** |
| 200 | ReLU & Sigmoid | Adam | Default | 50 | 16 | 18.12s | 0.8854 | 0.2808 | 0.8394 | 0.4131 |
| 200 | ReLU & Sigmoid | Adam | Default | 50 | 32 | 13.47s | 0.8919 | 0.2950 | 0.8541 | 0.3782 |
| 200 | ReLU & Sigmoid | Adam | Default | 50 | 64 | 10.29s | 0.8520 | 0.3920 | 0.8290 | 0.4613 |
| 200 | ReLU & Sigmoid | Adam | Default | 100 | - | 25.14s | 0.9378 | 0.1629 | 0.8835 | 0.3164 |
| 200 | ReLU & Sigmoid | Adam | Default | 100 | 16 | 35.93s | 0.9649 | 0.0984 | 0.8934 | 0.3308 |
| 100 | Swish | Default | Default | 10 | - | 3.74s | 0.6014 | 0.9517 | 0.5864 | 0.9876 |
| 100 | Swish | Default | Default | 10 | 16 | 4.86s | 0.6513 | 0.8476 | 0.6452 | 0.8717 |
| 100 | Swish | Default | Default | 50 | **-** | 9.79s | 0.8173 | 0.4564 | 0.7864 | 0.5404 |
| 100 | Swish | Default | Default | 50 | 16 | 18.24s | 0.8818 | 0.3020 | 0.8475 | 0.4337 |
| 100 | Swish | Default | Default | 50 | 32 | 11.86s | 0.8173 | 0.4564 | 0.7864 | 0.5404 |
| 100 | Swish | Default | Default | 50 | 64 | 9.42s | 0.7528 | 0.6101 | 0.7243 | 0.6719 |
| 100 | Swish | Default | Default | 100 | - | 18.56s | 0.9090 | 0.2270 | 0.8527 | 0.4420 |
| 100 | Swish | Default | Default | 100 | 16 | 33.79s | 0.9276 | 0.1905 | 0.8598 | 0.5392 |
| 100 | Swish | Adam | Default | 10 | - | 3.78s | 0.7264 | 0.6975 | 0.7059 | 0.7375 |
| 100 | Swish | Adam | Default | 10 | 16 | 4.90s | 0.7525 | 0.6032 | 0.7314 | 0.6373 |
| 100 | Swish | Adam | Default | 50 | - | 12.38s | 0.8646 | 0.3531 | 0.8247 | 0.4446 |
| **100** | **Swish** | **Adam** | **Default** | **50** | **16** | **18.18s** | **0.8928** | **0.2820** | **0.8494** | **0.4142** |
| 100 | Swish | Adam | Default | 50 | 32 | 12.74s | 0.8646 | 0.3531 | 0.8247 | 0.4446 |
| 100 | Swish | Adam | Default | 50 | 64 | 9.80s | 0.8524 | 0.3926 | 0.8138 | 0.4683 |
| 100 | Swish | Adam | Default | 100 | - | 22.58s | 0.9339 | 0.1823 | 0.8749 | 0.3872 |
| 100 | Swish | Adam | Default | 100 | 16 | 34.28s | 0.9522 | 0.1288 | 0.8759 | 0.4472 |

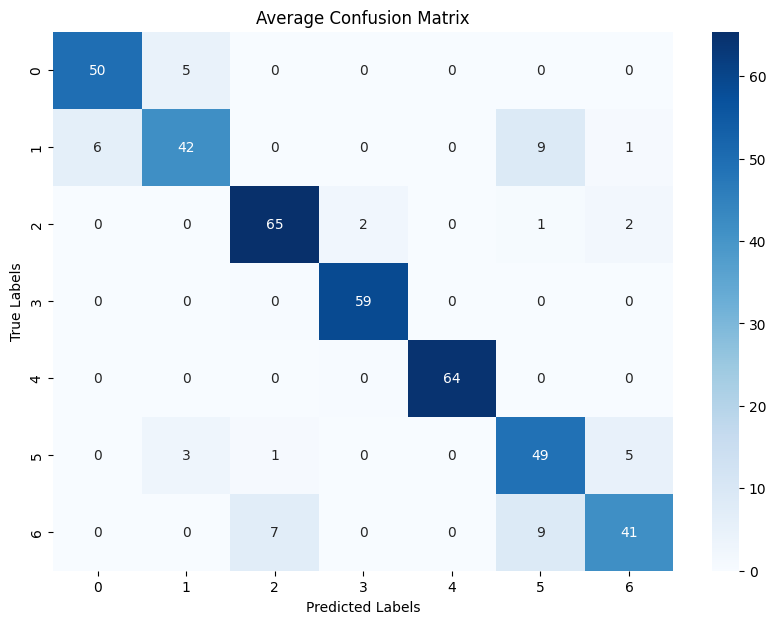
\*Με κόκκινο φαίνονται τα μοντέλα που υπερπροσαρμόζονται ή τείνουν να το κάνουν

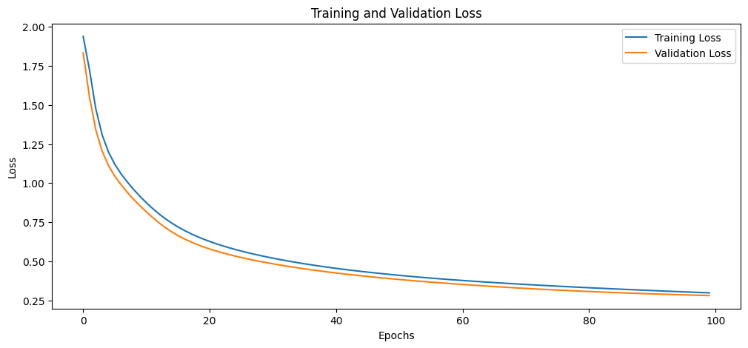
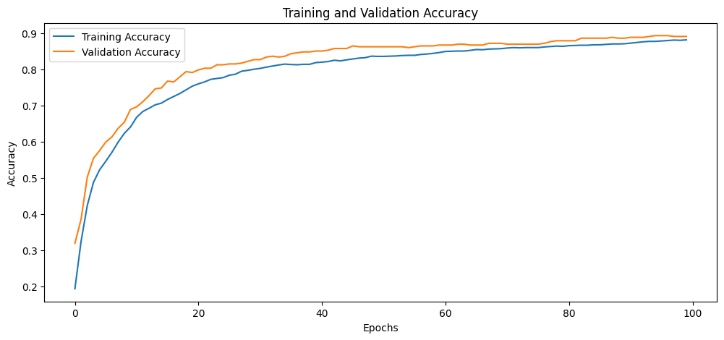
Μια πρώτη παρατήρηση είναι ότι με την αύξηση του αριθμού των εποχών σε όλα τα μοντέλα, υπάρχει σημαντική βελτίωση της ακρίβειας αλλά και του χρόνου εκπαίδευσης. Αυτό δείχνει ότι περισσότερες εποχές επιτρέπουν στο μοντέλο να μαθαίνει καλύτερα από τα δεδομένα. Ωστόσο, σε ορισμένες περιπτώσεις, η αύξηση των εποχών (από 50 σε 100) οδηγεί σε μεγαλύτερη πιθανότητα εμφάνισης overfitting, καθώς το training accuracy αυξάνεται σημαντικά, αλλά το test accuracy δεν αυξάνεται ανάλογα. Παράδειγμα: Με 100 εποχές και Adam optimizer, η εκπαίδευση επιτυγχάνει υψηλό training accuracy (0.9133) αλλά το test accuracy σταθεροποιείται γύρω στο 0.8750. Συμπερασματικά, η χρήση **50 εποχών είναι ιδανική** εφόσον προκύπτουν καλύτερα αποτελέσματα σε καλό χρόνο εκπαίδευσης και οι 10 δεν αρκούν ενώ οι 100 παρουσιάζουν υπερπροσαρμογή ή τάσεις υπερπροσαρμογής.

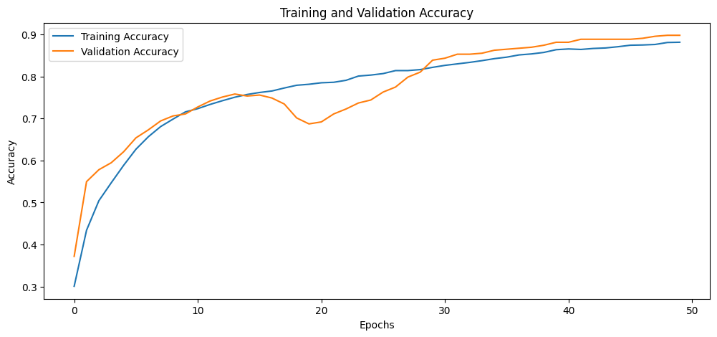
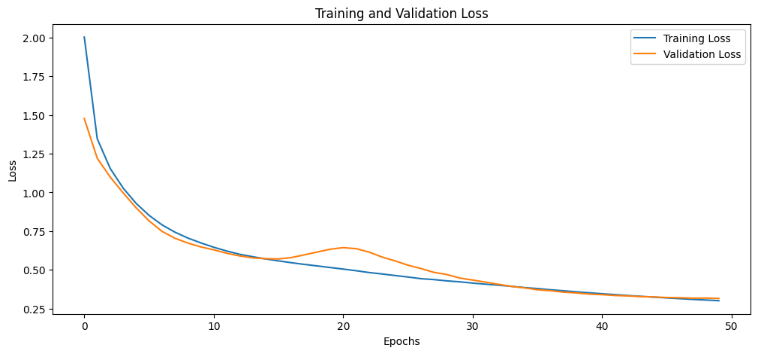
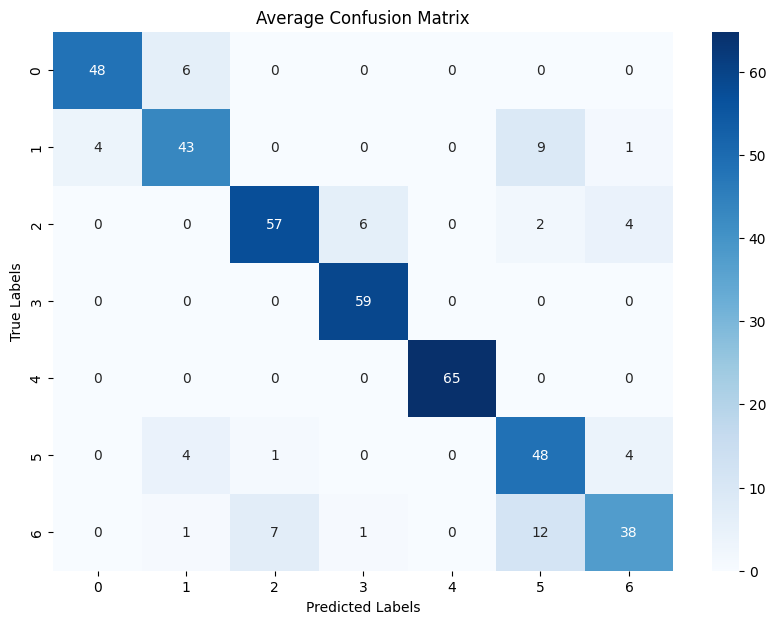
Στη συνέχεια, παρατηρείται ότι μικρό batch size (16) χρειάζεται περισσότερο χρόνο εκπαίδευσης αλλά συχνά προσφέρει καλύτερη απόδοση στο test set. Με χρήση μεγαλύτερου batch size (64), η εκπαίδευση είναι ταχύτερη αλλά τα μοντέλα δεν γενικεύουν τόσο καλά μειώνοντας το test accuracy. Χωρίς τη χρήση καθόλου batch size η επίδοση των μοντέλων ταυτίζεται με αυτή της χρήσης batch size 32 αλλά σε λιγότερο χρόνο εκπαίδευσης. Συμπερασματικά, η χρήση **batch size 16 είναι ιδανική** για το πρόβλημα εφόσον πετυχαίνει τις καλύτερες επιδόσεις και βελτιώνει τη γενίκευση, με μόνο αρνητικό το χρόνο εκπαίδευσης που αυξάνεται αλλά όχι απαγορευτικά.

Πιο συγκεκριμένα, για να υπάρχει ισορροπία μεταξύ γενίκευσης και καλής επίδοσης, για τη συνάρτηση ενεργοποίησης Tanh ο καλύτερος συνδυασμός είναι Adam Optimizer, 50 Epochs, Batch Size 16 (0.8366). Για τη Sigmoid ο καλύτερος συνδυασμός είναι Adam Optimizer, 100 Epochs, Batch Size – (0.8768). Για τη ReLU & Sigmoid είναι Adam Optimizer, 50 Epochs, Batch Size – (0.8541). Για τη Swish είναι Adam Optimizer, 50 Epochs, Batch size 16 (0.8494).

* **Καλύτερη συνολική επιλογή:** **Sigmoid με Adam Optimizer, 100 Epochs, Batch Size – (0.8768)**. Αν και υπάρχει ρίσκο υπερπροσαρμογής, είναι το πιο αποδοτικό μοντέλο.
* **Πολύ καλή ισορροπία:** **Swish με Adam Optimizer, 50 Epochs, Batch Size 16 (0.8494)**. Είναι μια ασφαλής επιλογή με καλή γενίκευση.

Sigmoid:



Swish:

Από τα παραπάνω, εξαιρώντας τα μοντέλα που εμφανίζουν πρόβλημα στη γενίκευση, παρατηρείται ότι κανένα από τα δύο μοντέλα δεν εμφανίζει τάση υπερπροσαρμογής. Το **καλύτερο μοντέλο** είναι το εξής: **100 Νευρώνες, Sigmoid, Adam optimizer, 100 Epochs, όχι batch size με test accuracy ίσο με 0.8768 και χρόνο εκπαίδευσης 19.30s.** Παρόλα αυτά αξίζει να δοκιμαστεί περαιτέρω και το μοντέλο με Swish.

## Αρχιτεκτονική με 10 κρυφά επίπεδα

Για αυτή την αρχιτεκτονική, διερευνήθηκαν οι **επιδόσεις ως προς τον αριθμό των νευρώνων ανά κρυφό επίπεδο και της συνάρτησης ενεργοποίησής τους**. Διατηρήθηκε, ο Adam optimizer με default ρυθμό μάθησης (0.0001), batch size 32 και **αριθμό εποχών 50**.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 10 | Sigmoid | Adam | Default | 32 | 15.22s | 0.2078 | 1.8277 | 0.1942 | 1.8358 |
| 30 | Sigmoid | Adam | Default | 32 | 14.81s | 0.4639 | 1.2672 | 0.4581 | 1.2818 |
| 60 | Sigmoid | Adam | Default | 32 | 17.08s | 0.6923 | 0.7600 | 0.6868 | 0.7733 |
| 100 | Sigmoid | Adam | Default | 32 | 18.51s | 0.6020 | 0.9150 | 0.5878 | 0.9349 |
| 10 | ReLU | Adam | Default | 32 | 14.91s | 0.7251 | 0.6292 | 0.7039 | 0.6292 |
| 30 | ReLU | Adam | Default | 32 | 16.59s | 0.7411 | 0.6165 | 0.7200 | 0.6367 |
| **60** | **ReLU** | **Adam** | **Default** | **32** | **17.14s** | **0.8002** | **0.4581** | **0.7849** | **0.5265** |
| 100 | ReLU | Adam | Default | 32 | 18.45s | 0.7958 | 0.4966 | 0.7555 | 0.5973 |
| 10 | Tanh | Adam | Default | 32 | 16.04s | 0.6466 | 0.7882 | 0.6410 | 0.8204 |
| 30 | Tanh | Adam | Default | 32 | 15.82s | 0.8455 | 0.3842 | 0.8138 | 0.4662 |
| **60** | **Tanh** | **Adam** | **Default** | **32** | **17.72s** | **0.8883** | **0.2874** | **0.8517** | **0.4002** |
| 100 | Tanh | Adam | Default | 32 | 18.25s | 0.8849 | 0.3005 | 0.8318 | 0.4366 |
| 10 | Swish | Adam | Default | 32 | 16.20s | 0.7564 | 0.5379 | 0.7375 | 0.5681 |
| 30 | Swish | Adam | Default | 32 | 15.51s | 0.8140 | 0.4207 | 0.7887 | 0.4796 |
| **60** | **Swish** | **Adam** | **Default** | **32** | **19.54s** | **0.8291** | **0.3881** | **0.7906** | **0.4777** |
| 100 | Swish | Adam | Default | 32 | 18.96s | 0.8388 | 0.3882 | 0.7878 | 0.5339 |

\*Με κόκκινο φαίνονται τα μοντέλα που υπερπροσαρμόζονται

Μια σημαντική παρατήρηση είναι ότι η αύξηση του αριθμού των επιπέδων δεν συνοδεύεται από βελτίωση της ακρίβειας, ενώ αυξάνεται ο χρόνος εκπαίδευσης. Αντίθετα, η υπερβολική αύξηση της πολυπλοκότητας του μοντέλου, μέσω περισσότερων επιπέδων, μπορεί να οδηγήσει σε μείωση της ακρίβειας και χειρότερη γενίκευση. Παρόλαυτα, η ReLU και η Tanh έχουν καλύτερες επιδόσεις σε σχέση με αυτές με χρήση 2 κρυφών επιπέδων γεγονός που αναμενόταν εφόσον είναι γνωστό ότι τα πηγαίνουν καλύτερα σε πιο βαθιές αρχιτεκτονικές.

Επιπλέον, παρατηρείται ότι η χρήση **60 νευρώνων ανά κρυφό επίπεδο** αποτελεί το βέλτιστο σημείο για τα περισσότερα μοντέλα. Με την αύξηση του αριθμού των νευρώνων πέρα από αυτό το όριο, η ακρίβεια είτε σταθεροποιείται είτε μειώνεται, ενώ το μοντέλο αρχίζει να υπερπροσαρμόζεται.

Πιο συγκεκριμένα, η **Sigmoid** δεν αποδίδει καλά σε δίκτυα με πολλά επίπεδα λόγω της τάσης της να προκαλεί **vanishing gradients**. Το gradient γίνεται πολύ μικρό για τα πρώτα επίπεδα, με αποτέλεσμα οι βελτιώσεις να είναι ελάχιστες ή να μη μεταδίδονται επαρκώς στα αρχικά επίπεδα. Επομένως, κρίνεται ακατάλληλο για 10 κρυφά επίπεδα. Με τη σειρά της η **ReLU**, αποδίδει καλύτερα λόγω της απλότητας και της ικανότητάς της να διαχειρίζεται βαθιά δίκτυα χωρίς να κορεστεί. Το ίδιο και η **Tanh**, διότι έχει καλύτερη ευαισθησία στις μικρές διαφορές εισόδου και σταθεροποιεί τα gradients.Τέλος, Η **Swish** αποδίδει ικανοποιητικά αλλά πετυχαίνοντας χαμηλότερο accuracy, καθώς προσαρμόζει δυναμικά την ενεργοποίηση κάθε νευρώνα.

Τελικά, το καλύτερο μοντέλο είναι : **Tanh** καθώς προσφέρει τη μέγιστη ακρίβεια **(85.17%)** με καλή ισορροπία μεταξύ train/test accuracy και χαμηλό loss.

Πριν την επόμενη φάση πειραματισμού, θα ήταν καλό να δοκιμαστεί συνδυασμός αυτών των τριών συναρτήσεων ενεργοποίησης ώστε να εξεταστούν τα αποτελέσματα που επιφέρουν. **Επομένως, θα εξεταστεί ένα μοντέλο το οποίο θα έχει στα τρία πρώτα επίπεδα συνάρτηση ενεργοποίησης ReLU, στα τρία επόμενα Tanh και στα τρία τελευταία Swish.**

ReLU για γρήγορη σύγλιση, Tanh για σύλληψη πιο περίπλοκων σχέσεων με εξισορρόπηση και τέλος Swish για πιο λεπτομερείς αποφάσεις κοντά στην έξοδο. Προκύπτουν τα παρακάτω αποτελέσματα :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 10 | Συνδυασμός | Adam | Default | 32 | 20.61s | 0.7453 | 0.5651 | 0.7262 | 0.5931 |
| 30 | Συνδυασμός | Adam | Default | 32 | 22.38s | 0.7893 | 0.4851 | 0.7608 | 0.5390 |
| **60** | **Συνδυασμός** | **Adam** | **Default** | **32** | **20.74s** | **0.7910** | **0.4768** | **0.7750** | **0.5342** |
| 100 | Συνδυασμός | Adam | Default | 32 | 21.48s | 0.8362 | 0.3858 | 0.8029 | 0.4799 |

Τελικά**, δεν επιφέρει τα αναμενόμενα αποτελέσματα εφόσον το accuracy είναι ικανοποιητικό αλλά παραμένει χαμηλό** σε σχέση με της συναρτήσεις ενεργοποίησης όταν αυτές χρησιμοποιούνται μεμονωμένα.

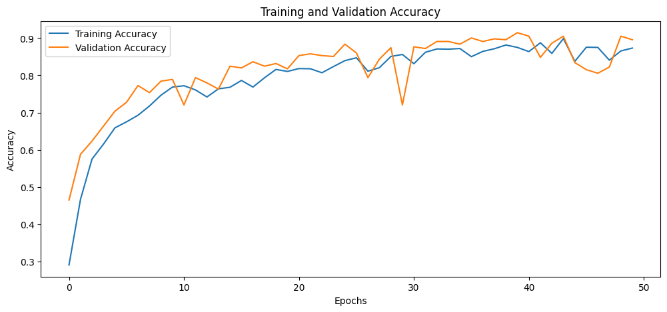
Στην επόμενη φάση εξετάστηκαν οι επιδόσεις των μοντέλων με διαφορετικά learning rates χρησιμοποιώντας optimizer Adam.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| **60** | **Συνδυασμός** | **Adam** | **Default** | **32** | **20.74s** | **0.7910** | **0.4768** | **0.7750** | **0.5342** |
| 60 | Συνδυασμός | Adam | 0.1 | 32 | 21.15s | 0.1535 | 1.9507 | 0.1535 | 1.9527 |
| 60 | Συνδυασμός | Adam | 0.01 | 32 | 20.82s | 0.1577 | 1.9443 | 0.1492 | 1.9453 |
| 60 | Συνδυασμός | Adam | 0.0001 | 32 | 20.87s | 0.7662 | 0.5170 | 0.7437 | 0.5550 |
| **60** | **ReLU** | **Adam** | **Default** | **32** | **17.14s** | **0.8002** | **0.4581** | **0.7849** | **0.5265** |
| 60 | ReLU | Adam | 0.1 | 32 | 21.27s | 0.1535 | 1.9507 | 0.1535 | 1.9527 |
| 60 | ReLU | Adam | 0.01 | 32 | 20.88s | 0.7842 | 0.4842 | 0.7589 | 0.5171 |
| 60 | ReLU | Adam | 0.0001 | 32 | 19.83s | 0.7185 | 0.6576 | 0.7105 | 0.6788 |
| **60** | **Tanh** | **Adam** | **Default** | **32** | **17.72s** | **0.8883** | **0.2874** | **0.8517** | **0.4002** |
| 60 | Tanh | Adam | 0.01 | 32 | 19.84s | 0.1509 | 2.0227 | 0.1511 | 2.0267 |
| 60 | Tanh | Adam | 0.0001 | 32 | 21.23s | 0.8031 | 0.4868 | 0.7826 | 0.5321 |
| **60** | **Swish** | **Adam** | **Default** | **32** | **19.54s** | **0.8291** | **0.3881** | **0.7906** | **0.4777** |
| 60 | Swish | Adam | 0.01 | 32 | 20.92s | 0.3047 | 1.6495 | 0.2975 | 1.6764 |
| 60 | Swish | Adam | 0.0001 | 32 | 22.95s | 0.7752 | 0.5015 | 0.7617 | 0.5257 |

Για ακόμα μια φορά το learning rate που είναι κατάλληλο για όλα τα μοντέλα είναι το default.

Στην συνέχεια θα εξεταστεί αν υπάρχει βελτίωση της ακρίβειας των μοντέλων με διαφορετικό αριθμό εποχών και batch size.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες ανά κρυφό επίπεδο** | **Συνάρτηση Ενεργοποίησης** | **Optimizer** | **Learning rate** | **Epochs** | **Batch size** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 60 | Tanh | Adam | Default | 10 | - | 8.19s | 0.7423 | 0.5964 | 0.7210 | 0.6456 |
| 60 | Tanh | Adam | Default | 10 | 16 | 9.38 s | 0.7203 | 0.6490 | 0.6985 | 0.6860 |
| 60 | Tanh | Adam | Default | 50 | - | 21.07s | 0.8883 | 0.2874 | 0.8517 | 0.4002 |
| 60 | Tanh | Adam | Default | 50 | 16 | 32.39s | 0.8431 | 0.4105 | 0.8029 | 0.5149 |
| **60** | **Tanh** | **Adam** | **Default** | **50** | **32** | **17.72s** | **0.8883** | **0.2874** | **0.8517** | **0.4002** |
| 60 | Tanh | Adam | Default | 50 | 64 | 13.73s | 0.8810 | 0.3067 | 0.8446 | 0.4137 |
| 60 | Tanh | Adam | Default | 100 | - | 37.52s | 0.8698 | 0.3416 | 0.8210 | 0.5688 |
| 60 | ReLU | Adam | Default | 10 | - | 7.49s | 0.6324 | 0.8448 | 0.6144 | 0.8674 |
| 60 | ReLU | Adam | Default | 10 | 16 | 11.74s | 0.6735 | 0.7512 | 0.6523 | 0.7770 |
| 60 | ReLU | Adam | Default | 50 | - | 19.14s | 0.8002 | 0.4581 | 0.7849 | 0.5265 |
| 60 | ReLU | Adam | Default | 50 | 16 | 34.09s | 0.8239 | 0.4055 | 0.7920 | 0.4915 |
| **60** | **ReLU** | **Adam** | **Default** | **50** | **32** | **17.14s** | **0.8002** | **0.4581** | **0.7849** | **0.5265** |
| 60 | ReLU | Adam | Default | 50 | 64 | 14.26s | 0.7936 | 0.4776 | 0.7731 | 0.5220 |
| 60 | ReLU | Adam | Default | 100 | - | 40.73s | 0.8558 | 0.3598 | 0.8195 | 0.5272 |
| 60 | ReLU | Adam | Default | 100 | 16 | 64.88s | 0.8829 | 0.3008 | 0.8190 | 0.5265 |
| 60 | Συνδυασμός | Default | Default | 10 | - | 6.16s | 0.6682 | 0.7365 | 0.6651 | 0.7458 |
| 60 | Συνδυασμός | Default | Default | 10 | 16 | 8.95s | 0.7339 | 0.5910 | 0.7234 | 0.6119 |
| **60** | **Συνδυασμός** | **Default** | **Default** | **50** | **-** | **25.74s** | **0.7910** | **0.4768** | **0.7750** | **0.5342** |
| 60 | Συνδυασμός | Default | Default | 50 | 16 | 28.45s | 0.7988 | 0.4539 | 0.7735 | 0.5396 |
| 60 | Συνδυασμός | Default | Default | 50 | 32 | 20.74s | 0.7910 | 0.4768 | 0.7750 | 0.5342 |
| 60 | Συνδυασμός | Default | Default | 50 | 64 | 13.41s | 0.7827 | 0.4987 | 0.7617 | 0.5408 |
| 60 | Συνδυασμός | Default | Default | 100 | - | 34.17s | 0.9009 | 0.2529 | 0.8456 | 0.4281 |
| 60 | Swish | Adam | Default | 10 | - | 7.37s | 0.7277 | 0.5922 | 0.7205 | 0.6089 |
| 60 | Swish | Adam | Default | 10 | 16 | 12.44s | 0.7369 | 0.5659 | 0.7181 | 0.5998 |
| 60 | Swish | Adam | Default | 50 | - | 22.54s | 0.8291 | 0.3881 | 0.7906 | 0.4777 |
| **60** | **Swish** | **Adam** | **Default** | **50** | **16** | **31.36s** | **0.8331** | **0.4018** | **0.7949** | **0.5459** |
| 60 | Swish | Adam | Default | 50 | 32 | 19.54s | 0.8291 | 0.3881 | 0.7906 | 0.4777 |
| 60 | Swish | Adam | Default | 50 | 64 | 13.35s | 0.7836 | 0.5097 | 0.7627 | 0.5532 |
| 60 | Swish | Adam | Default | 100 | - | 35.72s | 0.8971 | 0.2548 | 0.8347 | 0.5348 |

Προκύπτουν τα ίδια συμπεράσματα για τις εποχές και τα batch size όπως στα μοντέλα με δυο κρυφά επίπεδα. Επομένως, και για αυτή την αρχιτεκτονική **ιδανικός αριθμός εποχών είναι 50** και ιδανικό **batch size 32.** Τις καλύτερες επιδόσεις χωρίς την εμφάνιση υπερπροσαρμογής έχει το εξής μοντέλο **60 Νευρώνες, Tanh, Adam optimizer, 50 Epochs, 32 batch size με test accuracy ίσο με 0.8517 και χρόνο εκπαίδευσης 17.72s** τα γραφήματα της οποίας φαίνονται παρακάτω:

**Επειδή οι καμπύλες παρουσιάζουν κάποια αστάθεια όπως φαίνεται, εξετάστηκε η μείωση του learning rate σε 0.0001. Αυτό έχει ως αποτέλεσμα την εξομάλυνση των καμπυλών αλλά πετυχαίνει χαμηλότερη ακρίβεια 0.7826 με καλή γενίκευση**.

## Παρατηρήσεις

Παρατηρείται σε κάθε περίπτωση ότι η **καμπύλη του test loss είναι κάτω από αυτή του train loss**, ενώ **η καμπύλη του test accuracy υπερβαίνει αυτή του train accuracy**. Αυτό πιθανώς οφείλεται στη χρήση cross-validation, όπου τα test folds μπορεί να περιλαμβάνουν πιο "εύκολα" ή αντιπροσωπευτικά δείγματα. Αν το training set περιέχει πιο δύσκολα ή θορυβώδη δεδομένα, το μοντέλο ενδέχεται να παρουσιάσει χαμηλότερη απόδοση στην εκπαίδευση. Επίσης, η ισορροπημένη κατανομή των κλάσεων στα test folds ενισχύει τη γενίκευση, οδηγώντας σε υψηλότερη ακρίβεια και μικρότερο loss στα test δεδομένα. Αυτή η διαφορά είναι φυσιολογική και υποδηλώνει καλή ποιότητα του test set.

## Σύγκριση δύο Αρχιτεκτονικών

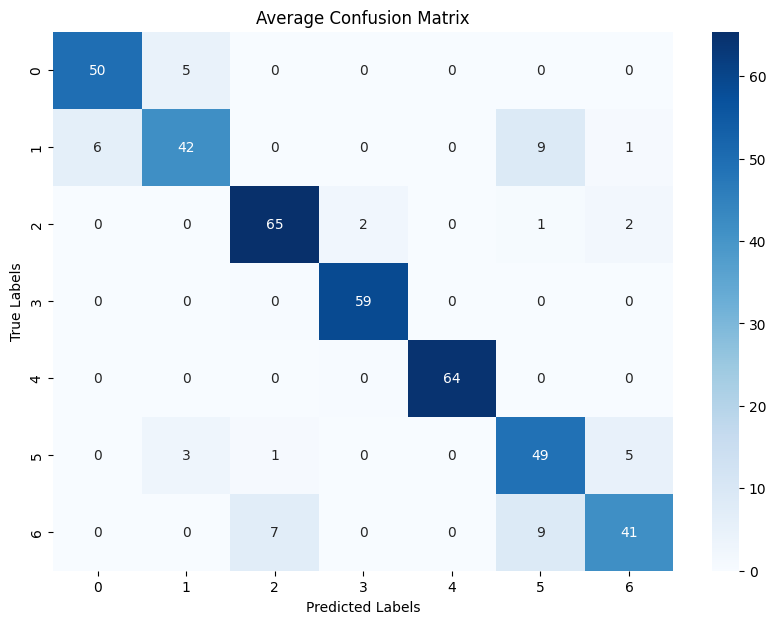
Η σύγκριση των δύο αρχιτεκτονικών αποκαλύπτει σημαντικές διαφορές όσον αφορά την απόδοση, τον χρόνο εκπαίδευσης και τη γενίκευση των μοντέλων.

Αρχικά, η αρχιτεκτονική με 2 κρυφά επίπεδα παρουσιάζει υψηλή ακρίβεια στο test set, π.χ., με τη χρήση **Sigmoid** και **Adam optimizer, 100 epochs,** φτάνει test accuracy **0.8768** με train accuracy 0.9057 και training time μόλις **19.30s**. Το training και validation loss παραμένουν χαμηλά και σταθερά, δείχνοντας **καλή ισορροπία και γενίκευση**. Επιπλέον, ο χρόνος εκπαίδευσης είναι σημαντικά χαμηλότερος σε σύγκριση με την πιο βαθιά αρχιτεκτονική, καθιστώντας το μοντέλο **ιδανικό για περιπτώσεις όπου η ταχύτητα εκπαίδευσης είναι κρίσιμη**.

Αντίθετα, η αρχιτεκτονική με 10 κρυφά επίπεδα μπορεί να επιτύχει **υψηλότερη train accuracy αλλά η διαφορά μεταξύ train και test accuracy δείχνει τάση υπερπροσαρμογής**, ιδιαίτερα σε περιπτώσεις όπου χρησιμοποιούνται πολλές εποχές ή μεγάλος αριθμός νευρώνων ανά επίπεδο. Ο **χρόνος εκπαίδευσης** είναι επίσης πολύ **μεγαλύτερος** (π.χ., 37.52s), λόγω της αυξημένης πολυπλοκότητας και του μεγαλύτερου αριθμού παραμέτρων. Επιπλέον, παρατηρούνται μεγαλύτερες διακυμάνσεις στο validation loss, γεγονός που υποδηλώνει **αστάθεια στη γενίκευση.**

Συνοψίζοντας, για τη συγκεκριμένη περίπτωση με 2111 δείγματα, η **αρχιτεκτονική με 2 κρυφά επίπεδα είναι η καλύτερη επιλογή**, **λόγω της ισορροπίας που προσφέρει μεταξύ ακρίβειας, χρόνου εκπαίδευσης και γενίκευσης**.

## Παραδείγματα Ορθής και Εσφαλμένης Κατηγοριοποίησης

Για το καλύτερο μοντέλο που προέκυψε από τον πειραματισμό, δηλαδή το **Sigmoid** και **Adam optimizer, 100 epochs με 100 νευρώνες ανά κρυφό επίπεδο,** έχουμε το εξής confusion matrix που δείχνει την ορθή και εσφαλμένη κατηγοριοποίηση:

Το μοντέλο παρουσιάζει καλή απόδοση στις περισσότερες κλάσεις, όπως φαίνεται από τις υψηλές τιμές των σωστών προβλέψεων στη διαγώνιο του confusion matrix. Για παράδειγμα, στην κλάση 0, κατά μέσο όρο 50 δείγματα ταξινομούνται σωστά, γεγονός που δείχνει την ικανότητα του μοντέλου να διακρίνει αυτήν την κατηγορία. Ωστόσο, παρατηρείται σύγχυση σε ορισμένες κλάσεις, όπως φαίνεται από τις μη μηδενικές τιμές εκτός της διαγωνίου. Ειδικότερα: Στην κλάση 6, 9 δείγματα έχουν ταξινομηθεί λανθασμένα ως κλάση 5 και 7 δείγματα ως κλάση 2 ενώ στην κλάση 1, 6 δείγματα έχουν ταξινομηθεί ως κλάση 0 και 9 δείγματα ως κλάση 5.

Αξιοσημείωτο είναι ότι οι κλάσεις 3(Obesity\_Type\_II) και 4 (Obesity\_Type\_III) φαίνεται να είναι οι καλύτερα διαχωρίσιμες, με μηδενική σύγχυση με άλλες κατηγορίες. Από την άλλη πλευρά, υπάρχει αυξημένη σύγχυση μεταξύ των κλάσεων 5(Overweight\_Level\_I) και 6(Overweight\_Level\_II), κάτι που μπορεί να οφείλεται σε παρόμοια χαρακτηριστικά μεταξύ αυτών.

### Αντιστοιχίες κλάσεων

* Insufficient Weight = 0
* Normal Weight = 1
* Obesity\_Type\_I = 2
* Obesity\_Type\_II = 3
* Obesity\_Type\_III = 4
* Overweight\_Level\_I = 5
* Overweight\_Level\_II = 6

## Σύγκριση με ΚΝΝ και NC

Αρχικά, είναι φανερό ότι όλα τα μοντέλα που εξετάστηκαν **ξεπερνάνε κατά πολύ την απόδοση του NC** κάτι που αναδεικνύει την υπεροχή των νευρωνικών δικτύων στην επίλυση σύνθετων προβλημάτων. Το **νευρωνικό δίκτυο με 2 κρυφά επίπεδα** υπερέχει, καθώς προσφέρει ισορροπία μεταξύ train και test accuracy (test accuracy: **0.8768**, train accuracy: 0.9057), με **χαμηλότερη τάση για υπερπροσαρμογή** από το KNN-1 και **καλύτερη απόδοση** από το KNN-3 (test accuracy: 0.8674). Επιπλέον, μπορεί να χρειάζεται περισσότερο χρόνο εκπαίδευσής (19.30s) αλλά αυτός είναι αποδεκτός. Είναι σημαντικό να σημειωθεί ότι για τα νευρωνικά δίκτυα χρησιμοποιήθηκε cross-validation, το οποίο παρέχει πιο αξιόπιστη εκτίμηση της απόδοσης σε σχέση με το KNN, όπου τα αποτελέσματα βασίζονται μόνο σε ένα split train και test. Συμπερασματικά, **το νευρωνικό δίκτυο αποδεικνύεται πιο αξιόπιστο και αποδοτικό** για το πρόβλημα που εξετάζεται.

# Δημιουργία και Εκπαίδευση SVM

Για τη δημιουργία του SVM, χρησιμοποιήθηκε οι βιβλιοθήκες **SVC και LinearSVC της sklearn.svm.** Για να συγκριθούν τα αποτελέσματα που θα προκύψουν με τα αποτελέσματα των κατηγοριοποιητών KNN και NC καθώς και με MLP, οι δοκιμές γίνονται με **20 χαρακτηριστικά** (one-hot-encoding) **χωρίς κανονικοποίηση** και με χρήση cross validation με k=5 folds. Τέλος, δεν χρησιμοποιείται one-hot-encoding της μεταβλητής στόχου.

## SVC μοντέλο

Για την εύρεση του καλύτερου SVC μοντέλου, εξετάστηκαν τα αποτελέσματα με βάση τις διαφορετικές τιμές των εξής παραμέτρων :

* Kernel (linear, rbf, poly)
* Decision function (one-vs-all, one-vs-one)
* C (0.001, 0.01, 0.1, 1 ,2 ,6 ,10)
* Gamma ( 0.001, 0.01, 0.05(default), 0.1, 1, 10) για rbf και poly kernel
* Degree (default, 2, 3, 4) για rbf και poly kernel

Αρχικά, εξετάστηκε η επίδοση του **πυρήνα** με βάση διαφορετικές τιμές του **C και degree** (για τον poly kernel), διατηρώντας τις υπόλοιπες παραμέτρους με τις **default τιμές** τους (**gamma** = 1/(n\_features \* X\_var()), **decision function** = one vs rest) δηλαδή, για κάθε κλάση, το SVM εκπαιδεύει έναν ταξινομητή που διαχωρίζει αυτήν την κλάση από όλες τις υπόλοιπες.. Στα αποτελέσματα που φαίνονται παρακάτω, περιλαμβάνεται επίσης ο αριθμός των **support vectors**, ο οποίος υποδεικνύει πόσα δεδομένα παίζουν ενεργό ρόλο στην απόφαση του μοντέλου. Αυτό είναι σημαντικό, διότι ένας **μικρότερος αριθμός support vectors** συνήθως υποδηλώνει ένα **πιο αποδοτικό και γενικεύσιμο μοντέλ**ο, ενώ ένας **μεγαλύτερος αριθμός μπορεί να είναι ένδειξη υπερπροσαρμογής**.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kernel** | **Decision Function** | **C** | **Gamma** | **Degree** | **Support Vectors** | **Training Time** | **Train Accuracy (AVG)** | **Test Accuracy (AVG)** |
| Linear | Ovr | 0.001 | - | - | [122.4 229.2 270.8 191.2 180.8 232.0 230.4] | 0.16s | 0.6292 | 0.6182 |
| Linear | Ovr | 0.01 | - | - | [101.2 211.2 249. 145.4 118.6 221.4 228.2] | 0.14s | 0.7623 | 0.7518 |
| Linear | Ovr | 0.1 | - | - | [ 80.8 172. 189.6 68.2 35.8 185.6 206.2] | 0.25s | 0.8452 | 0.8247 |
| Linear | Ovr | 1(default) | - | - | [ 68.0 138.2 128.8 34.4 11.0 151.8 167.6] | 0.29s | 0.9019 | 0.8816 |
| Linear | Ovr | 2 | - | - | [ 63.4 129.2 112.6 30.6 9.4 144.6 156.2] | 0.99s | 0.9254 | 0.9034 |
| **Linear** | **Ovr** | **6** | **-** | **-** | **[ 51.2 106.2 87.4 27.0 9.4 129.4 134.2]** | **1.68s** | **0.9569** | **0.9346** |
| Linear | Ovr | 10 | - | - | [ 45.0 94.2 76.6 26.0 9.4 117.2 120.6] | 1.84s | 0.9666 | 0.9479 |
| Linear | Ovr | 20 | - | - | [ 37.2 79. 64.8 23.6 9.4 101.6 101.4] | 2.27s | 0.9716 | 0.9545 |
| Rbf | Ovr | 0.001 | Default(1/20) | - | [217.6 229.6 275.2 237.6 259.2 232.0 232.0 ] | 0.32s | 0.1663 | 0.1663 |
| Rbf | Ovr | 0.01 | Default(1/20) | - | [217.6 229.6 276.4 237.6 259.2 232.0 232.0 ] | 0.37s | 0.1819 | 0.1791 |
| **Rbf** | **Ovr** | **0.1** | **Default(1/20)** | **-** | **[162.0 226.0 250.8 176.8 159.6 220.2 228.2]** | **0.31s** | **0.7917** | **0.7646** |
| Rbf | Ovr | 1 | Default(1/20) | - | [ 96.2 204.8 169.4 89.0 56.4 163. 185.6] | 0.24s | 0.9443 | 0.8906 |
| Rbf | Ovr | 2 | Default(1/20) | - | [ 87. 202.4 154.6 79.8 49.4 151.2 175.6] | 0.39s | 0.9732 | 0.9090 |
| Rbf | Ovr | 6 | Default(1/20) | - | [ 79.8 196.6 145.6 77.0 46.8 139.6 159.2] | 0.22s | 0.9927 | 0.9147 |
| Rbf | Ovr | 10 | Default(1/20) | - | [ 77.2 193.2 145.8 76.8 46.4 136.4 154.6] | 0.20s | 0.9970 | 0.9133 |
| Rbf | Ovr | 20 | Default(1/20) | - | [ 73.8 192.8 146.4 76.8 46.2 132.0 154.0 ] | 0.20s | 0.9993 | 0.9161 |
| Poly | Ovr | 0.001 | Default(1/20) | 2 | [ 93.8 196.8 211.0 72.2 47.8 207.2 223.0 ] | 0.13s | 0.8157 | 0.7973 |
| Poly | Ovr | 0.001 | Default(1/20) | 3 | [ 57.4 109.0 56.8 19.4 8.0 111.0 96.4] | 4.55s | 0.9735 | 0.9427 |
| Poly | Ovr | 0.001 | Default(1/20) | 4 | [28.8 65.4 42.6 17.8 8.4 61.8 54.8] | 19.05s | 1.000 | 0.9460 |
| Poly | Ovr | 0.01 | Default(1/20) | 2 | [ 76. 157.4 145.2 34.6 11.6 168.4 178.6] | 0.22s | 0.8748 | 0.8541 |
| Poly | Ovr | 0.01 | Default(1/20) | 3 | [35.2 69.8 43.0 17.4 8.0 79.2 60.6] | 13.47s | 0.9937 | 0.9507 |
| Poly | Ovr | 0.01 | Default(1/20) | 4 | [28.8 65.4 42.6 17.8 8.4 61.8 54.8] | 19.39s | 1.000 | 0.9460 |
| Poly | Ovr | 0.1 | Default(1/20) | 2 | [ 66.2 128. 90.8 24.2 9. 140.8 138.6] | 1.80s | 0.9417 | 0.9143 |
| Poly | Ovr | 0.1 | Default(1/20) | 3 | [26.6 61.0 43.4 17.4 8.0 59.8 52.0 ] | 15.95s | 1.000 | 0.9498 |
| Poly | Ovr | 0.1 | Default(1/20) | 4 | [28.8 65.4 42.6 17.8 8.4 61.8 54.8] | 20.77s | 1.000 | 0.9460 |
| Poly | Ovr | 1 | Default(1/20) | 2 | [39.2 82.4 52.6 19.2 9. 94.2 82.6] | 5.61s | 0.9835 | 0.9559 |
| Poly | Ovr | 1 | Default(1/20) | 3 | [26.6 61. 43.4 17.4 8. 59.8 52.2] | 20.10s | 1.000 | 0.9484 |
| Poly | Ovr | 1 | Default(1/20) | 4 | [28.8 65.4 42.6 17.8 8.4 61.8 54.8] | 21.27s | 1.000 | 0.9460 |
| **Poly** | **Ovr** | **2** | **Default(1/20)** | **2** | **[34.2 69.2 45.2 17.6 9.0 85.0 69.4]** | **5.96s** | **0.9879** | **0.9583** |
| Poly | Ovr | 6 | Default(1/20) | 2 | [31.0 57.2 41.6 16.8 9.0 73.2 59.2] | 8.64s | 0.9943 | 0.9569 |
| Poly | Ovr | 10 | Default(1/20) | 2 | [26.8 54.2 40.0 16.8 9.0 67.6 55.2] | 9.83s | 0.9959 | 0.9555 |
| Poly | Ovr | 20 | Default(1/20) | 2 | [24.8 51.6 40.2 16.8 9.0 59.2 54.0 ] | 10.62s | 0.9975 | 0.9545 |

Γενικά, παρατηρούνται πολύ ταχύτεροι χρόνοι εκπαίδευσης με υψηλότερη ακρίβεια για σωστή επιλογή παραμέτρων, σε σχέση με τους κατηγοριοποιητές και τα νευρωνικά δίκτυα που εξετάστηκαν. Αυτό ήταν αναμενόμενο εφόσον τα SVM τείνουν να αποδίδουν καλύτερα από τα νερυωνικά σε μικρού μεγέθους datasets όπως το συγκεκριμένο. Πιο συγκεκριμένα για κάθε πυρήνα προκύπτουν τα εξής :

* Linear: Για χαμηλές τιμές **C** (**0.001 και 0.01**), το μοντέλο δίνει προτεραιότητα στη μεγιστοποίηση του margin και έτσι δέχεται μεγαλύτερο αριθμό υποστηρικτικών διανυσμάτων. Αυτό σημαίνει ότι το μοντέλο είναι λιγότερο ευαίσθητο στο θόρυβο των δεδομένων και έχει **μικρότερη** **πιθανότητα υπερπροσαρμογής**. Ωστόσο, **η ακρίβεια του είναι χαμηλή** και δεν αρκεί εφόσον δεν προσαρμόζεται στα δεδομένα. Με την αύξηση του C αυξάνεται ο χρόνος εκπαίδευσης αλλά και η ακρίβεια του μοντέλου αισθητά ειδικά από 0.01 σε 0.1. Παρατηρείται μάλιστα, ότι για C=2, καταφέρνει να ξεπεράσει με πολύ μικρό χρόνο εκπαίδευσης, τόσο την επίδοση των απλών κατηγοριοποιητών όσο και τα μοντέλα νευρωνικών δικτύων που εξετάστηκαν. Για υψηλές τιμές του **C (6, 10, 20)**, το μοντέλο γίνεται πιο ευαίσθητο στα δεδομένα εκπαίδευσης, καθώς μειώνει το margin και προσπαθεί να κατηγοριοποιήσει σωστά περισσότερα δείγματα. Αυτό έχει ως αποτέλεσμα να πετυχαίνει **υψηλότερη ακρίβεια** (π.χ. 0.9545 για C=20), όμως αυτό είναι κίνδυνος **υπερπροσαρμογής** και όχι καλής γενίκευσης σε καινούρια δείγματα. Παράλληλα, ο αριθμός των υποστηρικτικών διανυσμάτων μειώνεται, καθώς το μοντέλο βασίζεται σε πιο συγκεκριμένα δείγματα για να διαμορφώσει την υπερεπιφάνεια απόφασης. Παρόλαυτα δεν εμφανίζεται έντονη υπερπροσαρμογή ακόμα και για μεγάλες τιμές του C. Συμπερασματικά, **για C=6 παρατηρείται καλή ισορροπία test και train accuracy με test accuracy = 0.9346 και χρόνο εκπαίδευσης 1s**.
* Rbf: Ο χρόνος εκπαίδευσης είναι χαμηλότερος από αυτή του linear kernel αλλά δεν επιτυγχάνεται τόση υψηλή ακρίβεια. Παρατηρείται ότι το C επηρεάζει άμεσα την επίδοση και τη πολυπλοκότητα του μοντέλου . Για χαμηλές τιμές του **C** (**0.001 και 0.01**), το μοντέλο έχει training και το test accuracy πολύ χαμηλά (π.χ., 0.1663 ). Αυτό δείχνει ότι το μοντέλο **υποπροσαρμόζεται**. Για **C >= 1**, το **training accuracy** αγγίζει σχεδόν το 100%, ενώ το **test accuracy** βελτιώνεται, αλλά όχι με τον ίδιο ρυθμό. Αυτό δείχνει **υπερπροσαρμογή του μοντέλου**. Έτσι το μοντέλο είναι ευαίσθητο στο θόρυβο και λιγότερο ικανό να γενικεύει. Συμπερασματικά, **για C=0.1 παρατηρείται καλή ισορροπία test και train accuracy με test accuracy = 0.7646 και χρόνο εκπαίδευσης 0.31s**.
* Poly: Για χαμηλές τιμές του **C**, όπως **C=0.001**, και χαμηλό βαθμό (**degree=2**), το μοντέλο καταφέρνει ήδη να πετύχει αξιοσημείωτη απόδοση με **test accuracy 0.7973**, ξεπερνώντας τόσο τον **rbf** όσο και τον **linear** kernel. Ο χρόνος εκπαίδευσης είναι εξαιρετικά μικρός (**0.13s**), καθιστώντας το μοντέλο αποδοτικό για αρχική ανάλυση δεδομένων. Παρατηρείται ότι όσο αυξάνεται το C η ακρίβεια του μοντέλου βελτιώνεται και ο χρόνος εκαπαίδευσης αυξάνεται. Η αύξηση του βαθμού του πολυωνύμου προσθέτει πολυπλοκότητα στο μοντέλο, με αποτέλεσμα την καλύτερη προσαρμογή στα δεδομένα εκπαίδευσης. Για **degree=3** ή **4**, το **train accuracy** φτάνει το **100%**, όμως το **test accuracy** σταθεροποιείται ή μειώνεται ελαφρώς (π.χ., **94.60%**). Αυτό αποτελεί σαφή ένδειξη **υπερπροσαρμογής**, καθώς το μοντέλο αποτυγχάνει να γενικεύσει σωστά. Η υπερπροσαρμογή αυτή είναι αναμενόμενη, καθώς οι πολυωνυμικές συναρτήσεις υψηλού βαθμού είναι πιο ευέλικτες και μπορούν να προσαρμοστούν τέλεια ακόμα και σε θορυβώδη δεδομένα, αγκαλιάζοντας όλες τις λεπτομέρειες των σημείων εκπαίδευσης. Επομένως για τις μεγαλύτερες τιμές του C κρίθηκε περιττός ο πειραματισμός με αυτούς του βαθμούς πολυωνύμου. **Η υψηλότερη απόδοση** παρατηρείται για **C=2** και **degree=2**, με test accuracy **0.9583** και χρόνο εκπαίδευσης **5.96s χωρίς εμφάνιση υπερπροσαρμογής** .Συμπερασματικά, η υπερβολική αύξηση του βαθμού ή του **C** δεν προσφέρει ουσιαστικά πλεονεκτήματα, ενώ μπορεί να οδηγήσει σε αυξημένο χρόνο εκπαίδευσης και απώλεια της ικανότητας γενίκευσης.

Τελικά, ο **Polynomial kernel** προσφέρει την υψηλότερη ακρίβεια (**95.83%**) για **C=2** και **degree=2**, αλλά με μεγαλύτερο χρόνο εκπαίδευσης συγκριτικά με τους άλλους πυρήνες (**5.96s**). Ο **Linear kernel** είναι ιδανικός για περιπτώσεις όπου η ταχύτητα εκπαίδευσης είναι κρίσιμη, ενώ ταυτόχρονα διατηρεί καλή γενίκευση (**C=6**, **test accuracy 93.46%**) σε **1s**. Ο **RBF kernel** είναι ταχύτερος, αλλά λιγότερο αποδοτικός εμφανίζοντας ευαισθησία στο θόρυβο.

Στη συνέχεια, εξετάστηκαν οι ίδιοι παράμετροι αλλά αυτή τη φορά με **decision function one vs one** δηλαδή Για **K κλάσεις, εκπαιδεύονται (Κ(Κ-1))/2μοντέλα.** Για κάθε ζεύγος κλάσεων, το SVM εκπαιδεύει έναν ταξινομητή που διαχωρίζει αυτές τις δύο κλάσεις.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kernel** | **Decision Function** | **C** | **Gamma** | **Degree** | **Support Vectors** | **Training Time** | **Train Accuracy (AVG)** | **Test Accuracy (AVG)** |
| Linear | Ovo | 0.001 | - | - | [122.4 229.2 270.8 191.2 180.8 232.0 230.4] | 0.05s | 0.6292 | 0.6182 |
| Linear | Ovo | 0.01 | - | - | [101.2 211.2 249.0 145.4 118.6 221.4 228.2] | 0.10s | 0.7623 | 0.7518 |
| Linear | Ovo | 0.1 | - | - | [ 80.8 172.0 189.6 68.2 35.8 185.6 206.2] | 0.39s | 0.8452 | 0.8247 |
| Linear | Ovo | 1(default) | - | - | [ 68.01 138.2 128.8 34.4 11.0 151.8 167.6] | 0.32s | 0.9019 | 0.8816 |
| Linear | Ovo | 2 | - | - | [ 63.4 129.2 112.6 30.6 9.4 144.6 156.2] | 0.51s | 0.9254 | 0.9034 |
| Linear | Ovo | 6 | - | - | [ 51.2 106.2 87.4 27.0 9.4 129.4 134.2] | 1.53s | 0.9569 | 0.9346 |
| Linear | Ovo | 10 | - | - | [ 45. 94.2 76.6 26. 9.4 117.2 120.6] | 2.05s | 0.9666 | 0.9479 |
| Linear | Ovo | 20 | - | - | [ 37.2 79. 64.8 23.6 9.4 101.6 101.4] | 3.29s | 0.9716 | 0.9545 |
| Rbf | Ovo | 0.001 | Default(1/20) | - | [217.6 229.6 275.2 237.6 259.2 232. 232.0 ] | 0.54s | 0.1663 | 0.1663 |
| Rbf | Ovo | 0.01 | Default(1/20) | - | [217.6 229.6 276.4 237.6 259.2 232. 232.0 ] | 0.72s | 0.1819 | 0.1791 |
| **Rbf** | **Ovo** | **0.1** | **Default(1/20)** | **-** | **[162. 226. 250.8 176.8 159.6 220.2 228.2]** | **0.26s** | **0.7917** | **0.7646** |
| Rbf | Ovo | 1 | Default(1/20) | - | [ 96.2 204.8 169.4 89. 56.4 163. 185.6] | 0.22s | 0.9443 | 0.8906 |
| Rbf | Ovo | 2 | Default(1/20) | - | [ 87. 202.4 154.6 79.8 49.4 151.2 175.6] | 0.30s | 0.9732 | 0.9090 |
| Rbf | Ovo | 6 | Default(1/20) | - | [ 77.2 193.2 145.8 76.8 46.4 136.4 154.6] | 0.18s | 0.9927 | 0.9147 |
| Rbf | Ovo | 10 | Default(1/20) | - | [ 73.8 192.8 146.4 76.8 46.2 132. 154.0 ] | 0.24s | 0.9970 | 0.9133 |
| Rbf | Ovo | 20 | Default(1/20) | - | [ 93.8 196.8 211.0 72.2 47.8 207.2 223.0 ] | 0.23s | 0.9993 | 0.9161 |
| Poly | Ovo | 0.001 | Default(1/20) | 2 | [ 57.4 109.0 56.8 19.4 8.0 111.0 96.4] | 0.26s | 0.8157 | 0.7973 |
| Poly | Ovo | 0.001 | Default(1/20) | 3 | [28.8 65.4 42.6 17.8 8.4 61.8 54.8] | 3.77s | 0.9735 | 0.9427 |
| Poly | Ovo | 0.001 | Default(1/20) | 4 | [ 76. 157.4 145.2 34.6 11.6 168.4 178.6] | 20.43s | 1.000 | 0.9460 |
| Poly | Ovo | 0.01 | Default(1/20) | 2 | [35.2 69.8 43. 17.4 8.0 79.2 60.6] | 0.22s | 0.8748 | 0.8541 |
| Poly | Ovo | 0.01 | Default(1/20) | 3 | [ 66.2 128.0 90.8 24.2 9.0 140.8 138.6] | 10.02s | 0.9937 | 0.9507 |
| Poly | Ovo | 0.1 | Default(1/20) | 2 | [39.2 82.4 52.6 19.2 9.0 94.2 82.6] | 1.06s | 0.9417 | 0.9143 |
| Poly | Ovo | 1 | Default(1/20) | 2 | [34.2 69.2 45.2 17.6 9.0 85. 69.4] | 4.90s | 0.9835 | 0.9559 |
| **Poly** | **Ovo** | **2** | **Default(1/20)** | **2** | **[31.0 57.2 41.6 16.8 9.0 73.2 59.2]** | **5.57s** | **0.9879** | **0.9583** |
| Poly | Ovo | 6 | Default(1/20) | 2 | [26.8 54.2 40. 16.8 9.0 67.6 55.2] | 8.13s | 0.9943 | 0.9569 |
| Poly | Ovo | 10 | Default(1/20) | 2 | [24.8 51.6 40.2 16.8 9.0 59.2 54.0 ] | 9.34s | 0.9959 | 0.9555 |
| Poly | Ovo | 20 | Default(1/20) | 2 | [24.8 51.6 40.2 16.8 9.0 59.2 54.0 ] | 10.62s | 0.9975 | 0.9545 |

Παρατηρείται ότι η χρήση one vs one αποφέρει ακριβώς τα ίδια αποτελέσματα με τον one vs all με ελαφρώς **μεγαλύτερους χρόνους εκπαίδευσης**. Είναι λογικό ο χρόνος εκπαίδευσης να είναι μεγαλύτερος εφόσον η **πολυπλοκότητα του μοντέλου αυξάνεται**. Παρόλαυτα το γεγονός ότι πετυχαίνουν ίδια ακρίβεια δείχνει ότι το dataset είναι καλά διαχωρίσιμο, με χαμηλό θόρυβο και κατάλληλες παραμέτρους. **Με το one vs one, κάθε ταξινομητής βλέπει μόνο τα δεδομένα δύο κλάσεων, επομένως ο αριθμός των υποστηρικτικών διανυσμάτων ανά ταξινομητή είναι μικρότερος**. Ωστόσο, ο συνολικός αριθμός SVs για όλα τα μοντέλα μπορεί να είναι μεγαλύτερος από το OvR. Συμπερασματικά, στην περίπτωση αυτή, επιλέγεται η **one vs all** για αποδοτικότερη εκπαίδευση και πρόβλεψη, εφόσον έχει **λιγότερη υπολογιστική πολυπλοκότητα και ο αριθμός των κλάσεων είναι μικρός**.

Στη συνέχεια, επιλέχθηκαν τα **καλύτερα μοντέλα** που προέκυψαν από τον παραπάνω πειραματισμό, ώστε να εξεταστούν ως προς τις διαφορετικές τιμές **gamma** (**Rbf και poly**).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kernel** | **Decision Function** | **C** | **Gamma** | **Degree** | **Support Vectors** | **Training Time** | **Train Accuracy (AVG)** | **Test Accuracy (AVG)** |
| Rbf | Ovr | 0.1 | 0.001 | - | [158.8 229.6 280.8 220.8 227. 232. 232. ] | 0.09s | 0.5944 | 0.5680 |
| Rbf | Ovr | 0.1 | 0.01 | - | [135.6 227.4 258.6 150. 166.8 231.8 229. ] | 0.26s | 0.6920 | 0.6798 |
| Rbf | Ovr | 0.1 | Default(1/20) | - | [162. 226. 250.8 176.8 159.6 220.2 228.2] | 0.31s | 0.7917 | 0.7646 |
| Rbf | Ovr | 0.1 | 0.08 | - | [178.2 228.8 256.4 189.8 161.8 220.2 231.2] | 0.28s | 0.8194 | 0.7759 |
| Rbf | Ovr | 0.1 | 0.1 | - | [186.6 229.2 260.4 196.6 163.6 221.4 231.8] | 0.38s | 0.8211 | 0.7669 |
| Rbf | Ovr | 0.1 | 1 | - | [217.6 229.6 267. 237.6 226.6 232. 232. ] | 0.56s | 0.2755 | 0.2610 |
| Rbf | Ovr | 0.1 | 10 | - | [217.6 229.6 279. 237.6 255. 232. 232. ] | 0.60s | 0.2022 | 0.1985 |
| Rbf | Ovr | 1 | 0.001 | - | [117.8 229.4 254.2 159. 186.4 232. 231.8] | 0.22s | 0.6847 | 0.6755 |
| **Rbf** | **Ovr** | **1** | **0.01** | **-** | **[102.8 208.4 209. 90.2 83.8 199.8 210.6]** | **0.16s** | **0.8329** | **0.8048** |
| Rbf | Ovr | 1 | Default(1/20) | - | [ 96.2 204.8 169.4 89. 56.4 163. 185.6] | 0.24s | 0.9443 | 0.8906 |
| Rbf | Ovr | 1 | 0.08 | - | [101.2 215. 177.4 91.6 59.6 162.6 185.4] | 0.25s | 0.9685 | 0.9015 |
| Rbf | Ovr | 1 | 0.1 | - | [107. 220.4 180.2 96.6 61. 166.6 183.2] | 0.20s | 0.9777 | 0.9095 |
| Rbf | Ovr | 1 | 1 | - | [181.8 227. 255.6 186.6 119.4 205.6 219.4] | 0.85s | 1.000 | 0.7688 |
| Rbf | Ovr | 1 | 10 | - | [214.8 229.6 279. 236.2 193. 222.4 232. ] | 0.72s | 1.000 | 0.3695 |
| Poly | Ovr | 2 | 0.001 | 2 | [ 96.6 198.8 216.2 78. 53.8 210.4 225.4] | 0.12s | 0.8080 | 0.7864 |
| Poly | Ovr | 2 | 0.01 | 2 | [ 66.6 132.4 96.2 25. 8.8 143.6 142.4] | 1.02s | 0.9338 | 0.9076 |
| **Poly** | **Ovr** | **2** | **Default(1/20)** | **2** | **[34.2 69.2 45.2 17.6 9. 85. 69.4]** | **5.96s** | **0.9879** | **0.9583** |
| Poly | Ovr | 2 | 0.08 | 2 | [31.8 58.6 42.2 16.8 8.8 72.6 59.6] | 10.01s | 0.9929 | 0.9578 |
| Poly | Ovr | 2 | 0.1 | 2 | [28.6 54. 40.8 16.8 9. 70.2 57. ] | 10.74s | 0.9951 | 0.9550 |
| Poly | Ovr | 2 | 1 | 2 | [23.4 52.2 40.2 17.2 9. 52.4 48.4] | 23.52s | 0.9960 | 0.9507 |
| Poly | Ovr | 2 | 10 | 2 | [22.8 52. 39.4 16.8 9. 54.8 46.6] | 51.53s | 0.9953 | 0.9531 |

Παρατηρείται ότι τo gamma, διαδραματίζει βασικό ρόλο στον τρόπο με τον οποίο το SVM διαμορφώνει την απόφαση και σχετίζεται με την επιρροή των δειγμάτων εκπαίδευσης στον χώρο χαρακτηριστικών. Επηρεάζει τόσο την αποδοτικότητα όσο και τη πολυπλοκότητα του μοντέλου. Πιο συγκεκριμένα, για τον rbf πυρήνα, το gamma ελέγχει το εύρος επιρροής ενός σημείου, επηρεάζοντας την τοπική προσαρμογή του μοντέλου. Η αλλαγές στο gamma , οδήγησαν σε καλύτερο accuracy χωρίς να επηρεάζεται σημαντικά ο χρόνος εκπαίδευσης. Για μικρές τιμές του **gamma** (0.001), το μοντέλο δεν υπερπροσαρμόζεται, εμφανίζοντας όμως χαμηλή απόδοση τόσο στο training όσο και στο test set εφόσον οι καμπύλες διαχωρισμού είναι πιο ομαλές. Αντίθετα, όταν το **gamma** αυξάνεται πολύ (0.08 έως 10), το μοντέλο **υπερπροσαρμόζεται υπερβολικά** στα δεδομένα εκπαίδευσης, **με το training accuracy να φτάνει το 100%**, αλλά το **test accuracy να μειώνεται δραματικά**, δείχνοντας **έλλειψη γενίκευσης**.

Για τον poly πυρήνα, το gamma ελέγχει τη "βαρύτητα" της σχέσης μεταξύ των δειγμάτων, καθορίζοντας πόσο απότομες είναι οι πολυωνυμικές συναρτήσεις. Φαίνεται να αποδίδει καλύτερα από τον rbf. Παρόλαυτα , οι διαφορετικές τιμές του gamma επηρεάζουν αρκετά το χρόνο εκπαίδευσης του μοντέλου ειδικά όταν αυτό είναι μεγάλο (π.χ. 10). Η εύρεση του κατάλληλου gamma οδήγησε τόσο σε καλύτερο test accuracy όσο και σε μείωση του αριθμού των υποστηρικτικών διανυσμάτων.

Συμπερασματικά, ο πυρήνας του SVC που αποφέρει καλύτερα αποτελέσματα είναι ο **Polynomial kernel** με **C=2**, **degree=2**, **default** **gamma, decision function one vs all** με **test accuracy = 0.9583** και training time = 5.96s.

## LinearSVC μοντέλο

Το μοντέλο LinearSVC βασίζεται αποκλειστικά και μόνο στο γραμμικό διαχωρισμό των κλάσεων. Για την εύρεση του καλύτερου μοντέλου, διερευνήθηκαν τα αποτελέσματα με βάση τις διαφορετικές τιμές των εξής παραμέτρων :

* C (0.001, 0.01, 0.1, 1 ,2 ,6 ,10)
* Max\_iter ( Μέγιστος αριθμός επαναλήψεων του αλγορίθμου) (5, 10, 50)
* Penalty (Τύπος κανονικοποίησης) (L1, L2)

Οι παρακάτω παράμετροι κρατήθηκαν ως εξής με βάση το συγκεκριμένο πρόβλημα :

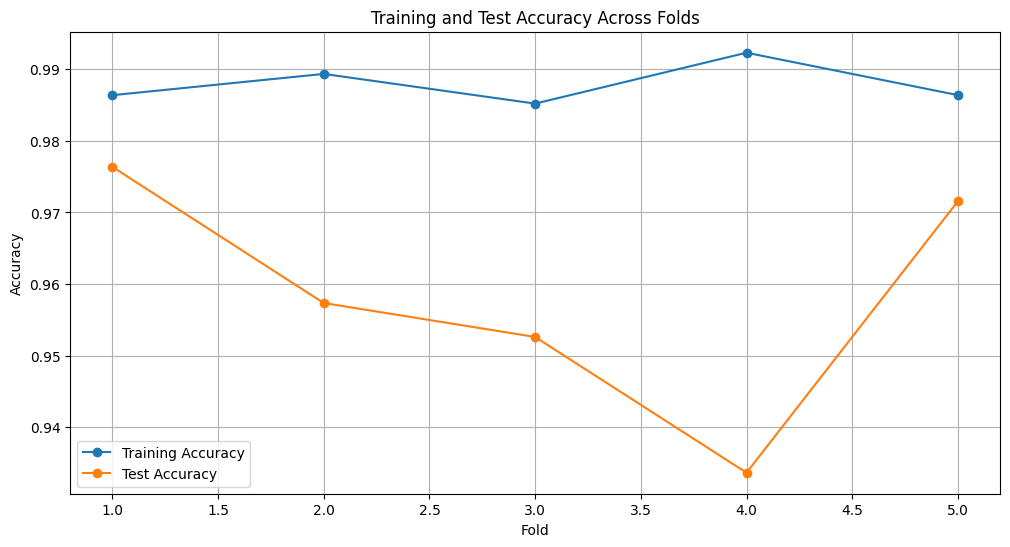
* Loss = squared\_hinge η οποία δίνει περισσότερη έμφαση σε παραβάσεις που απέχουν περισσότερο.
* Dual = False εφόσον το πλήθος των δειγμάτων είναι μεγαλύτερο από το πλήθος των χαρακτηριστικών.
* Class\_weight = None εφόσον δεν υπάρχουν κλάσεις με ανισορροπία δεδομένων.

Επομένως εξετάστηκε η επίδοση του μοντέλου για διαφορετικές τιμές του **C** σε συνδυασμό με το **πλήθος των επαναλήψεων,** και το τύπο **κανονικοποίησης**. Παρακάτω φαίνονται τα αποτελέσματα :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **C** | **Max\_iter** | **Penalty** | **Training Time** | **Train Accuracy (AVG)** | **Test Accuracy (AVG)** |
| 0.001 | 5 | L2 | 0.02s | 0.4063 | 0.4036 |
| 0.01 | 5 | L2 | 0.02s | 0.4035 | 0.3984 |
| 0.1 | 5 | L2 | 0.04s | 0.4040 | 0.3993 |
| 1 | 5 | L2 | 0.02s | 0.4038 | 0.3993 |
| 2 | 5 | L2 | 0.03s | 0.4038 | 0.3993 |
| 6 | 5 | L2 | 0.03s | 0.4038 | 0.3993 |
| 10 | 5 | L2 | 0.03s | 0.4038 | 0.3993 |
| 0.001 | 10 | L2 | 0.03s | 0.5985 | 0.5836 |
| 0.01 | 10 | L2 | 0.03s | 0.6592 | 0.6499 |
| 0.1 | 10 | L2 | 0.07s | 0.6680 | 0.6575 |
| 1 | 10 | L2 | 0.06s | 0.6675 | 0.6580 |
| 2 | 10 | L2 | 0.03s | 0.6678 | 0.6570 |
| 6 | 10 | L2 | 0.03s | 0.6678 | 0.6575 |
| 10 | 10 | L2 | 0.04s | 0.6679 | 0.6575 |
| 0.001 | 50 | L2 | 0.06s | 0.5995 | 0.5850 |
| 0.01 | 50 | L2 | 0.04s | 0.6850 | 0.6760 |
| 0.1 | 50 | L2 | 0.14s | 0.7366 | 0.7186 |
| 1 | 50 | L2 | 0.15s | 0.7692 | 0.7508 |
| 2 | 50 | L2 | 0.14s | 0.7730 | 0.7546 |
| 6 | 50 | L2 | 0.10s | 0.7783 | 0.7608 |
| **10** | **50** | **L2** | **0.14s** | **0.7798** | **0.7641** |

Μια πρώτη παρατήρηση που γίνεται είναι ότι το test accuracy που επιτυγχάνει το μοντέλο δεν ξεπερνάει το 0.76 γεγονός που υποδεικνύει ότι **τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα**. Έτσι **δεν ξεπερνάει την απόδοση του SVC με poly kernel**. Επίσης, αν και έχει πολύ μικρότερο χρόνο εκπαίδευσης και καθόλου εμφάνιση υπερπροσαρμογής **δεν καταφέρνει να ξεπεράσει ούτε τον SVC με linear kernel**. Επιλέχθηκε max\_iter =50 εφόσον πάνω από αυτό δεν παρατηρείται καμία αλλαγή στην ακρίβεια του μοντέλου αφού καταφέρνει να συγκλίνει γρηγορότερα. Τέλος, με penalty L1 λαμβάνονται τα ίδια αποτελέσματα γεγονός που οφείλεται στην ομοιομορφία των δεδομένων και της χαμηλής πολυπλοκότητας του dataset.

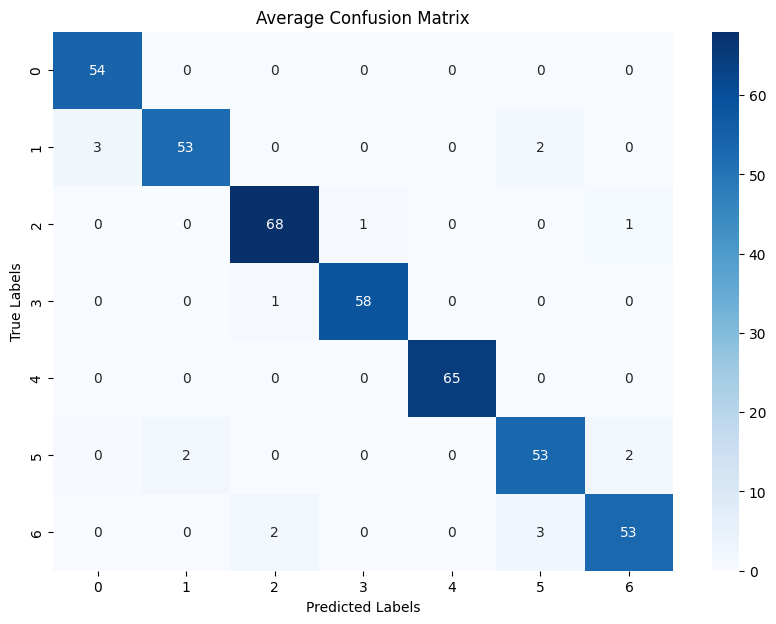
## Συμπεράσματα

Το καλύτερο SVM μοντέλο που προέκυψε από την ανάλυση είναι το **SVC με Polynomial kernel**, χρησιμοποιώντας τις παραμέτρους **C=2**, **degree=2**, **default gamma** και **decision function one-vs-all**. Το μοντέλο πέτυχε εξαιρετική ακρίβεια στο test set με **test accuracy = 0.9583** και χρόνο εκπαίδευσης μόλις **5.96s**, συνδυάζοντας υψηλή απόδοση και αποδοτική χρήση υπολογιστικών πόρων. Παρακάτω παρουσιάζονται οι καμπύλες ακρίβειας και το confusion matrix του μοντέλου.

Από το γράφημα παρατηρούμε ότι το μοντέλο παρουσιάζει υψηλή ακρίβεια στο training set (0.99) και ελαφρώς χαμηλότερη στο test set (0.96-0.97), κάτι που υποδεικνύει καλή γενίκευση με μικρή πιθανότητα overfitting. Υπάρχουν διακυμάνσεις στην απόδοση του test set μεταξύ των folds, με τη μεγαλύτερη πτώση να παρατηρείται στο 4ο fold, πιθανώς λόγω διαφορετικής κατανομής ή θορύβου στα δεδομένα. Το "gap" μεταξύ training και test accuracy παραμένει μικρό και σταθερό, δείχνοντας ότι το μοντέλο δεν υπερπροσαρμόζεται.

Στο κώδικα έχω κρατήσει τις τιμές των παραμέτρων για το καλύτερο SVC μοντέλο.

### Παραδείγματα Ορθής και Εσφαλμένης Κατηγοριοποίησης

Φαίνεται πως καταφέρνει να κατηγοριοποιήσει με μεγάλη επιτυχία όλες τις κλάσεις με μικρές απώλειας στη κλάση 1 και τη κλάση 6 (Normal\_Weight και Overweight\_Level\_II αντίστοιχα).

### Σύγκριση με ΚΝΝ και NC

Τα SVM αποδεικνύονται ανώτερα σε απόδοση σε σύγκριση τόσο με τον NC όσο και με τον KNN, όταν ρυθμίζονται κατάλληλα οι παράμετροί τους. Επιτυγχάνουν υψηλότερη ακρίβεια χωρίς να εμφανίζουν υπερπροσαρμογή, ενώ παράλληλα διατηρούν τον χρόνο εκπαίδευσης σε χαμηλά επίπεδα. Αν και ο χρόνος εκπαίδευσης του SVM είναι ελαφρώς μεγαλύτερος από αυτόν των KNN και NC, το πλεονέκτημα στην απόδοση είναι σημαντικό. Συγκεκριμένα, το καλύτερο **SVM** πέτυχε **test accuracy = 0.9583**, υπερβαίνοντας κατά πολύ την ακρίβεια του KNN με k=1 **test accuracy = 0.8899** και του NC **test accuracy = 0.5443**. Αξίζει να σημειωθεί ότι, σε αντίθεση με τους απλούς κατηγοριοποιητές (NC και KNN), στα SVM εφαρμόστηκε **cross-validation**, ενισχύοντας την αξιοπιστία και τη γενίκευση των αποτελεσμάτων.

### Σύγκριση με MLP

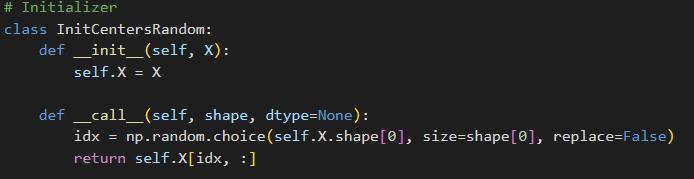
Το **Multilayer Perceptron** είναι ένα **feedforward** νευρωνικό δίκτυο που έχει τη δυνατότητα να μαθαίνει μη γραμμικές σχέσεις μεταξύ των χαρακτηριστικών και των εξόδων. Δημιουργήθηκε ένα τέτοιο δίκτυο με **ένα κρυφό επίπεδο** και με **hinge loss** για βελτιστοποίηση. Με τη χρήση hinge ( η οποία χρησιμοποιείται στα SVM) μεταφέρεται η έννοια της μεγιστοποίησης του margin σε ένα MLP. Η hinge loss απαιτεί το **τελευταίο επίπεδο ενεργοποίησης** να είναι **linear** ώστε να παίρνει τιμές -1 ή 1 για κάθε κατηγορία. Επίσης, χρησιμοποιήθηκαν **100 εποχές** , **συνάρτηση ενεργοποίησης sigmoid** και **Adam optimizer**. Το μοντέλο πέτυχε πάρα πολύ χαμηλή **ακρίβεια ίση με 0.1957** σε **χρόνο εκπαίδευσης** ίσο με **25.76s** .Η χαμηλή ακρίβεια οφείλεται στο γεγονός ότι η hinge loss είναι σχεδιασμένη για δυαδική κατηγοριοποίηση ενώ στο συγκεκριμένο dataset υπάρχουν 7 κλάσεις. Για να έχουμε καλύτερα αποτελέσματα θα πρέπει να χρησιμοποιηθεί one vs all στρατηγική. Προφανώς, το SVM ξεπερνάει κατά πολύ την επίδοση του MLP μοντέλου. Προκειμένου να εξεταστεί καλύτερα το MLP μοντέλο με hinge loss, χρησιμοποιήθηκε one-vs-all στρατηγική για την αξιολόγηση των αποτελεσμάτων. Με αυτό τον τρόπο, για κάθε κλάση εκπαιδεύεται ένα μοντέλο που μαθαίνει να διαχωρίζει αυτή από όλες τις υπόλοιπες χρησιμοποιώντας hinge loss για την ενημέρωση των βαρών. Παρατηρείται πως ο χρόνος εκπαίδευσης αυξάνεται εκθετικά γιατί χρειάζεται πολύ χρόνο μέχρι να εκπαιδευτεί ένα μοντέλο για κάθε κλάση. Συγκεκριμένα ο συνολικός χρόνος ήταν **2.5 λεπτά** πετυχαίνοντας **υψηλό test accuracy** για κάθε κλάση (**0.8337 – 0.8991**). Παρόλαυτα ακόμα και τώρα **δεν κατάφερε να ξεπεράσει τα αποτελέσματα του SVM** ενώ ο χρόνος εκπαίδευσης του είναι απαγορευτικός για περισσότερες κλάσεις ή περισσότερα folds.

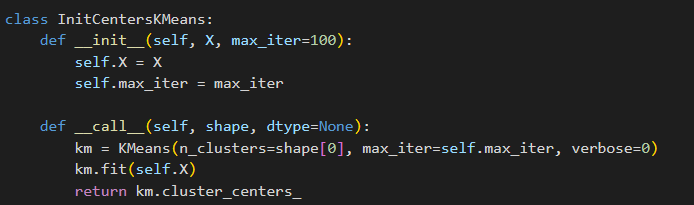
# Δημιουργία RBF Νευρωνικού Δικτύου

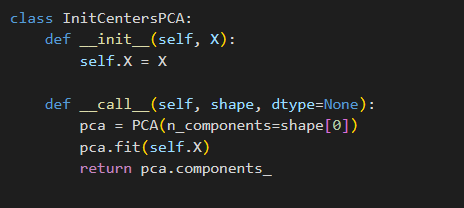
Η αρχιτεκτονική ενός **RBF Neural Network** περιλαμβάνει **τρία** βασικά **επίπεδα**: το επίπεδο **εισόδου**, ένα **κρυφό** **επίπεδο RBF νευρώνων**, και ένα **επίπεδο εξόδου**. Στο **κρυφό επίπεδο**, **κάθε νευρώνας χρησιμοποιεί μια συνάρτηση βάσης (όπως Gaussian) για να επεξεργαστεί τα εισερχόμενα δεδομένα, με την απάντηση κάθε νευρώνα να εξαρτάται από την απόσταση των δεδομένων εισόδου από τα κέντρα των RBF μονάδων**. Η **εκπαίδευση** του δικτύου γίνεται μέσω της **προσαρμογής των κέντρων , του πλάτους των RBF συναρτήσεων και των βαρών** που συνδέουν το κρυφό με το επίπεδο εξόδου. Η διαδικασία αυτή επιτρέπει τη μοντελοποίηση τοπικών χαρακτηριστικών των δεδομένων, σε αντίθεση με τις παραδοσιακές τεχνικές που στοχεύουν στη γραμμική διαχωρισιμότητα.

Για τη δημιουργία του RBF νευρωνικού δικτύου, χρησιμοποιήθηκαν οι βιβλιοθήκες keras και tensoflow εφόσον παρέχουν τη **δυνατότητα δημιουργίας** και προσθήκης ενός **custom layer** στο Sequential μοντέλο. Για το **RBF layer** έγινε **χρήση έτοιμης υλοποίησης** που βρίσκεται στο **link** που δίνεται παρακάτω μέσω GitHub : [repository](https://github.com/PetraVidnerova/rbf_keras) . Η συγκεκριμένη υλοποίηση περιλαμβάνει και δύο μεθόδους εκπαίδευσης ( **random** επιλογή κέντρων, **k-means** επιλογή κέντρων) . Προστέθηκε μία ακόμα μέθοδος εκπαίδευσης η **PCA** initializer, η οποία βασίζεται στην ανάλυση κύριων συνιστωσών για να επιλέξει τα κέντρα προκειμένου να αποτυπώνονται οι πιο σημαντικές διαστάσεις της πληροφορίας. Αν το dataset έχει d διαστάσεις, η PCA μπορεί να εξαγάγει το πολύ d κύριες συνιστώσες.

Για την εκπαίδευση του RBF layer υπάρχουν μέσα στον κώδικα υλοποιημένοι **τρεις initializers**:

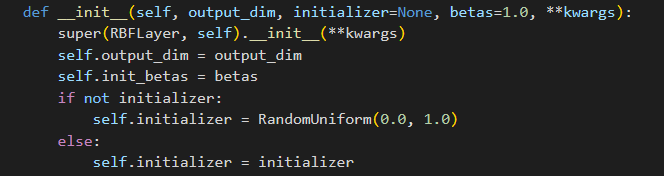
* **InitCentersRandom**: Διαλέγει τυχαία δείγματα μέσα από ένα σύνολο που του δίνεται (X\_train), τα οποία θα χρησιμοποιηθούν από το RBF layer για να γίνει η αρχικοποίηση των κέντρων. Ο αριθμός των επιλεγμένων δειγμάτων καθορίζεται από το μέγεθος που δίνεται στη μέθοδο (shape[0]). Τα τυχαία επιλεγμένα δείγματα επιστρέφονται και χρησιμοποιούνται ως κέντρα για τους RBF νευρώνες.
* **InitCentersKMeans**: Εφαρμόζει τον αλγόριθμο KMeans της sklearn.cluster για k ίσο με τον αριθμό των κρυφών νευρώνων στο RBF layer (shape[0]) πάνω στο σύνολο που του δίνεται (X\_train). Έπειτα επιστρέφει τα δείγματα που αποτελούν κέντρα των k clusters που δημιούργησε ο KMeans, ώστε να χρησιμοποιηθούν ως κέντρα των νευρώνων στο RBF layer, εξασφαλίζοντας αντιπροσωπευτική κατανομή των νευρώνων στα δεδομένα εκπαίδευσης.

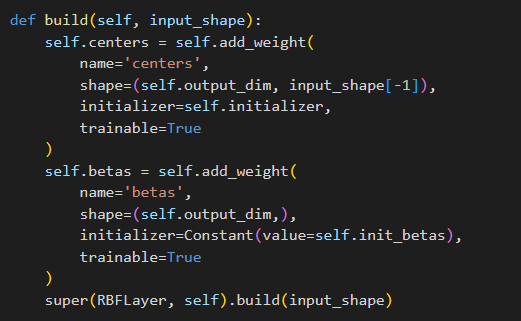
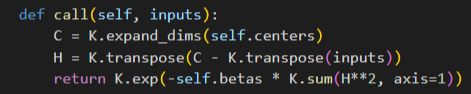


* **InitCentersPCA**: Χρησιμοποιεί την ανάλυση κύριων από τη βιβλιοθήκη sklearn.decomposition για να υπολογίσει τα αρχικά κέντρα των RBF νευρώνων. Ο αριθμός των κύριων συνιστωσών καθορίζεται από τον αριθμό των κρυφών νευρώνων (shape[0]) που απαιτούνται στο RBF layer. Αυτή η μέθοδος εξασφαλίζει ότι τα κέντρα των RBF νευρώνων βρίσκονται σε θέσεις που μεγιστοποιούν την κατανομή της διακύμανσης των δεδομένων.

Η συγκεκριμένη υλοποίηση του RBF layer **βασίζεται στις εξής συναρτήσεις** (το implementation των οποίων είναι απαραίτητο για την κατασκευή ενός custom layer με χρήση της Keras):

* **\_\_init\_\_**: Αποτελεί τον κονστράκτορα του RBF layer, μέσα στον οποίο αρχικοποιούνται με βάση τα ορίσματα ο αριθμός των κρυφών νευρώνων (output\_dim), το εύρος των γκαουσιανών (betas) και ο τρόπος εκπαίδευσης (initializer).



* **build**: Καλείται μία φορά όταν κατασκευάζεται το μοντέλο ώστε να προστεθούν trainable κέντρα και βάρη στο RBF layer.
* **call**: Υλοποιεί την λογική του **forward pass**, υπολογίζοντας την έξοδο με βάση μία είσοδο x. Η radial basis συνάρτηση που χρησιμοποιεί είναι η **gaussian**:

Προκειμένου τα αποτελέσματα να συγκριθούν με τους κατηγοριοποιητές KNN και NC που αναλύθηκαν παραπάνω, οι δοκιμές γίνονται με **20 χαρακτηριστικά** (one-hot-encoding) **με κανονικοποίηση**. Για την αξιολόγηση της απόδοσης χρησιμοποιήθηκε η τεχνική **cross-validation**. Πριν την εκπαίδευση, και λόγω χρήσης της συνάρτησης απώλειας **cross-entropy**, είναι απαραίτητη η μετατροπή της μεταβλητής **στόχου** σε **one-hot encoding**. Τα αποτελέσματα διερευνήθηκαν με βάση τις διαφορετικές τιμές των εξής παραμέτρων :

* **Αριθμό Νευρώνων στο RBF layer** (10,50,100,200)
* **Initializers** (random, k-means, PCA)
* **Betas** (εύρος της Gaussian basis function) (0.5,1.0,2.0,5.0,10.0)
* **Optimizer** (Adam, SGD, default(RMSprop))
* **Epochs** (10,50,100)

Αρχικά, διερευνήθηκαν οι **επιδόσεις ως προς τον αριθμό των νευρώνων στο RBF layer** και τους **initializers**. Διατηρήθηκε, ο default optimizer (RMSprop), αριθμό εποχών = 50 καιη παράμετρος **betas = 2.0**. Παρακάτω φαίνονται τα αποτελέσματα :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες στο κρυφό επίπεδο** | **Initializer** | **Betas** | **Optimizer** | **Epochs** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 10 | Random | 2 | RMSprop | 50 | 13.65s | 0.2020 | 1.9009 | 0.1885 | 1.9083 |
| 50 | Random | 2 | RMSprop | 50 | 14.48s | 0.2956 | 1.7780 | 0.2733 | 1.8030 |
| 100 | Random | 2 | RMSprop | 50 | 16.92s | 0.3732 | 1.6510 | 0.3477 | 1.6865 |
| 200 | Random | 2 | RMSprop | 50 | 20.14s | 0.4549 | 1.5332 | 0.4159 | 1.5887 |
| 500 | Random | 2 | RMSprop | 50 | 25.77s | 0.6017 | 1.3245 | 0.5125 | 1.4095 |
| 700 | Random | 2 | RMSprop | 50 | 33.54s | 0.6611 | 1.2368 | 0.5405 | 1.3583 |
| 10 | K-means | 2 | RMSprop | 50 | 11.72s | 0.1911 | 1.9214 | 0.1909 | 1.9305 |
| 50 | K-means | 2 | RMSprop | 50 | 14.84s | 0.3375 | 1.7390 | 0.3150 | 1.7671 |
| 100 | K-means | 2 | RMSprop | 50 | 17.39s | 0.4327 | 1.6342 | 0.3974 | 1.6726 |
| 200 | K-means | 2 | RMSprop | 50 | 19.04s | 0.5289 | 1.5217 | 0.4822 | 1.5844 |
| 500 | K-means | 2 | RMSprop | 50 | 25.74s | 0.6763 | 1.3809 | 0.5386 | 1.4934 |
| 700 | K-means | 2 | RMSprop | 50 | 34.20s | 0.7740 | 1.3107 | 0.5452 | 1.4556 |
| 2 | PCA | 2 | RMSprop | 50 | 11.90s | 0.1663 | 1.9422 | 0.1663 | 1.9467 |
| 5 | PCA | 2 | RMSprop | 50 | 12.91s | 0.1663 | 1.9422 | 0.1663 | 1.9467 |
| 10 | PCA | 2 | RMSprop | 50 | 13.70s | 0.1663 | 1.9422 | 0.1663 | 1.9467 |

Ο αριθμός των κρυφών νευρώνων στο RBF επίπεδο έχει άμεση επίδραση στο accuracy των μοντέλων. Παρατηρείται πως η αύξηση του αριθμού των νευρώνων έχει ως αποτέλεσμα τόσο την αύξηση των train και test accuracy, όσο και την μείωση των train και test loss. Αυτό σημαίνει πως η **αύξηση της πολυπλοκότητας** του RBF layer είναι **απαραίτητη** ώστε να μπορέσει το μοντέλο να ανταπεξέλθει καλύτερα στο συγκεκριμένο πρόβλημα κατηγοριοποίησης. Ωστόσο, για τη μέθοδο PCA αυτό δεν ισχύει εφόσον παρατηρείται ότι όλα παραμένουν σταθερά. Αυτό συμβαίνει γιατί η PCA βασίζεται αποκλειστικά σε γραμμικές συνδυαστικές σχέσεις και αγνοεί τη μη γραμμικότητα των δεδομένων, γεγονός που περιορίζει την προσαρμοστικότητα του μοντέλου. Σε αντίθεση με άλλες μεθόδους, όπως οι Random και K-Means, **η PCA δεν προσαρμόζει δυναμικά τα κέντρα με βάση τη δομή των δεδομένων**. Συμπερασματικά, χρειάζεται αρκετούς κρυφούς νευρώνες για να αποδίδει καλά γεγονός που οφείλεται στο ότι το RBF δίκτυο βασίζεται στη χωρική κάλυψη του χώρου εισόδου μέσω τοπικών συναρτήσεων ενεργοποίησης. Επομένως, για ή υψηλής διάστασης δεδομένα, χρειάζεται μεγαλύτερο αριθμό κρυφών νευρώνων ώστε να καταστεί ικανό να μοντελοποιήσει τις σχέσεις μεταξύ των εισόδων και των εξόδων με ακρίβεια. Ωστόσο, η **υπερβολική αύξηση των νευρώνων να οδηγεί σε υπερπροσαρμογή** και σημαντική αύξηση του χρόνου εκπαίδευσης. Έτσι στη συνέχεια επιλέχθηκαν τα μοντέλα με **περισσότερο από 100 κρυφούς νευρώνες.**

Παράλληλα σημαντικό ρόλο παίζει η **μέθοδος εκπαίδευσης**, για την οποία προκύπτει ότι ο **K-means αποφέρει καλύτερα αποτελέσματα**. Χρειάζεται περαιτέρω ρύθμιση των παραμέτρων εφόσον η ακρίβεια που επιτυγχάνεται είναι πολύ χαμηλή ενώ το loss παραμένει υψηλό.

Στη συνέχεια, επιλέχθηκαν τα μοντέλα με τις καλύτερες επιδόσεις για 200, 500 και 700 νευρώνες στο RBF επίπεδο, ώστε να εξεταστούν οι επιδόσεις τους ως προς **δύο διαφορετικούς optimizer**, τον **Adam** και τον **SGD**, διατηρώντας τις υπόλοιπες τιμές ίδιες.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες στο κρυφό επίπεδο** | **Initializer** | **Betas** | **Optimizer** | **Epochs** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 200 | Random | 2 | Adam | 50 | 18.08s | 0.7210 | 0.7575 | 0.6940 | 0.8163 |
| 500 | Random | 2 | Adam | 50 | 28.44s | 0.7974 | 0.6138 | 0.7527 | 0.7020 |
| 700 | Random | 2 | Adam | 50 | 35.11s | 0.8239 | 0.5925 | 0.7466 | 0.7231 |
| **200** | **K-means** | **2** | **Adam** | **50** | **17.99s** | **0.7803** | **0.6963** | **0.7442** | **0.7749** |
| 500 | K-means | 2 | Adam | 50 | 29.33s | 0.8483 | 0.5806 | 0.7542 | 0.7283 |
| 700 | K-means | 2 | Adam | 50 | 35.80s | 0.8683 | 0.5356 | 0.7726 | 0.6994 |

Παρατηρείται ότι με **optimizer Adam** το RBF δίκτυο εμφανίζει καλύτερα αποτελέσματα **αυξάνοντας την ακρίβεια** και μειώνοντας την απώλεια. Επιπλέον, βελτιώνει το χρόνο εκπαίδευσης σε σχέση με τους άλλους optimizer. Ωστόσο, πρέπει να σημειωθεί ότι με αύξηση των κρυφών νευρώνων παρατηρούνται τάσεις υπερπροσαρμογής. Αντίθετα, με optimizer SGD τα αποτελέσματα είναι χειρότερα και δεν υπάρχει λόγος για τη περεταίρω εξέταση του.

Στη συνέχεια, εξετάστηκαν τα καλύτερα μοντέλα για διαφορετικές τιμές **betas**. Παρακάτω φαίνονται τα αποτελέσματα:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες στο κρυφό επίπεδο** | **Initializer** | **Betas** | **Optimizer** | **Epochs** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 200 | Random | 0.1 | Adam | 50 | 17.97s | 0.8010 | 0.5588 | 0.7778 | 0.6023 |
| 200 | Random | 1.0 | Adam | 50 | 18.49s | 0.8302 | 0.4933 | 0.8029 | 0.5377 |
| 200 | Random | 2.0 | Adam | 50 | 18.08s | 0.7210 | 0.7575 | 0.6940 | 0.8163 |
| 200 | Random | 5.0 | Adam | 50 | 17.74s | 0.38831 | 1.6706 | 0.3174 | 1.7423 |
| 200 | Random | 10.0 | Adam | 50 | 18.35s | 0.3444 | 1.7577 | 0.2538 | 1.8292 |
| 200 | K-means | 0.1 | Adam | 50 | 18.40s | 0.8160 | 0.5402 | 0.7911 | 0.5880 |
| **200** | **K-means** | **1.0** | **Adam** | **50** | **17.54s** | **0.8672** | **0.4365** | **0.8351** | **0.4942** |
| 200 | K-means | 2.0 | Adam | 50 | 17.99s | 0.7803 | 0.6963 | 0.7442 | 0.7749 |
| 200 | K-means | 5.0 | Adam | 50 | 18.84s | 0.4224 | 1.6681 | 0.3330 | 1.7749 |
| 200 | K-means | 10.0 | Adam | 50 | 17.68s | 0.3244 | 1.7816 | 0.2534 | 1.8595 |

Παρατηρείται ότι τα μοντέλα **αποφέρουν καλύτερα αποτελέσματα για τις μικρότερες τιμές της παραμέτρου betas**. Αυτό σημαίνει πως **τα clusters** που σχηματίζουν τα δείγματα της κάθε κλάσης **είναι κοντά το ένα με το άλλο**, με αποτέλεσμα να **έχουμε καλύτερη κατηγοριοποίηση όταν οι γκαουσιανές έχουν μικρότερο εύρος**, το οποίο καθιστά τους νευρώνες του RBF επιπέδου πιο ευαίσθητους σε δείγματα που βρίσκονται πιο κοντά στο κέντρο που τους έχει ανατεθεί. Συγκεκριμένα και οι δύο μέθοδοι αποδίδουν καλύτερα για **betas = 1.0**.

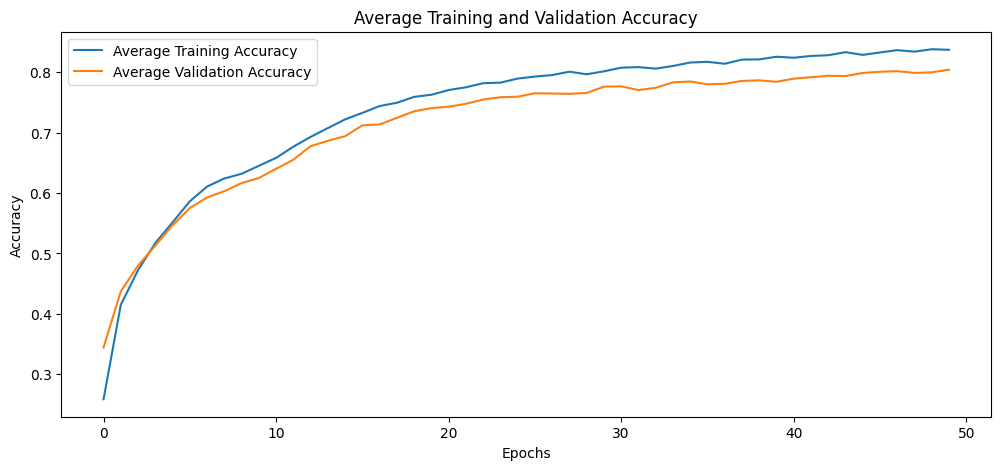
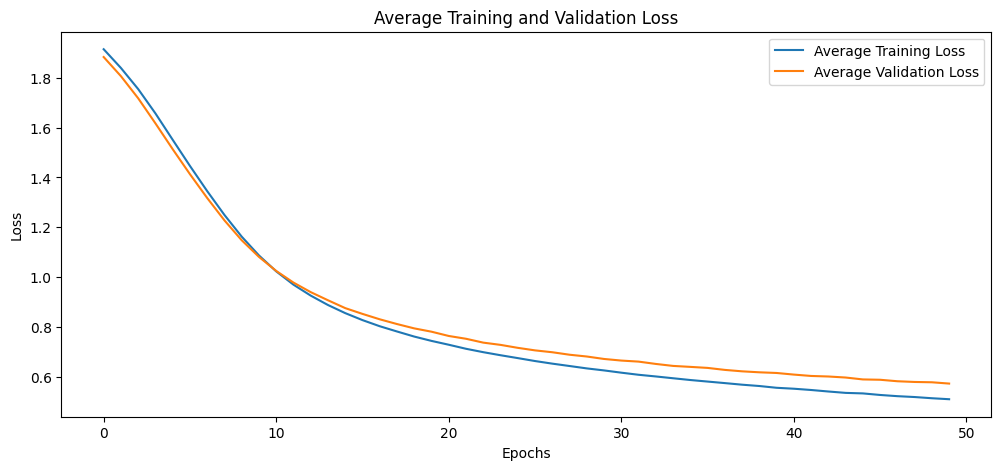
Τέλος, προκειμένου να εξεταστεί αν η ακρίβεια των καλύτερων μοντέλων μπορεί να βελτιωθεί, παρατηρήθηκε αν ο **αριθμός των εποχών** επιδρά αποτελεσματικά στα μοντέλα.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Νευρώνες στο κρυφό επίπεδο** | **Initializer** | **Betas** | **Optimizer** | **Epochs** | **Training Time** | **Train Accuracy (AVG)** | **Train Loss (AVG)** | **Test Accuracy (AVG)** | **Test Loss (AVG)** |
| 200 | Random | 1.0 | Adam | 20 | 10.70s | 0.6967 | 0.8613 | 0.6911 | 0.8855 |
| 200 | Random | 1.0 | Adam | 30 | 11.80s | 0.7590 | 0.6714 | 0.7281 | 0.7154 |
| 200 | Random | 1.0 | Adam | 50 | 18.49s | 0.8302 | 0.4933 | 0.8029 | 0.5377 |
| **200** | **Random** | **1.0** | **Adam** | **100** | **35.07s** | **0.8797** | **0.366** | **0.8494** | **0.4380** |
| 200 | K-means | 1.0 | Adam | 20 | 8.08s | 0.7328 | 0.8686 | 0.7087 | 0.9145 |
| 200 | K-means | 1.0 | Adam | 30 | 12.00s | 0.8084 | 0.6070 | 0.7778 | 0.6635 |
| 200 | K-means | 1.0 | Adam | 50 | 17.54s | 0.8672 | 0.4365 | 0.8351 | 0.4942 |
| 200 | K-means | 1.0 | Adam | 100 | 34.01s | 0.7498 | 0.6340 | 0.7932 | 0.5308 |

Παρατηρείται πως η αύξηση του αριθμού των εποχών, συνεπάγεται την αύξηση του χρόνου εκπαίδευσης αλλά και την αύξηση της ακριβείας και στις δύο μεθόδους. Πιο συγκεκριμένα, για τον **K-means** φαίνεται πως μέγιστη ακρίβεια φτάνει **0.8351 με 50 εποχές**. Για τον **Random**, η αύξηση του αριθμού των εποχών έχει ως αποτέλεσμα τη βελτίωση της ακρίβειας πετυχαίνοντας το μέγιστο για τα RBF μοντέλα που εξετάστηκαν. Ωστόσο, **ο χρόνος εκπαίδευσης αυξάνεται εκθετικά για να επιτευχθεί ακρίβεια 0.8494**.

## Συμπεράσματα

Το καλύτερο RBF μοντέλο που προέκυψε από την ανάλυση είναι με **200 κρυφούς νευρώνες** , **initializer τον KMeans, optimizer τον Adam με default (0.001) learning rate, 50 εποχές κατά την εκπαίδευση και εύρος (betas) ίσο με 1** .Το μοντέλο πέτυχε ικανοποιητική ακρίβεια στο test set με **test accuracy = 0.8351** και χρόνο εκπαίδευσης **17.54s**, χωρίς να υπερπροσαρμόζεται στα δεδομένα. Παρατίθενται οι καμπύλες ακρίβειας και κόστους όπου παρατηρείται μία ανοδική και μια καθοδική τάση αντίστοιχα.



Στο κώδικα έχω κρατήσει τις τιμές των παραμέτρων για το καλύτερο RBF μοντέλο.

### Παραδείγματα Ορθής και Εσφαλμένης Κατηγοριοποίησης

Φαίνεται πως καταφέρνει να κατηγοριοποιήσει με επιτυχία όλες τις κλάσεις ενώ υπάρχει μια ιδιαίτερη σύγχυση των κλάσεων 1 και 6 (Normal\_Weight και Overweight\_Level\_II αντίστοιχα).

### Σύγκριση με ΚΝΝ και NC

Τα RBF Neural Networks αποδεικνύονται μια ικανοποιητική μέθοδος ταξινόμησης, επιδεικνύοντας σημαντικά καλύτερη απόδοση σε σύγκριση με τους απλούστερους κατηγοριοποιητές, όπως ο KNN και ο NC. Το μοντέλο πέτυχε **test accuracy = 0.8351** και χρόνο εκπαίδευσης **17.54 δευτερόλεπτα**, χωρίς να εμφανίζει υπερπροσαρμογή. Σε σύγκριση με τον KNN, ο οποίος με **k=1** πέτυχε **test accuracy = 0.8899**, το RBF πλησίασε σε απόδοση αλλά απαιτεί περισσότερο χρόνο εκπαίδευσης. Παρ' όλα αυτά, παρέχει τη δυνατότητα γενίκευσης μη γραμμικών σχέσεων, κάτι που ο KNN δεν μπορεί να πετύχει τόσο εύκολα σε πιο σύνθετα datasets. Αυτή η διαφορά αποδεικνύει την ανωτερότητα των RBF σε περιβάλλοντα με πιο περίπλοκα όρια απόφασης, όπου οι απλοί κατηγοριοποιητές, όπως ο NC, αποτυγχάνουν να αποδώσουν.

# Συνολικά Αποτελέσματα

**Κ-Πλησιέστεροι Γείτονες (KNN):**

* Το καλύτερο αποτέλεσμα επιτεύχθηκε με 20 διαστάσεις (one-hot encoding) χωρίς κανονικοποίηση δεδομένων και k=1, όπου η ακρίβεια στο test set ήταν **0.8899**.
* Η προσθήκη κανονικοποίησης δεδομένων είχε αρνητικό αντίκτυπο στην απόδοση, μειώνοντας την ακρίβεια στο test set.

**Nearest Centroid (NC):**

* Το καλύτερο αποτέλεσμα επιτεύχθηκε με 20 διαστάσεις (one-hot encoding) και κανονικοποίηση δεδομένων, όπου η ακρίβεια στο test set ήταν **0.5443**.
* Το NC παρουσίασε σαφή βελτίωση με την κανονικοποίηση, ωστόσο, η ακρίβεια του παρέμεινε χαμηλότερη σε σύγκριση με τους άλλους κατηγοριοποιητές.

**Neural Network:**

* **Καλύτερη Αρχιτεκτονική**:
  + Αρχιτεκτονική : 2 κρυφά επίπεδα
  + Συνάρτηση Ενεργοποίησης: Sigmoid
  + Optimizer: Adam
  + Epochs: 100
  + Test Accuracy: **0.8768**
  + Train Accuracy: 0.9057
  + Training Time: **19.30s**
* **Συμπεράσματα:** Παρουσίασε εξαιρετική ισορροπία μεταξύ απόδοσης και χρόνου εκπαίδευσης ενώ η χαμηλή τιμή training loss και η σταθερότητα του validation loss δείχνουν ότι το μοντέλο γενικεύει καλά χωρίς να εμφανίζει υπερπροσαρμογή.

**Support Vector Classifier (SVC):**

Το SVC με γραμμικό ή μη γραμμικό πυρήνα (π.χ. RBF) επιδεικνύει ανώτερες επιδόσεις:

* **Καλύτερο SVM**:
  + Πυρήνας: Πολυωνυμικός (degree=2)
  + C=2
  + Decision function: One-vs-all
  + Test Accuracy: **0.9583**
  + Train Accuracy: 0.9879
  + Training Time: **5.96s**
* **Συμπεράσματα**: Η υψηλή ακρίβεια του SVM αλλά και πολύ μικρός χρόνος εκπαίδευσης, σε συνδυασμό με την αντοχή του σε θόρυβο, το καθιστούν **ιδανικό για το συγκεκριμένο dataset**.

**RBF Neural Network:**

Το καλύτερο αποτέλεσμα επιτεύχθηκε με 20 διαστάσεις (one-hot encoding) με τη κανονικοποίηση δεδομένων στο Age και Weight διότι η λειτουργία του βασίζεται στην απόσταση μεταξύ των δεδομένων και των κέντρων των RBF νευρώνων. Αν τα χαρακτηριστικά έχουν διαφορετικές κλίμακες, τα χαρακτηριστικά με μεγαλύτερη τιμή θα κυριαρχούν στον υπολογισμό της απόστασης, αλλοιώνοντας τη συνεισφορά των υπόλοιπων χαρακτηριστικών.

* **Καλύτερο RBF**:
  + Αριθμός Κρυφών Νευρώνων: 200
  + Initializer: KMeans
  + Optimizer: Adam
  + Betas: 1.0
  + Test Accuracy: **0.8351**
  + Train Accuracy: 0.8797
  + Training Time: **17.54s**
* **Συμπεράσματα**: Απαιτεί περισσότερους κρυφούς νευρώνες για να φτάσει υψηλά επίπεδα ακρίβειας σε σχέση με το MLP. Παρουσιάζει χαμηλότερη ακρίβεια από το νευρωνικό δίκτυο με 2 κρυφά επίπεδα (0.8351 έναντι 0.8768), δεν πλησιάζει την απόδοση του SVM.

**Συνολικά:**

Συγκρίνοντας τις διάφορες μεθόδους ταξινόμησης, παρατηρείται ότι το **Support Vector Classifier (SVC)** υπερέχει σε απόδοση στο συγκεκριμένο dataset, επιτυγχάνοντας **test accuracy 0.9583** με πολύ μικρό χρόνο εκπαίδευσης (**5.96s**). Η υψηλή του ακρίβεια, η αντοχή σε θόρυβο και η αποτελεσματικότητα στη γενίκευση το καθιστούν την **ιδανική επιλογή** μεταξύ των εξεταζόμενων μεθόδων.

Το **Neural Network με 2 κρυφά επίπεδα** κατέχει τη δεύτερη καλύτερη θέση, παρουσιάζοντας **test accuracy 0.8768** και συνδυάζοντας καλή ακρίβεια με σταθερότητα και χαμηλό χρόνο εκπαίδευσης (**19.30s**). Το χαμηλό training και validation loss δείχνουν ότι το μοντέλο γενικεύει καλά, χωρίς να εμφανίζει υπερπροσαρμογή.

Το **RBF Neural Network** σημειώνει **test accuracy 0.8351** με σημαντικά υψηλότερο χρόνο εκπαίδευσης (17.54s) και απαιτεί περισσότερους κρυφούς νευρώνες για να επιτύχει ικανοποιητική απόδοση. Ωστόσο, δεν πλησιάζει την ακρίβεια του MLP ούτε του SVM, υποδεικνύοντας ότι δεν είναι η βέλτιστη επιλογή για το συγκεκριμένο dataset.

Στις απλούστερες μεθόδους, ο **Κ-Πλησιέστερος Γείτονας (KNN)**, με **k=1**, επιτυγχάνει **test accuracy 0.8899**, ενώ ο χρόνος εκπαίδευσης είναι χαμηλός λόγω της απλότητας του αλγορίθμου. Ωστόσο εμφανίζει **υπερπροσαρμογή** στα δεδομένα. Από την άλλη, ο **Nearest Centroid (NC)** παρουσιάζει σημαντικά χαμηλότερη απόδοση (**test accuracy 0.5443**), ακόμη και με κανονικοποίηση δεδομένων, παραμένοντας υποδεέστερος σε σχέση με τις άλλες μεθόδους.

Συμπερασματικά, **το SVC και το Neural Network με 2 κρυφά επίπεδα** αποτελούν τις καλύτερες επιλογές για το dataset, με το **SVC να υπερέχει** λόγω της εξαιρετικής του απόδοσης και της χαμηλής υπολογιστικής πολυπλοκότητας.

# Αναφορές

[1][**sklearn.neighbors.KNeighborsClassifier documantation**](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)

[2][**sklearn.neighbors.NearestCentroid documentation**](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestCentroid.html)

[3] [**costum RBF layer using keras**](https://github.com/PetraVidnerova/rbf_keras)

[4]  [Radial Basis Functions: Types, Advantages, and Use Cases](https://hackernoon.com/radial-basis-functions-types-advantages-and-use-cases)