ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΛΥΣΗ ΓΟΝΙΔΙΑΚΗΣ ΕΚΦΡΑΣΗΣ ΜΕ ΤΗΝ ΧΡΗΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

ΠΑΝΑΓΙΩΤΑΚΗΣ ΓΕΩΡΓΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΜΑΡΙΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΗΡΑΚΛΕΙΟ
ΑΠΡΙΛΙΟΣ 2021

Ευχαριστίες:

Θα ήθελα να πω ένα μεγάλο ευχαριστώ στους καθηγητές μου Μαριά Κωνσταντίνο, Τσικνάκη Εμμανουήλ και Κουμάκη Ελευθέριο για την μεγάλη βοήθεια, κατανόηση και εμψύχωση που μου παρείχαν κατά την διάρκεια εκπόνησης την πτυχιακής μου εργασίας. Θα ήθελα να ευχαριστήσω και την κοπέλα μου για την ψυχολογική υποστήριξη και την επιβεβαίωση των δυνατοτήτων μου. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για την βοήθεια και την υποστήριξη που μου παρέχουν σε ότι και να κάνω.

Περίληψη:

Με την παρούσα εργασία επιδιώκεται η σκιαγράφηση των βασικών πτυχών της χρήσης τεχνητής νοημοσύνης στο πεδίο της βιοπληροφορικής και συγκεκριμένα της χρήσης νευρωνικών δικτύων για την ανάλυση γονιδιακών εκφράσεων. Πιο αναλυτικά, σε πρώτο στάδιο ορίζονται εισαγωγικές αλλά ιδιαίτερα χρήσιμες έννοιες για την παρούσα πτυχιακή εργασία. Έπειτα, παρουσιάζεται η διαδικασία συγκέντρωσης δεδομένων γονιδιακών εκφράσεων για 28 διαφορετικούς τύπους καρκίνου με 10.362 δείγματα από το TCGA, η οποία αποτελεί μια από τις πιο γνωστές βάσεις γονιδιακών εκφράσεων. Ακόμη, αναλύεται η διαδικασία προ επεξεργασίας των δεδομένων όπως η χρήση του αλγορίθμου SMOTE για την παραγωγή καινούργιων δειγμάτων και για την καλύτερη διεξαγωγή της ανάλυσης. Στην συνέχεια, παρουσιάζεται η εφαρμογή διαφορετικών ειδών της αρχιτεκτονικής Autoencoder και πιο συγκεκριμένα χρησιμοποιούνται τα μοντέλα Autoencoder, Denoising Autoencoder και Variational Autoencoder για την μείωση του αριθμού των χαρακτηριστικών. Μετά την μείωση των χαρακτηριστικών, τα δεδομένα τροφοδοτούνται σε δύο διαφορετικά δίκτυα ταξινόμησης, ένα απλό βαθύ νευρωνικό (DNN) δίκτυο και μια διαφορετική αρχιτεκτονική που ονομάζεται Deep Cross Model. Επιπρόσθετα, για την αξιολόγηση των αποτελεσμάτων των νευρωνικών δικτύων παρουσιάζεται και ένας διαφορετικός τρόπος ανάλυσης των αρχικών δεδομένων με την χρήση αλγορίθμων μηχανικής μάθησης. Συγκεκριμένα, για την μείωση των διαστάσεων χρησιμοποιείται ο αλγόριθμος PCA και για την διαδικασία της ταξινόμησης διαφορετικοί αλγόριθμοι ταξινόμησης, όπως ο αλγόριθμος SVM, Random Forests και άλλοι ακόμα. Εκτός από την ύπαρξη των δύο αναλύσεων παρουσιάζονται και δύο πρόσθετες αναλύσεις, οι οποίες αποτελούν μια μίξη των παραπάνω διαδικασιών. Ακόμη, παρουσιάζονται τα αποτελέσματα κάθε ανάλυσης και συγκρίνεται η μεταξύ τους απόδοση με ορισμένο μετρητή το ποσοστό ακρίβειας για την ταξινόμηση κάθε κλάσης. Τα αποτελέσματα από αυτές τις αναλύσεις ήταν, 97.4% για το ποσοστό ακρίβειας των αλγορίθμων μηχανικής μάθησης στην ταξινόμηση 2,210 χαρακτηριστικών σε σχέση με την ανάλυση των νευρωνικών δικτύων, που επιτεύχθηκε ποσοστό 95.4% για την ταξινόμηση 70 χαρακτηριστικών. Εκτός από το ποσοστό ακρίβειας , ένα μέτρο σύγκρισης ήταν και η διάρκεια εκπαίδευσης κάθε αλγορίθμου. Από τα πειράματά μας μπορούμε να συμπεράνουμε ότι τα μοντέλα βαθιάς μάθησης παρέχουν οριακά καλύτερα αποτελέσματα από τους παραδοσιακούς αλγόριθμους μηχανικής μάθησης για ένα τόσο περίπλοκο και μεγάλο σύνολο δεδομένων. Συμπερασματικά, αναφέρεται η διαδικασία συντονισμού υπερπαραμέτρων για την ανάλυση μηχανικής μάθησης και νευρωνικών δικτύων

χρησιμοποιώντας τον αλγόριθμο Μπεϋζιανής βελτιστοποίησης για την παροχή καλύτερων αποτελεσμάτων. Καταληκτικά, γίνεται οπτικοποίηση των πινάκων «σύγχυσης» και πραγματοποιείται μείωση των δεδομένων σε δύο χαρακτηριστικά με την χρήση των αλγορίθμων t-SNE, PCA καθώς και την χρήση Variational Autoencoder + PCA και Variational Autoencoder + t-SNE για την οπτικοποίηση και την παρουσίαση ομοιοτήτων μεταξύ δειγμάτων όλων των τύπων καρκίνου.

Λέξεις Κλειδιά:
Βιοπληροφορική, Γονίδιο, Γονιδιακή Έκφραση, Ανάλυση Γονιδιακής Έκφρασης, Γονιδιακή Έκφραση Καρκίνου, Νευρωνικά Δίκτυα, Μηχανική Μάθηση, Autoencoders, Denoising Autoencoders, Variational Autoencoders, Keras Tuner, Deep Cross Model

Abstract:

The present work outlines the basic aspects of artificial intelligence in the field of bioinformatics and in particular the use of neural networks for the analysis of gene expressions. In the first stage, introductory concepts for the present dissertation are defined. Next, the process of collecting gene expression data for 28 different types of cancer with 10.362 samples from TCGA is presented, which is one of the best-known gene expression databases. Furthermore, pre-processing methods for data are described, such as the SMOTE algorithm, that can produce new samples for better performance of the analysis. Next, the application of different types of Autoencoder architecture is presented and specifically the Autoencoder, Denoising Autoencoder and Variational Autoencoder models are used to reduce the number of features. After the feature reduction, the data were fed into two different classification networks, a simple deep neural network and a different architecture called Deep Cross Model. Also, an alternative way of analyzing the original data using machine learning algorithms is presented in order to evaluate and compare the results with the ones of the neural networks. Specifically, for the reduction of dimensions the PCA algorithm used and for the classification process different classification algorithms, such as the SVM algorithm, Random Forests and others employed. In addition, two different analyzes are presented, which are a mixture of the above procedures. Furthermore, the results of each analysis are presented and the performance between them is compared using the accuracy for the classification of each class. The results of these metrics were 97.4% accuracy score produced by the machine learning analysis for the classification of 2210 features while 95.4% accuracy produced by the deep learning task with number of features for each sample reduced to 70. In addition to the percentage of accuracy, a measure of comparison is the training duration of each algorithm. From our experiments we can conclude that the deep learning models provide marginally better results than the traditional machine learning algorithms for such a complex and big dataset. Concluding, the process of tuning hyperparameters for the analysis of machine learning and neural networks using the Bayesian optimization algorithm to provide better results is mentioned. Finally, the confusion matrices are visualized and the data were reduced to two features using the algorithms t-SNE, PCA as well as the use of Variational Autoencoder + PCA and Variational Autoencoder + t-SNE for their visualization and the presentation of similarities between of samples of all the types of cancer.

Keywords:
Bioinformatics, Gene, Gene Expression, Gene Expression Analysis, Cancer Gene Expression, Neural Network, Machine Learning, Autoencoders, Denoising Autoencoders, Variational Autoencoders, Keras Tuner, Deep Cross Model

Περιεχόμενα

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

1. Εισαγωγή στην βιοπληροφορική

1.1 Τι είναι βιοπληροφορική

Με τον όρο βιοπληροφορική εννοείται η χρήση της τεχνολογίας πάνω στο πεδίο της μοριακής βιολογίας για την ανάπτυξη εργαλείων για την βέλτιστη κατανόηση βιολογικών δεδομένων. Η βιοπληροφορική είναι συνδυασμός γνώσης από διάφορους επιστημονικούς κλάδους όπως στατιστική, μηχανική, επιστήμη υπολογιστών, μοριακή βιολογία και πολλούς ακόμα. Ο τομέας της βιοπληροφορικής υπάρχει εδώ και πολλά χρόνια αλλά τελευταία, η εξέλιξη της τεχνολογίας συνέβαλε στην ανάλυση του ανθρώπινου γονιδιώματος μέσω ενός επιστημονικού έργου (Human Genome Project) που ολοκληρώθηκε το 2003. Αν και η ανακάλυψη της δομής του DNA έγινε το 1962 από τους Τζέιμς Γουάτσον, Ρόζαλιν Φράνκλιν, και Φράνσις Κρικ ο πλήρης προσδιορισμός των ζευγών βάσεων που απαρτίζουν το ανθρώπινο DNA. καθώς και ο εντοπισμός και την χαρτογράφηση όλων των γονιδίων του ανθρώπινου γονιδιώματος ολοκληρώθηκε το 2003. Η βιοπληροφορική την σημερινή εποχή χρησιμοποιείται στην ανάλυση γονιδιακών εκφράσεων, στην ανάλυση ακολουθιών γονιδίων, στην πρόβλεψη κάποιας ασθένειας καθώς και στην ανακάλυψη και δημιουργία καινούριων φαρμάκων. Παραδείγματος χάρη μπορεί να χρησιμοποιηθεί για την ανάλυση γονιδιακής έκφρασης για διάφορες ασθένειες. Με τον τρόπο αυτό συμπεραίνεται ποια γονίδια προκαλούν μια ασθένεια, όπως ένα τύπο καρκίνου, και ποια τον καταπολεμούν καθώς και τις μεταξύ τους σχέσεις.[1]

1.2 Τι είναι γονιδιωματική;

Γονιδιωματική είναι η μελέτη του γονιδιώματος των οργανισμών και αφορά την εφαρμογή υπολογιστικών μεθόδων χαρτογράφησης γονιδίων, την αλληλεπίδραση μεταξύ των γονιδίων αυτών, την ανάλυση αλληλουχιών DNA καθώς και την σύγκριση και εξαγωγή συμπερασμάτων[2]. Η γονιδιωματική χωρίζεται σε διάφορα είδη, την δομική γονιδιωματική

[1] Cory Mitchell, "Bioinformatics", Investopedia, 10 December 2020, https://www.investopedia.com/terms/b/bioinformatics.asp (ανακτήθηκε: 15/1/2020)

[2] Τριανταφυλλίδης Α, "Ειδικά θέματα γενετικής" https://opencourses.auth.gr/modules/document/file.php/OCRS146/%CE%A0%CE%B1%CF%81%CE%BF%CF%85%CF%83%CE%B9%CE%AC%CF%83%CE%B5%CE%B9%CF%82/%CE%95%CE%BD%CF%8C%CF

(Structural Genomics) η οποία αναλύει την δομή του γονιδιώματος (π.χ. γενετική χαρτογράφηση) και την λειτουργική γονιδιωματική (Functional Genomics) η οποία εξετάζει την λειτουργία του γονιδιώματος. Η λειτουργική γονιδιωματική περιέχει την ανάλυση όλων των RNA που μεταγράφονται στο κύτταρο καθώς και όλων των πρωτεϊνών που κωδικοποιεί το γονιδίωμα. Η συγκριτική γονιδιωματική εφαρμόζει γνώση από έναν οργανισμό σε άλλους.

1.2.1 Τι είναι γονιδίωμα;

Γονιδίωμα είναι η συλλογή όλου του γενετικού υλικού που περιέχει ένας οργανισμός. Κάθε οργανισμός περιέχει γονίδια και άλλα στοιχεία του γενετικού υλικού που τον προσδιορίζουν. Το γονιδίωμα είναι διαφορετικό μεταξύ οργανισμών, για παράδειγμα σε μερικούς ιούς το γονιδίωμα κωδικοποιεί λιγότερα από 10 γονίδια ενώ σε ευκαριωτικούς οργανισμούς όπως ο άνθρωπος, το γονιδίωμα περιέχει δισεκατομμύρια βασικά ζευγάρια του γενετικού υλικού που κωδικοποιούν δεκάδες χιλιάδες γονίδια. Την σημερινή εποχή έχουμε προσδιορίσει την ακολουθία DNA από διάφορα γονιδιώματα και επικεντρωνόμαστε στο να προσδιορίσουμε την οργάνωση και την λειτουργία τους. Η αποκωδικοποίηση του ανθρώπινου γονιδιώματος το 2003 αποτέλεσε επιτυχία για τον επιστημονικό κλάδο. Μέσω ενός project (Human Genome Project) στο οποίο συμμετείχαν εκατοντάδες επιστήμονες από όλο τον κόσμο και ολοκληρώθηκε ακριβώς 50 χρόνια μετά από την δημοσίευση της δομής της διπλής ελικοειδής μορφής του DNA (Double-stranded helical structure of DNA) το 1953 από τους Crick και Watson.[3]

1.2.2 Τι είναι γονίδιο;

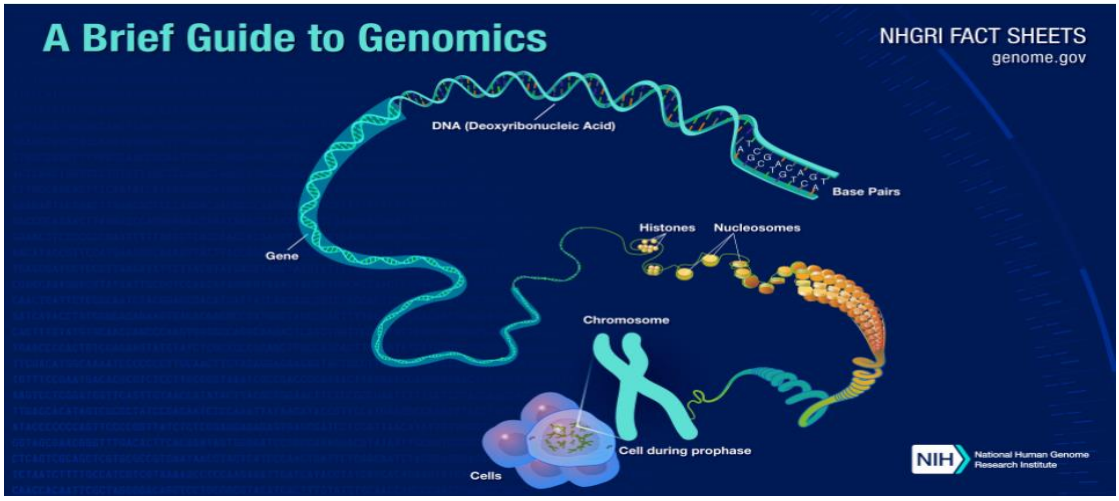
Γονίδιο είναι μία αλληλουχία DNA η οποία κωδικοποιεί ένα προϊόν RNA που μπορεί στην συνέχεια να μεταφραστεί σε πρωτεϊνικό προϊόν.[4] Συνήθως αναφερόμαστε ότι ένα γονίδιο είναι μια μονάδα DNA η οποία φέρει οδηγίες για την δημιουργία μιας πρωτεΐνης ή ενός

%84%CE%B7%CF%84%CE%B1%2001%3A%20%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE%20%CF%83%CF%84%CE%B7%20%CE%B3%CE%BF%CE%BD%CE%B9%CE%B4%CE%B9%CF%89%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AE.pdf (Ανακτήθηκε 21/1/2021)

[3] Jonathan Pevsner, ”Bioinformatics and functional genomics”, Wiley Blackwell, Third Edition, Singapore 2015

[4] Jonathan Pevsner, ”Bioinformatics and functional genomics”, Wiley Blackwell, Third Edition, Singapore 2015

συνόλου πρωτεϊνών. [5] Στο ανθρώπινο γονιδίωμα υπάρχουν περίπου 20.000-25.000 γονίδια από τα οποία το κάθε ένα κατά μέσο όρο κωδικοποιεί τρεις πρωτεΐνες. Τα γονίδια βρίσκονται σε 23 ζεύγη χρωμοσωμάτων στον πυρήνα του ανθρώπινου κυττάρου και κατευθύνουν την παραγωγή των πρωτεϊνών με την βοήθεια ενζύμων και αγγελιοφόρων μορίων. Στα ευκαρυωτικά κύτταρα τα γονίδια διαιρούνται σε πολλά κομμάτια αλλά συνεχίζουν να παράγουν μία πρωτεΐνη με συνοχή. Τα περισσότερα ευκαρυωτικά γονίδια περιέχουν εξώνια και εσώνια, από τα οποία τα ευκαρυωτικά κύτταρα αφαιρούν τα εσώνια από το μετάγραφο (Transcript) του RNA και συνενώσουν τα εξώνια πριν από την είσοδο του mRNA στο ριβόσωμα.[6]



Εικόνα 1.1: Εικόνα γονιδίου. Πηγή: National Human Genome Research Institute, "A brief Guide to genomics" https://www.genome.gov/about-genomics/fact-sheets/A-Brief-Guide-to-Genomics (Ανακτήθηκε 22/1/2021)

1.2.3 Τι είναι έκφραση γονιδίων;

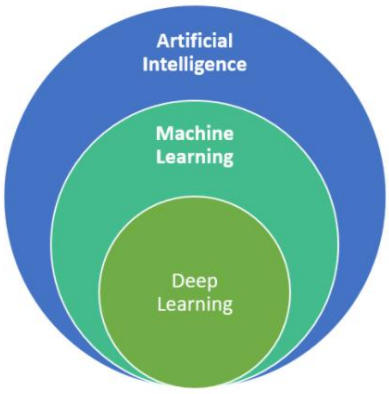
Έκφραση γονιδίων είναι η διαδικασία κατά την οποία η πληροφορία που είναι κωδικοποιημένη σε ένα γονίδιο χρησιμοποιείται για την συναρμολόγηση ενός μορίου πρωτεΐνης. Το κύτταρο «διαβάζει» την ακολουθία ενός γονιδίου σε γκρουπ των 3 βάσεων. Κάθε γκρουπ από τρεις βάσεις αντιστοιχεί σε ένα από τα 20 διαφορετικά αμινοξέα που

[5] National Human Genome Research Institute, "A brief guide to genomics", https://www.genome.gov/about-genomics/fact-sheets/A-Brief-Guide-to-Genomics (Ανακτήθηκε 21/1/2021)

[6] Neil C. Jones, Pavel A. Pevzner, "An introduction to Bioinformatics algorithms", Massachusetts Institute of technology, August 2004

χρησιμοποιούνται για να κατασκευάσουν μια πρωτεΐνη.[7] Δύο πειραματικές προσεγγίσεις έχουν υποστήριξη την ανάλυση των γονιδιακών εκφράσεων (mRNA Transcript levels): 1) οι μικροσυστοιχίες (Microarrays) και 2) η ανάλυση ακολουθίας επόμενης γενιάς (Next Generation Sequencing). Οι μικροσυστοιχίες ήταν στην κορυφή μέχρι τις αρχές του 2000 και μέσα στην επόμενη δεκαετία υπήρξε αρκετό ενδιαφέρον στην ανάλυση επόμενης γενιάς. Σε καθεμία από τις δύο περιπτώσεις το RNA εξάγεται από μια πηγή (π.χ. ανθρώπινος εγκέφαλος) και πραγματοποιούνται διάφορες συγκρίσεις (π.χ. αλλαγές που συμβαίνουν στην μεταγραφή RNA σε διάφορα στάδια ανάπτυξης). Ο κύριος σκοπός και των δύο τεχνολογιών είναι να αναγνωριστούν ποια γονίδια έχουν υψηλές ή χαμηλές τιμές. Το πλεονέκτημα της τεχνικής αλληλουχίας RNA (RNA Sequencing) έναντι των μικροσυστοιχιών είναι ότι δεν προεπιλέγεται το μετάγραφο (transcript) που θα αναλυθεί αλλά η αλληλουχία διευκρινίζει όλα τα είδη RNA που είναι παρόν σε κάθε δείγμα[8].

2.Τι είναι τεχνητή νοημοσύνη, μηχανική μάθηση και νευρωνικά δίκτυα;



Εικόνα 2.1: Εικόνα συσχέτισης τεχνητής νοημοσύνης, μηχανικής μάθησης και νευρωνικών δικτύων. Πηγή: Santosh, "Artificial Intelligence VS Machine Learning VS Deep Learning", Data Driven Investor, 27 March 2020 (Ανακτήθηκε 19/1/2021)

[7] National Human Genome Research Institute, "Gene Expression", https://www.genome.gov/genetics-glossary/Gene-Expression (Ανακτήθηκε 22/1/2021)

[8] Jonathan Pevsner, "Bioinformatics and functional genomics", Wiley Blackwell, Third Edition, Singapore 2015

2.1 Τι είναι τεχνητή νοημοσύνη;

Η τεχνητή νοημοσύνη είναι ο τρόπος ώστε οι υπολογιστές να «σκέφτονται» και να ενεργούν μόνοι τους. Είναι η διαδικασία εύρεσης μεθοδολογιών και θεωριών ώστε να καταλάβουν το πώς λειτουργεί ο κόσμος μας και να συμπεριφέρονται όπως ο άνθρωπος ή και καλύτερα.[9] Επιστήμονες πιστεύουν ότι το κλειδί για να δημιουργηθεί η τεχνητή νοημοσύνη είναι να αποκωδικοποιήσουν τη λειτουργία του ανθρώπινου εγκεφάλου. Για το λόγο αυτό και έχουν δημιουργηθεί κατάλληλοι αλγόριθμοι οι οποίοι αντιπροσωπεύουν κατά κάποιο βαθμό τον ανθρώπινο εγκέφαλο όπως τα νευρωνικά δίκτυα. Χάρη στην εξέλιξη της τεχνολογίας και της δημιουργίας υπολογιστών οι οποίοι μπορούν να εκτελέσουν τρισεκατομμύρια πράξεις το δευτερόλεπτο ο κλάδος της τεχνητής νοημοσύνης έχει προσελκύσει τους επιστήμονες να ασχοληθούν περισσότερο με αυτήν ενώ χρόνο με το χρόνο επενδύονται από κυβερνήσεις και μεγάλες εταιρείες δισεκατομμύρια για έρευνα και ανάπτυξη πάνω σε αυτό το κομμάτι.

2.1.1 Πώς χρησιμοποιείται η τεχνητή νοημοσύνη σήμερα;

Η τεχνητή νοημοσύνη χρησιμοποιείται καθημερινά σε διάφορους κλάδους, όπως, στην ιατρική, για την καταπολέμηση ασθενειών, στην μηχανική για την δημιουργία οχημάτων τα οποία οδηγούν μόνα τους, καθώς και στην κατασκευή ρομπότ και σε πολλούς άλλους ακόμα. Φυσικά, δεν βρισκόμαστε στο σημείο όπου οι υπολογιστές μπορούν να σκέφτονται όπως ο άνθρωπος, όμως βρισκόμαστε σε καλό δρόμο για να φτάσουμε σε αυτό. Η τεχνητή νοημοσύνη βρίσκει εφαρμογή στον πραγματικό κόσμο με διαφορετικούς τρόπους όπως:

Υπολογιστική Όραση: είναι η ανάπτυξη συστημάτων τα οποία επεξεργάζονται βίντεο και εικόνες από τα οποία εξάγουν πληροφορίες και μοτίβα όπως ο άνθρωπος. Για παράδειγμα η Google χρησιμοποιεί αντίστροφη αναζήτηση εικόνας για να βρει παρόμοιες εικόνες στο διαδίκτυο.[10]

Επεξεργασία φυσικής γλώσσας: είναι η διαδικασία κατά την οποία, ένας υπολογιστής καταλαβαίνει την φυσική γλώσσα και ενεργεί με βάση αυτήν. Ένα παράδειγμα είναι η μηχανή

[9] Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017"

[10] Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017

αναζήτησης της Google στην οποία ο χρήστης κάνει αναζήτηση για κάτι σε φυσική γλώσσα και η μηχανή γυρνάει τα σωστά αποτελέσματα.[11]

Αναγνώριση ομιλίας: είναι η ικανότητα μια μηχανή να μπορεί να «ακούσει» και να καταλάβει τι λέμε. Για παράδειγμα ψηφιακοί βοηθοί, όπως η Siri της Apple, η οποία καταλαβαίνει σε ένα βαθμό τι λέμε και ενεργεί κατάλληλα.[12]

Έμπειρα συστήματα: τα οποία χρησιμοποιούνται για να μας παρέχουν συμβουλές και να μας βοηθούν να παίρνουμε αποφάσεις. Στο μάρκετινγκ χρησιμοποιούνται για την καλύτερη εύρεση στρατηγικής, για να προωθηθεί μια εταιρεία.[13]

Παιχνίδια: Στα παιχνίδια χρησιμοποιείται ώστε να μπορούν να ανταγωνιστούν τον άνθρωπο. Παραδείγματος χάρη το "Deep blue" σύστημα το οποίο νίκησε τον παγκόσμιο πρωταθλητή στο σκάκι Garry Kasparov το 1997.[14] [15]

Ρομποτική: Για την κατασκευή ρομπότ τα οποία συνδυάζουν πολλές λειτουργίες από τις παραπάνω και είναι σε θέση να εκτελούν πολλά διαφορετικά καθήκοντα.[16]

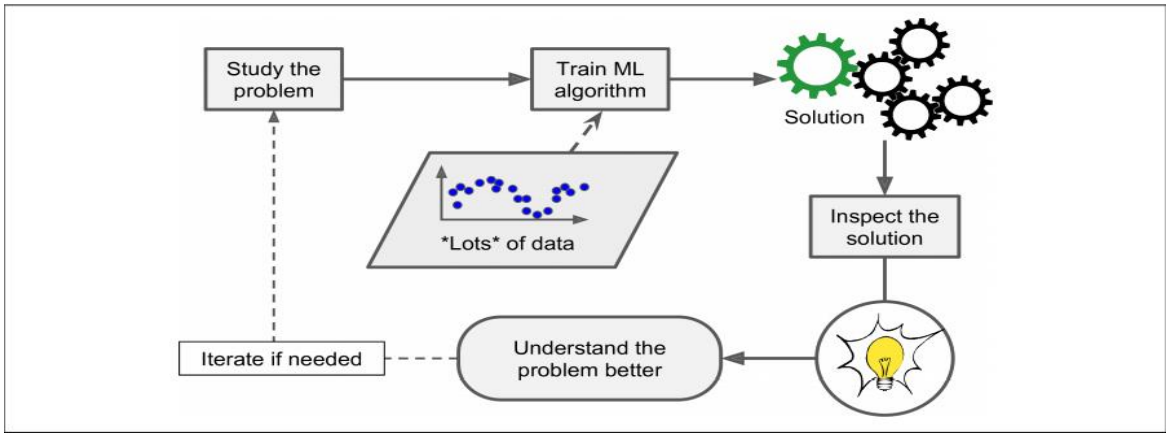
2.2 Τι είναι η μηχανική μάθηση και πως χρησιμοποιείται;

Η Μηχανική μάθηση είναι ένα υποσύνολο της τεχνητής νοημοσύνης και πρόκειται για τη διαδικασία κατά την οποία ένας υπολογιστής μπορεί να «μάθει» από δεδομένα.[17] Για παράδειγμα την ταξινόμηση ενός email αν είναι ανεπιθύμητο ή όχι. Για το παραπάνω πρόβλημα τροφοδοτούμε ένα σύστημα με δεδομένα το οποίο «μαθαίνει» από αυτά και είναι σε θέση να ανακαλύψει μοτίβα ώστε να μπορεί να πραγματοποιήσει την ταξινόμηση.[18]

Παρακάτω αναλύεται μια τυπική χρήση της μηχανικής μάθησης. Η διαδικασία ξεκινάει με την κατανόηση του προβλήματος, την συλλογή των δεδομένων που θα χρειαστούν για την εύρεση λύσης πάνω στο πρόβλημα. Έπειτα χρησιμοποιείται ένας ή περισσότεροι αλγόριθμοι μηχανικής μάθησης, στην συνέχεια αξιολογούνται τα αποτελέσματα και δίνουν

[11] Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017
[12] Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017
[13]Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017
[14] Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017
[15] Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015
[16] Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017
[17] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019
[18] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019

περισσότερες πληροφορίες για το πρόβλημα. Τέλος επαναλαμβάνονται τα παραπάνω βήματα αν το αποτέλεσμα δεν είναι ικανοποιητικό.



Εικόνα 2.2: Εικόνα διαδικασίας λύσης ενός προβλήματος μηχανικής μάθησης. Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019 σελίδα 9

2.2.1 Τύποι μηχανικής μάθησης

Οι τύποι της μηχανικής μάθησης χωρίζονται σε επιβλεπόμενη μάθηση, μη επιβλεπόμενη μάθηση, ήμι-επιβλεπόμενη μάθηση και ενισχυτική μάθηση. Παρακάτω θα δούμε τον κάθε τύπο ξεχωριστά.

Επιβλεπόμενη μάθηση (**Supervised learning**) κατά την οποία τα δεδομένα που περνάμε στο σύστημα για να εκπαιδευτεί περιέχουν και τα κατάλληλα αποτελέσματα τα οποία τα λέμε ετικέτες (Labels).[19] Στόχος της επιβλεπόμενης μάθησης είναι να χρησιμοποιήσουμε τα δεδομένα μας για να παράγουμε ένα μοντέλο το οποίο δέχεται σαν είσοδο ένα πίνακα με χαρακτηριστικά $\boldsymbol{X}$ και εξάγει πληροφορία η οποία αντιστοιχεί στην ετικέτα του αντίστοιχου πίνακα χαρακτηριστικών $\boldsymbol{X}$ [20]**.** Η επιβλεπόμενη μάθηση, που αποτελεί και τον τομέα της μηχανικής μάθησης με τους περισσότερους αλγορίθμους, χωρίζεται σε 2 υποκατηγορίες:

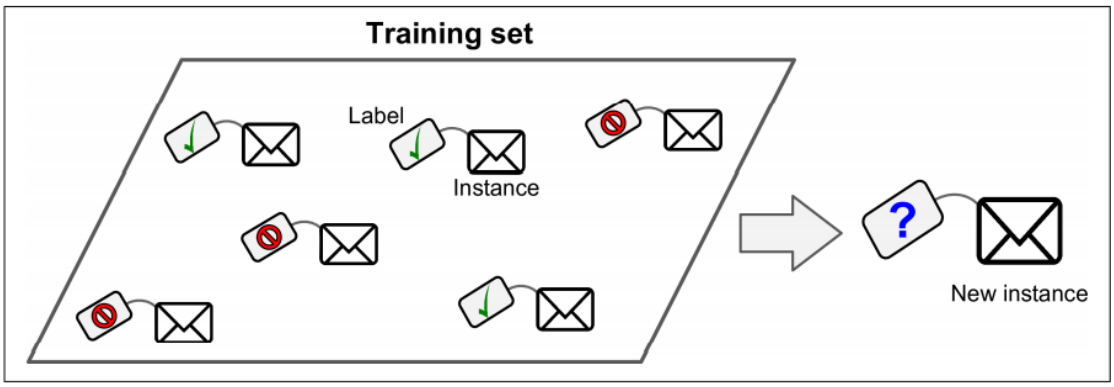
- Την **Ταξινόμηση,** σύμφωνα με την οποία το σύστημα πρέπει να «έρθει» σε ένα αποτέλεσμα από ένα διακριτό σύνολο ετικετών. Το παραπάνω παράδειγμα που την ταξινόμηση ενός email σε επιθυμητό ή όχι είναι ένα παράδειγμα ταξινόμησης. Ένα πρόβλημα ταξινόμησης λύνεται

[19] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019

[20] Andriy Burkov, "The hundred page – machine learning", January 2019

με τους αλγόριθμους ταξινόμησης οι οποίοι δέχονται σαν όρισμα μια συλλογή από δεδομένα με ετικέτες και παράγει ένα μοντέλο το οποίο δέχεται σαν όρισμα κάποια χαρακτηριστικά χωρίς ετικέτα και δίνει αποτέλεσμα μια ετικέτα. Οι πιο γνωστοί αλγόριθμοι ταξινόμησης είναι:[21]

- **Support Vector Machines (SVM)**
- **K – nearest neighbor (KNN)**
- **Logistic Regression**
- **Decision trees και Random Forests**
- **Νευρωνικά δίκτυα (NN)**



Εικόνα 2.3: Εικόνα παραδείγματος ταξινόμησης. Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019 σελίδα 8

- Την **Παλινδρόμηση,** όπου το σύστημα πρέπει να «έρθει» σε ένα αποτέλεσμα με μεγάλο και μη διακριτό εύρος τιμών. Ένα παράδειγμα είναι η εκτίμηση τιμής μια κατοικίας με χαρακτηριστικά όπως η περιοχή, ο αριθμός των δωματίων, ο αριθμός μπάνιων και άλλα ακόμα.

[21] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019

Εικόνα 2.4: Εικόνα παραδείγματος παλινδρόμησης. Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019 σελίδα 9

Μη επιβλεπόμενη μάθηση (**Unsupervised learning**) είναι η διαδικασία κατά την οποία τα δεδομένα αποτελούν μια συλλογή από παραδείγματα χωρίς την ύπαρξη ετικετών. Όπως και στην επιβλεπόμενη μάθηση, υπάρχει ένας πίνακας χαρακτηριστικών $\boldsymbol{X}$ και ο στόχος της μη επιβλεπόμενης μάθησης είναι να κατασκευαστεί ένα μοντέλο το οποίο θα δέχεται σαν είσοδο έναν πίνακα χαρακτηριστικών $\boldsymbol{X}$ και είτε τον ανακατασκευάζει σε άλλο πίνακα είτε σε μια τιμή η οποία θα μας βοηθήσει να την χρησιμοποιήσουμε για την επίλυση πρακτικών προβλημάτων. Για παράδειγμα, στην **μείωση διαστάσεων (dimensionality reduction**) το αποτέλεσμα του μοντέλου είναι ένας καινούργιος πίνακας ο οποίος θα περιέχει λιγότερα χαρακτηριστικά από τον αρχικό πίνακα χαρακτηριστικών. Άλλο παράδειγμα είναι στην ανίχνευση ανωμαλιών (Anomaly Detection) όπου το αποτέλεσμα είναι ένας αριθμός ο οποίος προσδιορίζει το πόσο διαφορετικό είναι το $\boldsymbol{X}$ από άλλα «τυπικά» παραδείγματα στα δεδομένα μας.[22] Τέλος ένα σημαντικό παράδειγμα μη επιβλεπόμενης μάθησης είναι η ομαδοποίηση (Clustering) κατά την οποία το μοντέλο μας δίνει ένα αναγνωριστικό (Id) για την ομάδα που ανήκει κάθε πίνακας χαρακτηριστικών $\boldsymbol{X}$. Οι πιο γνωστοί αλγόριθμοι μη επιβλεπόμενης μάθησης είναι :[23]

- **Ομαδοποίηση (Clustering)**
  - K-means

[22] Andriy Burkov, "The hundred page – maching learning", January 2019

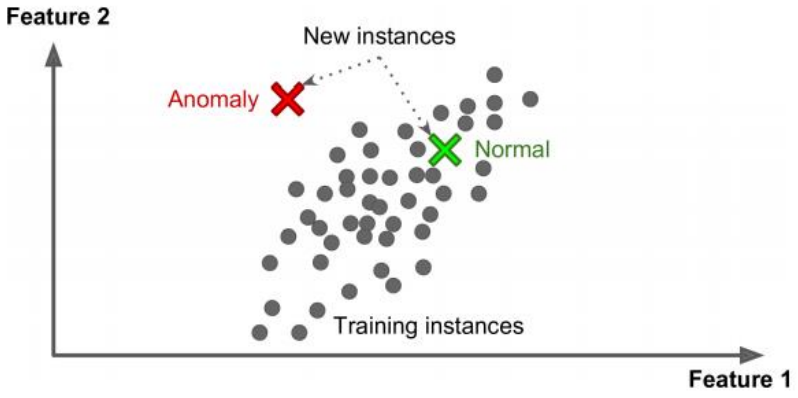
[23] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019

- DBScan
- Hierarchical Cluster Analysis (HCA)

- **Ανίχνευση ανωμαλιών (Anomaly Detection)**
  - One-class SVM
  - Isolation Forest

- **Για παρουσίαση και μείωση διαστάσεων (Visualization and dimensionality reduction)**
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - T-distributed Stochastic Neighbor Embedding (t-SNE)

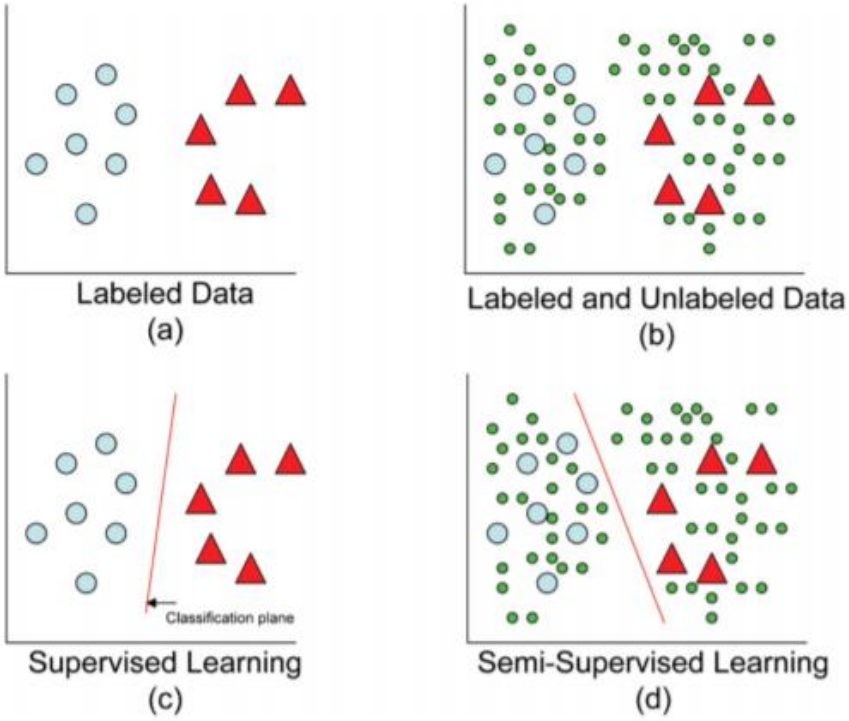
- **Εκμάθηση κανόνων συσχέτισης (Association Rule learning)**
  - Apriori
  - Eclat



Εικόνα 2.5: Εικόνα παραδείγματος ανίχνευσης ανωμαλιών. Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019 σελίδα 13

Ήμι-επιβλεπόμενη μάθηση (**Semi-supervised learning**) είναι η μίξη των δύο παραπάνω ειδών μηχανικής μάθησης δηλαδή τα δεδομένα περιέχουν και παραδείγματα με ή χωρίς ετικέτες. Συνήθως, το πλήθος των παραδειγμάτων χωρίς ετικέτες είναι μεγαλύτερο από τα δεδομένα με ετικέτες. Ο στόχος αυτής της τεχνικής είναι ο ίδιος με τον στόχο την

επιβλεπόμενης μάθησης με την ελπίδα ότι με την χρήση παραδειγμάτων χωρίς ετικέτες θα βοηθήσουν το μοντέλο να γίνει καλύτερο. Μερικοί αλγόριθμοι ημι-επιβλεπόμενης μάθησης είναι τα **Deep Belief Networks** (**DBNs**) τα οποία βασίζονται σε ένα άλλο δίκτυο μη επιβλεπόμενης μάθησης τα **Restricted Boltzmann Machines** (**RBMs**) τα οποία στοιβάζονται το ένα πάνω στο άλλο. Αυτά τα **RBMs** εκπαιδεύονται διαδοχικά με έναν μη επιβλεπόμενο τρόπο και έπειτα το σύστημα **ρυθμίζεται** (**Fine tuning**) ώστε να χρησιμοποιηθεί σε επιβλεπόμενες τεχνικές μάθησης. [24]

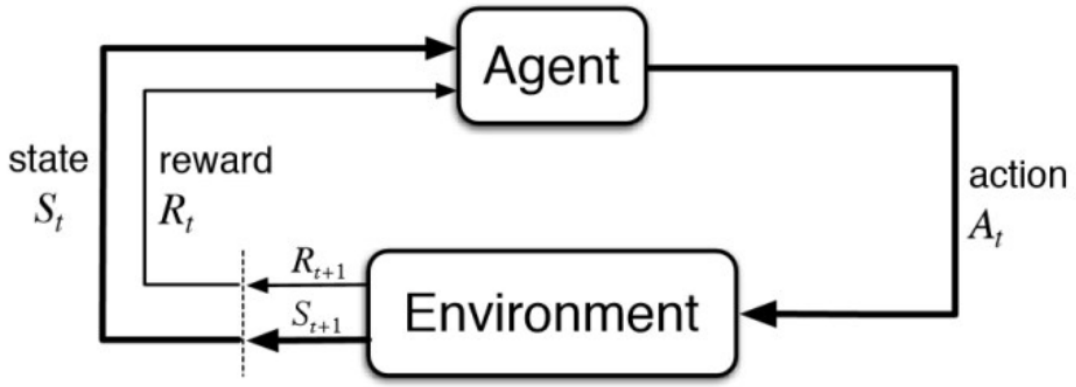


Εικόνα 2.6: Εικόνα ημι- επιβλεπόμνης μάθησης. Πηγή:Diedrik P. Kingma,Danilo J. Rezende,Shakir Mohamed,Max Welling, "Semi supervised learning with deep generative models" http://llcao.net/cu-deeplearning15/presentation/Semi-supervised%20Learning%20with%20Deep%20Generative%20Models.pdf (Ανακτήθηκε 22/1/2021)

Ενισχυτική μάθηση (**Reinforcement learning**) είναι ένα διαφορετικό είδος από τους παραπάνω τύπους. Πρόκειται για τη διαδικασία κατά την οποία η μηχανή (**Agent**) «ζει» σε ένα περιβάλλον και είναι σε θέση να αντιλαμβάνεται την κατάσταση του περιβάλλοντος που «ζει» σαν ένα πίνακα χαρακτηριστικών. Σε κάθε διαφορετική κατάσταση η μηχανή είναι σε θέση να εκτελεί ορισμένες ενέργειες. Διαφορετικές ενέργειες φέρνουν διαφορετικές

[24] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019

ανταμοιβές και μπορούν να μεταφέρουν την μηχανή σε διαφορετική κατάσταση. Στόχος της ενισχυτικής μάθησης είναι η μηχανή να «μάθει» μια πολιτική (Policy), η οποία είναι συνάρτηση F και δέχεται σαν είσοδο ένα πίνακα χαρακτηριστικών που προκύπτει από την κατάσταση που βρίσκεται η μηχανή και εκτελεί μια «ιδανική» ενέργεια για την συγκεκριμένη κατάσταση. Η ενέργεια θεωρείται ιδανική αν μεγιστοποιεί την ανταμοιβή που θα δεχτεί.



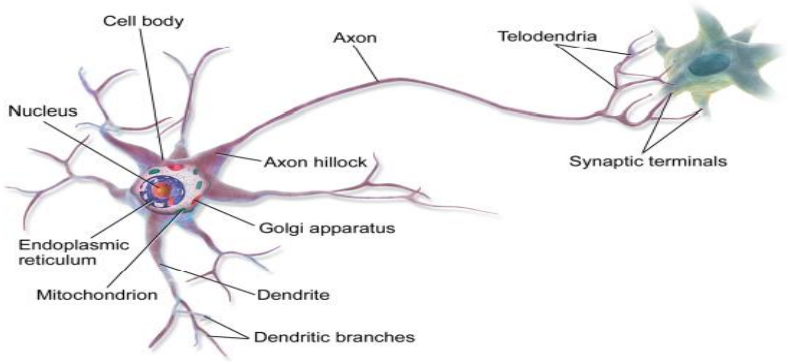
Εικόνα 2.7: Εικόνα διαδικασίας ενισχυτικής διαδικασίας. Πηγή:Shwetta Bhatt, "5 things you need to know about Reinforcemt learning", https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html (Ανακτήθηκε 21/1/2021)

2.3 Τι είναι τα τεχνητά νευρωνικά δίκτυα και πως χρησιμοποιούνται;

Τα τεχνητά νευρωνικά δίκτυα είναι αλγόριθμοι μηχανικής μάθησης. Αποτελούν ένα υπολογιστικό σύστημα το οποίο περιλαμβάνει ένα δίκτυο από συναρτήσεις για την κατανόηση δεδομένων που δέχεται σαν είσοδο όπως οι υπόλοιποι αλγόριθμοι μηχανικής μάθησης και την κατάληξη σε ένα ικανοποιητικό αποτέλεσμα[25]. Από που όμως προήλθε αυτό το είδος και γιατί προέκυψε; Η δημιουργία των τεχνητών νευρωνικών δικτύων εμπνεύστηκε από τον ανθρώπινο εγκέφαλο και τη λειτουργία του. Ο εγκέφαλος αποτελείται από ένα δίκτυο κυττάρων τα οποία καλούνται νευρώνες. Ο κάθε βιολογικός νευρώνας αποτελείται από ένα κυτταρικό σώμα το οποίο περιέχει τον πυρήνα και ένα μεγάλο αριθμό από άλλα οργανίδια και από πολλές επεκτάσεις οι οποίες λέγονται δενδρίτες (Dendrites), καθώς και μια μεγάλη επέκταση η οποία λέγεται νευροάξονας (axon). Το μήκος του νευροάξονα μπορεί να είναι μερικές φορές

[25] Deepai, «Neural Network», https://deepai.org/machine-learning-glossary-and-terms/neural-network (Ανακτήθηκε 29/1/2021)

μεγαλύτερο από το κυτταρικό σώμα έως και χιλιάδες φορές μεγαλύτερο από αυτό. Κοντά στο άκρο του ο νευροάξονας χωρίζεται σε πολλά κλαδιά τα οποία ονομάζονται τελόνδρια (Telondria), στην άκρη αυτών των κλαδιών υπάρχουν μικροσκοπικές δομές οι οποίες λέγονται συνοπτικοί ακροδέκτες που συνδέονται με τους δενδρίτες (ή απευθείας με το κυτταρικό σώμα). Οι βιολογικοί νευρώνες λαμβάνουν σύντομα ηλεκτρικά φορτία που ονομάζονται σήματα, (Signals) από άλλους νευρώνες μέσω αυτών των συνάψεων. Όταν ένας νευρώνας λαμβάνει επαρκή αριθμό σημάτων από άλλους νευρώνες τότε πυροδοτεί τα δικά του σήματα. Οι βιολογικοί νευρώνες φαίνεται να συμπεριφέρονται με απλό τρόπο, αλλά είναι οργανωμένοι σε ένα δίκτυο δισεκατομμυρίων νευρώνων, από τους οποίους ο κάθε νευρώνας συνδέεται με χιλιάδες άλλους νευρώνες[26].



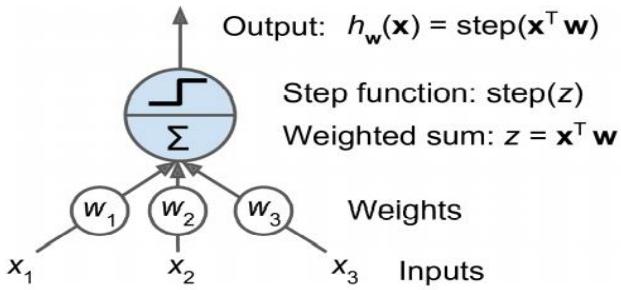
Εικόνα 2.8: Εικόνα κυτταρικού νευρώνα. Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019 σελίδα 280

Παρόλο που η θεωρία των τεχνητών νευρωνικών δίκτυών υπάρχει αρκετές δεκαετίες, τώρα πρόσφατα άρχισαν να εξερευνιούνται σε μεγαλύτερο βαθμό και αυτό συνέβη για δύο λόγους. Τα νευρωνικά δίκτυα απαιτούν μεγάλες ποσότητες δεδομένων για την σωστή εκπαίδευση τους σε σχέση με άλλους αλγορίθμους, για παράδειγμα για την δημιουργία οχημάτων που οδηγούν μόνα τους χρειάζονται εκατομμύρια εικόνες και χιλιάδες ώρες από βίντεο για την εκπαίδευσή τους. Επιπλέον, ένας λόγος είναι ότι για την εκπαίδευση τους χρειάζεται αρκετή υπολογιστική δύναμη για να μειωθεί ο χρόνος εκπαίδευσης. Πλέον, έχουν αναπτυχθεί κάρτες γραφικών που μας επιτρέπουν παράλληλες εκπαιδεύσεις και σε

[26] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019

μεγαλύτερες ταχύτητες[27]. Τα τεχνητά νευρωνικά δίκτυα αποτελούνται από νευρώνες που συνδέονται μεταξύ τους, ο κάθε νευρώνας είναι μια μαθηματική συνάρτηση. Ένα από τα πιο απλά μοντέλα είναι ο **αναγνωριστής** (**Perceptron**), ο οποίος εφευρέθηκε το 1957 από τον Frank Rosenblatt και βασίζεται σε ένα διαφορετικό νευρώνα, την **λογική μονάδα κατωφλίου** (**TLU**) όπου οι τιμές εισόδου και οι τιμές εξόδου είναι αριθμοί και κάθε είσοδος συνδέεται με ένα **βάρος** (**Weight**). Στην συνέχεια ο αλγόριθμος υπολογίζει ένα άθροισμα των εισόδων πολλαπλασιασμένων με το εκάστοτε βάρος της κάθε εισόδου

$z = w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n = X^T W$ και τέλος υπολογίζεται μια **συνάρτηση βήματος** (**Step Function**), η οποία δέχεται σαν είσοδο το άθροισμα που υπολογίσθηκε προηγουμένως h(x) = step(z)[28].

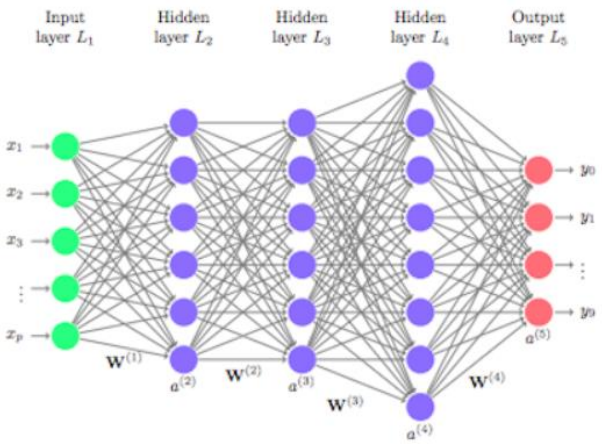


Εικόνα 2.9: Εικόνα τεχνητού νευρώνα Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, Ιούνιος 2019 σελίδα 282

Στα τεχνητά νευρωνικά δίκτυα συνηθίζεται να χρησιμοποιείται ο όρος **βαθιά μάθηση** (**Deep Learning**). Με αυτήν την έννοια, εννοείται ένα μοντέλο το οποίο αποτελείται από πολλά **κρυφά πεδία** (**Hidden Layers**), όπου το κάθε πεδίο αποτελείται από νευρώνες και ο κάθε νευρώνας λειτουργεί όπως ο αναγνωριστής που αναφέραμε παραπάνω. Πιο συγκεκριμένα, δέχεται κάποιες τιμές εισόδου στις οποίες εφαρμόζει μια γραμμική συνάρτηση και έπειτα στο αποτέλεσμα της γραμμικής συνάρτησης εφαρμόζει συνήθως μια μη γραμμική **συνάρτηση ενεργοποίησης** (**Activation Function**).

[27] Mathworks, "What Deep Learning?", "3 things you need to know", https://www.mathworks.com/discovery/deep-learning.html (Ανακτήθηκε 11/2/2021)

[28] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019



Εικόνα 2.10: Εικόνα βαθύ νευρωνικού δικτύου. Πηγή: Sunpark, "It's Deep Learning Times:A New Frontier of Data", Δεκέμβριος 2019, https://towardsdatascience.com/its-deep-learning-times-a-new-frontier-of-data-a1e9ef9fe9a8 (Ανακτήθηκε 11/2/2021)

Τα νευρωνικά δίκτυα βρίσκουν εφαρμογή σε διάφορες βιομηχανίες από την αυτόματη οδήγηση μέχρι και σε ιατρικές υπηρεσίες. Ορισμένες εφαρμογές που θα αναφέρουμε σαν παράδειγμα είναι οι εξής[29]:

- **Αυτόματη οδήγηση**: τα νευρωνικά δίκτυα χρησιμοποιούνται για να μάθουν πως να παρατηρούν αντικείμενα όπως πινακίδες, άλλα οχήματα, περαστικούς αλλά και άλλα.

- **Ιατρική έρευνα**: Ερευνητές χρησιμοποιούν νευρωνικά δίκτυα για να ανιχνεύσουν καρκινικά κύτταρα. Ένα παράδειγμα είναι οι ομάδες στο πανεπιστήμιο της Καλιφόρνιας (UCLA) που κατασκεύασαν ένα μικροσκόπιο το οποίο έχει την δυνατότητα να προσδιορίζει μια συλλογή από δεδομένα μεγάλων διαστάσεων για να εκπαιδεύσουν ένα νευρωνικό δίκτυο το οποίο προσδιορίζει καρκινικά κύτταρα.
- **Ηλεκτρονικές συσκευές**: Τα νευρωνικά δίκτυα χρησιμοποιούνται για την αυτόματη ανάλυση και επεξεργασίας ήχου και ομιλίας. Για παράδειγμα, οικιακές συσκευές που αντιλαμβάνονται την φωνή και είτε απαντούν είτε εκτελούν εντολές που τους δίνονται.

[29] Mathworks, "What Deep Learning?", "3 things you need to know", https://www.mathworks.com/discovery/deep-learning.html (Ανακτήθηκε 11/2/2021)

Τα νευρωνικά δίκτυα χρησιμοποιούνται σε πολλά ακόμη πεδία που δεν δύναται να απαριθμηστούν στην παρούσα έρευνα. Ωστόσο, το πλέον σημαντικό και αξιοσημείωτο είναι ότι τα νευρωνικά δίκτυα βρίσκονται ακόμα στην αρχή και πως στο μέλλον θα εξελιχθούν σε μεγαλύτερο βαθμό. Θα βοηθήσουν στην δημιουργία μιας γέφυρας για τον άνθρωπο και την τεχνητή νοημοσύνη και θα τον φέρουν πιο κοντά στην πραγματική μορφή της.

3. Ήδη υπάρχουσες τεχνολογίες στην ανάλυση γονιδιακής έκφρασης

Παρακάτω αναφέρονται ορισμένες αναλύσεις που έχουν πραγματοποιηθεί πάνω σε δεδομένα γονιδιακής έκφρασης με την χρήση νευρωνικών δικτύων και θα παρατηρηθεί τι προσφέρουν διαφορετικές υλοποιήσεις αυτών.

3.1 Deep Net

Το Deep Net είναι ένα πρότζεκτ που υλοποιήθηκε από τους D.Urda, J.Montes-Torres, F.Moreno, L.Franco και J.M.Jerez και αναφέρεται σε μια σύγκριση τεχνικών μηχανικής μάθησης. Συγκεκριμένα συγκρίνονται δύο διαφορετικά μοντέλα νευρωνικών δικτύων τα οποία διαφέρουν μόνο στην διαδικασία επιλογής χαρακτηριστικών (Feature Selection), καθώς και σε ένα πιο απλό αλγόριθμο μηχανικής μάθησης με όνομα LASSO (Least absolute shrinkage and selection operator) ο οποίος έχει σαν κύριος σκοπό την επιλογή χαρακτηριστικών και την γενίκευση (Regularization). Χρησιμοποιήθηκαν τρεις δημόσιες βάσεις δεδομένων καρκίνου για την συγκέντρωση των δεδομένων και την υλοποίηση της έρευνας τους. Αναλυτικότερα επεξεργάστηκαν δεδομένα RNA ακολουθιών από το TCGA και αναλύθηκαν 3 σύνολα δεδομένων τα οποία είχαν ήδη επεξεργαστεί μέσω της RSEM διαδικασίας. Τα δεδομένα που συγκέντρωσαν βασίζονται σε τρεις κατηγορίες καρκίνου, τον καρκίνο του μαστού (BRCA), αδενοκαρκίνωμα του εντέρου (COAD) και καρκίνο των νεφρών (KIPAN). Παρακάτω διαφορετικές υλοποιήσεις των μοντέλων των νευρωνικών δικτύων:

-Στο πρώτο μοντέλο χρησιμοποιούν την στατιστική διαδικασία t-test για να συγκρίνουν γονιδιακές εκφράσεις. Τα γονίδια που είχαν κατώφλι (P-value threshold) λιγότερο από 0.001 διατηρούνταν και έπειτα εισέρχονταν σε μια διαδικασία μείωσης χαρακτηριστικών συσχέτισης μέχρι ο αριθμός γονιδίων που διατηρούνταν ήταν ίσος με τον αριθμό γονιδίων που διατηρούνταν από τον αλγόριθμο LASSO.

-Στο δεύτερο μοντέλο χρησιμοποίησαν τον αλγόριθμο LASSO για να επιλέξουν τα πιο σημαντικά γονίδια. Εφόσον είχαν επιλεχτεί τα γονίδια στην συνέχεια δόθηκαν σαν είσοδο σε κοινό προς τα εμπρός τροφοδοτούμενο νευρωνικό δίκτυο (Feed Forward Neural Net) με πολλά κρυφά στρώματα (Hidden Layers).

Το αποτέλεσμα ήταν ότι ο αλγόριθμος LASSO είχε πολύ καλύτερο χρόνο εκτέλεσης από τις άλλες δύο υλοποιήσεις και καλύτερες ή ίδιες τιμές αποτελεσμάτων[30].

Dataset	Model	AUC	#genes	time (mins.)
BRCA	*Lasso*	0.65 [0.62, 0.67]	285.54 ± 25.83	501.79
	$DeepNet_i$	0.62 [0.58, 0.65]	242.02 ± 8.01	2294.83
	$DeepNet_{ii}$	0.65 [0.63, 0.68]	285.54 ± 25.83	9768.37
COAD	*Lasso*	0.57 [0.52, 0.63]	69.64 ± 11.63	30.89
	$DeepNet_i$	0.58 [0.54, 0.62]	37.29 ± 1.52	2699.84
	$DeepNet_{ii}$	0.57 [0.52, 0.61]	69.64 ± 11.63	2370.15
KIPAN	*Lasso*	0.77 [0.76, 0.78]	268.81 ± 32.54	93.60
	$DeepNet_i$	0.72 [0.68, 0.75]	201.64 ± 3.44	2633.52
	$DeepNet_{ii}$	0.75 [0.73, 0.78]	268.81 ± 32.54	9281.08

Εικόνα 3.1: Εικόνα αποτελεσμάτων Deep Net. Πηγή: D.Urda, J.Montes-Torres, F.Moreno, L.Franco, J.M.Jerez, "Deep Learning to Analyze RNA-Seq Gene Expression Data", https://link.springer.com/chapter/10.1007/978-3-319-59147-6_5, 18 Μαΐου 2017 (Ανακτήθηκε 15/2/2021)

3.2 Deep Chrome

Το Deep Chrome βασίζεται σε ένα ειδικό είδος νευρωνικών δικτύων που ονομάζεται **Συνελικτικά Νευρωνικά Δίκτυα.** (**Convolutional Neural Networks**). Σκοπό έχουν την εκμάθηση αλληλεπιδράσεων μεταξύ **Ιστονικών Τροποποιήσεων** (**Histone Modifications**) για την πρόβλεψη γονιδιακών εκφράσεων. Τα συνελικτικά νευρωνικά δίκτυα έχουν γίνει αρκετά δημοφιλή στις μέρες μας και χρησιμοποιούνται συνήθως για την ανάλυση εικόνων, την επεξεργασία φυσική γλώσσας και σε πολλούς τομείς ακόμα[31]. Έχουν την ικανότητα να μειώσουν σε αρκετά ικανοποιητικό βαθμό τον αριθμό των παραμέτρων σε ένα νευρωνικό δίκτυο χωρίς να χαθεί σημαντική πληροφορία σε σχέση με ένα απλό εμπρός τροφοδοτούμενο

[30] D.Urda, J.Montes-Torres, F.Moreno, L.Franco, J.M.Jerez, "Deep Learning to Analyze RNA-Seq Gene Expression Data", https://link.springer.com/chapter/10.1007/978-3-319-59147-6_5, 18 May 2017 (Ανακτήθηκε 15/2/2021)

[31] Ritambhara Singh, Jack Lanchantin, Gabriel Robins, Yanjun Qi, "DeepChrome: deep-learning for predicting gene expression from histone modifications", 1 September 2016, https://academic.oup.com/bioinformatics/article/32/17/i639/2450757?login=true (Ανακτήθηκε 3/7/2021)

νευρωνικό δίκτυο (Feed Forward Neural Network)[32]. Υποστηρίζεται ότι το δίκτυο είναι σε θέση να «αντιληφθεί» γειτονικές αλλά και μεγαλύτερου εύρους αλληλεπιδράσεις μεταξύ των δεδομένων εισόδου και ταυτόχρονα να εξαγάγει σημαντικά χαρακτηριστικά. Για να μπορούν να κατανοήσουν τις αλληλεπιδράσεις μεταξύ των ιστονικών σημείων, εφάρμοσαν μια μέθοδο βασισμένη στην βελτιστοποίηση για την απεικόνιση συνδυαστικών σχέσεων από τα μοντέλα. Μερικές συνοπτικές για το Deep Chrome είναι οι εξής:

- Αποτελεί το πρώτο μοντέλο νευρωνικών δικτύων που επικεντρώνεται στην πρόβλεψη γονιδιακών εκφράσεων χρησιμοποιώντας ιστονικές τροποποιήσεις. Χρησιμοποιείται σε σήματα ιστονικών τροποποιήσεων από 56 διαφορετικά είδη κυττάρων.
- Ξεπερνάει σε απόδοση άλλα μοντέλα μηχανικής μάθησης όπως SVM και Random Forests που έχουν χρησιμοποιηθεί για την πρόβλεψη 56 έργων.
- Μπορεί να οπτικοποιήσει συνδυαστικές σχέσεις μεταξύ διαφορετικών σημάτων ιστονικών τροποποιήσεων.

Για την ανάλυση που πραγματοποιήθηκε χρησιμοποιήθηκαν 10.000 ζευγάρια βάσεων από την περιοχή του DNA γύρω από το σημείο που ξεκινάει η διαδικασία της (TSS -Transcription starting site) **μεταγραφής** (**Transcription**) για κάθε γονίδιο, τα οποία ζευγάρια διαιρέθηκαν σε ομάδες μεγέθους 100 ζευγαριών βάσεων. Έπειτα, επιλέχτηκαν πέντε βασικές υπογραφές ιστονικών τροποποιήσεων από την REMC βάση δεδομένων. Επιλέχθηκαν οι συγκεκριμένες υπογραφές διότι βρίσκονται σε όλα τα είδη κυττάρων που χρησιμοποιήθηκαν. Αυτό είχε ως αποτέλεσμα, κάθε γονίδιο εισόδου να αποτελείται από έναν πίνακα 5×100, ώστε οι γραμμές να είναι οι ιστονικές τροποποιήσεις και οι κολώνες να αποτελούν τις διαφορετικές ομάδες μεγέθους 100. Οι ετικέτες που το σύστημα έπρεπε να προβλέψει ήταν 1 και -1. Αυτές καθόριζαν το αν το επίπεδο της γονιδιακής έκφρασης είναι υψηλό ή χαμηλό αντίστοιχα. Το Deep Chrome αποτελείται από **συνελικτικά επίπεδα** (**Convolution layers**) που χρησιμοποιούν κάποια φίλτρα, τα οποία συνδυαστικά με τα δείγματα εισόδου παράγουν κάποιους **χάρτες χαρακτηριστικών** (**Feature Maps**). Έπειτα εφαρμόζεται η μη γραμμική συνάρτηση της **διορθωμένης γραμμικής μονάδας** (**Rectified Linear Unit** – **ReLU**) μετά από κάθε συνελικτικό επίπεδο. Στην συνέχεια εφαρμόστηκε ένα επίπεδο **μέγιστης ομαδοποίησης** (**Max-Pooling**) στο αποτέλεσμα που προήλθε από την παραπάνω μη

[32] Andriy Burkov, "The Hundred page – Machine Learning", January 2019

γραμμική συνάρτηση και το οποίο έχει την ιδιότητα να διατηρεί τα πιο εκφρασμένα χαρακτηριστικά. Ακόμη, χρησιμοποιήθηκε ένα **επίπεδο εγκατάλειψης** (**Dropout Layer**) το οποίο τυχαία μηδένιζε ένα ποσοστό των αποτελεσμάτων που προέρχονταν από το επίπεδο **μέγιστης ομαδοποίησης**. Τέλος, τα αποτελέσματα από το επίπεδο εγκατάλειψης περάστηκαν σαν είσοδο σε ένα **εμπρός τροφοδοτούμενο νευρωνικό δίκτυο** (**Feed Forward Neural Network**), το οποίο ταξινομούσε τα δείγματα με την χρήση της συνάρτησης **SoftMax**. Για να προκύψει ένα συνολικό αποτέλεσμα, έπρεπε να συγκριθεί το συνελικτικό νευρωνικό δίκτυο που υλοποιήθηκε με άλλους αλγορίθμους μηχανικής μάθησης. Επιλέχθηκαν δύο άλλοι αλγόριθμοι μηχανικής μάθησης ο **SVM** και ο **Random Forest.** Το αποτέλεσμα μετρήθηκε με την διαδικασία της **περιοχής κάτω από την καμπύλη** (**Area Under The Curve – AUC**) με το συνελικτικό δίκτυο να δίνει μέση μέτρηση 0.80, ο αλγόριθμος SVM είχε καλύτερη μέτρηση στην τιμή 0.66 και ο αλγόριθμος **Random Forest** με τιμή 0.59, έδινε το χειρότερο αποτέλεσμα[33].

3.3 Dee Pathology

Το Dee Pathology, είναι μια ερεύνα η οποία βασίζεται στην χρήση μιας αρχιτεκτονικής νευρωνικών δικτύων με όνομα **Autoencoders**. Αυτή η αρχιτεκτονική αποτελείται από δύο μέρη, τον **κωδικοποιητή** (**encoder**) και τον **αποκωδικοποιητή** (**decoder**). Στόχος του δικτύου, είναι να δεχθεί σαν είσοδο τα αρχικά χαρακτηριστικά και να προσπαθήσει να τα συμπιέσει σε μικρότερο χώρο διαστάσεων (**latent space**) χωρίς να χαθεί σημαντική πληροφορία. Στην συνέχεια, από τον χώρο αυτό, προσπαθεί να δημιουργήσει από την αρχή τα δεδομένα που του δόθηκαν σαν είσοδο.

$$E \rightarrow f(x) = l$$

$$D \rightarrow G(l) \sim X$$

Όπου $\boldsymbol{E}$ είναι το 1ο μέρος του κωδικοποιητή το οποίο αποτελείται από μια συνάρτηση $\boldsymbol{f}$ με αποτέλεσμα $\boldsymbol{l}$. Το $\boldsymbol{D}$ είναι ο αποκωδικοποιητής που δέχεται το $\boldsymbol{l}$ σαν είσοδο στην συνάρτηση $\boldsymbol{G}$ και δίνει αποτέλεσμα όσο πιο κοντά γίνεται στα αρχικά δεδομένα.

[33] Ritambhara Singh, Jack Lanchantin, Gabriel Robins, Yanjun Qi, "DeepChrome:deep-learning for predicting gene expression from histone modifications", 29 August 2016, https://bit.ly/38R3Vtb (Ανακτήθηκε 17/3/2021)

Η έρευνα που πραγματοποιήθηκε, βασίστηκε σε τέσσερις διαφορετικές υλοποιήσεις των Autoencoders, οι οποίες δεχόντουσαν σαν είσοδο δεδομένα mRNA μεταγραφών από 10,750 κλινικά δείγματα από 34 κατηγορίες (33 διαφορετικά είδη καρκίνου και μια κατηγορία υγιών δειγμάτων) και στην συνέχεια τα κωδικοποιούσαν σε έναν διανυσματικό πίνακα μεγέθους 8 θέσεων. Έπειτα, από τον διανυσματικό πίνακα προσπαθούσαν να ανακτήσουν τα προφίλ έκφρασης mRNA και miRNA, τον ιστό και τον τύπο ασθένειας. Συγκεκριμένα χρησιμοποιήθηκαν:

1. **Contractive Autoencoders**: Αποτελεί εναλλακτική τεχνικής γενίκευσης των κλασσικών Autoencoder. Ο γενικευμένος όρος των δικτύων αυτών, είναι το σύνολο των τετραγώνων όλων των μερικών διαφορικών εξισώσεων του διανυσματικού πίνακα (latent vector), με σεβασμό της κάθε διάστασης του προηγούμενου επιπέδου. Αποτελείται από ένα συντελεστή εξισορρόπησης **λ** και από έναν όρο $\left|\left|J_{\psi}(x)\right|\right|_{f}^{2}$ ο οποίος είναι ο κανόνας Frobenius ενός πίνακα Τζακόμπι (Jacobian matrix). Στόχος αυτής της τεχνικής είναι το αποτέλεσμα να είναι ευέλικτο σε μικρές παραλλαγές των δεδομένων εισόδου.

$$g = \sum_{x \in D} L\left(x, \tilde{x}(\varphi, \psi, \chi)\right) + \lambda \left|\left|J_{\psi}(x)\right|\right|_{f}^{2}$$

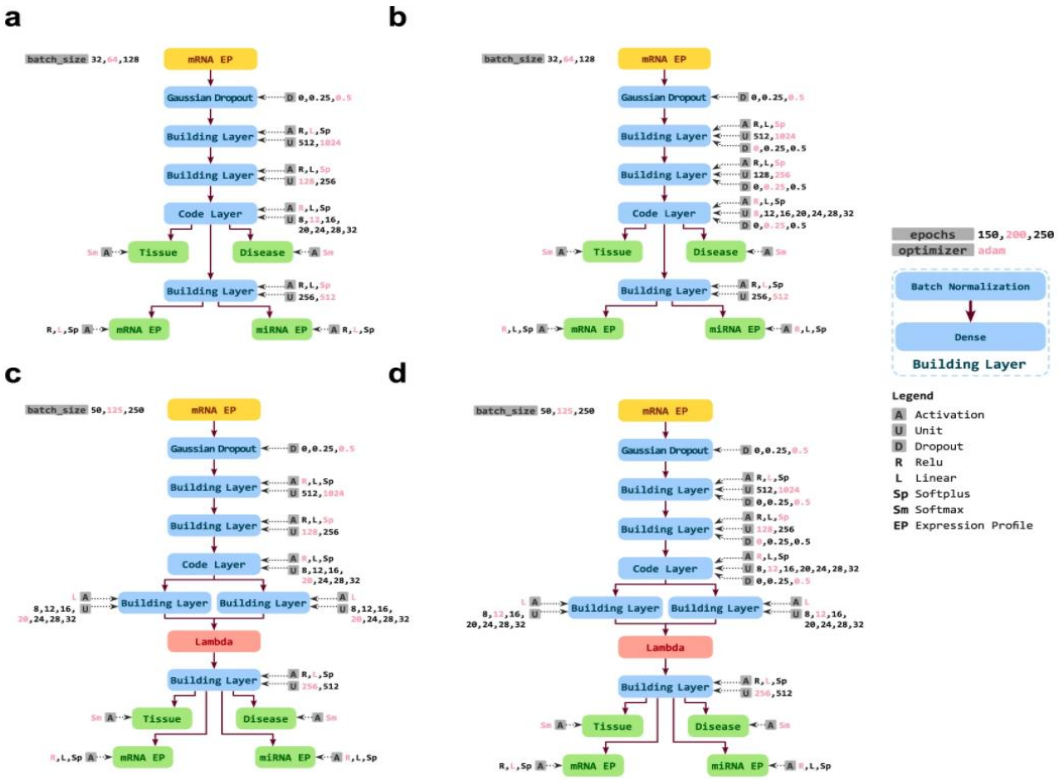
2. **Contractive Dropout Autoencoders:** Η διαφορά αυτής της αρχιτεκτονικής με την παραπάνω είναι ότι χρησιμοποιήθηκε ένα ειδικό επίπεδο γενίκευσης (Regularization) με όνομα **Επίπεδο εγκατάλειψης** (**Dropout Layer**). Η ιδιαιτερότητα αυτών των επιπέδων είναι να θέτουν κατά την διαδικασία της εκπαίδευσης κάποιους νευρώνες σαν μη διαθέσιμους, με αποτέλεσμα να μη εκπαιδεύονται. Ο λόγος που αυτή η διαδικασία είναι χρήσιμη, είναι για να γενικευτεί ένα δίκτυο και να μειωθούν οι αλληλεξαρτήσεις μεταξύ των νευρώνων[34]. Η εγκατάλειψη αποτελεί έναν από τους πιο γνωστούς τρόπους γενίκευσης για την αποφυγή της **υπερβολικής προσαρμογής** (**Over fitting**).

3. **Variational Autoencoders**: Είναι ένας ειδικός τύπος **Autoencoder** με περισσότερους περιορισμούς πάνω στην κωδικοποιημένη αναπαράσταση. Θεωρούν ότι υπάρχει μια

[34] Amar Budhiraja, "Dropout in (Deep) Machine Learning", 15 December 2016, https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (Ανακτήθηκε 3/7/2021)

τυχαία μεταβλητή η οποία μπορεί να μας δώσει τα δεδομένα εισόδου μέσω μια στοχαστικής χαρτογράφησης. Γενικά, οι Autoencoders δεν δύναται να παράγουν ικανοποιητικά αποτελέσματα, όμως οι **Variational Autoencoders** μπορούν να κατανοήσουν ένα μοντέλο το οποίο μπορεί να παράγει καινούργια δείγματα από τον διανυσματικό χώρο για αυτό και κατατάσσονται στα παραγωγικά μοντέλα (**Generative Models**).

4. **Variational Dropout Autoencoders**: Η διαφορά με το παραπάνω μοντέλο είναι η ίδια με την αρχιτεκτονική των **Contractive Autoencoders**. Εφαρμόζεται επίσης ένα επίπεδο εγκατάλειψης ανάμεσα από τα **ολικά συνδεδεμένα επίπεδα** (**Fully Connected Layers**).



Εικόνα 3.2: Εικόνα διαφορετικών αρχιτεκτονικών Dee Pathology. Πηγή: Behrooz Azarkhalili, Ali Saberi, Hamidreza Chitsaz, Ali Sharifi-Zarchi, "DeePathology: Deep Multi-Task Learning for Inferring Molecular Pathology from Cancer Transcriptome" , https://www.nature.com/articles/s41598-019-52937-5#Sec15 (Ανακτήθηκε 23/2/2021)

Το αποτέλεσμα ήταν ότι οι αρχιτεκτονικές CAE και DCAE είχαν καλύτερη απόδοση από τις άλλες δύο αρχιτεκτονικές. Η τεχνική DCAE, είχε την καλύτερη κωδικοποίηση των χαρακτηριστικών εισόδου αρχικού μεγέθους 19,671 θέσεων σε διανυσματικό πίνακα μεγέθους 8. Στην διαδικασία ταξινόμησης, όλοι οι αλγόριθμοι είχαν σαν είσοδο όλο το mRNA ενός δείγματος, μετά την δοκιμή διάφορων γνωστών αλγορίθμων μηχανικής μάθησης και ενός νευρωνικού δικτύου με πολλά επίπεδα το αποτέλεσμα ήταν ότι το νευρωνικό δίκτυο είχε καλύτερο ποσοστό ακρίβειας (accuracy). Συγκεκριμένα, το νευρωνικό δίκτυο έφτασε ποσοστό 98.1% για την ταξινόμηση ιστού και 95.2% για ταξινόμηση ασθενειών σε σχέση με 95.1% και 90.9% που είχαν οι τεχνικές μηχανικής μάθησης αντίστοιχα. Στην συνέχεια, χρησιμοποίησαν αλγόριθμους όπως τον PCA, Kernel PCA και άλλους, για να μειώσουν τις διαστάσεις της εισόδου σε 8 και το σύγκριναν με το διανυσματικό χώρο μεγέθους 8 που είχαν κατασκευάσει με την χρήση των Autoencoders. Το αποτέλεσμα ήταν το μέγιστο ποσοστό ακρίβειας στην διαδικασία ταξινόμησης, όταν σαν είσοδο υπήρχαν τα δεδομένα που είχαν παραχθεί από τον PCA σε ποσοστό 38.7% και 33.2% για την ταξινόμηση ιστού και ασθένειας αντίστοιχα. Ωστόσο, με την χρήση των δεδομένων που είχαν παραχθεί από τους Autoencoders και την χρήση **μοντέλων συνολικής μάθησης (Ensemble learning models)** το ποσοστό ακρίβειας έφτασε το 93.7% και 89.3% για ταξινόμηση ιστού και ασθένειας. Τα ποσοστά ακρίβειας ανήλθαν σε >=99% για 9 τύπους καρκίνου και >=95% για 25 τύπους καρκίνου μαζί με τα υγιή δείγματα. Υπήρξαν μόνο τρείς τύποι καρκίνου από τους 33 που είχαν ποσοστό ακρίβειας μικρότερο από 90%. Τέλος, για την εισαγωγή των δεδομένων εισόδου στα υπολογιστικά μοντέλα, προστέθηκε ένα επίπεδο εγκατάλειψης, με αποτέλεσμα, τυχαίες τιμές εισόδου να τεθούν με την τιμή μηδέν. Με την διαδικασία αυτή, παρατηρήθηκε ότι τα νευρωνικά δίκτυα μπορούν να αντισταθούν στον θόρυβο και σε τιμές που μπορεί να μην υπάρχουν για κάποια χαρακτηριστικά και να εξάγουν ένα αρκετά ικανοποιητικό ποσοστό ακρίβειας[35].

[35] Behrooz Azarkhalili, Ali Saberi, Hamidreza Chitsaz, Ali Sharifi-Zarchi, "DeePathology: Deep Multi-Task Learning for Inferring Molecular Pathology from Cancer Transcriptome" **,** **https://www.nature.com/articles/s41598-019-52937-5#Sec15** **(** Ανακτήθηκε 23/2/2021 **)**

4. Η ανάλυση του προβλήματος

Γνωρίζουμε ότι τα γονίδια είναι υπεύθυνα για πολλές λειτουργίες των οργανισμών και η ανάλυση της έκφρασης τους είναι ένας τρόπος να κατανοήσουμε για ποιες λειτουργίες του οργανισμού αλλά και για ποιες ασθένειες, ποια γονίδια, είναι υπεύθυνα. Η έκφραση γονιδίων, είναι η διαδικασία που επιτρέπει σε ένα κύτταρο να προσαρμόζεται στις αλλαγές του περιβάλλοντος του[36]. Λειτουργεί σαν διακόπτης που ελέγχει πότε δημιουργούνται πρωτεΐνες στον οργανισμό και σε τι βαθμό. Με την ανάλυση της, έχουμε την δυνατότητα να εξάγουμε περισσότερες πληροφορίες για τις κυτταρικές λειτουργίες[37]. Φυσικά, η ανάλυση της έκφρασης γονιδίων κάθε χρόνο γίνεται ευκολότερη. Εξαιτίας της τεχνολογικής εξέλιξης, πολλοί αλγόριθμοι στον κλάδο της μηχανικής μάθησης χρησιμοποιούνται για την ανάλυση των γονιδίων, όπως και μέθοδοι στατιστικής που προσφέρουν μια ξεκάθαρή εικόνα για την λειτουργία των γονιδίων και τις μεταξύ τους σχέσεις. Στην περίπτωση μας, θα επικεντρωθούμε στην συγκέντρωση και ανάλυση γονιδιακών εκφράσεων για διαφορετικούς τύπους καρκίνου με την χρήση νευρωνικών δικτύων. Εκτός από την χρήση των νευρωνικών δικτύων θα χρησιμοποιηθούν και άλλοι αλγόριθμοι μηχανικής μάθησης και στο τέλος θα συγκριθούν ποιοι τρόποι ανάλυσης είναι καλύτεροι για το συγκεκριμένο πρόβλημα. Αυτή η σύγκριση είναι σημαντική για την αξιολόγηση της αποτελεσματικότητας των νευρωνικών δικτύων. Είναι γνωστό ότι τα νευρωνικά δίκτυα είναι κατάλληλα για την ανάλυση μεγάλου όγκου δεδομένων και στο μεγαλύτερο ποσοστό υπερτερούν σε απόδοση από τους κλασσικούς αλγορίθμους μηχανικής μάθησης. Οι γονιδιακές εκφράσεις αποτελούν ένα πρόβλημα μεγάλου όγκου δεδομένων διότι ένα δείγμα μπορεί να αποτελείται από χιλιάδες γονίδια. Για αυτό το λόγο θα αναλυθεί η απόδοση των νευρωνικών δικτύων στις γονιδιακές εκφράσεις και πόσο καλά θα αντιμετωπίσουν το πρόβλημα της κατάρας των διαστάσεων σε σχέση με τους αλγορίθμους της μηχανικής μάθησης. Συγκεκριμένα οι τύποι καρκίνου που θα αναλυθούν είναι οι παρακάτω[38]:

[36] YourGenome, "What is gene expression", https://www.yourgenome.org/facts/what-is-gene-expression (Ανακτήθηκε 11/2/2021)

[37] Bio-rad, «What is gene expression analysis», https://www.bio-rad.com/en-gr/applications-technologies/what-gene-expression-analysis?ID=LUSNINKSY (Ανακτήθηκε 11/2/2021)

[38] Broad Institute, Firehose http://gdac.broadinstitute.org/ (Ανακτήθηκε 16/1/2021)

1. Adrenocortical Carcinoma (ACC)	2. Bladder Urothelial Carcinoma (BLCA)
3. Breast Invasive Carcinoma (BRCA)	4. Cervical and Endocervical Cancers (CESC)
5. Cholangiocarcinoma (CHOL)	6. Colon Adenocarcinoma (COAD)
7. Lymphoid Neoplasm Diffuse Large B-cell Lymphoma (DLBC)	8. Esophageal Carcinoma (ESCA)
9. Glioblastoma multiforme (GBM)	10. Head and neck squamous cell carcinoma (HNSC)
11. Kidney Chromophobe (KICH)	12. Kidney renal clear cell carcinoma (KIRC)
13. Acute Myeloid Leukemia (LAML)	14. Brain Lower Grade Glioma (LGG)
15. Liver hepatocellular carcinoma (LIHC)	16. Lung adenocarcinoma (LUAD)
17. Lung Squamous cell carcinoma (LUSC)	18. Ovarian serous cystadenocarcinoma (OV)
19. Pheochromocytoma and Paraganglioma (PCPG)	20. Prostate adenocarcinoma (PRAD)
21. Rectum adenocarcinoma (READ)	22. Sarcoma (SARC)
23. Skin Cutaneous Melanoma (SKCM)	24. Stomach adenocarcinoma (STAD)
25. Testicular Germ Cell Tumors (TGCT)	26. Thyroid carcinoma (THCA)
27. Thymoma (THYM)	28. Uterine Corpus Endometrial Carcinoma (UCEC)

Πίνακας 4.1: Διαφορετικά είδη καρκίνου.

5. Η προτεινόμενη λύση

Για την αντιμετώπιση του προβλήματος, η ανάλυση θα επικεντρωθεί στην κατασκευή μοντέλων νευρωνικών δικτύων για την μείωση των χαρακτηριστικών και την ταξινόμηση αυτών. Για αυτό τον σκοπό θα χρησιμοποιηθούν διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων, που ειδικεύονται στην μείωση των χαρακτηριστικών. Συγκεκριμένα επιλέγονται οι αρχιτεκτονικές Autoencoders, Denoising Autoencoders, Variational Autoencoders. Εφόσον μειωθεί ο αριθμός των χαρακτηριστικών γίνεται χρήση δύο αρχιτεκτονικών νευρωνικών δικτύων για την διαδικασία ταξινόμησης, ένα απλό βαθύ νευρωνικό δίκτυο και ένα μοντέλο με όνομα Deep Cross. Για την αξιολόγηση των αποτελεσμάτων, θα χρησιμοποιηθούν και αλγόριθμοι μηχανικής μάθησης που θα εκτελέσουν τις παραπάνω διαδικασίες ώστε να συγκριθούν με τα νευρωνικά δίκτυα για να αναφερθούν διαφορές αλλά και ποια τεχνική είναι "καλύτερη".

6. Η υλοποίηση του Μοντέλου

6.1 Επιλογή και συλλογή δεδομένων

Για την ανάλυση που θα πραγματοποιηθεί χρησιμοποιήθηκαν δεδομένα από το **The Cancer Genome Atlas Program (TCGA)**, το οποίο αποτελεί μια τεράστια συλλογή από γονιδιωματικά δεδομένα. Το **TCGA** περιέχει μοριακά δεδομένα από πολλαπλές αναλύσεις όπως, **DNA SEQUENCING**, **RNA SEQUENCING**,**DNA METHYLATION**, **COPY NUMBER**, **ARRAY EXPRESSION** και διάφορες άλλες[39]. Για την διεξαγωγή της ανάλυσης θα χρησιμοποιηθούν δεδομένα για εκφράσεις γονιδίων τα οποία έχουν αναλυθεί με RNA sequence μέθοδο. Πρόκειται για μια τεχνική, με την οποία μπορεί να εξεταστεί η ποσότητα και η ακολουθία του RNA σε ένα δείγμα χρησιμοποιώντας **ακολουθίες επομένης γενιάς (next generation sequencing)**. Αναλύεται το **μεταγράφωμα (transcriptome)** από εκφράσεις γονιδίων που είναι κωδικοποιημένες στο RNA και αποτελεί τη βασική τεχνική που χρησιμοποιείται στις μέρες μας[40]. Τα δεδομένα που θα αναλυθούν αφορούν γονιδιακές

[39] SevenBridges, "The Cancer Genome Atlas" https://www.sevenbridges.com/tcga/ (Ανακτήθηκε στις 16/1/2021)

[40] Ruairi J MacKenzie, "RNA-seq:Basics, Applications and Protocol", Apr 06, 2018 https://www.technologynetworks.com/genomics/articles/rna-seq-basics-applications-and-protocol-299461 (Ανακτήθηκε στις 16/1/2021)

εκφράσεις από ασθενείς με καρκίνο. Παρέχουν πληροφορίες για περίπου 33 διαφορετικούς τύπους καρκίνου από 11.328 ασθενείς. Κατεβάσαμε δεδομένα από το http://gdac.broadinstitute.org/, για τους 28 τύπους καρκίνου που θα αναλυθούν, όπου το κάθε αρχείο περιέχει συγκεντρωτικά δεδομένα από το TCGA για το συγκεκριμένο είδος καρκίνου. Τα δεδομένα έχουν παραχθεί από το **HiSeq 2000** της **Illumina**, το οποίο είναι ένα μηχάνημα ακολουθιών επόμενης γενιάς, φιλικό προς τον χρήστη και αποτελείται από ένα εύρος διάφορων εφαρμογών. Μια από αυτές, είναι η επεξεργασία ακολουθιών σε πραγματικό χρόνο[41]. Έχει την δυνατότητα να επεξεργάζεται έναν αρκετά μεγάλο αριθμό δειγμάτων και να αποκωδικοποιεί περίπλοκα και πιο μεγάλα γονιδιώματα[42]. Στην συνέχεια τα δεδομένα που παράχθηκαν επεξεργάστηκαν από το εργαλείο **RSEM,** το οποίο είναι ένα εργαλείο λογισμικού φιλικό προς τον χρήστη και χρησιμοποιείται για τον ποσοτικό προσδιορισμό της μεταγραφής από δεδομένα ακολουθιών RNA. Χρησιμοποιεί τον αλγόριθμο προσδοκίας-μεγιστοποίησης (Expectation-Maximization) και μπορεί να λειτουργήσει με ή χωρίς αναφορά γονιδιώματος, καθώς αναφέρει μεταγραφές ανά εκατομμύρια χαρτογραφημένες αναγνώσεις (TPM)[43]. Το RSEM επέτρεψε έναν πιο αποδοτικό σχεδιασμό ποσοτικοποίησης ακολουθιών RNA ο οποίος αποτελεί μια σχετικά μεγάλη σε κόστος διαδικασία.[44] Τέλος, τα δεδομένα έχουν υποστεί ομαλοποίηση λογαριθμικά με βάση το δύο (**Log2 normalization** ή **Row normalization**). Ο λόγος που πραγματοποιείται αυτή η ομαλοποίηση είναι επειδή τα δείγματα μπορεί να έχουν διαφορετικό αριθμό αναγνώσεων (**Read Counts**) και η γονιδιακή έκφραση του κάθε δείγματος να είναι ανάλογη με τον αριθμό αναγνώσεων. Λόγω των διαφορετικών τιμών των γονιδίων ανάμεσα σε διαφορετικά δείγματα, οι περισσότερο εκφρασμένες τιμές θα μπορούσαν να κυριαρχήσουν μεταξύ των δειγμάτων. Για το λόγο αυτό, χρησιμοποιείται ο λογαριθμικός μετασχηματισμός, ο οποίος φέρνει τα γονίδια προς τα πάνω ή προς τα κάτω στην ίδια κλίμακα[4546]. Επομένως, ο λογαριθμικός μετασχηματισμός κάνει τα δείγματα πιο συμμετρικά

---

[41] TCGA,"HiSeq 2000 System User Guide",https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga/using-tcga/technology/illumina-hiseq2000-data-sheet (Ανακτήθηκε 3/7/2021)

[42] https://www.illumina.com/documents/products/datasheets/datasheet_hiseq2000.pdf

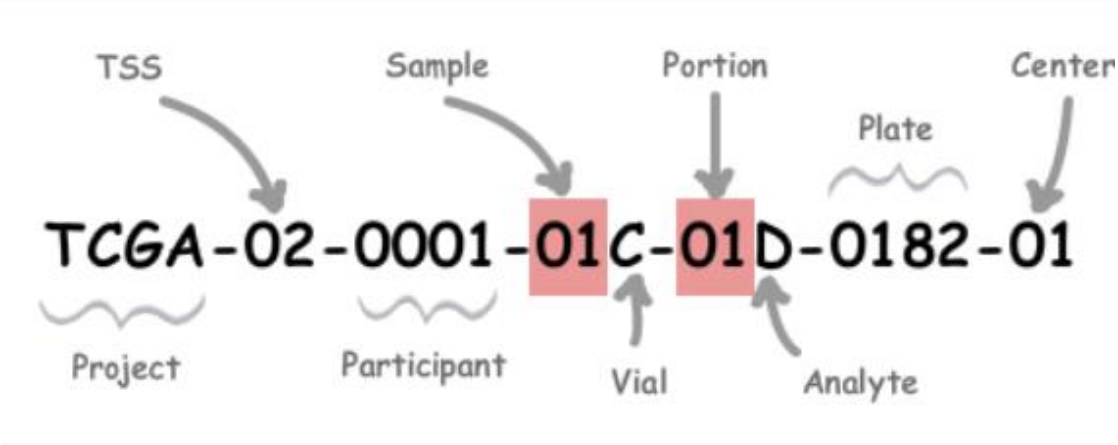
[43] BioinformaticsHome, https://bioinformaticshome.com/tools/rna-seq/descriptions/RSEM.html (Ανακτήθηκε 3/7/2021)

[44] Bo Li & Colin N Dewey, "RSEM :accurate transcript quanitification from RNA-Seq data with or without a reference genome", August 4 2011 https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-323 (Ανακτήθηκε 21/1/2021)

[45] Joshua Gould,Broad Institute,"Preprocess Dataset", https://genepattern.org/modules/docs/PreprocessDataset/5 (Ανακτήθηκε 14/3/2021)

[46]Statquest with Josh Starmer ,"StatQuest:DESeq2, part1, Library Normalization", 27 Μαρτίου 2017 https://www.youtube.com/watch?v=UFB993xufUU&list=PLblh5JKOoLUJo2Q6xK4tZElbIvAACEykp&index=13 (Ανακτήθηκε 14/3/2021)

ώστε στατιστικά τεστ να μπορούν να δώσουν πιο ακριβείς αποτελέσματα.[47] Όλα τα αρχεία περιέχουν τους κωδικούς των συμμετεχόντων που έχουν παραχθεί από το TCGA στις κολώνες και είναι της μορφής TCGA-[TSS]-[PARTICIPANT]-[SAMPLE][VIAL]-[PORTION][ANALYTE]-[PLATE] [CENTER][48] ενώ οι γραμμές περιέχουν τις εκφράσεις από 20,532 γονίδια. Συγκεκριμένα, η κάθε κολώνα περιέχει την έκφραση όλων των γονιδίων ενός συμμετέχοντα.



Εικόνα 6.1: Εικόνα κωδικού δείγματος από το TCGA. Πηγή: https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/

ybridization RE	TCGA-CS-4938-01B-11R-1896-07	TCGA-CS-4941-01A-01R-1470-07	TCGA-CS-4942-01A-01R-1470-07	TCGA-CS-4943-01A-01R-1470-07	TCGA-CS-4944-01A-01R-1470-07	TCGA-CS-5390-01A-02R-1470-07	TCGA-CS-5393-01A-01R-1470-07
?|441362	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
?|442388	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
?|553137	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
?|57714	315.0829	541.4079	387.8893	404.9844	91.7917	285.1666	336.4938
?|645851	9.6497	22.0842	6.5744	13.3690	3.0261	10.2236	13.3145
?|652919	26.4317	4.8275	33.9066	40.3158	0.0000	0.0913	0.0000
?|653553	58.3176	120.7729	38.7543	24.4556	166.9399	18.6216	27.4359
?|728045	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
?|728603	0.0000	0.0000	0.0000	0.3261	0.0000	0.0000	0.0000
?|728788	2.0978	1.3803	0.3460	4.5650	0.0000	0.3651	4.8416
?|729884	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
?|8225	572.2677	734.2995	674.7405	777.0352	830.6645	769.6942	422.4329
?|90288	33.5641	108.3506	13.1488	4.2390	14.1218	2.1908	3.2278
A1BG|1	94.1095	72.2326	74.4533	29.9858	24.7132	85.4696	46.3789
A1CF|29974	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A2BP1|54715	22.6558	524.4997	368.5121	44.9983	105.4092	309.2652	323.5828
A2LD1|87769	13.4844	144.0856	51.4083	13.9821	18.0154	19.5308	29.0942
A2ML1|144568	146.0038	521.3941	174.3945	179.0148	159.3746	74.4865	164.6157
A2M|2	14783.7130	17944.7205	19269.8893	11719.7554	10894.9590	11046.5979	16480.1130
A4GALT|53947	73.4214	159.7654	44.2907	35.5421	114.9918	68.2793	76.2558
A4GNT|51146	0.0000	1.0352	0.3460	1.3043	0.5044	1.0954	0.0000
AAA1|404744	0.0000	0.0000	0.0000	0.0000	0.0000	0.3651	0.0000
AAAS|8086	584.8542	359.9034	560.8997	1241.6912	534.1067	840.5294	727.8596
AACSL|729522	0.0000	0.0000	0.0000	0.0000	0.0000	5.4770	0.8069

Εικόνα 6.2: Δεδομένα πριν την επεξεργασία.

[47] Jo Vandesompele,"Seven tips for bio-statistical analysis of gene expression data",11 December ,2013, https://blog.qbaseplus.com/seven-tips-for-bio-statistical-analysis-of-gene-expression-data (Ανακτήθηκε 14/3/2021)

[48] https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/ (Ανακτήθηκε 21/1/2021)

6.2 Επεξεργασία δεδομένων

Για την επεξεργασία των δεδομένων χρησιμοποιήσαμε την γλώσσα προγραμματισμού **python** και συγκεκριμένα τις παρακάτω βιβλιοθήκες:

- **Pandas:** Αποτελεί μια από τις πιο βασικές βιβλιοθήκες της Python για την ανάλυση δεδομένων, παρέχει πολύπλοκες δομές δεδομένων και συναρτήσεις που ταυτόχρονα είναι εύκολες στην χρήση και αποτελεσματικές. Επιτρέπει την φόρτωση και την επεξεργασία εγγράφων σε μορφή **CSV**,**XML**,**HTML**,**MICROSOFT EXCEL**, **JSON** και άλλες ακόμα καθώς και την εξαγωγή δεδομένων στις παραπάνω μορφές. Ένα τελευταίο πλεονέκτημα που παρέχεται εκτός από τις παραπάνω μορφές, είναι ότι μπορεί να μετατρέψει τα δεδομένα σε **SQL** και να χρησιμοποιηθεί για την ανταλλαγή πληροφοριών μεταξύ μιας βάσης δεδομένων και του συστήματος[49].
- **NumPy:** Ίσως η πιο βασική βιβλιοθήκη της **Python** στην οποία βασίζονται άλλες γνωστές βιβλιοθήκες. Το όνομα **NumPy** προέρχεται από το **Numerical Python** (**Αριθμητική python**) και παρέχει στον χρήστη, ένα πλήθος διάφορων δομών δεδομένων, (όπως έναν πολυδιάστατο πίνακα πολύ πιο γρήγορο στο να επεξεργαστεί δεδομένα από αυτούς που παρέχονται στην **Python**) καθώς και συναρτήσεις υψηλής απόδοσης που δεν παρέχει η **Python**. Εκτός από την **Python** είναι διαθέσιμη και σε άλλες προγραμματίστηκες γλώσσες όπως **C** , **C++, Fortran** και άλλες ακόμα[50].
- **Tensorflow:** Είναι βιβλιοθήκη που χρησιμοποιείται για αριθμητικούς υπολογισμούς. Αναπτύχθηκε από την **Google Brain team** και χρησιμοποιείται κατά κόρον σε διάφορες εφαρμογές της **Google,** όπως την μηχανή αναζήτηση της**. Το Tensorflow** εφαρμόζεται σε διάφορους τομείς στην μηχανική μάθηση όπως την **ταξινόμηση εικόνων** (**Image Classification**), **προτεινόμενα συστήματα** (**Recommender Systems**) και άλλους ακόμα. Ένα από τα πιο χρήσιμα πλεονεκτήματα είναι ότι υποστηρίζει αριθμητικούς υπολογισμούς με την χρήση **GPU.** ποστηρίζει ακόμα κατανεμημένους υπολογισμούς (**Distributed Computing**) μεταξύ πολλών συσκευών και Servers. Μπορεί να χρησιμοποιηθεί σε διάφορα λογισμικά συστήματα όπως **Windows**, **Linux**, **MacOS** αλλά ακόμα και σε κινητές συσκευές (**Android**, **IOS**) και υποστηρίζεται σε διάφορες γλώσσες προγραμματισμού όπως **Python**, **C++, Java**, **Go**, **SWIFT** και **Javascript**. Τέλος με την έκδοση του **Tensorflow 2.0** δίνεται η δυνατότητα να χρησιμοποιηθεί σαν διεπαφή του Tensorflow μια

[49] Fabio Nelli, ”Python Data Analytics- with Pandas, NumPy and Matplotlib”, Second Edition, Apress, 2018

[50] Fabio Nelli, ”Python Data Analytics- with Pandas, NumPy and Matplotlib”, Second Edition, Apress, 2018

άλλη γνωστή και πιο φιλική για τον χρήστη βιβλιοθήκη για την κατασκευή νευρωνικών δικτύων με όνομα **Keras**[51]**.**

- **Scikit-Learn (Sklearn):** Είναι η πιο γνωστή βιβλιοθήκη μηχανικής μάθησης. Αποτελείται από ένα μεγάλο πλήθος υλοποιημένων αλγορίθμων μηχανικής μάθησης για σκοπούς ταξινόμησης, παλινδρόμησης, ομαδοποίησης, μείωσης διαστάσεων, προ επεξεργασίας δεδομένων, εξαγωγής χαρακτηριστικών και άλλων ακόμα[52]. Είναι μέρος μιας άλλης βιβλιοθήκης με όνομα **SciPy** (**Scientific Python**), η οποία δημιουργήθηκε για επιστημονικό σκοπό και συγκεκριμένα για την ανάλυση δεδομένων[53].
- **Matplotlib:** Είναι μια δημοφιλής βιβλιοθήκη της Python, η οποία παρέχει δυνατότητες οπτικοποίησης των δεδομένων με την δημιουργία γραφημάτων. Μερικά από τα χαρακτηριστικά της είναι η ευκολία στην χρήση της, ο έλεγχος που δίνεται στον χρήση στις γραφικές παραστάσεις και η δυνατότητα να εξάγονται τα γραφήματα σε μορφή όπως **PNG**,**PDF**,**SVG** και **EPS**[54].
- **Seaborn:** Αποτελεί άλλη μια σημαντική βιβλιοθήκη για την οπτικοποίηση δεδομένων που είναι βασισμένη στις διαμορφώσεις της βιβλιοθήκης **Matplotlib**. Παρέχει στους χρήστες διάφορες διεργασίες οπτικοποίησης για την απεικόνιση δεδομένων όπως χαρτογράφηση. Συνήθως, η βιβλιοθήκη **Seaborn** έχει μια πιο ολοκληρωμένη συνεργασία με την βιβλιοθήκη **Pandas** για την οπτικοποίηση πλαισίων δεδομένων (**DataFrames**)[55].
- **Keras Tuner:** Είναι βιβλιοθήκη η οποία κατασκευάστηκε από την ίδια ομάδα που κατασκεύασε και την βιβλιοθήκη **Keras** δίνει την δυνατότητα να επιλεχτούν οι ιδανικοί υπερπαραμέτροι για τους αλγόριθμους που θα υλοποιηθούν. Παρέχει διάφορους **δέκτες** (**Tuners**) για την επιλογή υπερπαραμέτρων όπως **BayesianOptimization** tuner, **HyperBand** tuner, **RandomSearch** tuner, **Sklearn** tuner.[56] Η διαδικασία λειτουργίας του είναι απλή, πρώτα επιλέγεται ο **δέκτης (Tuner)** που θα χρησιμοποιηθεί, δεύτερον

[51] Aurélien Géron, “Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow”, O’Reilly Media, June 2019

[52] Gavin Hackeling, “Mastering Machine Learning with Scikit Learn”, Second Edition, Packt, July 2017

[53] Fabio Nelli, ”Python Data Analytics- with Pandas, NumPy and Matplotlib”, Second Edition, Apress, 2018

[54] Fabio Nelli, ”Python Data Analytics- with Pandas, NumPy and Matplotlib”, Second Edition, Apress, 2018

[55] Ali Hassan Sial, Syed Yahya, Shah Rashdi, Dr.Hafeez Khan, ”Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python”, February 2021 https://bit.ly/3vucEv6 (Ανακτήθηκε 16/3/2021)

[56] https://keras-team.github.io/keras-tuner/ (Ανακτήθηκε 14/3/2021)

ορίζονται οι παράμετροι που θα εξετάσει ο **δέκτης (Tuner)** και τέλος, η αρμοδιότητα του είναι να ανακαλύψει τον καλύτερο συνδυασμό των παραμέτρων που έχουν οριστεί για το καλύτερο αποτέλεσμα[57].

Σε πρώτη φάση, εφόσον φορτώσαμε τα αρχεία, μετατοπίσαμε τις γραμμές και τις στήλες σε κάθε αρχείο (δηλαδή οι γραμμές έγιναν στήλες και οι στήλες γραμμές), ώστε να υπάρχουν στις γραμμές οι κωδικοί των συμμετεχόντων και στις στήλες τα γονίδια. Έπειτα, παραλείψαμε τα 30 πρώτα γονίδια τα οποία δεν έχουν όνομα και αποτελούν δείγματα ελέγχου (**control samples**) που προσθέτει το σύστημα για την καλύτερη εξαγωγή των γονιδιακών εκφράσεων. Στη συνέχεια αναλύσαμε ποια από τα δεδομένα από κάθε αρχείο ανήκουν σε υγιείς ανθρώπους και ποια σε ανθρώπους που έχουν την ασθένεια και έπειτα προσθέσαμε σε κάθε δείγμα την αντίστοιχη ετικέτα.

Index	ZW10|9183	ZWILCH|55055	ZWINT|11130	ZXDA|7789	ZXDB|158586	ZXDC|79364	ZYG11A|440590	ZYG11B|79699	ZYX|7791	ZZEF1|23140	ZZZ3|26009	psiTPTE22|38759	tAKR|389932	label
TCGA-CS-4938-01B-1…	76.06460000	80.16360000	50.34610000	64.19130000	523.18020000	1311.51670000	0.00000000	1495.69960000	1598.90920000	1171.80620000	836.16530000	777.84770000	0.00000000	LGG
TCGA-CS-4941-01A-0…	01.24220000	187.29810000	110.42100000	105.93510000	467.56380000	1226.70810000	0.34510000	1751.20770000	3889.92410000	1707.72950000	1015.52800000	18.97860000	0.00000000	LGG
TCGA-CS-4942-01A-0…	20.41520000	120.03110000	186.50520000	115.22490000	454.32530000	1102.42210000	0.34600000	1586.15920000	2214.53290000	1423.87540000	706.92040000	1706.22840000	0.00000000	LGG
TCGA-CS-4943-01A-0…	36.93970000	667.44820000	1183.65000000	89.01830000	491.06800000	1288.31990000	3.26070000	1760.14950000	2131.54820000	1916.99120000	689.32120000	268.03310000	0.00000000	LGG
TCGA-CS-4944-01A-0…	72.85340000	57.32440000	56.48720000	42.36540000	323.79270000	1012.73480000	0.50440000	1225.06620000	1692.59870000	729.79450000	456.94110000	19.66970000	0.00000000	LGG
TCGA-CS-5390-01A-0…	37.38020000	191.28430000	422.82060000	78.13780000	887.63120000	1622.63810000	0.00000000	1065.81470000	1954.90640000	2093.29070000	457.87310000	42.72020000	0.00000000	LGG
TCGA-CS-5393-01A-0…	40.87150000	77.36940000	70.20380000	52.04760000	392.57620000	1318.94290000	0.00000000	1648.98120000	1749.04180000	1979.42300000	845.26930000	850.91790000	0.00000000	LGG
TCGA-CS-5394-01A-0…	63.63330000	228.80900000	218.13320000	128.80560000	609.23390000	1498.82900000	0.33460000	1402.47570000	1807.62800000	2348.27700000	423.21850000	276.01200000	0.00000000	LGG
TCGA-CS-5395-01A-0…	69.91700000	244.34340000	169.20720000	89.60210000	425.42480000	1070.41170000	0.00000000	1749.83250000	1755.75660000	1288.12250000	851.59010000	14.81030000	0.00000000	LGG
TCGA-CS-5396-01A-0…	26.71980000	105.02340000	193.87990000	152.15890000	662.62750000	1255.00420000	0.00000000	900.06900000	2137.58720000	2317.04890000	397.57650000	7.36250000	0.00000000	LGG
TCGA-CS-5397-01A-0…	46.43820000	213.41400000	220.09410000	73.58870000	404.56990000	1109.20700000	0.00000000	1864.24730000	2406.92200000	1413.30650000	776.88170000	36.96240000	0.00000000	LGG
TCGA-CS-6186-01A-1…	23.46750000	121.11950000	89.65940000	50.52630000	333.87000000	1288.42110000	0.49540000	1338.45200000	1402.35290000	1685.20120000	630.58820000	15.35600000	0.00000000	LGG
TCGA-CS-6188-01A-1…	50.09040000	219.03800000	170.34360000	42.67630000	364.55700000	1103.43580000	0.00000000	1633.27310000	9556.23870000	1482.82100000	861.12120000	25.67810000	0.00000000	LGG
TCGA-CS-6290-01A-1…	74.59230000	85.23570000	61.64320000	64.66070000	373.73870000	1097.07620000	0.43110000	1124.66470000	2311.83470000	1366.06460000	660.40100000	1632.46660000	0.00000000	LGG
TCGA-CS-6665-01A-1…	57.56650000	189.78140000	351.80480000	50.16100000	436.87510000	965.59910000	0.33890000	1467.54790000	2663.95530000	1649.55090000	975.76680000	137.60380000	0.00000000	LGG
TCGA-CS-6666-01A-1…	70.27190000	197.25780000	215.90320000	107.53800000	498.39730000	982.31830000	0.00000000	1301.62340000	2898.56270000	1748.73330000	983.14550000	923.99960000	0.00000000	LGG
TCGA-CS-6667-01A-1…	10.80000000	93.82480000	73.71310000	68.39730000	358.99710000	1247.45280000	0.35440000	1421.81270000	1850.62460000	1775.13950000	756.97710000	532.64820000	0.00000000	LGG
TCGA-CS-6668-01A-1…	07.45630000	106.46440000	138.38940000	273.71110000	933.61740000	2351.59780000	0.00000000	1571.36770000	1194.37580000	3224.88280000	578.44060000	297.91220000	0.00000000	LGG
TCGA-CS-6669-01A-1…	17.11420000	244.96130000	83.75060000	48.70280000	488.39330000	849.79520000	0.91030000	3256.71370000	870.27770000	1106.50890000	534.82020000	39.14430000	0.00000000	LGG
TCGA-CS-6670-01A-1…	49.15450000	92.42770000	160.84180000	99.21080000	505.07330000	1535.13720000	0.75160000	1164.22400000	2186.02030000	1653.88950000	374.67120000	246.52390000	0.00000000	LGG
TCGA-DB-5270-01A-0…	67.98500000	154.50390000	122.40030000	49.43740000	395.15850000	864.30280000	1.02280000	2210.36480000	2230.13980000	1170.13300000	553.69930000	1086.60080000	0.00000000	LGG
TCGA-DB-5273-01A-0…	95.48280000	132.54870000	193.94940000	57.60460000	295.89720000	922.50310000	0.41440000	1392.04310000	2327.80770000	1171.98510000	604.22710000	2341.06920000	0.00000000	LGG
TCGA-DB-5274-01A-0…	86.20690000	180.92530000	400.57470000	33.33330000	624.13790000	1045.40230000	0.00000000	689.08050000	2964.94250000	705.74710000	232.18390000	17.24140000	0.00000000	LGG
TCGA-DB-5275-01A-0…	95.59280000	168.89110000	230.27580000	62.98430000	534.86630000	1020.41160000	0.66650000	1519.95330000	2330.75060000	2060.81810000	589.85250000	148.62950000	0.00000000	LGG

Εικόνα 6.3: Δεδομένα πρώτης φάσης επεξεργασίας.

Τέλος, συγκεντρώθηκαν όλα τα αρχεία σε ένα και δημιουργήθηκε ένα αρχείο το οποίο περιείχε 10,362 γραμμές (κάθε γραμμή είναι ο κάθε συμμετέχων στην ανάλυση) και 20,503 κολώνες. Οι 20,502 κολώνες αποτελούσαν τα γονίδια που θα αναλύονταν, ενώ η τελευταία κολώνα περιείχε τις ετικέτες του κάθε συμμετέχοντα (αν περιέχει κάποιο είδος καρκίνου ή αν είναι υγιείς). Στην παρακάτω φωτογραφία, εμφανίζονται μόνο οι πρώτες 600 γραμμές δεδομένων. Για προφανείς λόγους έχουν συμπληρωθεί με τελίτσες οι ενδιάμεσες γραμμές, καθώς θα χρειαζόντουσαν αρκετές σελίδες για να τυπωθούν και οι 600 γραμμές.

[57] Julie Prost, “Hands on hyperparameters tuning with Keras Tuner”, https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner (Ανακτήθηκε 14/3/2021)

```
In [23]: data.head(600)
Out[23]:
                               A1BG|1  A1CF|29974  ...  tAKR|389932  label
TCGA-CS-4938-01B-11R-1896-07   94.1095         0.0  ...       0.0000    LGG
TCGA-CS-4941-01A-01R-1470-07   72.2326         0.0  ...       0.0000    LGG
TCGA-CS-4942-01A-01R-1470-07   74.4533         0.0  ...       0.0000    LGG
TCGA-CS-4943-01A-01R-1470-07   29.9858         0.0  ...       0.0000    LGG
TCGA-CS-4944-01A-01R-1470-07   24.7132         0.0  ...       0.0000    LGG
...                                ...         ...  ...          ...    ...
TCGA-23-1111-01A-01R-1567-13  128.9958         0.0  ...       0.0000     OV
TCGA-23-1114-01B-01R-1566-13   89.2616         0.0  ...       0.0000     OV
TCGA-23-1120-01A-02R-1565-13  337.9484         0.0  ...       0.0000     OV
TCGA-23-1122-01A-01R-1565-13  129.8043         0.0  ...       0.0000     OV
TCGA-23-1123-01A-01R-1565-13   58.0075         0.0  ...       0.5335     OV

[600 rows x 20503 columns]
```

Εικόνα 6.4: Εικόνα δεδομένων.

Συνολικά υπάρχουν 29 κατηγορίες και παρακάτω θα αναφερθούν πόσα δείγματα στα δεδομένα υπάρχουν σε κάθε κατηγορία:

ACC:79	**ESCA:185**	**LIHC:373**	**SARC:263**	**UCEC:177**
BLCA: 408	**GBM:166**	**LUAD:517**	**SKCM:472**	
BRCA: 1100	**HNSC:522**	**LUSC:501**	**STAD:415**	
CESC: 306	**KICH:66**	**OV:307**	**NORMAL:735**	
CHOL: 36	**KIRC:534**	**PCPG:184**	**TGCT:156**	
COAD:287	**LAML:173**	**PRAD:498**	**THCA:509**	
DLBC:48	**LGG:530**	**READ:95**	**THYM:120**	

Όπως παρατηρήθηκε σε ορισμένες κατηγορίες υπάρχουν περισσότερα δείγματα από άλλες κατηγορίες (π.χ CHOL αποτελείται από 36 δείγματα), αυτό προκαλεί ένα ανισόρροπο σύνολο δεδομένων (Imbalanced dataset). Αν και υπάρχουν αλγόριθμοι που δεν επηρεάζονται από ανισόρροπα σύνολα (π.χ. Naïve Bayes), θα εκτελεστούν ορισμένες διαδικασίες παρακάτω που θα βοηθήσουν να διορθωθεί αυτό το πρόβλημα.

Για να αντιμετωπιστεί το πρόβλημα με το ανισόρροπο σύνολο δεδομένων χρησιμοποιήθηκε ο αλγόριθμος **Συνθετικής Τεχνικής Δειγματοληψίας Μειονοτήτων** (**Synthetic Minority Oversampling Technique – SMOTE**), ο οποίος αυξάνει το σύνολο δεδομένων με το να αντιγράφει και να επαναχρησιμοποιεί δεδομένα από τις κλάσεις με τον μικρότερο αριθμό δειγμάτων. Όμως, επειδή αυτή η αντιγραφή δεν προσφέρει περισσότερες πληροφορίες από τις

ήδη υπάρχουσες για τις μειονότητες, ο αλγόριθμος προσπαθεί να κατασκευάσει καινούργιες πληροφορίες με την σύνθεση των ήδη υπάρχοντών δειγμάτων. Ο τρόπος που λειτουργεί ο αλγόριθμος αυτός σχετίζεται με την επιλογή των δειγμάτων που είναι πολύ κοντά στον χώρο των χαρακτηριστικών (feature space), διαχωρίζοντας τα δείγματα με μια γραμμή στον χώρο των χαρακτηριστικών και στην συνέχεια δημιουργώντας καινούργια δείγματα κατά μήκος της γραμμής αυτής. Αρχικά, επιλέγεται ένα τυχαίο δείγμα **a** και στην συνέχεια ο αλγόριθμος βρίσκει τους **n** πιο κοντινούς γείτονες τους τυχαίου δείγματος που επιλέχτηκε, επιλέγεται ένας γείτονας **b** και τα συνθετικά δείγματα δημιουργούνται ως ένα συνδυασμός των **a** και **b**[58]. Παρακάτω, θα δείξουμε μετά την χρήση το αλγορίθμου πόσα δείγματα έχουμε συνολικά.

	Sampled_data	Original Data
LGG	530	530
OV	400	307
THCA	509	509
NORMAL	735	735
BRCA	1100	1100
COAD	400	287
UCEC	400	177
ESCA	400	185
SKCM	472	472
ACC	400	79
BLCA	408	408
CESC	306	306
CHOL	400	36
LAML	400	173
DLBC	400	48
GBM	400	166
THYM	400	120
HNSC	522	522
KICH	400	66
KIRC	534	534
PRAD	498	498
PCPG	400	184
STAD	415	415
SARC	400	263
TGCT	400	156
READ	400	95
LUSC	501	501
LIHC	373	373
LUAD	517	517

Εικονα 6.5: Δεδομένα μετά την εφαρμογή της τεχνικής SMOTE.

Ο συνολικός αριθμός των δεδομένων ανήλθε από 10.362 δείγματα σε 13.420 δείγματα.

Εφόσον αυξήθηκαν τα δεδομένα, έπειτα κωδικοποιήθηκε ο πίνακας με τις ετικέτες του κάθε δείγματος, 29 ετικέτες κωδικοποιήθηκαν σε αριθμούς και έτσι κάθε ετικέτα αντιστοιχεί σε έναν αριθμό.

```
['0-ACC', '1-BLCA', '2-BRCA', '3-CESC', '4-CHOL', '5-COAD', '6-DLBC', '7-ESCA', '8-GBM', '9-HNSC', '10-KICH', '11-KIRC', '12-LAML', '13-LGG', '14-LIHC', '15-LUAD', '16-LUSC', '17-NORMAL', '18-OV', '19-PCPG', '20-PRAD', '21-READ', '22-SARC', '23-SKCM', '24-STAD', '25-TGCT', '26-THCA', '27-THYM', '28-UCEC']
```

[58] Jason Brownlee, «SMOTE for imbalanced classification with python», January 17,2020 https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ (Ανακτήθηκε 2/3/2021)

Εικόνα 6.6: Αντιστοιχία αριθμών και κλάσεων μετά την χρήση του Label Encoder.

Στην συνέχεια, για όλες τις αναλύσεις που πραγματοποιήθηκαν χωρίστηκε το σύνολο δεδομένων σε **δεδομένα εκπαίδευσης** (**Training Set**) και **δεδομένα δοκιμής** (**Test Set**) με την συνάρτηση που παρέχεται από την βιβλιοθήκη **Scikit-Learn** . Ο λόγος που πρέπει να γίνεται ο διαχωρισμός αυτός είναι διότι οι αλγόριθμοι πρέπει να «εκπαιδευτούν» με δεδομένα και μετά να δοκιμαστεί η αποτελεσματικότητα τους σε δεδομένα που δεν έχουν ξανά συναντήσει. Κάποιες προϋποθέσεις που πρέπει να συμπληρώνει το σύνολο με τα δεδομένα δοκιμής είναι: 1) να είναι αρκετά μεγάλο ώστε να μπορεί να αποφέρει σημαντικά στατιστικά αποτελέσματα και 2) να μοιράζεται τα ίδια χαρακτηριστικά με τα δεδομένα εκπαίδευσης. Στόχος είναι, να κατασκευαστεί ένα σύστημα το οποίο θα δίνει ικανοποιητικά αποτελέσματα σε δεδομένα που δεν έχει ξανά συναντήσει και η διαδικασία αυτή συμβάλει σε αυτό[59]. Επίσης, για να γίνει πιο δίκαια η διαχώριση των δεδομένων, χρησιμοποιήθηκε η διαδικασία της **στρωμάτωσης** (**Stratification**), η οποία χωρίζει τις κατηγορίες των δεδομένων με το ίδιο ποσοστό. Για παράδειγμα, το σύνολο δεδομένων περιέχεται από 13,420 δείγματα και 29 κλάσεις, τα δεδομένα χωρίζονται σε ποσοστό ώστε το 80% να πηγαίνουν στα δεδομένα εκπαίδευσης και το 20% στα δεδομένα δοκιμής. Αυτό που κάνει η **στρωμάτωση** είναι ότι το 20% που αποτελεί τα δεδομένα δοκιμής θα αποτελούνται από το 20% των δεδομένων κάθε κλάσης.

```
X_train:  (10736, 20502)
X_test:  (2684, 20502)
y_train:  (10736,)
y_test:  (2684,)
```

Εικόνα 6.7: Αριθμός δεδομένων στο σύνολο εκπαίδευσης και δοκιμής.

Οι αριθμοί που αναφέρονται παραπάνω δείχνουν ότι τα δεδομένα εκπαίδευσης αποτελούνται από 10,736 δείγματα τα οποία είναι το 80% του συνολικού αριθμού δειγμάτων και τα δεδομένα δοκιμής από 2,684 δείγματα τα οποία είναι το υπόλοιπο 20%.

Από την παρατήρηση των στατιστικών των δεδομένων φαίνεται ότι, πολλά γονίδια έχουν μεγαλύτερο εύρος τιμών. Για παράδειγμα, το γονίδιο **A1BG** αποτελείται από τιμές που κυμαίνονται από 0.94 – 542,015 σε σχέση με το γονίδιο **AAA1** που έχει εύρος τιμών 0-244.

[59] Google Developers," Training and Test Sets: Splitting Data", https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data (Ανακτήθηκε 7/3/2021)

	A1BG|1	A1CF|29974	A2BP1|54715	A2LD1|87769	A2ML1|144568	A2M|2	A4GALT|53947	A4GNT|51146	AAA1|404744	AAAS|8086	...	ZXDA|7789
count	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	...	11216.000000
mean	1712.934363	89.734769	46.549926	122.070542	945.258547	17705.135187	628.768559	14.428525	0.905598	903.752554	...	56.885784
std	12215.414292	339.193486	256.156364	155.089116	5776.323393	31014.301322	838.770597	103.182170	4.478547	378.474701	...	31.697995
min	0.947900	0.000000	0.000000	2.309700	0.000000	0.000000	0.000000	0.000000	0.000000	79.928600	...	0.000000
25%	34.923981	0.000000	0.000000	62.051225	0.337900	4984.031600	142.407550	0.000000	0.000000	654.748575	...	33.704725
50%	83.865100	0.000000	0.738450	95.636450	3.087891	9122.958161	339.913750	0.591400	0.000000	815.072850	...	51.822468
75%	189.892588	2.721298	8.179413	144.861194	75.846326	18082.861575	793.100000	1.929300	0.348085	1048.391482	...	74.244196
max	542015.610700	4611.237300	6153.670800	7435.957500	146583.234500	716942.016800	9830.241500	3136.419800	244.699600	4477.112700	...	343.777200

8 rows × 20502 columns

Εικόνα 6.8: Στατιστική περιγραφή των δεδομένων πριν την χρήση ομαλοποίησης.

Αυτή η διαφορά τιμών των γονιδίων αποτελεί πρόβλημα για έναν αριθμό αλγορίθμων αλλά υπάρχουν και αλγόριθμοι που δεν τους αφορά η διακύμανση των τιμών. Επειδή η ανάλυση που θα διεξαχθεί βασίζεται στην χρησιμοποίηση νευρωνικών δικτύων τα οποία βασίζονται στην βελτιστοποίηση με τον **αλγόριθμο διαβάθμισης κλίσης** (**Gradient Descent Algorithm**)

$$\boldsymbol{\theta_j} := \boldsymbol{\theta_j} - \boldsymbol{a}\left(\frac{\boldsymbol{1}}{\boldsymbol{m}}\right)\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\left(\boldsymbol{h_\theta}\left(\boldsymbol{x^{(i)}}\right) - \boldsymbol{y^{(i)}}\right)\boldsymbol{x_j}^{(i)}$$

οι αλγόριθμοι αυτού του είδους είναι ευάλωτοι σε χαρακτηριστικά με διαφορετικό εύρος τιμών. Αυτό το πρόβλημα διορθώνεται με την διαδικασία **κλιμάκωσης των χαρακτηριστικών** (**Feature Scaling**). Δύο πιο χαρακτηριστικοί τρόποι κλιμάκωσης είναι η **ομαλοποίηση** (**Normalization**) και η **τυποποίηση** (**Standardization**).

Ομαλοποίηση, είναι η διαδικασία κατά την οποία οι τιμές θα ανακατασκευαστούν και θα καταλήξουν να έχουν εύρος συνήθως από 0 – 1 για όλα τα χαρακτηριστικά. Η διαδικασία που ακολουθείται για να πραγματοποιηθεί αυτή ανακατασκευή είναι η εξής: αφαιρείται η ελάχιστη τιμή ενός χαρακτηριστικού (**γονιδίου** στην συγκεκριμένη περίπτωση) για όλα τα δείγματα από την τιμή του εκάστοτε δείγματος και στην συνέχεια διαιρείται με την διαφορά της μεγαλύτερης τιμή με της μικρότερης του χαρακτηριστικού[60].

$$\boldsymbol{X'} = \frac{\boldsymbol{X - X_{min}}}{\boldsymbol{X_{max} - X_{min}}}$$

Τυποποίηση είναι άλλος ένας τρόπος να κλιμακώσουμε τα δεδομένα σύμφωνα με τον οποίο από μια τιμή ενός χαρακτηριστικού αφαιρούμε την μέση τιμή και διαιρούμε με την

[60] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019

τυπική απόκλιση. Έτσι τα χαρακτηριστικά επικεντρώνονται γύρω από την μέση τιμή με τυπική απόκλιση μονάδας[61].

$$X' = \frac{X - \mu}{\sigma}$$

Συνήθως, τα νευρωνικά δίκτυα έχουν καλύτερη απόδοση με την διαδικασία της ομαλοποίησης. Για το λόγο αυτό, στις αναλύσεις που θα γίνουν, τα δεδομένα θα είναι ομαλοποιημένα. Θα χρησιμοποιηθεί ένας **μετασχηματιστής** (**Transformer**) που παρέχεται στην βιβλιοθήκη **Scikit-Learn** με όνομα **MinMaxScaler.**

Ο μετασχηματιστής (MinMaxScaler) θα εφαρμοστεί στα δεδομένα εκπαίδευσης μόνο για να μη υπάρχει περίπτωση διαρροής πληροφορίας από «μελλοντικά» δεδομένα και στην συνέχεια θα χρησιμοποιηθεί για να μετασχηματίσει τα δεδομένα δοκιμής[62]. Παρακάτω φαίνεται πως μετατράπηκαν τα δεδομένα μετά την εφαρμογή του μετασχηματιστή. Όπως βλέπουμε οι τιμές των χαρακτηριστικών κυμαίνονται από **0** μέχρι **1.**

count	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	11216.000000	...	11216.000000	1121
mean	0.003159	0.019460	0.007565	0.016111	0.006449	0.024695	0.063963	0.004600	0.003701	0.187353	...	0.165473	
std	0.022537	0.073558	0.041627	0.020863	0.039406	0.043259	0.085326	0.032898	0.018302	0.086072	...	0.092205	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	
25%	0.000063	0.000000	0.000000	0.008037	0.000002	0.006952	0.014487	0.000000	0.000000	0.130725	...	0.098042	
50%	0.000153	0.000000	0.000120	0.012555	0.000021	0.012725	0.034578	0.000189	0.000000	0.167185	...	0.150744	
75%	0.000349	0.000590	0.001329	0.019177	0.000517	0.025222	0.080680	0.000615	0.001423	0.220246	...	0.215966	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	

8 rows × 20502 columns

Εικόνα 6.9: Στατιστική περιγραφή των δεδομένων μετά την χρήση ομαλοποίησης.

6.3 Κατασκευή των Μοντέλων

Η επιλογή του μοντέλου πραγματοποιήθηκε με την χρήση τριών δικτύων. Ο σκοπός του **πρώτου** νευρωνικού δικτύου ήταν να **μειωθεί** ο αριθμός των χαρακτηριστικών (**dimensionality reduction**) για να αντιμετωπιστεί το πρόβλημα της «**κατάρας**» **των διαστάσεων** (**Curse of dimensionality**). Το πρόβλημα δημιουργείται όταν τα δεδομένα έχουν πολλά χαρακτηριστικά συνήθως χιλιάδες ή εκατομμύρια σε κάθε δείγμα. Θεωρείται πρόβλημα διότι με το να υπάρχουν τόσα πολλά χαρακτηριστικά το σύστημα θα είναι είτε

[61] ANIRUDDHA BHANDARI," Feature Scaling for Machine Learning: Understarding the Difference Between the Normalization vs. Standardization", April 3,2020, https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/ (Ανακτήθηκε 3/7/2021)

[62] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019

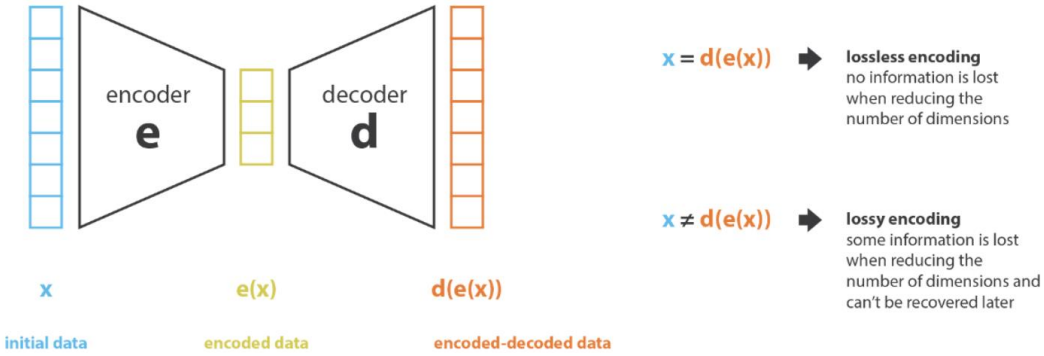
αρκετά αργό, με τα τωρινά υπολογιστικά δεδομένα, είτε πιο δύσκολο το να βρει μια καλή λύση[63]. Άλλο ένα πρόβλημα είναι η οπτικοποίηση των δεδομένων, πλέον είναι δύσκολο να απεικονίσουμε δεδομένα περισσοτέρων από τριών χαρακτηριστικών, οπότε αδύνατον να οπτικοποιηθούν δεδομένα με χιλιάδες χαρακτηριστικά. Στην συνέχεια, το **δεύτερο** νευρωνικό δίκτυο χρησιμοποιείται για την ταξινόμηση των χαρακτηριστικών που εξάγονται από το πρώτο δίκτυο. Τέλος, η ανάλυση ολοκληρώνεται από μια **τρίτη** αρχιτεκτονική νευρωνικών δικτύων[64], η οποία επαναλαμβάνει την διαδικασία της δεύτερης αρχιτεκτονικής, συγκρίνοντας τα αποτελέσματα τους.

Το πρώτο νευρωνικό δίκτυο υλοποιείται με τρεις διαφορετικές αρχιτεκτονικές, τους **Autoencoders**, τους **Denoising Autoencoders** και τέλος τους **Variational Autoencoders.** Επειδή η κάθε αρχιτεκτονική ανακατασκευάζει χαρακτηριστικά με τον δικό της τρόπο, το δεύτερο και το τρίτο δίκτυο που χρησιμοποιείται για την ταξινόμηση δεν είναι το ίδιο και στις τρεις περιπτώσεις αλλά διαφέρει στις **υπερπαραμέτρους** (**Hyperparameters**). Για την εύρεση των κατάλληλων υπερπαραμέτρων και για τα τρία δίκτυα, χρησιμοποιήθηκε η μέθοδος **Μπεϋζιανής βελτιστοποίησης,** (**Bayesian Optimization**) την οποία παρείχε υλοποιημένη σαν **δέκτη** (**Tuner**) η βιβλιοθήκη **Keras Tuner.** Επομένως, υπήρξαν τρείς διαφορετικοί τρόποι ανάλυσης με την χρήση νευρωνικών δικτύων. Παρακάτω, αναλύονται λεπτομέρειες σχετικά με σημεία που θα αναφερθούν στην ανάλυση.

Οι **Autoencoders** αποτελούν μια μορφή ενός νευρωνικού δικτύου που συμπεριλαμβάνει δύο κύρια μέρη τον κωδικοποιητή (Encoder) $\boldsymbol{h} = \boldsymbol{f}(\boldsymbol{x})$ και τον αποκωδικοποιητή (Decoder) $\boldsymbol{r} = \boldsymbol{g}(\boldsymbol{h})$ [65]. Ο κύριος σκοπός τους είναι να αντιγράψουν την είσοδο στην έξοδο τους. Η αρμοδιότητα του **κωδικοποιητή** είναι να **κωδικοποιήσει** τα δεδομένα εισόδου σε έναν **τανυστή** (**Tensor**) μικρότερων διαστάσεων[66]. Η αρμοδιότητα του **αποκωδικοποιητή** είναι να δεχτεί σαν είσοδο τον τανυστή που προήλθε από τον κωδικοποιητή και να κατασκευάσει την αρχική είσοδο[67]. Έχουν σχεδιαστεί για να μην μπορούν να αντιγράψουν ακριβώς την είσοδο στην έξοδο. Τις περισσότερες φορές ορίζονται περιορισμοί ώστε να μπορούν να

[63] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019
[64] Khalid Salama,"Structured data learning with Wide,Deep and Cross Networks", 31/12/2020, https://keras.io/examples/structured_data/wide_deep_cross_networks/ (Ανακτήθηκε 19/3/2021)
[65] Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015
[66] Rowel Atienza, "Advanced Deep Learning with Tensorflow 2 and Keras", Second Edition, Packt Publishing, February 2020
[67] Rowel Atienza, "Advanced Deep Learning with Tensorflow 2 and Keras", Second Edition, Packt Publishing, February 2020

αντιγράψουν ένα ποσοστό της αρχικής εισόδου κατά προσέγγιση και να αντιγραφούν δεδομένα από την είσοδο που να μοιάζουν στα δεδομένα εκπαίδευσης. Οπότε το μοντέλο υποχρεώνεται να αντιγράψει χρήσιμες πληροφορίες για τα δεδομένα[68]. Τέλος, η διαφορά μεταξύ των δεδομένων εισόδου με αυτά της εξόδου μετριόνται με μια **συνάρτηση απώλειας** (**Loss Function**)[69] την οποία προσπαθεί να ελαχιστοποιήσει.



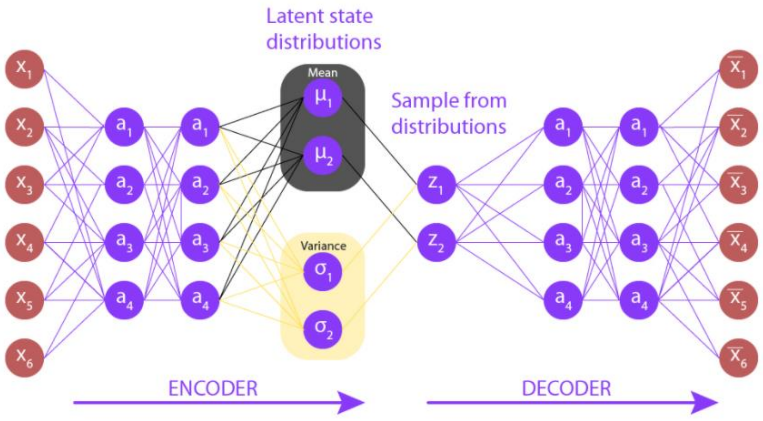
Εικόνα 6.10: Εικόνα Autoencoder. Πηγή: Joseph Rocca « Understarding Variational Autoencoders (VAEs)», towards data science, 24 September 2019, https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 (Ανακτήθηκε 23/2/2019)

Οι **Denoising Autoencoders** είναι μια επέκταση των **Autoencoders**. Ο σκοπός τους είναι να μειώσουν την **υπερβολική προσαρμογή** (**Overfitting**) και να αποτρέψουν τον Autoencoder να μάθει μια απλή **ταυτοτική συνάρτηση** (**Identity Function**)[70]. Οι denoising autoencoders, προσθέτουν ένα ποσοστό θορύβου $C(\tilde{x} \mid x)$ στα δεδομένα εισόδου και εκπαιδεύονται στο να ανακατασκευάσουν τα δεδομένα χωρίς τον θόρυβο υπολογίζοντας μια συνάρτηση απώλειας. Γενικά, όσο ο κωδικοποιητής είναι ντετερμινιστικός, ο denoising autoencoder μπορεί να εκπαιδευτεί με τις ίδιες τεχνικές όπως κάθε προς τα **εμπρός τροφοδοτούμενο νευρωνικό δίκτυο** (**Feed Forward Neural Network**) αφού αποτελεί ένα τέτοιο δίκτυο[71].

Οι **Variational Autoencoders** είναι μια ειδική υπό κατηγορία στην γενική έννοια των **Autoencoders.** Έχουν προσελκύσει το ενδιαφέρον διότι ανήκουν στα **παραγωγικά μοντέλα** (**Generative Models**). Η διαδικασία με την οποία λειτουργούν σχετίζεται με τον

[68] Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015
[69] Rowel Atienza, "Advanced Deep Learning with Tensorflow 2 and Keras", Second Edition, Packt Publishing, February 2020
[70] Adrian Rosebrock, "Denoising autoencoders with Keras,Tensorflow, and Deep Learning",14 February 2020, https://www.pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/ (Ανακτήθηκε 16/3/2021)
[71] Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015

κωδικοποιητή (**Encoder**). Συγκεκριμένα ο κωδικοποιητής δεν αποτελεί έναν διανυσματικό πίνακα μεγέθους $\boldsymbol{n}$ όπως τα υπόλοιπα είδη **Autoencoders,** αλλά υπολογίζονται δύο διαφορετικοί διανυσματικοί πίνακες. Ο **πρώτος** αποτελεί έναν **διανυσματικό πίνακα των μέσων** (**Vector of Means**) μ και ο **δεύτερος** αποτελεί έναν **διανυσματικό πίνακα των τυπικών αποκλίσεων** (**Vector of Standard Deviations**) σ**.** Στην συνέχεια, σχηματίζεται ένας διανυσματικός πίνακας με τυχαίες τιμές και μεγέθους $\boldsymbol{n}$**.** Για την παραγωγή της κάθε τυχαίας τιμής στην θέση $\boldsymbol{i}$ του πίνακα λαμβάνεται υπόψη η τιμή στην θέση $\boldsymbol{i}$ των πινάκων μ και σ και η τυχαία τιμή αποτελεί δείγμα της φυσικής κατανομής μεταξύ αυτών. Για την εκπαίδευση των δικτύων αυτών προστίθεται στην συνάρτηση απώλειας ένας υπολογισμός της απόκλισης Kullback Leibler, μεταξύ της κανονικής κατανομής και της υποτιθέμενης κατανομής**.** Η συνάρτηση Kullback Leibler μεταξύ δύο κατανομών πιθανότητας όπως των παραπάνω μετρά το πόσο αυτές διαφέρουν μεταξύ τους. Αυτή η συνάρτηση, βοηθάει στην κατασκευή καινούργιων δειγμάτων που είναι και η βασική λειτουργία των Variational Autoencoder[72].



Εικόνα 6.11: Εικόνα Variational Autoencoder. Πηγή: GeekforGeeks, ”Variational Autoencoder”, 17 July 2021, https://www.geeksforgeeks.org/variational-autoencoders/ (Ανακτήθηκε 20/3/2021)

Η **Μπεϋζιανή Βελτιστοποίηση (Bayesian Optimization)** είναι διαδικασία για την βελτιστοποίηση λειτουργιών που απαιτούν πολύ χρόνο ή είναι περίπλοκες. Η διαδικασία

[72] Irhum Shafkat, “Intuively Understanding Variational Autoencoder”, Towards Data Science, 4 February 2018, https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf (Ανακτήθηκε 20/3/2021)

ξεκινάει με την δημιουργία ενός μοντέλου πιθανοτήτων της **αντικειμενικής συνάρτησης** (**Objective function**) που ονομάζεται **συνάρτηση αντικατάστασης** (**Surrogate Function**). Έπειτα, ποσοτικοποιείται η αβεβαιότητα μέσω του Μπεϋζιανού αλγορίθμου μηχανικής μάθησης με όνομα **Γκαουσιανή διαδικασία παλινδρόμησης** (**Gaussian Process Regression**) για την αναζήτηση των υποψήφιων δειγμάτων που θα χρησιμοποιηθούν στην συνάρτηση αντικατάστασης[7374]. Συνήθως, χρησιμοποιείται για την βελτιστοποίηση των υπερπαραμέτρων σε προβλήματα μηχανικής μάθησης για την επίτευξη ενός αντικειμενικού στόχου (π.χ. μεγιστοποίηση της ακρίβειας ενός μοντέλου). Ο στόχος της διαδικασίας είναι να καταναλώσει παραπάνω χρόνο για την επιλογή των υπερπαραμέτρων ώστε να καλέσει λιγότερες φορές την αντικειμενική συνάρτηση. Συνοπτικά, τα βήματα για την εύρεση υπερπαραμέτρων είναι: 1) Δημιουργία ενός μοντέλου πιθανοτήτων την αντικειμενικής συνάρτησης (Objective Function), 2) Εύρεση των υπερπαραμέτρων που έχουν την καλύτερη απόδοση στην συνάρτηση αντικατάστασης, 3) Εφαρμογή των καλύτερων υπερπαραμέτρων στην αντικειμενική συνάρτηση, 4) Ανανέωση των δεδομένων και της συνάρτησης αντικατάστασης (Surrogate Function).

Τέλος επαναλαμβάνονται τα βήματα 2-4 όσες φορές έχει ορίσει ο χρήστης (**Max Trials**)[7576].

Επίπεδο Εγκατάλειψης (**Dropout Layer**), αποτελεί μια διαδικασία γενίκευσης για την εκπαίδευση των νευρωνικών δικτύων. Ο τρόπος που δουλεύει ο αλγόριθμος είναι τυχαία να «**εγκαταλείπει**» (μηδενίζει) νευρώνες κατά της διάρκεια της εκπαίδευσης, θέτοντας ένα ποσοστό νευρώνων που δεν θα χρησιμοποιηθούν σε ένα επίπεδο ώστε να αποφευχθεί η υπερπροσαρμογή των νευρώνων που ανιχνεύουν χαρακτηριστικά. Συνήθως, χρησιμοποιείται στον μηδενισμό νευρώνων ή συνδέσεων στα κρυφά δίκτυα αλλά μπορεί να χρησιμοποιηθεί και στα δεδομένα εισόδου ώστε να προστεθεί ένα είδος θορύβου. Μπορεί να χρησιμοποιηθεί σε διαφορετικούς τύπους επιπέδων όπως **πυκνά επίπεδα** (**Dense Layers**), **συνελικτικά επίπεδα** (**Convolutional Layers**) και **επαναλαμβανόμενα επίπεδα** (**Recurrent Layers**).

---

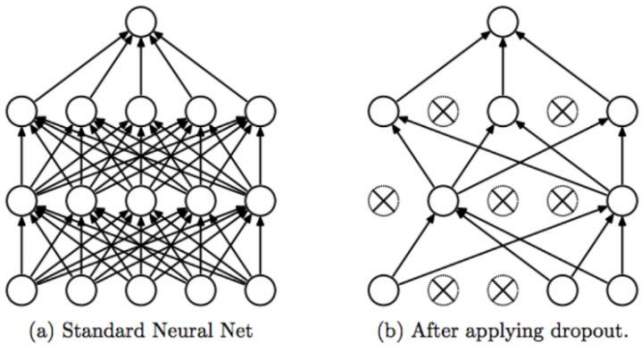
[73] Peter I. Frazier, "A tutorial on Bayesian Optimization", Arxiv, 10 July 2018, https://arxiv.org/pdf/1807.02811.pdf (Ανακτήθηκε 21/3/2021)

[74] Jason Brownlee, "How to Implement Bayesian Optimization from scratch in Python", Machine Learning Mastery, 9 October 2019, https://machinelearningmastery.com/what-is-bayesian-optimization/ (Ανακτήθηκε 21/3/2021)

[75] Will Koehrsen, "A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning", Towards Data Science, 24 June 2018, https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f (Ανακτήθηκε 21/3/2021)

[76] Jason Brownlee, "How to Implement Bayesian Optimization from scratch in Python", Machine Learning Mastery, 9 October 2019, https://machinelearningmastery.com/what-is-bayesian-optimization/ (Ανακτήθηκε 21/3/2021)

Πρέπει να σημειωθεί ότι η εγκατάλειψη δεν χρησιμοποιείται όταν έχει τελειώσει η εκπαίδευση του δικτύου και τη πρόβλεψη δειγμάτων.[77][78]



Εικόνα 6.12: Εικόνα επιπέδου εγκατάλειψης (Dropout Layer). Πηγή: Amar Budhiraja, "Dropout in (Deep) Machine Learning", 15 December 2016, https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (Ανακτήθηκε 3/7/2021)

Batch Normalization (Επίπεδο ομαλοποίησης παρτίδας) ονομάζεται μια τεχνική γενίκευσης που προσφέρει μια πιο σταθερή και πιο γρήγορη εκπαίδευση ενός νευρωνικού δικτύου[79]. Ο τρόπος που λειτουργεί είναι όπως της διαδικασίας της τυποποίησης (Standardization), ώστε το αποτέλεσμα από ένα επίπεδο να μετασχηματίζεται ανά παρτίδες και να έχει μέση τιμή ίση με το μηδέν και τυπική απόκλιση ίση με ένα. Αυτό το επίπεδο μπορεί να χρησιμοποιηθεί και στα δεδομένα που εισάγονται στο δίκτυο ή στα αποτελέσματα μετά την εφαρμογή των συναρτήσεων ενεργοποίησης σε κάθε κρυφό επίπεδο. Το επίπεδο αυτό, μπορεί να εφαρμοστεί σε τιμές διαφορετικών επιπέδων όπως πυκνά επίπεδα (Dense Layer), συνελικτικά επίπεδα (Convolutional Layer) και επαναλαμβανόμενα επίπεδα (Recurrent Layer). Υπάρχουν δύο τρόποι εφαρμογής του επιπέδου, είτε πριν την συνάρτηση ενεργοποίησης, είτε μετά. Προτείνεται να χρησιμοποιείται **μετά** την συνάρτηση ενεργοποίησης για τις συναρτήσεις που έχουν σχήμα S (πχ tanh και sigmoid) και **πριν** από την συνάρτηση ενεργοποίησης για συναρτήσεις όπως Relu[80].

[77] Jason Brownlee, "A gentle Introduction to Dropout for Regularizing Deep Neural Networks", Machine Learning Mastery, 3 December 2018, https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (Ανακτήθηκε 28/3/3021)
[78] Pierre Baldi, Peter Sadowski, "Understanding Dropout", https://bit.ly/2PFaUP5 (Ανακτήθηκε 28/3/2021)
[79] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry, "How does Batch Normalization Help Optimization", https://arxiv.org/pdf/1805.11604.pdf (Ανακτήθηκε 29/3/2021)
[80] Jason Brownlee, "A gentle Introduction to Batch Normalization for Deep Neural Networks", Machine Learning Mastery, 16 January 2019, https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/ (Ανακτήθηκε 29/3/2021)

Ο **Adam – Adaptive Moments** είναι ένας αλγόριθμος βελτιστοποίησης παραμέτρων, ο οποίος είναι υπολογιστικά αποτελεσματικός και δεν έχει μεγάλες απαιτήσεις μνήμης. Ο αλγόριθμος αυτός μοιάζει με ένα συνδυασμό μεταξύ του αλγορίθμου RMSProp και του αλγορίθμου Momentum με ορισμένες διαφορές[81]. Είναι αποτελεσματικός στην περίπτωση που το πρόβλημα περιέχει έναν μεγάλο αριθμό από δεδομένα ή μεγάλο αριθμών παραμέτρων. Ακόμη, αυτός ο αλγόριθμος είναι ιδανικός για την βελτιστοποίηση σε δεδομένα όπου υπάρχει θόρυβος ή αραίωση στις βαθμίδες(Gradients). Μια διαφορά που έχει ο αλγόριθμος αυτός σε σχέση με έναν βασικό αλγόριθμο βελτιστοποίησης τον **Stochastic Gradient Descent**, είναι η δυνατότητά του να ανανεώνει το **ποσοστό εκμάθησης** (**Learning Rate**) κατά την διάρκεια της διαδικασίας ανανέωσης των παραμέτρων, σε σχέση με τον αλγόριθμο Stochastic Gradient Descent που το ποσοστό εκμάθησης παραμένει σταθερό. Πλέον, ο αλγόριθμος Adam έχει συσταθεί ως ο προκαθορισμένος αλγόριθμος βελτιστοποίησης παραμέτρων σε αρχιτεκτονικές νευρωνικών δικτύων, εξαιτίας του ότι σε μεγαλύτερο ποσοστό των περιπτώσεων παρατηρείται καλύτερη απόδοση σε σχέση με άλλους αλγορίθμους βελτιστοποίησης[82].

ReLU/SELU/ELU πρόκειται για μη γραμμικές συναρτήσεις ενεργοποίησης. Χρησιμοποιούνται μετά την εφαρμογή μιας γραμμικής συνάρτησης και ο σκοπός τους είναι το δίκτυο να διευκρινίσει ποιοι νευρώνες θα είναι ενεργοί και ποιοι όχι με το να «μάθει» μη γραμμικές συσχετίσεις εκτός από γραμμικές. Η λειτουργία τους βασίζεται στο πόσο σημαντικός είναι ένας νευρώνας για την πρόβλεψη μιας τιμής[83]. Η πιο διαδεδομένη συνάρτηση που χρησιμοποιείται κατά κόρον είναι η συνάρτηση ReLU. Πρόκειται για μια αρκετά απλή συνάρτηση που εκφράζεται από τον τύπο $\mathbf{max}(\mathbf{0}, \boldsymbol{x})$. Όπου $\boldsymbol{x}$ είναι η τιμή από την γραμμική συνάρτηση που προήλθε. Θετικό της συνάρτησης αυτής, είναι η αποτελεσματικότητα της και ο εύκολος υπολογισμός της, ώστε το δίκτυο να συγκλίνει στο καλύτερο αποτέλεσμα πιο γρήγορα. Είναι αποτελεσματική στο πρόβλημα της **εξαφάνισης των βαθμίδων (Vanishing Gradient)** . Το πρόβλημα προκύπτει σε δίκτυα που αποτελούνται από πολλά επίπεδα και ο υπολογισμός των παραγώγων που είναι μικρότεροι από ένα, πολλαπλασιάζονται μεταξύ τους με αποτέλεσμα οι βαθμίδες να εξαφανιστούν κατά την διάρκεια της οπισθοδιάδοσης (Backpropagation). Ο τρόπος λειτουργίας του αλγορίθμου είναι ότι, για

[81] Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015
[82] Jason Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning", Machine Learning Mastery, 3 July 2017, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (Ανακτήθηκε 22/3/2021)
[83] Sebai Dorsaf, "Comprehensive synthesis of the main activation functions pros and cons", Medium, 2 April 2020, https://medium.com/analytics-vidhya/comprehensive-synthesis-of-the-main-activation-functions-pros-and-cons-dab105fe4b3b (Ανακτήθηκε 29/3/2021)

θετικές τιμές εισόδου έχει βαθμίδες (Gradients) ίσες με ένα και μηδέν για τις αρνητικές. Ένα πρόβλημα που προκαλείται είναι το λεγόμενο "Dying ReLU" κατά το οποίο οι βαθμίδες που είναι μηδέν επηρεάζουν το δίκτυο, με αποτέλεσμα να υπάρχουν επιπτώσεις στην βελτίωση της μάθησης. Η συνάρτηση **ELU (Exponential Linear Unit)** η οποία δίνεται από τον τύπο

$$\boldsymbol{elu(x)} = \begin{cases} \boldsymbol{a(\exp(x) - 1)}, & \boldsymbol{x \leq 0} \\ \boldsymbol{x}, & \boldsymbol{x > 0} \end{cases}$$

και αποτελεί μια παραλλαγή από την συνάρτηση ReLU. Στόχος της συνάρτησης είναι, να κεντροποιήσει στο μηδέν τις τιμές ενεργοποίησης και να καταπολεμήσει το πρόβλημα που παρουσιάστηκε στην συνάρτηση ReLU. Η τελευταία συνάρτηση ονομάζεται **SELU (Scaled Exponential Linear Unit)** η οποία δίνεται από τον παρακάτω τύπο.

$$\boldsymbol{selu(x)} = \boldsymbol{\lambda} \begin{cases} \boldsymbol{a(\exp(x) - 1)}, & \boldsymbol{x \leq 0} \\ \boldsymbol{x}, & \boldsymbol{x > 0} \end{cases}$$

Η συνάρτηση αυτή είναι μια επέκταση της συνάρτησης ELU και έχει ιδιότητες αυτοομαλοποίησης (Self-Normalization). Αυτό επιτυγχάνεται επειδή οι τιμές ενεργοποίησης που έχουν μέση τιμή κοντά στο μηδέν και διακύμανση κοντά στο ένα, βοηθιούνται από τη συνάρτηση, ώστε η μέση τιμή (Mean) να συγκλίνει στο μηδέν και η διακύμανση (Variance) στο ένα. Αυτό κάνει πιο ευέλικτο το δίκτυο και επιτρέπει την εκπαίδευση ενός δικτύου που αποτελείται από πολλά επίπεδα[84].

Η **Γενίκευση (Regularization)** αποτελεί έναν αποτελεσματικό τρόπο βελτίωσης ενός νευρωνικού δικτύου ώστε να αποφευχθεί η υπερπροσαρμογή (Overfitting). Αυτή η τεχνική αποθαρρύνει την εκμάθηση ενός περίπλοκου ή ευέλικτου μοντέλου ώστε να μειώνει σημαντικά την διακύμανση (Variance) του μοντέλου αλλά ταυτόχρονα να μην αυξάνεται η μεροληψία (Bias). Δυο από τις τεχνικές αυτές είναι, η L1 γενίκευση (L1 Regularization) και η L2 γενίκευση (L2 Regularization). Η L2 γενίκευση γνωστή και ως Weight Decay ή Ridge παλινδρόμηση (Ridge Regression), προσθέτει έναν όρο γενίκευσης $\boldsymbol{\Omega} = \left(\frac{\boldsymbol{\lambda}}{\boldsymbol{2}}\right) \left|\left|\boldsymbol{\theta}\right|\right|_{\boldsymbol{2}}^{\boldsymbol{2}}$ στην συνάρτηση απώλειας.

$$\boldsymbol{J_{train}(\theta)} = \boldsymbol{J_{old_{train}}(\theta)} + \frac{\boldsymbol{\lambda}}{\boldsymbol{2}} \left|\left|\boldsymbol{\theta}\right|\right|_{\boldsymbol{2}}^{\boldsymbol{2}}$$

[84]Dabal Pedamonti , "Comparison of non-linear activations functions for deep neural networks on MNIST classification Task ", https://arxiv.org/pdf/1804.02763.pdf (Ανακτήθηκε 29/3/2021)

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\alpha}(\, \boldsymbol{\nabla_{\theta} J^{old}(\theta)} + \boldsymbol{\lambda \theta}\,)$$

Όπου $\boldsymbol{\alpha}$ είναι το ποσοστό εκμάθησης, $\boldsymbol{\lambda}$ είναι η υπερπαράμετρος της γενίκευσης και $\boldsymbol{\nabla_{\theta} J^{old}(\theta)}$ είναι η διαβάθμιση κλίσης των παραμέτρων $\boldsymbol{\Theta}$. Το αποτέλεσμα αυτής της τεχνικής είναι οι παράμετροι να τείνουν σιγά σιγά προς το μηδέν. Η L1 γενίκευση ή Lasso επίσης προσθέτει έναν όρο στην συνάρτηση απώλειας $\boldsymbol{\lambda}\left|\left|\boldsymbol{\theta}\right|\right|_{\mathbf{1}}$

$$\boldsymbol{J_{train}(\theta) = J_{old_{train}}(\theta) + \lambda}\left|\left|\boldsymbol{\theta}\right|\right|_{\mathbf{1}}$$

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\alpha}(\, \boldsymbol{\nabla_{\theta} J^{old}(\theta)} + \boldsymbol{\lambda\, sign(\theta)})$$

Το αποτέλεσμα από αυτήν την εφαρμογή είναι οι παράμετροι που είναι κοντά στο μηδέν να γίνουν μηδέν ώστε το συνολικό αποτέλεσμα να είναι πιο αραιό[85][86].

Η **Αρχικοποίηση βαρών (Weight Initialization)** είναι μια βασική διαδικασία για ένα δίκτυο ώστε να αποδώσει σωστά. Χωρίς την σωστή αρχικοποίηση των βαρών μπορεί να υπάρξει το πρόβλημα της έκρηξης ή της εξαφάνισης των βαθμίδων (Exploding/ Vanishing Gradient), όπου η τιμή των βαθμίδων απώλειας στην ανανέωση των βαρών είτε είναι πολύ μεγάλη είτε πάρα πολύ μικρή. Αυτό έχει ως αποτέλεσμα, το δίκτυο να χρειαστεί περισσότερο χρόνο για να συγκλίνει σε ένα καλό πόρισμα, εάν είναι σε θέση να το κάνει και δεν προκύψουν τα βάρη με μεγάλες τιμές (Exploding) όπου σε αυτήν την περίπτωση δεν θα μπορεί να εκπαιδευτεί[87]. Ένα αρκετά απλό νευρωνικό δίκτυο μπορεί να περιέχει εκατομμύρια βάρη, οπότε η σωστή αρχικοποίηση τους θα αποφύγει τα παραπάνω προβλήματα. Δύο βασικές τεχνικές αρχικοποίησης είναι η αρχικοποίηση **He** και **Xavier / Glorot**. Η αρχικοποίηση **Glorot** πήρε το όνομα της από τον **Xavier Glorot** και ο σκοπός της είναι να αρχικοποιηθούν τα βάρη από την Γκαουσιανή κατανομή με **μέση τιμή** (**Mean**) μηδέν και η **τυπική απόκλιση** (**Standard Deviation**) να δίνεται από τον τύπο:

[85] Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015

[86] Alfredo Canziani, "Week 14- Practicum: Overfitting and Regularization, and Bayesian Neural Nets", Youtube, 25 September 2020, https://www.youtube.com/watch?v=DL7iew823c0 (Ανακτήθηκε 29/3/2021)

[87] James Dellinger, "Weight Initialization in Neural Networks: A journey From the Basics to Kaiming", Towards Data Science, 3 April 2019, https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79 (Ανακτήθηκε 29/3/2021)

$$\boldsymbol{\sigma = \frac{\sqrt{2}}{\sqrt{n_{in} + n_{out}}}}$$

Όπου $\boldsymbol{n_{in}}$ $\boldsymbol{και}$ $\boldsymbol{n_{out}}$ είναι ο αριθμός των συνδέσεων εισόδου και εξόδου αντίστοιχα. Η αρχικοποίηση **He** πήρε το όνομα της από τον **Kaiming He** (η διαφορά της με την αρχικοποίηση Xavier είναι ένας πολλαπλασιασμός με το δύο στην διακύμανση) και η τυπική απόκλιση της ορίζεται από τον τύπο[88].

$$\boldsymbol{\sigma = \sqrt{2}\frac{\sqrt{2}}{\sqrt{n_{in} + n_{out}}}}$$

Όπως προαναφέρθηκε πραγματοποιήθηκαν τρεις διαφορετικοί τρόποι ανάλυσης της γονιδιακής έκφρασης με την χρήση των νευρωνικών δικτύων. Για τα δύο δίκτυα ταξινόμησης χρησιμοποιήθηκε η ίδια αρχιτεκτονική και ο συντονισμός των ίδιων υπερπαραμέτρων και για τις τρείς διαφορετικές αναλύσεις. Παρακάτω, θα παρουσιαστεί η επιλογή των τιμών για τον συντονισμό των υπερπαραμέτρων και για τις δυο διαφορετικές αρχιτεκτονικές.

6.3.1 Πρώτο δίκτυο ταξινόμησης – Deep Neural Network

Το πρώτο δίκτυο ταξινόμησης υλοποιείται από ένα απλό βαθύ νευρωνικό δίκτυο. Παρακάτω, αναφέρονται ποιες υπερπαραμέτροι συντονίστηκαν (Tuned), καθώς και το σύνολο τιμών που ορίστηκε για κάθε υπερπαράμετρο.

- Αριθμός των επιπέδων (Number of Layers): 1-10
- Αριθμός των Νευρώνων (Number of Units): 128, 256, 384, 512, 640, 768, 896, 1024
- Συναρτήσεις Ενεργοποίησης (Activation Functions): Relu, Selu, Elu, Linear
- Μέθοδος αρχικοποίησης στα βάρη (Kernel Initialization): Glorot Normal, He Normal, Glorot Uniform, He Uniform
- Ποσοστό επιπέδου εγκατάλειψης (Dropout Rate): 0.0,0.1, 0.2, 0.3, 0.4, 0.5
- Ποσοστό Εκμάθησης (Learning Rate): 0.1,0.01,0.001,0.0001
- Μέθοδος Γενίκευσης (Regularization Algorithm): L1, L2

[88] Mustafa Murat ARAT, "Weight Initialization Schemes – Xavier (Glorot) and He", Mustafa Murat ARAT, 5 February 2019, https://mmuratarat.github.io/2019-02-25/xavier-glorot-he-weight-init (Ανακτήθηκε 29/3/2021)

Μετά από κάθε πυκνό επίπεδο (Dense Layer), χρησιμοποιούνταν ένα επίπεδο ομαλοποίησης παρτίδας (Batch Normalization Layer) και στην συνέχεια έναν επίπεδο εγκατάλειψης (Dropout Layer). Επειδή η ανάλυση περιείχε την ταξινόμηση μεταξύ πολλαπλών ετικετών, χρησιμοποιήθηκε η συνάρτηση ενεργοποίησης **SoftMax** για το τελευταίο πυκνό επίπεδο. Για την εκπαίδευση του δικτύου χρησιμοποιήθηκε σαν συνάρτηση απώλειας η **αραιά κατηγορηματική εγκάρσια εντροπία (Sparse Categorical Cross Entropy).** Επιλέχθηκε η συνάρτηση αυτή επειδή οι ετικέτες των κλάσεων ήταν σε μορφή ακεραίου από 1-29. Για την ανανέωση στα βάρη χρησιμοποιήθηκε η συνάρτηση Adam και για να μετρηθεί η απόδοση του δικτύου ορίστηκε σαν **μέτρηση** (**Metrics**) η **αραιά κατηγορηματική ακρίβεια (Sparse Categorical Accuracy)**. Επειδή για τον συντονισμό των υπερπαραμέτρων χρησιμοποιήθηκε η Μπεϋζιανή βελτιστοποίηση, ορίστηκε σαν αντικειμενική συνάρτηση η μεγιστοποίηση της τιμής **της αραιάς κατηγορηματικής ακρίβειας των δεδομένων αξιολόγησης** (**Validation Sparse Categorical Accuracy**) και ο μέγιστος αριθμός δοκιμών τέθηκε στους τριάντα. Για κάθε δίκτυο που δημιουργούσε η Μπεϋζιανή βελτιστοποίηση, το δίκτυο αυτό εκπαιδευόταν στο 80% των δεδομένων που είχαν παραχθεί από τους διαφορετικούς τύπους των Autoencoder, αξιολογούταν στο 20% των δεδομένων για 15 «εποχές» (**Epochs**) και τα δεδομένα χωριζόντουσαν σε παρτίδες (Batch Size), με κάθε παρτίδα να περιέχει 256 δείγματα. Ταυτόχρονα χρησιμοποιήθηκε μια συνάρτηση επανάκλησης, η συνάρτηση **πρόωρης διακοπής** (**Early Stopping**). Σε αυτή τέθηκε η παρακολούθηση του **σφάλματος αξιολόγησης** (**Validation Loss**) και αν δεν βελτιωνόταν μετά από 4 «εποχές» σταματούσε την εκπαίδευση. Στην συνέχεια, μετά την εύρεση των καλύτερων υπερπαραμέτρων, ξανά εκπαιδεύτηκε το μοντέλο για 100 «εποχές» (**Epochs**) και ίδιο αριθμό παρτίδων.

6.3.2 Δεύτερο δίκτυο ταξινόμησης – Deep Cross Network

Το Deep Cross Network είναι μια αρχιτεκτονική νευρωνικών δικτύων που αποτελείται από ένα απλό βαθύ νευρωνικό δίκτυο (DNN) και ένα άλλο δίκτυο διασταύρωσης (Cross Network), τα οποία συνδυάζονται σε ένα άλλο επίπεδο συνένωσης (Combination Layer). Ο λόγος κατασκευής αυτής της αρχιτεκτονικής είναι ότι τα βαθιά νευρωνικά δίκτυα (DNNS) είναι σε θέση να μαθαίνουν αλληλεπιδράσεις μεταξύ των χαρακτηριστικών οι οποίες μπορεί να μην είναι τόσο αποτελεσματικές. Από την άλλη, πρότειναν τον συνδυασμό του DNN με ένα δίκτυο διασταύρωσης το οποίο είναι σε θέση για αποτελεσματική εκμάθηση αλληλεπιδράσεων μεταξύ των χαρακτηριστικών σε έναν οριακό βαθμό. Το δίκτυο διασταύρωσης αποτελείται από πολλαπλά επίπεδα στα οποία το βάθος των επιπέδων καθορίζει το επίπεδο των

αλληλεπιδράσεων[89]. Παρακάτω αναφέρονται για το δίκτυο αυτό, ποιοι υπερπαραμέτροι συντονίστηκαν:

- Αριθμός των επιπέδων (Number of Layers): 2-5
- Αριθμός των Νευρώνων (Number of Units): 32,64,128,256,512
- Ποσοστό επιπέδου εγκατάλειψης (Dropout Rate): 0.0,0.1, 0.2, 0.3, 0.4, 0.5
- Ποσοστό Εκμάθησης (Learning Rate): 0.1,0.01,0.001,0.0001,0.00001

Για την εκπαίδευση του συγκεκριμένου δικτύου χρησιμοποιήθηκε η ίδια μετρική, συνάρτηση απώλειας και συνάρτηση βελτιστοποίησης των παραμέτρων με το παραπάνω μοντέλο ταξινόμησης, καθώς και ο ίδιος αριθμός μέγιστων δοκιμών και η ίδια αντικειμενική συνάρτηση για την Μπεϋζιανή βελτιστοποίηση. Τα δίκτυα που δημιουργήθηκαν με τις εκάστοτε επιλεγμένες παραμέτρους εκπαιδεύτηκαν και αυτά με το 80% των δεδομένων που είχαν παραχθεί από όλους τους τύπους των Autoencoders, και αξιολογήθηκαν με τα υπόλοιπα δεδομένα που αποτελούσαν το 20%. Εκπαιδεύτηκαν για 10 «εποχές» (Epochs) και σε παρτίδες των 256 δειγμάτων. Μετά την εύρεση του δικτύου με το καλύτερο σύνολο υπερπαραμέτρων ξανά εκπαιδεύτηκε το δίκτυο για 50 «εποχές» (Epochs) εκτός στα δεδομένα που προήλθαν από το μοντέλο Variational Autoencoder που εκπαιδεύτηκε για 300 εποχές.

6.3.3 Δίκτυο Autoencoder

Για αυτήν την ανάλυση χρησιμοποιήθηκε για την μείωση των χαρακτηριστικών ένα απλό δίκτυο Autoencoder. Για την εύρεση των κατάλληλων υπερπαραμέτρων ορίστηκαν οι παρακάτω υπερπαραμέτροι:

- Αριθμός των επιπέδων (Number of Layers): 1-3
- Αριθμός των Νευρώνων (Number of Units): 256,512,768,1024
- Συναρτήσεις Ενεργοποίησης (Activation Functions): Relu, Selu, Elu, Linear
- Μέθοδος αρχικοποίησης στα βάρη (Kernel Initialization): Glorot Normal, He Normal, Glorot Uniform, He Uniform
- Ποσοστό επιπέδου εγκατάλειψης (Dropout Rate): 0.0,0.1, 0.2, 0.3, 0.4, 0.5
- Αριθμός Νευρώνων στον Λανθάνων Χώρο (Latent Space):10, 20, 30, 40, 50, 60, 70, 80, 90, 100

[89] Ruoxi Wang, Bin Fu, Gang Fu, Mingliang Wang,"Deep & Cross Network for Ad Click Predictions", Arxiv, 17 August 2017, https://arxiv.org/pdf/1708.05123.pdf (Ανακτήθηκε 22/3/2021)

- Συναρτήσεις ενεργοποίησης αποκωδικοποιητή (Activation Function Decoder): Sigmoid, Linear
- Ποσοστό Εκμάθησης (Learning Rate): 0.1,0.01,0.001,0.0001

Στην συνέχεια το αποτέλεσμα από το κάθε ένα πυκνό επίπεδο (Dense Layer), εισήλθε σαν είσοδος σε ένα επίπεδο ομαλοποίησης παρτίδας (Batch Normalization Layer) για την διαμόρφωση του αποτελέσματος. Τέλος, το αποτέλεσμα από το επίπεδο ομαλοποίησης, χρησιμοποιήθηκε σαν είσοδος σε ένα επίπεδο εγκατάλειψης (Dropout Layers) για το μηδενισμό ενός ποσοστού των δεδομένων. Χρησιμοποιήθηκε η Μπεϋζιανή βελτιστοποίηση με μέγιστο όριο δοκιμών (Max Trials) στις πενήντα επαναλήψεις και τέθηκε σαν αντικειμενικός στόχος, η ελαχιστοποίηση του σφάλματος αξιολόγησης (Validation Error). Σε κάθε επανάληψη δημιουργούταν ένα νευρωνικό δίκτυο με τις εκάστοτε επιλεγμένες παραμέτρους. Εκπαιδευόταν με το 80% των δεδομένων εκπαίδευσης και αξιολογούταν με το υπόλοιπο 20% για 10 «εποχές» (Epochs) και σε παρτίδες με μέγεθος κάθε παρτίδας (Batch Size) 256 δείγματα. Για την εκπαίδευση κάθε δικτύου εφαρμόστηκε σαν συνάρτηση απώλειας (Loss Function), η συνάρτηση του μέσου τετραγωνικού σφάλματος (Mean Squared Error) και για την βελτιστοποίηση και ανανέωση των παραμέτρων χρησιμοποιήθηκε ο αλγόριθμος Adam. Μετά τις 50 επαναλήψεις επιλέχθηκε το μοντέλο που παρήγαγε το μικρότερο ποσοστό σφάλματος και ξανά εκπαιδεύτηκε με όλο το σύνολο των δεδομένων εκπαίδευσης για 200 «εποχές» (Epochs) και με το ίδιο μέγεθος κάθε παρτίδας (Batch Size). Εφόσον είχε ετοιμαστεί το μοντέλο με τον Autoencoder χρησιμοποιήθηκε για να κωδικοποιήσει τα δεδομένα εισόδου, σε δεδομένα μικρότερου διανυσματικού χώρου και στην συνέχεια τα δεδομένα αυτά τροφοδοτήθηκαν στα δύο διαφορετικά δίκτυα ταξινόμησης.

6.3.4 Δίκτυο Denoising Autoencoder

Για την ανάλυση αυτή, χρησιμοποιήθηκε για την μείωση των χαρακτηριστικών ένα δίκτυο Denoising Autoencoder. Η μόνη διαφορά με το δίκτυο των απλών Autoencoder είναι ότι στα δεδομένα εισόδου προστέθηκε κατά 5% θόρυβος Γκάους (Gaussian Noise) και ένα 5% Γκαουσιανής Εγκατάλειψης (Gaussian Dropout), όπου ουσιαστικά και αυτή η διαδικασία προσθέτει ένα ποσοστό Γκαουσιανού θορύβου για τον μηδενισμό τιμών εισόδου[90]. Έτσι τα μοντέλα που δημιουργούνταν από την Μπεϋζιανή βελτιστοποίηση εκπαιδευόταν, με είσοδο τα

[90] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Rusian Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf (Ανακτήθηκε 22/3/2021)

δεδομένα που είχαν τροποποιηθεί με θόρυβο και με έξοδο τα αρχικά δεδομένα. Εκτός από αυτήν την διαφορά, η υλοποίηση του δικτύου είναι ακριβώς η ίδια με αυτήν των Autoencoder, με τον ίδιο συντονισμό των υπερπαραμέτρων και τις ίδιες συναρτήσεις βελτιστοποίησης, απώλειας και μέτρησης.

6.3.5 Δίκτυο Variational Autoencoder

Για αυτήν την ανάλυση χρησιμοποιήθηκε για την μείωση των χαρακτηριστικών ένα δίκτυο από Variational Autoencoders. Για την εύρεση των κατάλληλων υπερπαραμέτρων ορίστηκαν οι παρακάτω υπερπαραμέτροι:

- Αριθμός των επιπέδων (Number of Layers): 0-8
- Αριθμός των Νευρώνων (Number of Units): 256,512,768,1024
- Συναρτήσεις Ενεργοποίησης (Activation Functions): Relu, Selu, Elu, Linear
- Μέθοδος αρχικοποίησης στα βάρη (Kernel Initialization): Glorot Normal, He Normal, Glorot Uniform, He Uniform
- Αριθμός Νευρώνων στον Λανθάνων Χώρο (Latent Space):10, 20, 30, 40, 50, 60, 70, 80, 90, 100
- Συναρτήσεις ενεργοποίησης αποκωδικοποιητή (Activation Function Decoder): Sigmoid, Linear
- Ποσοστό Εκμάθησης (Learning Rate): 0.0001,0.00001

Σε σχέση με τα δύο προηγούμενα δίκτυα Autoencoder για την συνάρτηση απώλειας χρησιμοποιήθηκε το άθροισμα του μέσου τετραγωνικού σφάλματος (Mean Squared Error) και της απόκλισης Kullback-Leibler και για την συνάρτηση βελτιστοποίησης των παραμέτρων χρησιμοποιήθηκε η συνάρτηση Adam. Χρησιμοποιήθηκε όπως και στις προηγούμενες αρχιτεκτονικές η Μπεϋζιανή βελτιστοποίηση για την εύρεση των υπερπαραμέτρων. Ο αριθμός των μέγιστων δοκιμών είχε οριστεί στους πενήντα ενώ επίσης τέθηκε αντικειμενικός στόχος η ελαχιστοποίηση του σφάλματος αξιολόγησης (Validation Loss). Για την εκπαίδευση των δεδομένων χρησιμοποιήθηκε το 80% των δεδομένων εκπαίδευσης και το υπόλοιπο 20% των δεδομένων για την αξιολόγηση των δικτύων. Τα δίκτυα εκπαιδευόντουσαν για 10 «εποχές» (Epochs) και σε παρτίδες (Batch size) των 256 δειγμάτων όπως και στις προηγούμενες αρχιτεκτονικές. Μετά την εύρεση του καλύτερου μοντέλου που έδινε το λιγότερο σφάλμα αξιολόγησης, χρησιμοποιήθηκε αυτό το μοντέλο για να ξανά εκπαιδευτεί με όλα τα δεδομένα για 200 «εποχές» (Epochs) και το ίδιο μέγεθος παρτίδας (Batch Size). Τέλος, εφόσον είχε εκπαιδευτεί το μοντέλο Variational Autoencoder χρησιμοποιήθηκε ο κωδικοποιητής για να

κωδικοποιήσει τα δεδομένα, σε δεδομένα διανυσματικού χώρου λιγότερων διαστάσεων, τα οποία δεδομένα χρησιμοποιήθηκαν ως είσοδος στα δίκτυα ταξινόμησης.

6.4 Ανάλυση Αποτελεσμάτων

Παρακάτω θα αναφερθούν τα αποτελέσματα που παρατηρήθηκαν στο τέλος της εκπαίδευσης κάθε δικτύου.

6.4.1 Δίκτυο Autoencoder

Μετά την χρήση της Μπεϋζιανής βελτιστοποίησης, οι καλύτεροι υπερπαραμέτροι που επιλέχθηκαν ήταν δύο κρυφά επίπεδα (ένα πυκνό επίπεδο και ο λανθάνων χώρος) στον κωδικοποιητή και ένα στον αποκωδικοποιητή. Το πρώτο επίπεδο στον κωδικοποιητή και στον αποκωδικοποιητή αποτελούνταν από 256 νευρώνες (Units) και ο λανθάνων χώρος κωδικοποιήθηκε σε διανυσματικό πίνακα εκατό θέσεων. Το ποσοστό του επιπέδου εγκατάλειψης ανήλθε σε 0.0 για το πρώτο επίπεδο του κωδικοποιητή και 0.5 για τον αποκωδικοποιητή. Αυτό σημαίνει ότι το 0% των νευρώνων ήταν ανενεργοί στο επίπεδο του κωδικοποιητή (ουσιαστικά ήταν όλοι οι νευρώνες ενεργοί) και το 50% των νευρώνων του αποκωδικοποιητή ήταν ενεργοί. Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε από όλα τα επίπεδα εκτός του τελευταίου επιπέδου ήταν η συνάρτηση **ELU** σε συνδυασμό με την αρχικοποίηση των βαρών με την μέθοδο **He Uniform**. Για το τελευταίο επίπεδο χρησιμοποιήθηκε η γραμμική (**Linear**) συνάρτηση ενεργοποίησης και το ποσοστό εκμάθησης που είχε τεθεί ήταν στο 0.001.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_loss', direction='min')
Trial summary
Hyperparameters:
activations: elu
kernel_initializers: he_uniform
kernel_regularizer: l2
latent_space: 100
layers: 1
activations_decoder: linear
learning_rate: 0.001
units_encoder_0: 256
dropout_encoder_0: 0.0
units_decoder_0: 256
dropout_decoder_0: 0.5
```

Εικόνα 6.13: Εικόνα καλύτερων υπερπαραμέτρων Autoencoder.

Το ποσοστό σφάλματος στα δεδομένα εκπαίδευσης του Autoencoder ήταν στο 0.0011

```
mse(X_train,X_train_pred)
```

```
0.0011049408736204796
```

Εικόνα 6.14: Μέσο τετραγωνικό σφάλμα Autoencoder στα δεδομένα εκπαίδευσης..

καθώς χρησιμοποιήθηκε και η συνάρτηση μέσου τετραγωνικού σφάλματος (**Mean Squared Error**) μεταξύ των δεδομένων δοκιμής (**Test Data**) που δεν είχε ξανά δει προηγουμένως το μοντέλο, με ποσοστό σφάλματος στο 0.0013.

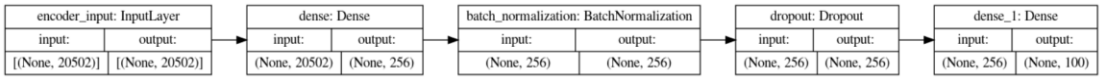
```
mse(X_test,X_test_pred)
```

```
0.0013934857470744822
```

Εικόνα 6.15: Μέσο τετραγωνικό σφάλμα Autoencoder στα δεδομένα δοκιμής.

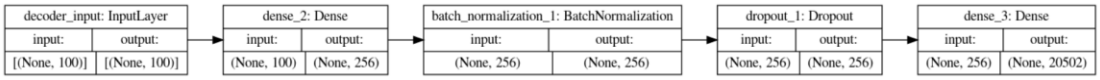
Παρακάτω παρουσιάζεται το δίκτυο του κωδικοποιητή και του αποκωδικοποιητή αντίστοιχα.

Κωδικοποιητής – Encoder



Εικόνα 6.16: Αρχιτεκτονική Κωδικοποιητή Autoencoder

Αποκωδικοποιητής – Decoder



Εικόνα 6.17: Αρχιτεκτονική Αποκωδικοποιητή Autoencoder

Ο συνολικός αριθμός παραμέτρων του μοντέλου του Autoencoder ήταν 10.571.386

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 20502)]           0
_________________________________________________________________
encoder (Functional)         (None, 100)               5275492
_________________________________________________________________
decoder (Functional)         (None, 20502)             5295894
=================================================================
Total params: 10,571,386
Trainable params: 10,570,362
Non-trainable params: 1,024
_________________________________________________________________
```

Εικόνα 6.18: Περίληψη Autoencoder

6.4.2 Δίκτυο Denoising Autoencoder

Οι καλύτεροι υπερπαραμέτροι που επιλέχθηκαν ήταν τρία κρυφά επίπεδα (δύο πυκνά επίπεδα και ο λανθάνων χώρος) στον κωδικοποιητή και δύο στον αποκωδικοποιητή, με το πρώτο επίπεδο στον κωδικοποιητή να αποτελείται από **1024** νευρώνες (**Units**) και στον αποκωδικοποιητή να αποτελείται από **768** νευρώνες (**Units**)και το δεύτερο επίπεδο να

αποτελείται από **1024** νευρώνες στον κωδικοποιητή σε σχέση με το δεύτερο επίπεδο στον αποκωδικοποιητή που αποτελείται από **256**. Το ποσοστό εγκατάλειψης στον πρώτο επίπεδο του κωδικοποιητή και του αποκωδικοποιητή είχε τεθεί στο **0.4** και **0.5** αντίστοιχα, καθώς το ποσοστό στο δεύτερο επίπεδο είχε τεθεί στο **0.0** και **0.2** για κωδικοποιητή και αποκωδικοποιητή εξίσου. Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε ήταν η **Relu** με αρχικοποίηση των βαρών με τον αλγόριθμο **He Normal.** Τέλος, ο λανθάνων χώρος κωδικοποιήθηκε σε διανυσματικό πίνακα **πενήντα** θέσεων.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_loss', direction='min')
Trial summary
Hyperparameters:
layers: 2
units_encoder_0: 1024
activations: relu
kernel_initializers: he_normal
dropout_encoder_0: 0.4
latent_space: 50
units_decoder_0: 768
dropout_decoder_0: 0.5
activations_decoder: linear
learning_rate: 0.01
units_encoder_1: 1024
dropout_encoder_1: 0.0
units_decoder_1: 256
dropout_decoder_1: 0.2
```

Εικόνα 6.19: Εικόνα καλύτερων υπερπαραμέτρων Denoising Autoencoder

Παρακάτω παρουσιάζεται το δίκτυο του κωδικοποιητή και του αποκωδικοποιητή.
Το ποσοστό σφάλματος στα δεδομένα εκπαίδευσης του Denoising Autoencoder ήταν στο 0.0010

```
mse(X_train,X_train_pred)
```

```
0.0010573799745008424
```

Εικόνα 6.20: Μέσο τετραγωνικό σφάλμα Denoising Autoencoder στα δεδομένα εκπαίδευσης.

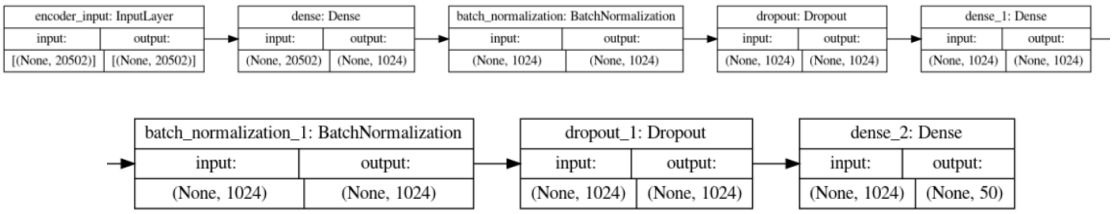
καθώς χρησιμοποιήθηκε και η συνάρτηση **μέσου τετραγωνικού σφάλματος** (**Mean Squared Error**) μεταξύ των δεδομένων δοκιμής (**Test Data**) που δεν είχε ξανά δει προηγουμένως το μοντέλο, με ποσοστό σφάλματος στο 0.0013.

```
mse(X_test,X_test_pred)
```

```
0.001367627496780833
```

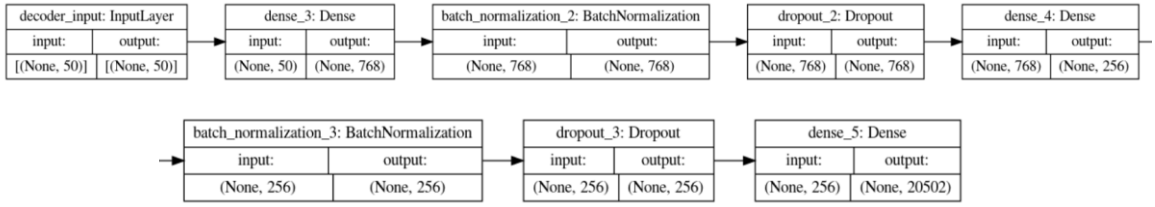
Εικόνα 6.21: Μέσο τετραγωνικό σφάλμα Denoising Autoencoder στα δεδομένα δοκιμής.

Κωδικοποιητής – Encoder



Εικόνα 6.22: Αρχιτεκτονική Κωδικοποιητή Denoising Autoencoder.

Αποκωδικοποιητής – Decoder



Εικόνα 6.23: Αρχιτεκτονική Αποκωδικοποιητή Denoising Autoencoder.

Ο αριθμός των παραμέτρων του δικτύου ήταν 27.613.256.

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 20502)]           0
_________________________________________________________________
encoder (Functional)         (None, 50)                22104114
_________________________________________________________________
decoder (Functional)         (None, 20502)             5509142
=================================================================
Total params: 27,613,256
Trainable params: 27,607,112
Non-trainable params: 6,144
_________________________________________________________________
```

Εικόνα 6.24: Περίληψη Denoising Autoencoder.

6.4.3 Δίκτυο Variational Autoencoder

Μετά την χρήση της Μπεϋζιανής βελτιστοποίησης, οι καλύτεροι υπερπαραμέτροι που επιλέχθηκαν ήταν τέσσερα κρυφά επίπεδα (ένα Dense Layer, ένα επίπεδο υπολογισμού του μέσου, ένα επίπεδο υπολογισμού της τυπικής απόκλισης και το τελευταίο επίπεδο ήταν η δειγματοληψία από τα δύο προηγούμενα επίπεδα) στον κωδικοποιητή και δύο στον αποκωδικοποιητή με το πρώτο κρυφό επίπεδο στον κωδικοποιητή και στον αποκωδικοποιητή να αποτελείται από 1024 νευρώνες (**Units**) εκτός από τα επίπεδα μέσου, τυπικής απόκλισης και δειγματοληψίας που είχαν μέγεθος λανθάνων χώρου (Latent Space) εβδομήντα θέσεων.

Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε από όλα τα επίπεδα εκτός του τελευταίου επιπέδου ήταν η συνάρτηση **ReLU** σε συνδυασμό με την αρχικοποίηση των βαρών με την μέθοδο **Glorot Uniform**. Για το τελευταίο επίπεδο χρησιμοποιήθηκε η γραμμική (**Linear**) συνάρτηση ενεργοποίησης και το ποσοστό εκμάθησης που είχε τεθεί ήταν στο 0.0001.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_loss', direction='min')
Trial summary
Hyperparameters:
layers: 1
latent_space: 70
units: 1024
activations: relu
kernel_initializers: glorot_uniform
activation_decoder: linear
learning_rate: 0.0001
```

Εικόνα 6.25: Εικόνα καλύτερων υπερπαραμέτρων Variational Autoencoder.

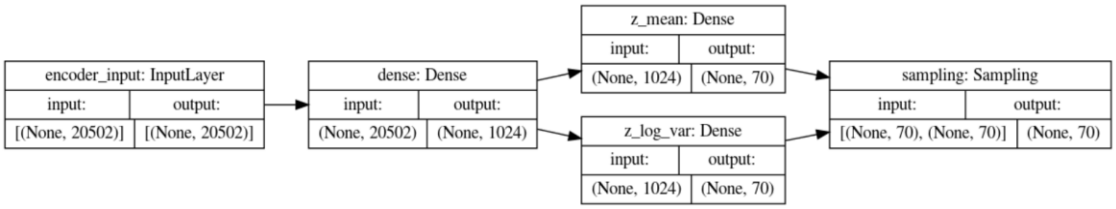
Το ποσοστό σφάλματος στα δεδομένα εκπαίδευσης και δοκιμής του Variational Autoencoder κατά την τελευταία «εποχή» ήταν 14.8 και 22.5 αντίστοιχα. Τα ποσοστά αυτά φαίνονται αρκετά μεγάλα, αλλά στην πραγματικότητα δεν είναι απλά η συνάρτηση απώλειας πολλαπλασιάστηκε με τον αρχικό αριθμό χαρακτηριστικών που υπήρχαν δηλαδή 20502.

```
Epoch 200/200
42/42 [==============================] - 2s 38ms/step - loss: 14.8114 - val_loss: 22.5108
```

Εικόνα 6.26: Αποτελέσματα σφάλματος Variational Autoencoder

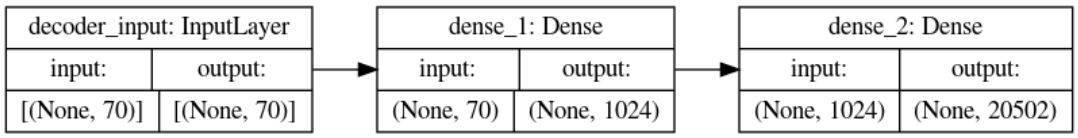
Προφανώς, αν δεν υπήρχε αυτός ο πολλαπλασιασμός το σφάλμα θα ήταν αρκετά μικρότερο. Παρακάτω παρουσιάζεται το δίκτυο του κωδικοποιητή και του αποκωδικοποιητή.

Κωδικοποιητής – Encoder



Εικόνα 6.27: Αρχιτεκτονική Κωδικοποιητή Variational Autoencoder

Αποκωδικοποιητής – Decoder



Εικόνα 6.28: Αρχιτεκτονική Αποκωδικοποιητή Variational Autoencoder

Ο συνολικός αριθμός παραμέτρων του μοντέλου του Variational Autoencoder ήταν 21.138.572 παράμετροι για τον κωδικοποιητή και 21.087.254 για τον αποκωδικοποιητή.

```
Model: "encoder"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
encoder_input (InputLayer)      [(None, 20502)]      0
__________________________________________________________________________________________________
dense (Dense)                   (None, 1024)         20995072    encoder_input[0][0]
__________________________________________________________________________________________________
z_mean (Dense)                  (None, 70)           71750       dense[0][0]
__________________________________________________________________________________________________
z_log_var (Dense)               (None, 70)           71750       dense[0][0]
__________________________________________________________________________________________________
sampling (Sampling)             (None, 70)           0           z_mean[0][0]
                                                                 z_log_var[0][0]
==================================================================================================
Total params: 21,138,572
Trainable params: 21,138,572
Non-trainable params: 0
__________________________________________________________________________________________________
```

Εικόνα 6.29: Περίληψη Κωδικοποιητή Variational Autoencoder

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
decoder_input (InputLayer)   [(None, 70)]              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              72704
_________________________________________________________________
dense_2 (Dense)              (None, 20502)             21014550
=================================================================
Total params: 21,087,254
Trainable params: 21,087,254
Non-trainable params: 0
_________________________________________________________________
```

Εικόνα 6.30: Περίληψη Αποκωδικοποιητή Variational Autoencoder

6.4.4 Πρώτο δίκτυο ταξινόμησης – Deep Neural Network

Για αυτό το δίκτυο θα αναφερθούν τα αποτελέσματα και οι καλύτεροι υπερπαραμέτροι που επιλέχθηκαν από την Μπεϋζιανή βελτιστοποίηση για κάθε τύπο Autoencoder.

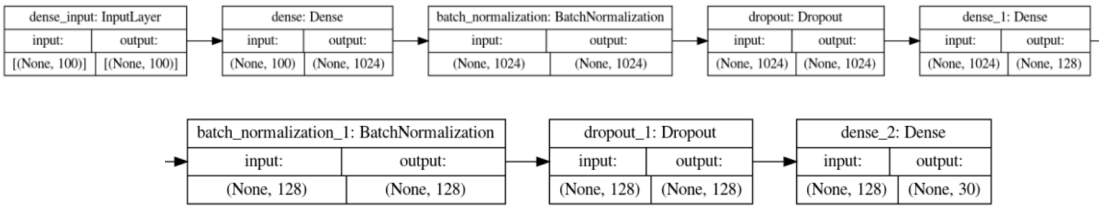
6.4.4.1 Simple Autoencoder data

Οι καλύτεροι υπερπαραμέτροι για το DNN με είσοδο τα δεδομένα που κωδικοποίησε το πρώτο δίκτυο Autoencoder, ήταν δύο κρυφά επίπεδα (Hidden Layers). Το πρώτο επίπεδο αποτελούταν από 1024 νευρώνες (Units), το δεύτερο (Units0) από 128 νευρώνες. Το ποσοστό εγκατάλειψης σε κάθε επίπεδο εγκατάλειψης (Dropout Layer) μετά από ένα πυκνό επίπεδο (Dense Layer) ήταν και για το πρώτο και για το δεύτερο επίπεδο εγκατάλειψης (Dropout) 0%. Η συνάρτηση ενεργοποίησης για τα όλα τα πυκνά επίπεδα (Dense Layers) πλην του τελευταίου, είχε τεθεί στην ELU με αρχικοποίηση των βαρών με την συνάρτηση Glorot Uniform. Μέθοδος γενίκευσης χρησιμοποιήθηκε η L2 γενίκευση και ποσοστό εκμάθησης ίσο με 0.0001.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
layers: 1
activations: elu
kernel_initializer: glorot_uniform
kernel_regularizers: l2
learning_rate: 0.0001
units: 1024
dropout: 0.0
units0: 128
dropout0: 0.0
```

Εικόνα 6.31: Εικόνα καλύτερων υπερπαραμέτρων DNN με δεδομένα από τον Autoencoder.

Παρακάτω στην πρώτη φωτογραφία παρουσιάζεται σχηματικά το μοντέλο και στην επόμενη με περισσότερες λεπτομέρειες.



Εικόνα 6.32: Αρχιτεκτονική δικτύου DNN με δεδομένα από τον Autoencoder.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              103424
_________________________________________________________________
batch_normalization (BatchNo (None, 1024)              4096
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               131200
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 30)                3870
=================================================================
Total params: 243,102
Trainable params: 240,798
Non-trainable params: 2,304
_________________________________________________________________
```

Εικόνα 6.33: Περίληψη δικτύου DNN με δεδομένα από τον Autoencoder.

Το συνολικό ποσοστό των παραμέτρων ήταν 243.102.

Το αποτέλεσμα του δικτύου ταξινόμησης με τα δεδομένα εκπαίδευσης και δοκιμής που είχαν κωδικοποιηθεί και δόθηκαν σαν είσοδος ήταν:

Για τα δεδομένα δοκιμής το ποσοστό ακρίβειας ήταν ίσο με 96%

```
print(accuracy_score(y_test,y_pred_auto))  # accuracy = correct/total
print(f1_score(y_test,y_pred_auto,average="micro"))
```

```
0.9605067064083458
0.9605067064083458
```

Εικόνα 6.33: Accuracy Autoencoder-DNN στα δεδομένα δοκιμής.

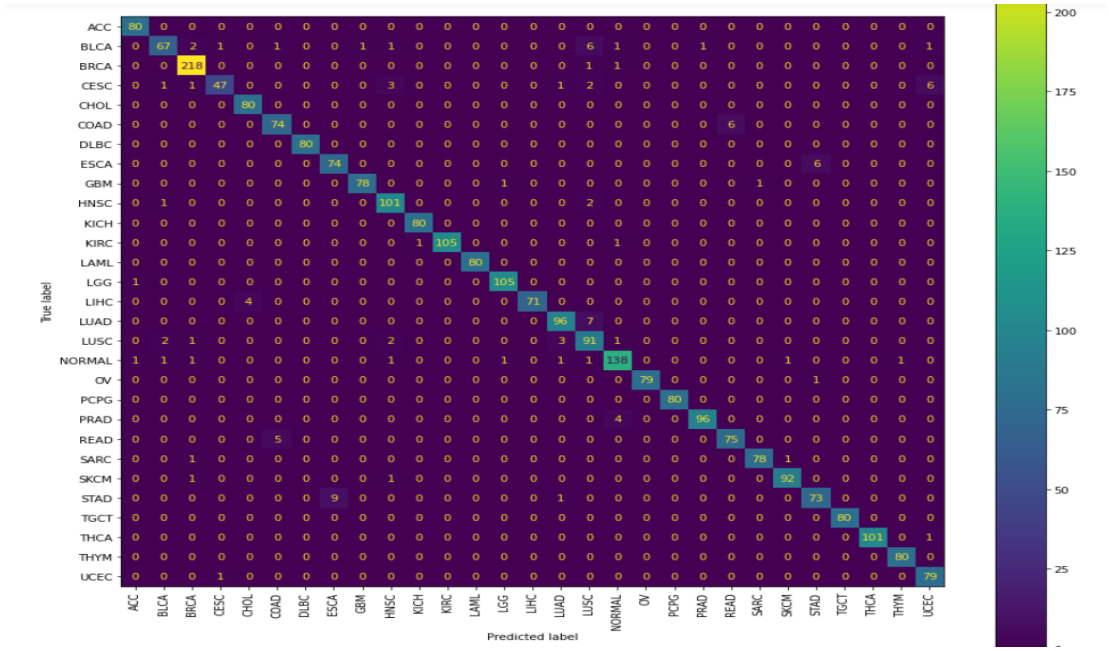
Ενώ για τα δεδομένα εκπαίδευσης το ποσοστό ακρίβειας ήταν ίσο με 98.2%

```
y_pred_train_auto = model_auto.predict(X_train_auto)
y_pred_train_auto = np.argmax(y_pred_train_auto,axis=1)
print(accuracy_score(y_train,y_pred_train_auto))
print(f1_score(y_train,y_pred_train_auto,average="micro"))
```

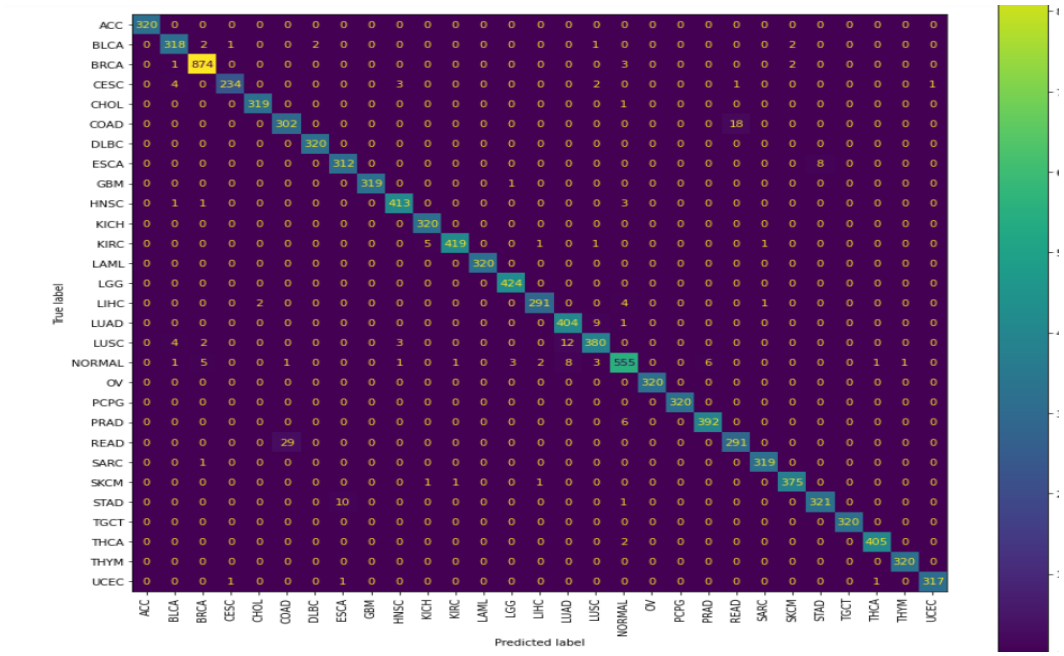
```
0.9821162444113264
0.9821162444113264
```

Εικόνα 6.34: Accuracy Autoencoder-DNN στα δεδομένα εκπαίδευσης.

Οι πίνακες σύγχυσης για τα δύο σύνολα δεδομένα παρουσιάζονται παρακάτω:



Εικόνα 6.35: Πίνακας «σύγχυσης» Autoencoder-DNN στα δεδομένα δοκιμής.



Εικόνα 6.36: Πίνακας «σύγχυσης» Autoencoder-DNN στα δεδομένα εκπαίδευσης.

6.4.4.2 Denoising Autoencoder data

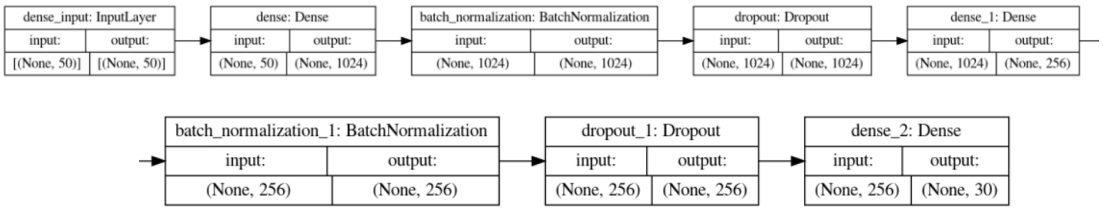
Οι καλύτεροι υπερπαραμέτροι για το δίκτυο αυτό που είχαν σαν είσοδο τα δεδομένα που προήλθαν από το Denoising Autoencoder ήταν δύο κρυφά επίπεδα, με το πρώτο να αποτελείται από 1024 νευρώνες (Units) και το δεύτερο από 256 νευρώνες. Το ποσοστό εγκατάλειψης μετά το πρώτο πυκνό επίπεδο ήταν 0% και μετά το δεύτερο ήταν 0.5. Η συνάρτηση

ενεργοποίησης για όλα τα επίπεδα που είχε επιλεχτεί ήταν η ELU με την μέθοδο αρχικοποίησης των βαρών να είναι η Glorot Uniform και η μέθοδος γενίκευσης να έχει τεθεί στην L2 γενίκευση. Τέλος, το ποσοστό εκμάθησης είχε επιλεχθεί στο 0.0001.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
units: 1024
activations: elu
kernel_initializer: glorot_uniform
kernel_regularizers: l2
dropout: 0.0
layers: 1
units0: 256
dropout0: 0.5
learning_rate: 0.0001
```

Εικόνα 6.37: Εικόνα καλύτερων υπερπαραμέτρων DNN με δεδομένα από τον Denoising Autoencoder.

Παρακάτω στην πρώτη φωτογραφία παρουσιάζεται σχηματικά το μοντέλο και στην επόμενη με περισσότερες λεπτομέρειες.



Εικόνα 6.38: Αρχιτεκτονική δικτύου DNN με δεδομένα από τον Denoising Autoencoder.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              52224
_________________________________________________________________
batch_normalization (BatchNo (None, 1024)              4096
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 30)                7710
=================================================================
Total params: 327,454
Trainable params: 324,894
Non-trainable params: 2,560
_________________________________________________________________
```

Εικόνα 6.39: Περίληψη δικτύου DNN με δεδομένα από τον Denoising Autoencoder.

Το συνολικό ποσοστό των παραμέτρων ήταν 327.454.

Το αποτέλεσμα του δικτύου ταξινόμησης με τα δεδομένα εκπαίδευσης και δοκιμής που είχαν κωδικοποιηθεί και δόθηκαν σαν είσοδος ήταν:

Για τα δεδομένα δοκιμής το ποσοστό ακρίβειας ήταν ίσο με 94.5%

```
print(accuracy_score(y_test,y_pred_deno))  # accuracy = correct/total
print(f1_score(y_test,y_pred_deno,average="micro"))
```

```
0.9456035767511177
0.9456035767511178
```

Εικόνα 6.40: Accuracy Denoising Autoencoder-DNN στα δεδομένα δοκιμής.

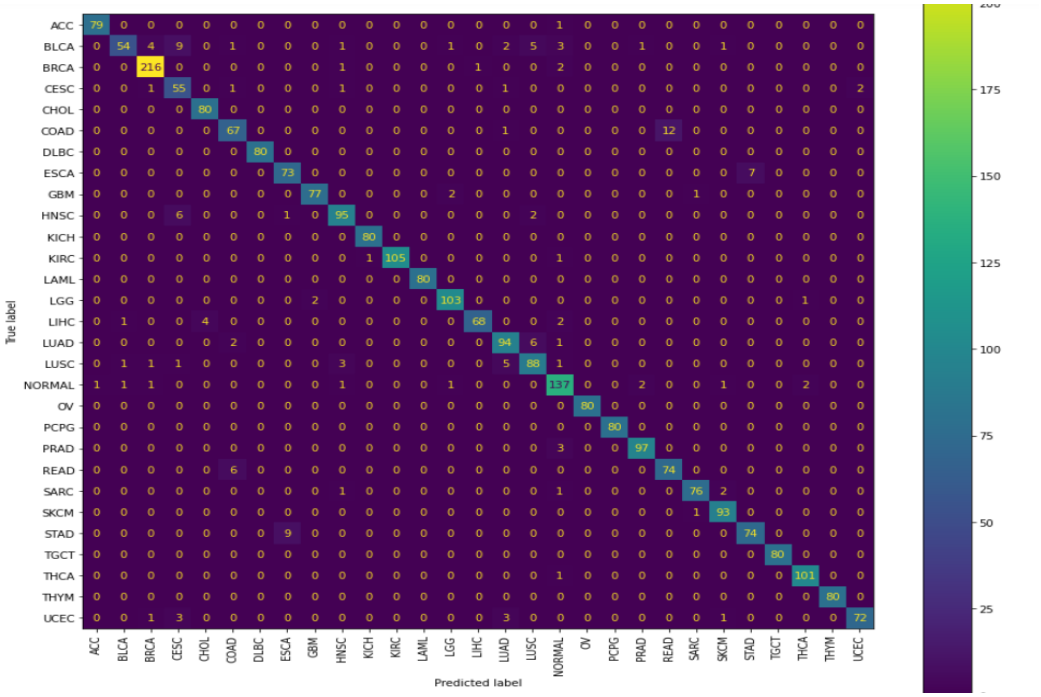
Ενώ για τα δεδομένα εκπαίδευσης το ποσοστό ακρίβειας ήταν ίσο με 98.7%

```
y_pred_train_deno = model_deno.predict(X_train_deno)
y_pred_train_deno = np.argmax(y_pred_train_deno,axis=1)
print(accuracy_score(y_train,y_pred_train_deno))  # accuracy = correct/total
print(f1_score(y_train,y_pred_train_deno,average="micro"))
```

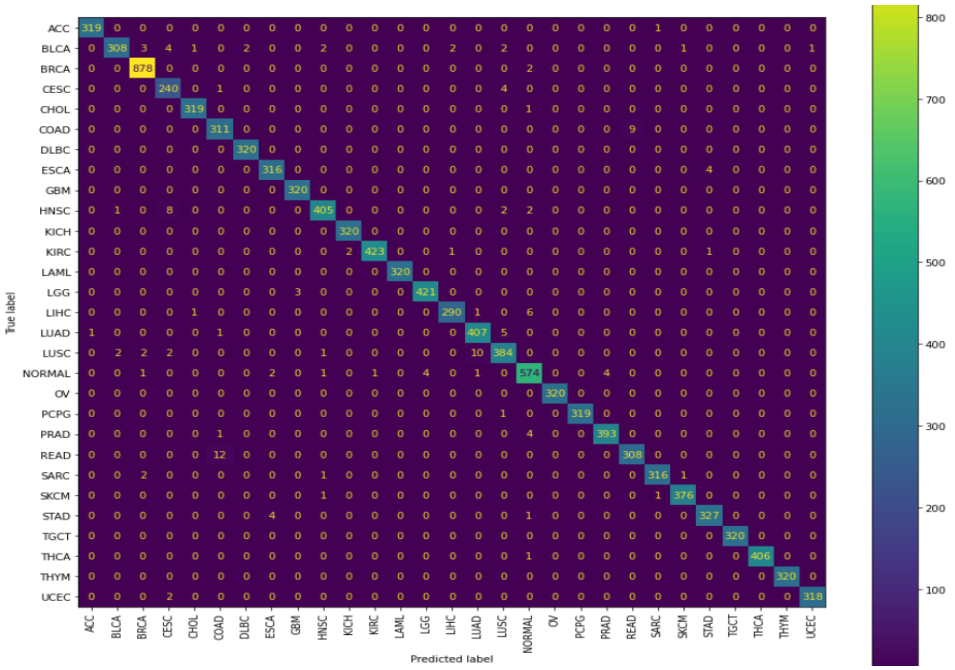
```
0.9871460506706409
0.9871460506706409
```

Εικόνα 6.41: Accuracy Denoising Autoencoder-DNN στα δεδομένα εκπαίδευσης.

Οι πίνακες σύγχυσης για τα δύο σύνολα δεδομένα παρουσιάζονται παρακάτω:



Εικόνα 6.42: Πίνακας «σύγχυσης» Denoising Autoencoder-DNN στα δεδομένα δοκιμής.



Εικόνα 6.43: Πίνακας «σύγχυσης» Denoising Autoencoder-DNN στα δεδομένα εκπαίδευσης.

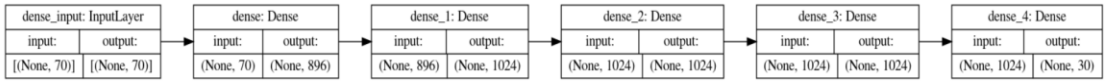
6.4.4.3 Variational Autoencoder data

Για αυτό δίκτυο, τα καλύτερα αποτελέσματα προέκυψαν χωρίς την χρήση των επιπέδων ομαλοποίησης παρτίδας και των επιπέδων εγκατάλειψης. Οι καλύτεροι υπερπαραμέτροι ήταν τέσσερα κρυφά επίπεδα με το πρώτο να αποτελείται από 896 νευρώνες (Units) και τα υπόλοιπα τρία επίπεδα από 1024 νευρώνες. Η συνάρτηση ενεργοποίησης για όλα τα επίπεδα που είχε επιλεχτεί ήταν η γραμμική (Linear) με την μέθοδο αρχικοποίησης των βαρών να είναι η He Normal και η μέθοδος γενίκευσης να έχει τεθεί στην L2 γενίκευση. Τέλος, το ποσοστό εκμάθησης είχε επιλεχθεί στο 0.0001.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
layers: 3
activations: linear
kernel_initializer: he_normal
kernel_regularizers: l2
learning_rate: 0.0001
units: 896
units0: 1024
units1: 1024
units2: 1024
```

Εικόνα 6.44: Εικόνα καλύτερων υπερπαραμέτρων DNN με δεδομένα από τον Variational Autoencoder.

Παρακάτω στην πρώτη φωτογραφία παρουσιάζεται σχηματικά το μοντέλο και στην επόμενη με περισσότερες λεπτομέρειες.



Εικόνα 6.45: Αρχιτεκτονική δικτύου DNN με δεδομένα από τον Variational Autoencoder.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 896)               63616
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              918528
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_4 (Dense)              (None, 30)                30750
=================================================================
Total params: 3,112,094
Trainable params: 3,112,094
Non-trainable params: 0
_________________________________________________________________
```

Εικόνα 6.46: Περίληψη δικτύου DNN με δεδομένα από τον Variational Autoencoder.

Το συνολικό ποσοστό των παραμέτρων ήταν 3,112.094 .

Το αποτέλεσμα του δικτύου ταξινόμησης με τα δεδομένα εκπαίδευσης και δοκιμής που είχαν κωδικοποιηθεί και δόθηκαν σαν είσοδος ήταν:

Για τα δεδομένα δοκιμής το ποσοστό ακρίβειας ήταν ίσο με 95%

```
print(accuracy_score(y_test,y_pred_vae))  # accuracy = correct/total
print(f1_score(y_test,y_pred_vae,average="micro"))
```

```
0.9500745156482862
0.9500745156482862
```

Εικόνα 6.47: Accuracy Variational Autoencoder-DNN στα δεδομένα δοκιμής.

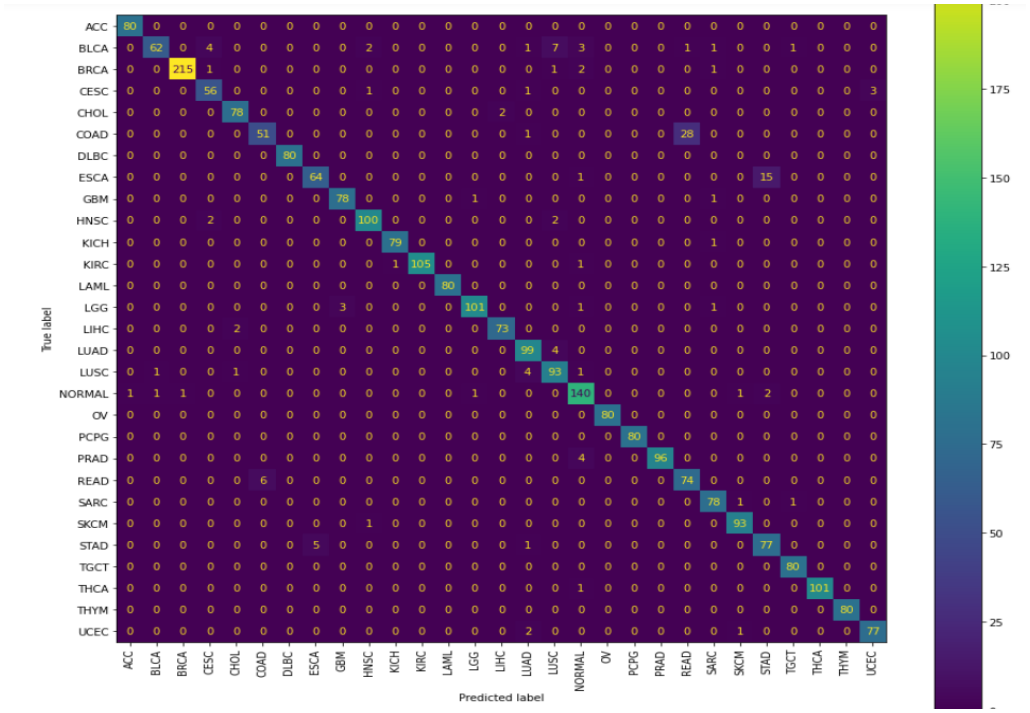
Ενώ για τα δεδομένα εκπαίδευσης το ποσοστό ακρίβειας ήταν ίσο με 96%

```
y_pred_train_vae = model_vae.predict(X_train_vae[2])
y_pred_train_vae = np.argmax(y_pred_train_vae,axis=1)
print(accuracy_score(y_train,y_pred_train_vae))  # accuracy = correct/total
print(f1_score(y_train,y_pred_train_vae,average="micro"))
```

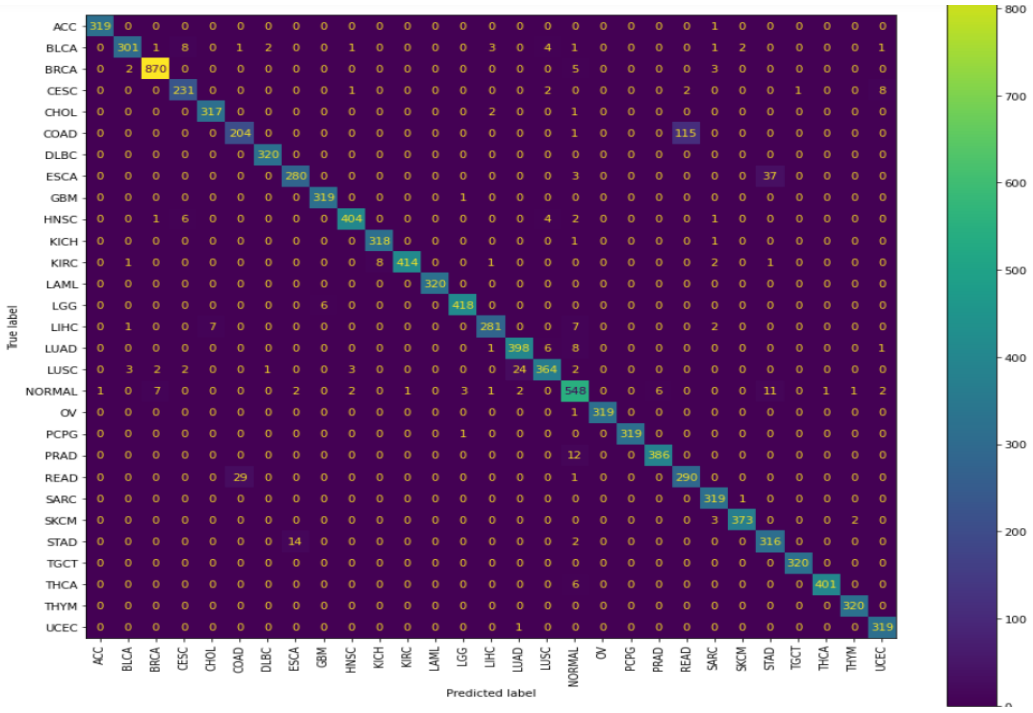
```
0.960134128166915
0.960134128166915
```

Εικόνα 6.48: Accuracy Variational Autoencoder-DNN στα δεδομένα εκπαίδευσης.

Οι πίακες σύγχυσης για τα δύο σύνολα δεδομένα παρουσιάζονται παρακάτω:



Εικόνα 6.49: Πίνακας «σύγχυσης» Variational Autoencoder-DNN στα δεδομένα δοκιμής.



Εικόνα 6.50: Πίνακας «σύγχυσης» Variational Autoencoder-DNN στα δεδομένα εκπαίδευσης.

6.4.5 Δεύτερο δίκτυο ταξινόμησης – Deep Cross Model

Για αυτό το δίκτυο θα αναφερθούν τα αποτελέσματα και οι καλύτεροι υπερπαραμέτροι που επιλέχθηκαν από την Μπεϋζιανή βελτιστοποίηση για κάθε τύπο Autoencoder.

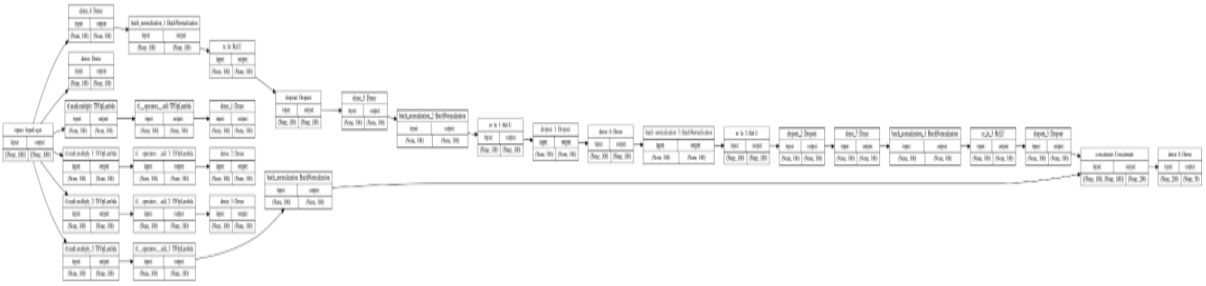
6.4.5.1 Simple Autoencoder data

Οι καλύτεροι υπερπαραμέτροι για το μοντέλο Deep Cross με είσοδο τα δεδομένα που κωδικοποίησε το πρώτο δίκτυο Autoencoder, ήταν τέσσερα κρυφά επίπεδα (Hidden Layers) οριζόντια και κάθετα τα οποία συνενώθηκαν. Ο αριθμός των νευρώνων (Units) στα κρυφά επίπεδα ήταν 64. Το ποσοστό εγκατάλειψης σε κάθε επίπεδο εγκατάλειψης (Dropout Layer) μετά από ένα πυκνό επίπεδο (Dense Layer) ήταν 0.3 ή 30% των νευρώνων να είναι ανενεργοί. Η συνάρτηση ενεργοποίησης για τα όλα τα πυκνά επίπεδα (Dense Layers) πλην του τελευταίου είχε τεθεί στην ELU με αρχικοποίηση των βαρών με την συνάρτηση He Uniform. Μέθοδος γενίκευσης χρησιμοποιήθηκε η L2 γενίκευση και ποσοστό εκμάθησης ίσο με 0.01.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
layers: 4
hidden_layers: 64
dropout: 0.3
learning_rate: 0.01
activations: elu
kernel_initializer: he_normal
kernel_regularizers: l2
```

Εικόνα 6.51: Εικόνα καλύτερων υπερπαραμέτρων Deep Cross Model με δεδομένα από τον Autoencoder.

Παρακάτω στην πρώτη φωτογραφία παρουσιάζεται σχηματικά το μοντέλο και στην επόμενη με περισσότερες λεπτομέρειες.



Εικόνα 6.52: Αρχιτεκτονική δικτύου Deep Cross με δεδομένα από τον Autoencoder.

Το αποτέλεσμα του δικτύου ταξινόμησης με τα δεδομένα εκπαίδευσης και δοκιμής που είχαν κωδικοποιηθεί και δόθηκαν σαν είσοδος ήταν:

Για τα δεδομένα δοκιμής το ποσοστό ακρίβειας ήταν ίσο με 94%

```
print(accuracy_score(y_test,y_pred_auto_deep_cross))
print(f1_score(y_test,y_pred_auto_deep_cross,average="micro"))
```

```
0.9407600596125186
0.9407600596125186
```

Εικόνα 6.53: Accuracy Autoencoder-Deep Cross Model στα δεδομένα δοκιμής.

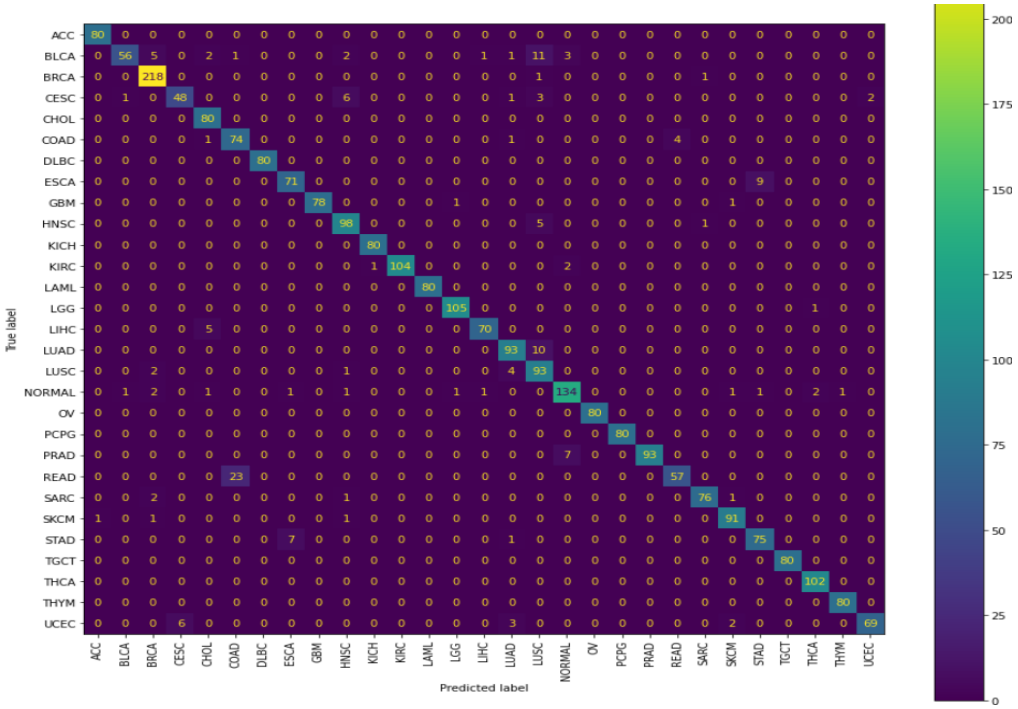
Ενώ για τα δεδομένα εκπαίδευσης το ποσοστό ακρίβειας ήταν ίσο με 95.8%

```
print(accuracy_score(y_train,y_pred_train_auto_deep_cross))
print(f1_score(y_train,y_pred_train_auto_deep_cross,average="micro"))
```

```
0.9589232488822653
0.9589232488822653
```

Εικόνα 6.54: Accuracy Autoencoder-Deep Cross Model στα δεδομένα εκπαίδευσης.

Οι πίνακες σύγχυσης για τα δύο σύνολα δεδομένα παρουσιάζονται παρακάτω:



Εικόνα 6.55: Πίνακας «σύγχυσης» Autoencoder-Deep Cross Model στα δεδομένα δοκιμής.

6.4.5.2 Denoising Autoencoder data

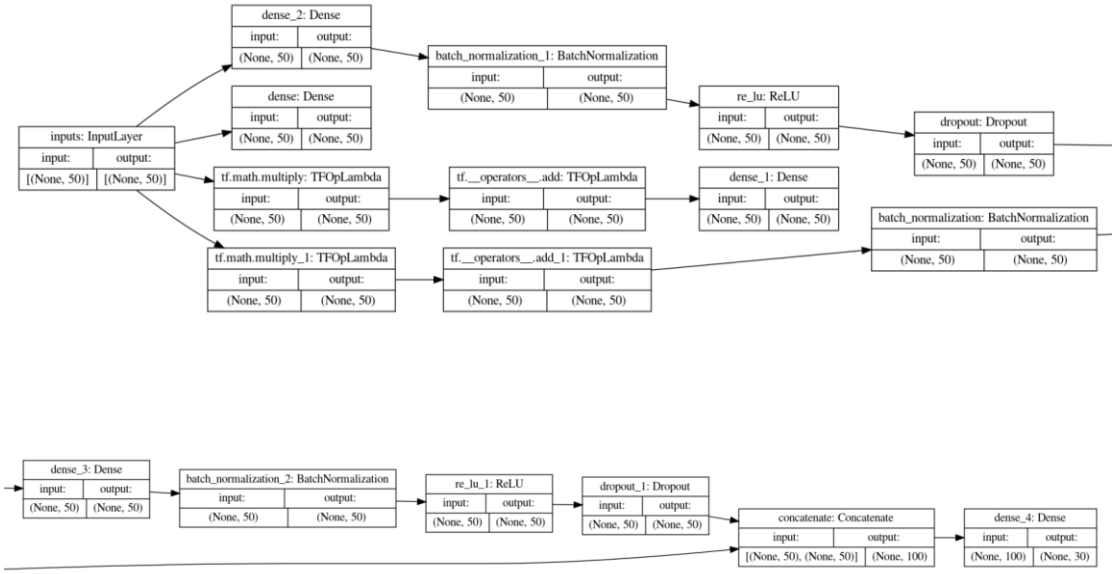
Οι καλύτεροι υπερπαραμέτροι για το δίκτυο αυτό που είχαν σαν είσοδο τα δεδομένα που προήλθαν από το Denoising Autoencoder ήταν δύο κρυφά επίπεδα οριζόντια και κάθετα με αριθμό νευρώνων το κάθε επίπεδο να είναι 512. Το ποσοστό εγκατάλειψης σε κάθε επίπεδο εγκατάλειψης ήταν 0.0 ή 0%. Η συνάρτηση ενεργοποίησης για όλα τα επίπεδα που είχε

επιλεχτεί ήταν η ELU με την μέθοδο αρχικοποίησης των βαρών να είναι η He Normal και η μέθοδος γενίκευσης να έχει τεθεί στην L2 γενίκευση. Τέλος, το ποσοστό εκμάθησης είχε επιλεχθεί στο 0.1.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
layers: 2
hidden_layers: 512
dropout: 0.0
learning_rate: 0.1
activations: elu
kernel_initializer: he_normal
kernel_regularizers: l2
```

Εικόνα 6.56: Εικόνα καλύτερων υπερπαραμέτρων Deep Cross Model με δεδομένα από τον Denoising Autoencoder.

Παρακάτω στην πρώτη φωτογραφία παρουσιάζεται σχηματικά το μοντέλο και στην επόμενη με περισσότερες λεπτομέρειες.



Εικόνα 6.57: Αρχιτεκτονική δικτύου Deep Cross με δεδομένα από τον Denoising Autoencoder.

Το αποτέλεσμα του δικτύου ταξινόμησης με τα δεδομένα εκπαίδευσης και δοκιμής που είχαν κωδικοποιηθεί και δόθηκαν σαν είσοδος ήταν:

Για τα δεδομένα δοκιμής το ποσοστό ακρίβειας ήταν ίσο με 81.2%

```
print(accuracy_score(y_test,y_pred_deno_deep_cross))
print(f1_score(y_test,y_pred_deno_deep_cross,average="micro"))
```

```
0.8125931445603577
0.8125931445603577
```

Εικόνα 6.58: Accuracy Denoising Autoencoder-Deep Cross Model στα δεδομένα δοκιμής.

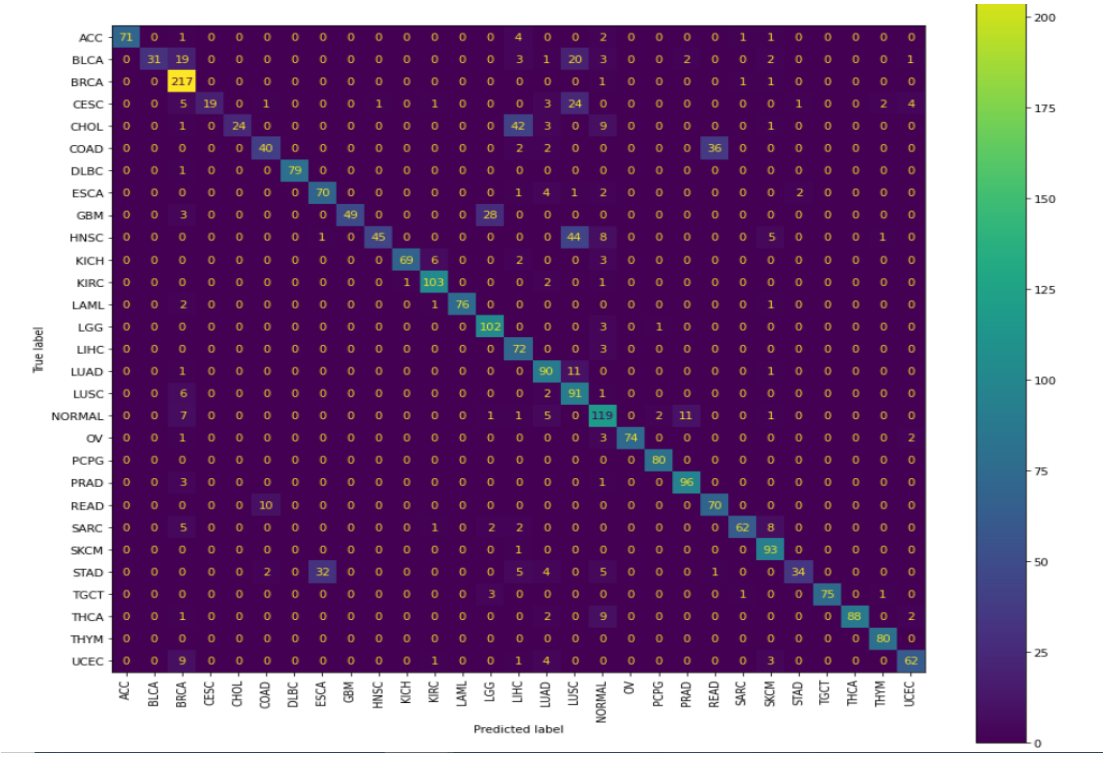
Ενώ για τα δεδομένα εκπαίδευσης το ποσοστό ακρίβειας ήταν ίσο με 83%

```
print(accuracy_score(y_train,y_pred_train_deno_deep_cross))
print(f1_score(y_train,y_pred_train_deno_deep_cross,average="micro"))
```

```
0.8306631892697467
0.8306631892697468
```

Εικόνα 6.59: Accuracy Denoising Autoencoder-Deep Cross Model στα δεδομένα εκπαίδευσης.

Οι πίνακες σύγχυσης για τα δύο σύνολα δεδομένα παρουσιάζονται παρακάτω:



Εικόνα 6.60: Πίνακας «σύγχυσης» Denoising Autoencoder-Deep Cross Model στα δεδομένα δοκιμής.

6.4.5.3 Variational Autoencoder data

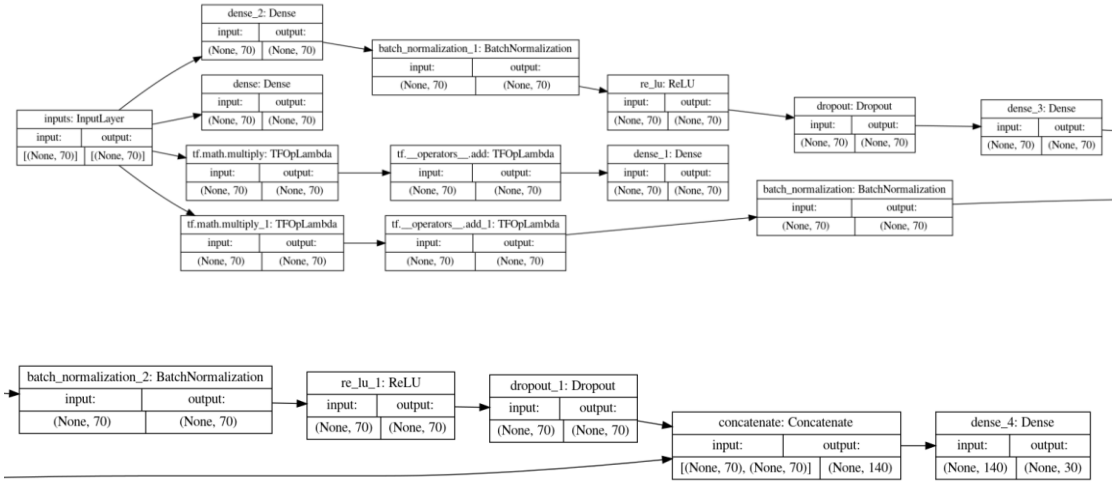
Για αυτό δίκτυο τα καλύτερα αποτελέσματα προέκυψαν χωρίς την χρήση των επιπέδων ομαλοποίησης παρτίδας και των επιπέδων εγκατάλειψης. Οι καλύτεροι υπερπαραμέτροι ήταν δύο κρυφά επίπεδα οριζόντια και κάθετα με 512 νευρώνες (Units) σε κάθε επίπεδο. Το ποσοστό εγκατάλειψης για κάθε επίπεδο εγκατάλειψης είχε επιλεχθεί στο 0.5 ή 50%. Η συνάρτηση ενεργοποίησης για όλα τα επίπεδα που είχε επιλεχτεί ήταν η γραμμική (Linear)

με την μέθοδο αρχικοποίησης των βαρών να είναι η Glorot Uniform και η μέθοδος γενίκευσης να έχει τεθεί στην L1 γενίκευση. Τέλος, το ποσοστό εκμάθησης είχε επιλεχθεί στο 0.01.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
layers: 2
hidden_layers: 512
dropout: 0.5
learning_rate: 0.01
activations: linear
kernel_initializer: glorot_uniform
kernel_regularizers: l1
```

Εικόνα 6.61: Εικόνα καλύτερων υπερπαραμέτρων Deep Cross Model με δεδομένα από τον Variational Autoencoder.

Παρακάτω στην πρώτη φωτογραφία παρουσιάζεται σχηματικά το μοντέλο και στην επόμενη με περισσότερες λεπτομέρειες.



Εικόνα 6.62: Αρχιτεκτονική δικτύου Deep Cross με δεδομένα από τον Variational Autoencoder.

Το αποτέλεσμα του δικτύου ταξινόμησης με τα δεδομένα εκπαίδευσης και δοκιμής που είχαν κωδικοποιηθεί και δόθηκαν σαν είσοδος ήταν:
Για τα δεδομένα δοκιμής το ποσοστό ακρίβειας ήταν ίσο με 95.4%

```
print(accuracy_score(y_test,y_pred_vae_deep_cross))
print(f1_score(y_test,y_pred_vae_deep_cross,average="micro"))
```

```
0.9541728763040238
0.9541728763040238
```

Εικόνα 6.63: Accuracy Variational Autoencoder-Deep Cross Model στα δεδομένα δοκιμής.

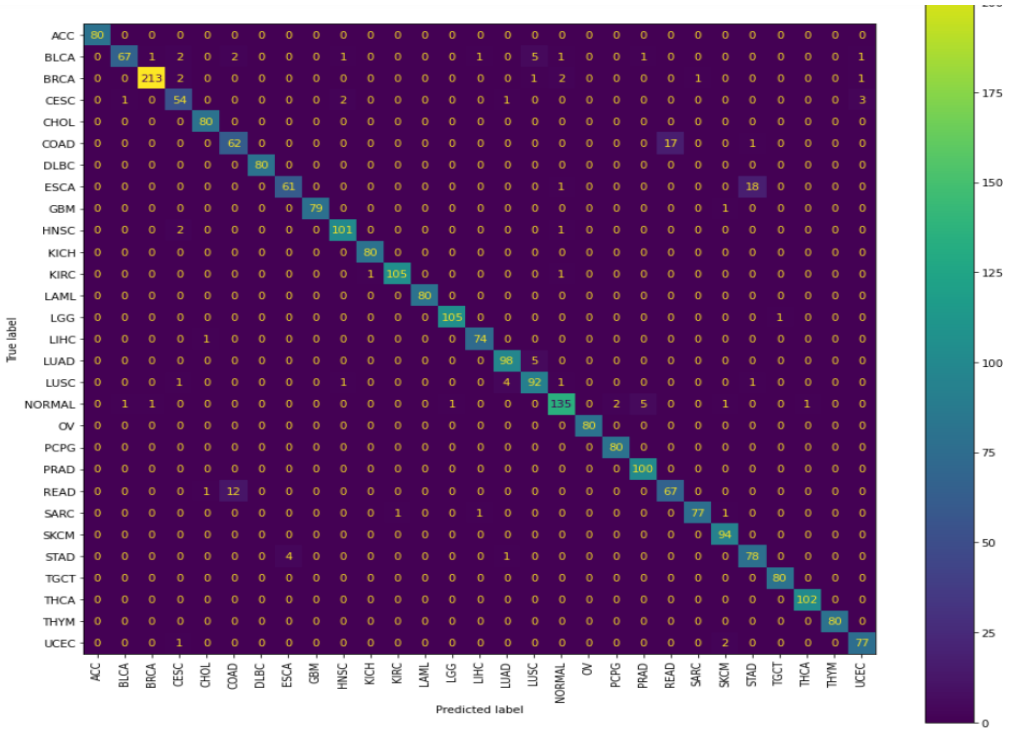
Ενώ για τα δεδομένα εκπαίδευσης το ποσοστό ακρίβειας ήταν ίσο με 97%

```
print(accuracy_score(y_train,y_pred_train_vae_deep_cross))
print(f1_score(y_train,y_pred_train_vae_deep_cross,average="micro"))
```

```
0.9702868852459017
0.9702868852459017
```

Εικόνα 6.64: Accuracy Variational Autoencoder-Deep Cross Model στα δεδομένα εκπαίδευσης.

Οι πίνακες σύγχυσης για τα δύο σύνολα δεδομένα παρουσιάζονται παρακάτω:



Εικόνα 6.65: Πίνακας «σύγχυσης» Variational Autoencoder-Deep Cross Model στα δεδομένα δοκιμής.

7. Σύγκριση αποτελεσμάτων ποιοτικά και ποσοτικά

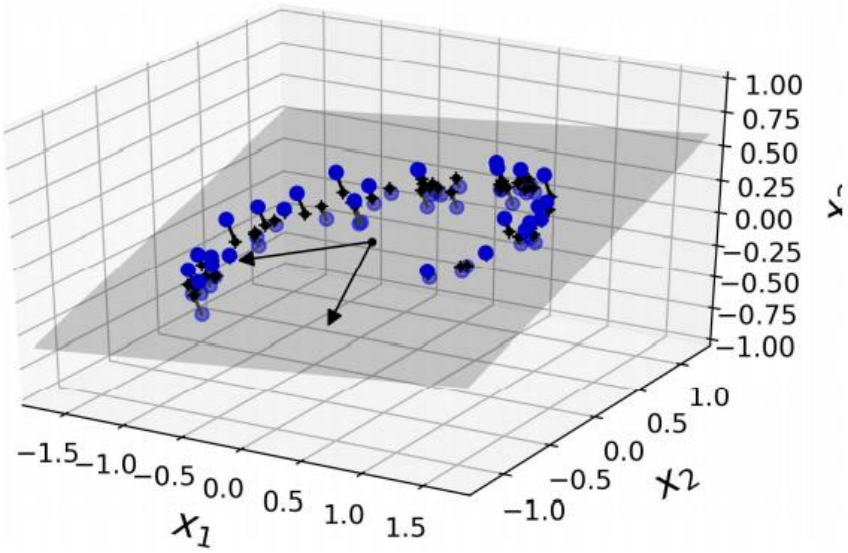
Για να επιτευχθεί η αξιολόγηση της απόδοσης των νευρωνικών δικτύων απαραίτητη προϋπόθεση ήταν η ύπαρξη ενός μέτρου σύγκρισης με διαφορετικές τεχνικές που να μπορούν να χρησιμοποιηθούν για το συγκεκριμένο πρόβλημα. Για αυτόν τον σκοπό έγινε χρήση διαφορετικών αλγορίθμων μηχανικής μάθησης. Ακολουθήθηκε η ίδια διαδικασία με αυτήν των νευρωνικών δικτύων. Δηλαδή επιλέχθηκε ένας αλγόριθμος για την μείωση των χαρακτηριστικών και στην συνέχεια χρησιμοποιήθηκαν διάφοροι αλγόριθμοι μηχανικής μάθησης για την διαδικασία της ταξινόμησης. Στην συνέχεια, όπως και με την ανάλυση των νευρωνικών δικτύων έτσι και εδώ, χρησιμοποιήθηκε η βιβλιοθήκη **Keras Tuner** για την

βελτιστοποίηση των υπερπαραμέτρων με την Μπεϋζιανή μέθοδο βελτιστοποίησης. Τέλος, τα αποτελέσματα συγκρίθηκαν ποιοτικά και ποσοτικά με τα αποτελέσματα που είχε δώσει η ανάλυση των νευρωνικών δικτύων.

7.1 Σύγκριση με άλλες τεχνικές μηχανικής μάθησης

Για την μείωση των χαρακτηριστικών χρησιμοποιήθηκε ο αλγόριθμος **Principal Component Analysis (PCA)** και για την ταξινόμηση χρησιμοποιήθηκαν οι αλγόριθμοι **Support Vector Machine, Random Forests, Naïve Bayes, Stochastic Gradient Descent, Extreme Gradient Boosting (XGB) και Ridge algorithm**. Παρακάτω θα περιγράφουν οι αλγόριθμοι και έπειτα θα συνεχιστεί η παρουσίαση των αποτελεσμάτων.

Ο αλγόριθμος **Principal Component Analysis (Ανάλυση Κύριων Συνιστωσών)** ή **PCA** είναι ο πιο γνωστός αλγόριθμος που χρησιμοποιείται για να μειωθεί ο αριθμός των χαρακτηριστικών. Ο τρόπος λειτουργίας του είναι να προσδιορίζει ένα υπερπλάνο κοντά στα δεδομένα και στην συνέχεια τα δεδομένα προβάλλονται σε αυτό το πλάνο.



Εικόνα 7.1: Εικόνα παραδείγματος υπερπλάνου PCA. Πηγή: Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019 σελίδα 218

Ο αλγόριθμος **PCA** χρησιμοποιείται στα δεδομένα και προσδιορίζει τους άξονες που διατηρούν το μεγαλύτερο μέρος της **διακύμανσης (Variance)** των δεδομένων αυτών. Ένας διαφορετικός τρόπος να δούμε την λειτουργία του αλγορίθμου, είναι η προσπάθεια του να μειώσει το μέσο του τετραγωνισμού της διαφοράς μεταξύ των αρχικών δεδομένων και της προβολής των δεδομένων σε αυτούς του άξονες. Για παράδειγμα για να προβάλουμε τα δεδομένα σε δύο διαστάσεις ο αλγόριθμος, προσδιορίζει τον άξονα με το μεγαλύτερο ποσοστό

διακύμανσης και στην συνέχεια προσδιορίζει έναν δεύτερο άξονα ορθογώνιο με τον πρώτο με το μεγαλύτερο ποσοστό της υπόλοιπης διακύμανσης. Πως δουλεύει όμως; Εφαρμόζεται μια μέθοδος που ονομάζεται **αποσύνθεση μοναδιαίας τιμής** (**Singular Value Decomposition**), η οποία είναι διαδικασία αποσύνθεσης ενός πίνακα $\boldsymbol{X}$ σε τρείς διαφορετικούς πίνακες που πολλαπλασιάζονται $\boldsymbol{U} * \boldsymbol{\Sigma} * \boldsymbol{V^T}$. Το $\boldsymbol{U}$ είναι ένας $m \times m$ πίνακας, το $\boldsymbol{\Sigma}$ **ε**ίναι ένας $m \times n$ διαγώνιος πίνακας και ο $\boldsymbol{V}$ είναι ο πίνακας $n \times n$ που περιέχει όλες τις κύριες συνιστώσες (Principal Components) που θέλουμε[91][92].

Ο αλγόριθμος **t-distributed Stochastic Neighbor Embedding** ή **t-SNE** είναι μια τεχνική μη επιβλεπόμενης μηχανικής μάθησης που κυρίως χρησιμοποιείται για την **οπτικοποίηση** δεδομένων μεγάλων διαστάσεων (**Data Visualization**) και για την **εξερεύνηση των δεδομένων** (**Data Exploration**). Ο αλγόριθμος t-SNE υπολογίζει ένα μέτρο ομοιότητας μεταξύ των ζευγών των σημείων στο χώρο των μεγάλων διαστάσεων με τον χώρο των μικρό διαστάσεων. Στόχος, είναι να ελαχιστοποιήσει την **απόκλιση Kullback-Leibler** μεταξύ των παραγομένων δεδομένων μικρότερης διάστασης από την αρχική με τα δεδομένα μεγαλύτερης διάστασης[93]. Εφόσον ο αλγόριθμος για την οπτικοποίηση μειώνει τον αριθμό των διαστάσεων ποια είναι η διαφορά του με τον αλγόριθμο **PCA;** Ο **PCA** μειώνει τα χαρακτηριστικά μέσω μιας γραμμικής διαδικασίας με στόχο την μεγιστοποίηση της διακύμανσης. Αυτή η τεχνική μπορεί να οδηγήσει σε μια κακή οπτικοποίηση των δεδομένων. Από την άλλη, ο αλγόριθμος **t-SNE** διατηρεί τοπικές ομοιότητες ή μικρά ζεύγη αποστάσεων με αποτέλεσμα να οπτικοποιεί γραμμικές και μη σχέσεις στα δεδομένα μας[94].

Ο αλγόριθμος **Support Vector Machines (Διανυσματικές Μηχανές Υποστήριξης)** είναι ένας από τους πιο σημαντικούς αλγόριθμους επιβλεπόμενης μηχανικής μάθησης και χρησιμοποιείται για την **ανίχνευση ακραίων τιμών** (**Outlier Detection**),την επίλυση προβλημάτων **παλινδρόμησης** αλλά και **ταξινόμησης** σε **γραμμικά** δεδομένα και **μη γραμμικά**[95]. Παρέχεται στην βιβλιοθήκη **Scikit Learn**. Πίσω από την υλοποίηση του

[91] Jason Brownlee, "How to Calculate the SVD from Scratch with Python", Machine Learning Mastery, 26 February 2018, https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/ (Ανακτήθηκε 21/3/2021)

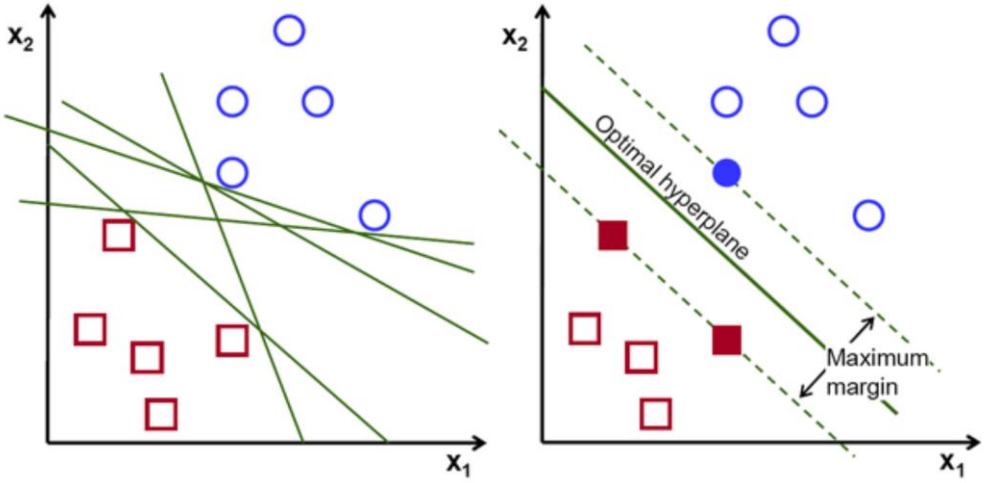
[92] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019

[93] Scikit Learn, "t-SNE ", https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html (Ανακτήθηκε 21/3/2021)

[94] Andre Violante, "An Introduction to t-SNE with Python Example", Towards Data Science,29 August 2018, https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1 (Ανακτήθηκε 21/3/2021)

[95] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019

αλγόριθμου βρίσκονται περίπλοκες μαθηματικές πράξεις που δυσκολεύουν την κατανόηση του[96]. Ο στόχος των διανυσματικών μηχανών υποστήριξης είναι να ανακαλύψουν ένα **υπερπλάνο** (**Hyperplane**) σε χώρο N διαστάσεων για την ταξινόμηση των δεδομένων. Επειδή μεταξύ των κλάσεων μπορούν να υπάρχουν πολλά πιθανά υπερπλάνα, ο αλγόριθμος ψάχνει να βρει το μοναδικό που έχει την μεγαλύτερη απόσταση μεταξύ των σημείων της κάθε κλάσης. Αυτό παρέχει στον αλγόριθμο μεγαλύτερο βαθμό αυτοπεποίθησης για την ταξινόμηση μελλοντικών δεδομένων. Όπως παρατηρείται και στην φωτογραφία παρακάτω, το υπερπλάνο ορίζεται σαν ένα **αποφασιστικό όριο** (**Decision Boundary**) μεταξύ των κλάσεων, όπου δεδομένα με συγκεκριμένα μοτίβα τιμών τοποθετούνται στην ανάλογη πλευρά του υπερπλάνου[97].

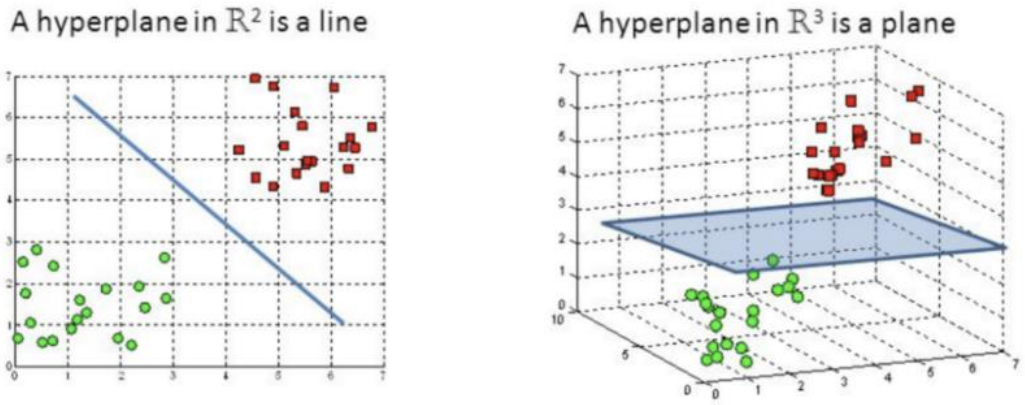


Εικόνα 7.2: Παράδειγμα υπερπλάνου SVM. Πηγή: Rohith Gandhi, "Support Vector Machine – Introduction to Machine Learning Algorithms", Towards Data Science, 7 July 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Ανακτήθηκε 21/3/2021)

Οι διαστάσεις του υπερπλάνου βασίζονται στον αριθμό των χαρακτηριστικών που υπάρχουν. Για παράδειγμα, αν ο αριθμός των χαρακτηριστικών είναι δύο τότε το υπερπλάνο θα είναι μια ευθεία γραμμή, σε περίπτωση που ο αριθμός των δεδομένων είναι τρία, τότε θα το υπερπλάνο θα σχηματίζεται σαν ένα σχήμα ή πλάνο δύο διαστάσεων. Όσο μεγαλώνει ο αριθμός των

[96] Shan Suthaharan, "Support Vector Machine", Springer Link, https://link.springer.com/chapter/10.1007/978-1-4899-7641-3_9 (Ανακτήθηκε 21/3/2021)
[97] Rohith Gandhi, "Support Vector Machine – Introduction to Machine Learning Algorithms", Towards Data Science, 7 June 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Ανακτήθηκε 21/3/2021)

διαστάσεων το πλάνο αποκτά μεγαλύτερη πολυπλοκότητα για αυτό είναι δύσκολο να φανταστούμε ένα πλάνο σε δεδομένα περισσοτέρων από τριών διαστάσεων.



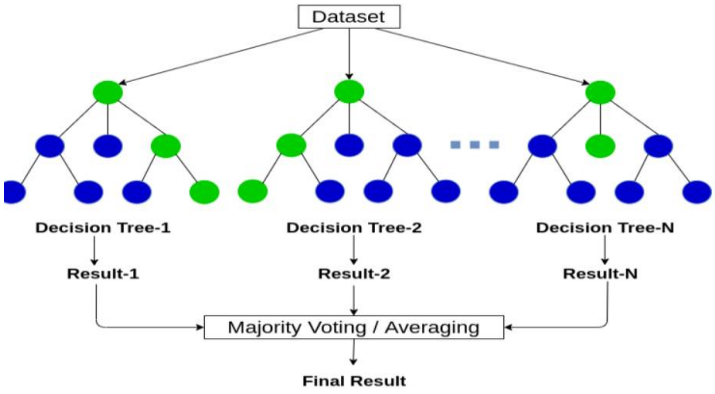
Εικόνα 7.3: Παράδειγμα υπερπλάνου SVM σε δύο και τρείς διαστάσεις. Πηγή: Rohith Gandhi, "Support Vector Machine – Introduction to Machine Learning Algorithms", Towards Data Science, 7 June 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Ανακτήθηκε 21/3/2021)

Πλεονεκτήματα του αλγορίθμου είναι η αποτελεσματικότητα του σε δεδομένα μεγάλων διαστάσεων, αποτελεσματικότητα σε δεδομένα που ο αριθμός των διαστάσεων είναι μεγαλύτερος από τον αριθμό δειγμάτων, χρησιμοποιεί υποσύνολα των δεδομένων εκπαίδευσης για την συνάρτηση απόφασης για αυτόν τον λόγο είναι αποτελεσματικός στην διαχείριση μνήμης. Μπορούν να χρησιμοποιηθούν διάφοροι **πυρήνες** (**Kernels**) για την εκμάθηση μη γραμμικών σχέσεων. Μειονεκτήματα είναι ότι αν ο αριθμός των χαρακτηριστικών είναι πολύ μεγάλος σε σχέση με τον αριθμό των δειγμάτων τότε μπορεί να υπάρξει το πρόβλημα της **υπερτοποθέτησης** (**Overfitting**) και για την αποφυγή αυτής πρέπει να επιλεχθούν διάφοροι όροι γενίκευσης και διαφορετικοί πυρήνες[98].

Ο αλγόριθμος **Random Forests** (**Τυχαία δάση**) είναι μια ομαδοποίηση πολλών μοντέλων ενός άλλου αλγόριθμού μηχανικής που ονομάζεται **δέντρα απόφασης** (**Decision Trees**) και αποτελεί έναν από τους πιο χρήσιμους αλγορίθμους στις μέρες μας. Ο αλγόριθμος παρέχει μια τυχαία υλοποίηση στην παραγωγή των δέντρων, συγκεντρώνεται στην εύρεση του καλύτερο χαρακτηριστικού ανάμεσα σε ένα τυχαίο υποσύνολο των χαρακτηριστικών για την παραγωγή

[98] Scikit Learn, "Support Vector Machine", https://scikit-learn.org/stable/modules/svm.html (Ανακτήθηκε 21/3/2021)

ενός δέντρου. Αυτή η διαδικασία έχει ως αποτέλεσμα την απόδοση ενός καλύτερου μοντέλου, επειδή ανταλλάζεται **μεγαλύτερη μεροληψία** (**Higher Bias**) για **μικρότερη διακύμανση** (**Lower Variance**)[99]. Ένα ακόμη πλεονέκτημα του αλγορίθμου, είναι ότι ωφελείται αν υπάρχει η διαθεσιμότητα διαφορετικών **κεντρικών μονάδων επεξεργασίας** (**CPUS**), για την αξιοποίηση τους με αποτέλεσμα να μειώνει τον χρόνο εκπαίδευσης[100].



Εικόνα 7.4: Παράδειγμα Random Forests. Πηγή: Abhishek Sharma, "Decision Tree vs. Random Forest – Which Algorithm Should you Use", Analytics Vidhya, 12 May 2020, https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/ (Ανακτήθηκε 21/3/2021)

Ο **Naïve Bayes** ή **Αφελής Μπέυζ** είναι ένας απλός και πολύ χρήσιμος αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται σε προβλήματα ταξινόμησης. Βασίζεται στο θεώρημα του **Μπέυζ**, με το θεώρημα να εκφράζει την εύρεση της πιθανότητας ενός γεγονότος $\boldsymbol{A}$ να συμβεί με προϋπόθεση να έχει συμβεί ένας γεγονός $\boldsymbol{B}$. Το $\boldsymbol{B}$ ορίζεται σαν το «στοιχείο» και το $\boldsymbol{A}$ σαν την υπόθεση.

$$\boldsymbol{P(A|B) = \frac{P(B \mid A) * P(A)}{P(B)}}$$

Ο αλγόριθμος στηρίζει την υπόθεση ότι υπάρχει μια ανεξαρτησία μεταξύ των χαρακτηριστικών. Δηλαδή, ότι ένα χαρακτηριστικό δεν επηρεάζει την τιμή ενός άλλου[101]. Πλεονεκτήματα αυτού του αλγορίθμου είναι η ευκολία στην υλοποίηση του και η ταχύτητα

[99] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, June 2019
[100] Andriy Burkov, "The hundred page – machine learning", January 2019
[101] Rohith Gandhi, "Naïve Bayes Classifier", Towards Data Science, 5 May 2018, https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (Ανακτήθηκε 21/3/2021)

του στην εκπαίδευση των δεδομένων καθώς και στην πρόβλεψη κλάσεων. Ακόμα, έχει πολύ καλή απόδοση σε προβλήματα που έχουν περισσότερες από δύο κλάσεις να προβλέψουν. Όταν ισχύει η υπόθεση ότι τα δεδομένα είναι ανεξάρτητα μεταξύ τους ο αλγόριθμος έχει καλύτερη απόδοση από άλλους αλγορίθμους όπως της λογιστικής παλινδρόμησης και χρειάζεται λιγότερα δεδομένα για να εκπαιδευτεί. Μειονεκτήματα του αλγορίθμου είναι ότι στην πραγματικότητα συχνά τα δεδομένα δεν είναι ανεξάρτητα μεταξύ τους. Μάλιστα, σε περίπτωση που υπάρχει μια κατηγορηματική μεταβλητή στα δεδομένα δοκιμής που δεν υπήρχε στα δεδομένα εκπαίδευσης, τότε το μοντέλο θα εκχωρήσει μηδενική πιθανότητα και δεν θα μπορεί να εκτελέσει κάποια πρόβλεψη[102]. Υπάρχουν διαφορετικοί τύποι του συγκεκριμένου αλγορίθμου, μερικοί από αυτούς είναι ο Bernoulli Naïve Bayes και ο Gaussian Naïve Bayes. Ο Bernoulli αλγόριθμος είναι κατάλληλος για διακριτά δεδομένα και έχει σχεδιαστεί για να «δουλεύει» με δυαδικές τιμές[103].

$$\boldsymbol{P(x_i \mid y) = P(i \mid y)x_i + \big(1 - P(i|y)\big)(1 - x_i)}$$

Από την άλλη ο αλγόριθμος Gaussian είναι κατάλληλος για συνεχή δεδομένα[104].

$$\boldsymbol{P(x_i|y) = \left(\frac{1}{\sqrt{2\pi\sigma_y^2}}\right) * exp\left(-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2}\right)}$$

Όπου μ είναι η μέση τιμή και σ είναι η τυπική απόκλιση και υπολογίζονται χρησιμοποιώντας τον αλγόριθμο μέγιστης πιθανοφάνειας (Maximum Likelihood)[105].

Ο αλγόριθμος **Stochastic Gradient Descent-SGD** (**Στοχαστική διαβάθμισή κλίσης**) είναι μια διαφορετική εφαρμογή του απλού αλγορίθμου **διαβάθμισης κλίσης** (**Gradient Descent**). Είναι ένας απλός αλλά ταυτόχρονα πολύ αποτελεσματικός αλγόριθμος για την κατασκευή γραμμικών **παλινδρομητών (Regressors)** και **ταξινομητών** (**Classifiers**) σε σχέση με

[102] Sunil Ray, "6 Easy Steps To Learn Naïve Bayes Algorithm with codes in Python and R", Analytics Vidhya, 11 September 2017, https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ (Ανακτήθηκε 21/3/2021)

[103] Scikit Learn, "Bernoulli Naïve Bayes", https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html (Ανακτήθηκε 21/3/2021)

[104] Rohith Gandhi, "Naïve Bayes Classifier", Towards Data Science, 5 May 2018, https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (Ανακτήθηκε 21/3/2021)

[105] Scikit Learn, "Naïve Bayes", https://scikit-learn.org/stable/modules/naive_bayes.html (Ανακτήθηκε 22/3/2021)

άλλους αλγόριθμους μηχανικής μάθησης όπως τις διανυσματικές μηχανές υποστήριξης και την λογιστική παλινδρόμηση. Ο αλγόριθμος SGD θεωρείται διαδικασία βελτιστοποίησης παρά διαδικασία ταξινόμησης αλλά η βιβλιοθήκη Scikit Learn παρέχει σαν μοντέλο τον αλγόριθμο SGD για την διεξαγωγή αναλύσεων. Τα πλεονεκτήματα του αλγορίθμου αυτού είναι η αποτελεσματικότητα του και η εύκολη υλοποίηση του. Τα μειονεκτήματα του είναι η ευαισθησία του στην κλιμάκωση των χαρακτηριστικών και ο υποχρεωτικός ορισμός μερικών υπερπαραμέτρων για την λειτουργία του[106].

Ο αλγόριθμος **Extreme Gradient Boosting-XGBoost** ανήκει στα μοντέλα συνόλων της μηχανικής μάθησης (**Ensemble Machine Learning Algorithm**) όπως ο αλγόριθμος Random Forest και μπορεί να χρησιμοποιηθεί είτε για προβλήματα παλινδρόμησης είτε για ταξινόμησης. Το μοντέλο αυτό αποτελείται από ένα σύνολο από δέντρα απόφασης τα οποία εισέρχονται στο σύνολο ένα – ένα κάθε φορά. Μετά την εισαγωγή ενός δέντρου ο αλγόριθμος εφαρμόζεται ξανά με στόχο να διορθώσει το σφάλμα πρόβλεψης που είχαν τα προηγούμενα μοντέλα. Ο αλγόριθμος πήρε το όνομα **Gradient Boosting** επειδή στο μοντέλο εφαρμόζεται ο αλγόριθμος **διαβάθμισης βαθμίδας** (**Gradient Descent**) για την βελτιστοποίηση του μοντέλου και ελαχιστοποιείται το σφάλμα κλίσης (Gradient). Ο αλγόριθμος για να μπορεί να γενικεύσει τα αποτελέσματα του, χρησιμοποιεί δύο **τεχνικές γενίκευσης,** την **L1 γενίκευση** (**L1 Regularization**) και την **L2 γενίκευση** (**L2 Regularization**). Δύο βασικά πλεονεκτήματα του αλγορίθμου αυτού, είναι η υπολογιστική ταχύτητα που προσφέρει και η πολύ καλή απόδοση του μοντέλου. Ένα μειονέκτημα που έχει, είναι ότι μπορεί να μην έχει καλή απόδοση αν στα δεδομένα εκπαίδευσης ο αριθμός των χαρακτηριστικών είναι μεγαλύτερος από τον αριθμό των δειγμάτων[107][108].

Το μοντέλο **Ridge** που παρέχει η βιβλιοθήκη **Scikit Learn** βασίζεται στον αλγόριθμο **Ridge Regression,** ο οποίος μετασχηματίζει τις ετικέτες σε τιμές μέσα στο διάστημα {-1,1} και μετατρέπει το πρόβλημα, από πρόβλημα ταξινόμησης σε πρόβλημα παλινδρόμησης[109]. Συγκεκριμένα ο αλγόριθμος **Ridge Regression** είναι μια μέθοδος για την ανάλυση δεδομένων που πάσχουν από πολυγραμμικότητα. Το πρόβλημα της πολυγραμμικότητας συμβαίνει όταν η

[106] Scikit Learn, " Stochastic Gradient Descent " (Ανακτήθηκε 21/3/2021)
[107] Jason Brownlee, "Extreme Gradient Boosting (XGBoost) Ensemble in Python", Machine Learning Mastery, 23 November 2020, https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/ (Ανακτήθηκε 21/3/2021)
[108] Neetika Khandelwal, "A Brief Introduction to XGBoost", Towards Data Science,7 July 2020, https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6 (Ανακτήθηκε 21/3/2021)
[109] Scikit Learn, "Ridge Classifier", https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html (Ανακτήθηκε 21/3/2021)

διακύμανση είναι μεγάλη και το αποτέλεσμα είναι ότι οι τιμές που προβλέπει το μοντέλο είναι αρκετά μακριά από τις πραγματικές τιμές. Για αυτόν τον λόγο ο αλγόριθμος χρησιμοποιεί και μια τεχνική γενίκευσης την **L2 γενίκευση** (**L2 Regularization**)[110]. Για την καλύτερη απόδοση του αλγόριθμου τα δεδομένα πρέπει να έχουν **τυποποιηθεί** (**Standardized**). Πλεονέκτημα του αλγορίθμου αυτού είναι ότι ανταλλάζει την **διακύμανση** (**Variance**) για την **μεροληψία** (**Bias**) και αποτρέπει την υπερπροσαρμογή (**Overfitting**). Μειονέκτημα είναι ότι μπορεί να αυξηθεί κατά πολύ η μεροληψία και να μην έχουμε τα επιθυμητά αποτελέσματα, πρέπει να επιλεχθεί η υπερπαράμετρος alpha και τέλος το μοντέλο να μην παρέχει ικανοποιητικά αποτελέσματα[111].

Για να μπορεί να υπάρχει ένα μέτρο σύγκρισης «ίσο» με αυτών των νευρωνικών δικτύων ορίστηκε το ποσοστό διατήρησης της διακύμανσης (Variance) των δεδομένων στο 95% για τον αλγόριθμο PCA (δηλαδή να διατηρηθεί στα καινούργια δεδομένα που θα παράγει ο αλγόριθμος PCA το 95% της πληροφορίας των αρχικών δεδομένων), με αποτέλεσμα ο αριθμός των χαρακτηριστικών να μειωθεί από 20,502 χαρακτηριστικά στα **2,210** χαρακτηριστικά.

```
pca = PCA(n_components=0.95,random_state=1)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

```
pd.DataFrame(X_train_pca).shape
```

```
(10736, 2210)
```

```
pd.DataFrame(X_test_pca).shape
```

```
(2684, 2210)
```

Εικόνα 7.5: Αριθμός χαρακτηριστικών μετά την χρήση PCA με το 95% διακύμανσης.

Στην συνέχεια υπήρξε ο συντονισμός ορισμένων υπερπαραμέτρων για τους αλγόριθμους μηχανικής μάθησης που προ αναφέρθηκαν πιο πάνω. Συγκεκριμένα, οι υπερπαραμέτροι που συντονίστηκαν είναι:

[110] Great Learning Team, "What is Ridge Regression", Great Learning, 15 October 2020, https://www.mygreatlearning.com/blog/what-is-ridge-regression/ (Ανακτήθηκε 21/3/2021)
[111] Gokul Elumalai, "Pros and Cons of common Machine Learning Algorithms", Medium, 19 November 2019, https://medium.com/@gokul.elumalai05/pros-and-cons-of-common-machine-learning-algorithms-45e05423264f (Ανακτήθηκε 21/3/2021)

- SVC – (Support Vector Classifier) :
    - Πυρήνας (kernel): linear, rbf, poly
    - C: 0.25,0.5,0.75,1.0
- Random Forests:
    - Κριτήριο (criterion): gini, entropy
    - Αριθμός εκτιμητών (n_estimators): 10, 30, 50, 70, 90, 110, 130, 150, 170, 190
    - Μέγιστο βάθος (max_depth): 3,4,5,6,7,8,9,10
- XGB-ExtremeGradientBoosting:
    - Ενισχυτής (Booster): gblinear, gbtree
    - Αριθμός εκτιμητών (n_estimators): 50, 100, 200
- SGD-Stochastic Gradient Descent:
    - Απώλεια (loss): hinge, log, perceptron
    - Ποινή (penalty): l1, l2, elasticnet
- Ridge:
    - Alpha: 0.001 – 1 (sampling=”log”)

Επίσης για όσους αλγορίθμους υπήρχε η επιλογή, τέθηκε το random_state = 1 ώστε να υπάρχει η δυνατότητα ανακατασκευής των αποτελεσμάτων και n_jobs = -1 για να χρησιμοποιηθούν όλοι οι πόροι του υπολογιστή για μεγαλύτερη υπολογιστική ταχύτητα. Έπειτα, για τους αλγορίθμους GaussianNB, BernoulliNB δεν υπήρξε ο συντονισμός για κάποια υπερπαράμετρο. Χρησιμοποιήθηκε για την βελτιστοποίηση των υπερπαραμέτρων η Μπεϋζιανή μέθοδος με αντικειμενικό στόχο την μεγιστοποίηση της ακρίβειας και ορίστηκε ο αριθμός μεγίστων δοκιμών στους τριάντα. Τέλος, για την αξιολόγηση κάθε αλγορίθμου χρησιμοποιήθηκε η συνάρτηση `StratifiedKFold` από την βιβλιοθήκη **Scikit Learn** για πέντε folds. Με αποτέλεσμα κάθε φορά που επιλεγόταν ένας αλγόριθμος με κάποιες επιλεγμένες παραμέτρους από την Μπεϋζιανή βελτιστοποίηση, αυτός ο αλγόριθμος εκπαιδευόταν πέντε φορές και κάθε φορά χώριζε τα δεδομένα εκπαίδευσης ώστε το 80% των δεδομένων να εκπαιδεύονται από τον αλγόριθμο και να αξιολογείται η απόδοση στο υπόλοιπο 20%. Ο χωρισμός που γινόταν κάθε φορά ήταν τυχαίος. Η διαδικασία αυτή ονομάζεται **K-Fold Cross Validation**. Στην ανάλυση που πραγματοποιήθηκε το K ισούταν με ***πέντε.*** Επίσης, δεν χρησιμοποιήθηκε η διαδικασία K-Fold Cross Validation αλλά μια επέκταση της, η **Stratified K-Fold Cross Validation**, η οποία χωρίζει τα δεδομένα με κάθε κλάση να διατηρεί το 80%

των δεδομένων της για εκπαίδευση και το 20% για την αξιολόγηση[112]. Εκτός όμως της σύγκρισης μεταξύ νευρωνικών δικτύων και των αλγορίθμων μηχανικής μάθησης, πραγματοποιήθηκαν δύο υβριδικές αναλύσεις μεταξύ των δύο διαφορετικών τεχνικών. Η πρώτη ανάλυση επικεντρώνεται στην χρήση των νευρωνικών δικτύων ταξινόμησης με είσοδο τα δεδομένα που κατασκεύασε ο αλγόριθμος **PCA**. Η δεύτερη, επικεντρώνεται στην χρήση των αλγορίθμων μηχανικής μάθησης με είσοδο τα δεδομένα που πάρθηκαν από τα δίκτυα των **Autoencoders**. Ο συντονισμός των υπερπαραμέτρων των νευρωνικών δικτύων και των αλγορίθμων μηχανικής μάθησης είναι ακριβώς ο ίδιος με τις προηγούμενες αναλύσεις.

7.2 Παρουσίαση αποτελεσμάτων μηχανικής μάθησης

Τα αποτελέσματα που παρατηρήθηκαν ήταν ότι ο καλύτερος αλγόριθμος που επιλέχθηκε στο τέλος της Μπεϋζιανής βελτιστοποίησης ήταν ο SVC με υπερπαραμέτρους:

- Πυρήνας (Kernel): linear
- C: 0.25

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='score', direction='max')
Trial summary
Hyperparameters:
model_type: svc
kernel: linear
C: 0.25
```

Εικόνα 7.6: Καλύτεροι υπερπαραμέτροι SVC με δεδομένα από τον PCA.

Το ποσοστό ακρίβειας που κατάφερε για την πρόβλεψη των δεδομένων εκπαίδευσης ήταν 99.22%.

```
print(accuracy_score(y_train,y_pred_train_pca))
print(f1_score(y_train,y_pred_train_pca,average="micro"))
```

```
0.9922690014903129
0.9922690014903129
```

[112] Scikit Learn, “Stratified K-Fold ”, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (Ανακτήθηκε 23/3/2021)

Εικόνα 7.7: Αποτελέσματα PCA-SVC στα δεδομένα εκπαίδευσης

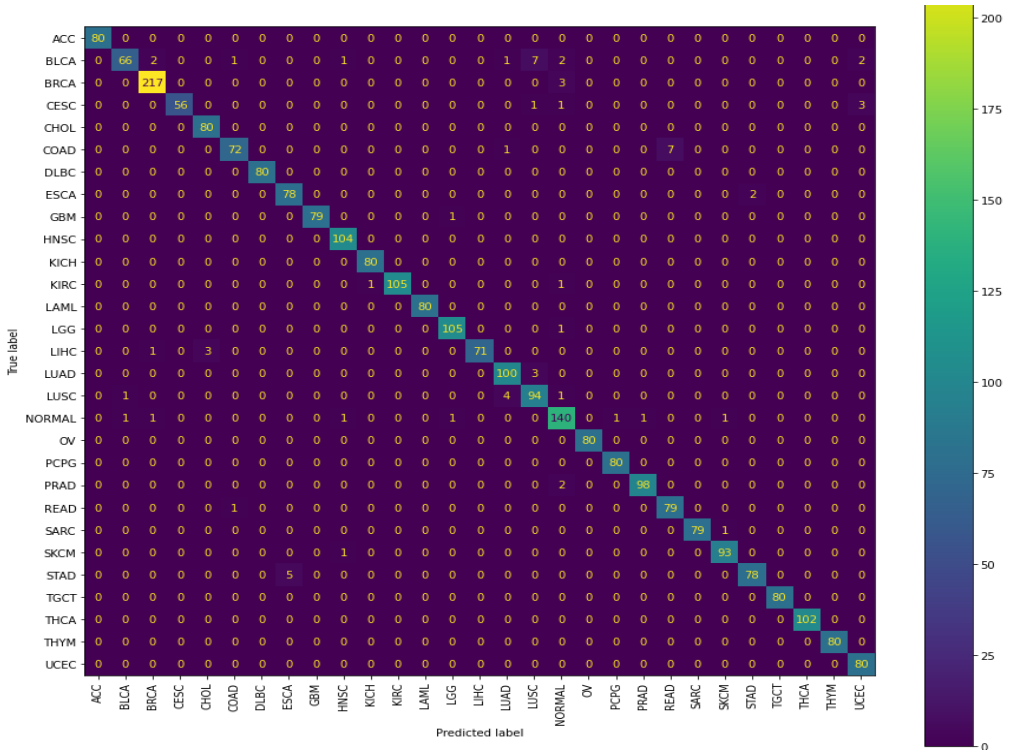
Για την πρόβλεψη των δεδομένων δοκιμής που δεν είχε ξανά συναντήσει το ποσοστό ακρίβειας ανήλθε σε 97.46%.

```
print(accuracy_score(y_test,y_pred_pca))
print(f1_score(y_test,y_pred_pca,average="micro"))
```

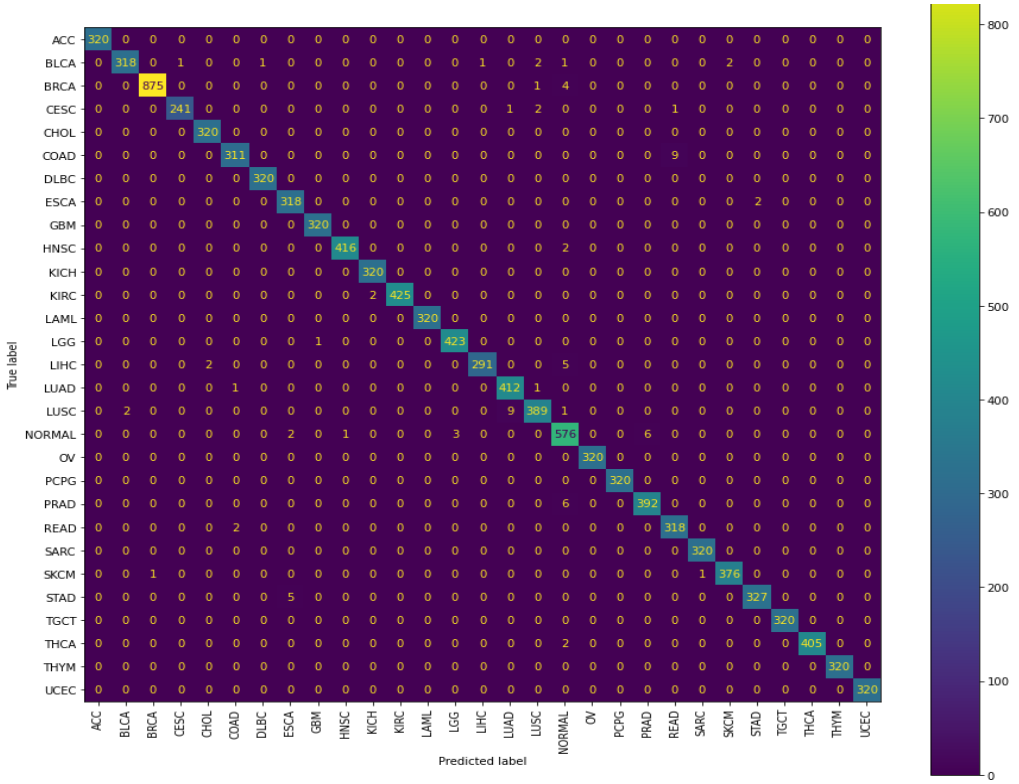
```
0.9746646795827124
0.9746646795827124
```

Εικόνα 7.8: Αποτελέσματα PCA-SVC στα δεδομένα δοκιμής

Παρακάτω φαίνονται οι πίνακες «σύγχυσης» (Confusion Matrix) για τα δεδομένα δοκιμής και τα δεδομένα εκπαίδευσης.



Εικόνα 7.9: Πίνακας «σύγχυσης» PCA-SVC στα δεδομένα δοκιμής



Εικόνα 7.10: Πίνακας «σύγχυσης» PCA-SVC στα δεδομένα εκπαίδευσης

7.2.1 Πρώτη Υβριδική Ανάλυση – PCA => DNN Ταξινόμησης

Σε αυτήν την περίπτωση χρησιμοποιήθηκαν τα παραπάνω χαρακτηριστικά που παράχθηκαν από τον αλγόριθμο PCA και τροφοδοτήθηκαν σε ένα βαθύ νευρωνικό δίκτυο. Επίσης, χρησιμοποιήθηκε η Μπεϋζιανή βελτιστοποίηση. Η διαφορά με τις υπερπαραμέτρους των παραπάνω νευρωνικών δικτύων ταξινόμησης ήταν ότι δεν εφαρμόστηκε κάποιο επίπεδο εγκατάλειψης (Dropout Layer). Οι καλύτεροι υπερπαραμέτροι που επιλέχθηκαν ήταν δύο κρυφά πυκνά επίπεδα (Dense Layer) με 256 και 1024 νευρώνες για το πρώτο και το δεύτερο κρυφό επίπεδο αντίστοιχα. Η συνάρτηση ενεργοποίησης που τέθηκε ήταν η ELU με αρχικοποίηση βαρών με την συνάρτηση Glorot Uniform**.** Χρησιμοποιήθηκε στα πυκνά επίπεδα η μέθοδος L2 γενίκευσης (L2 regularization). Τέλος, το ποσοστό εκμάθησης επιλέχθηκε στο 0.0001. Όπως και με τα παραπάνω νευρωνικά δίκτυα, μετά την επιλογή των καλύτερων υπερπαραμέτρων, το δίκτυο ξανά εκπαιδεύτηκε για 100 «εποχές» (Epochs). Παρακάτω παρουσιάζονται τα αποτελέσματα αυτής της υλοποίησης.

```
Results summary
Results in ./untitled_project
Showing 1 best trials
Objective(name='val_sparse_categorical_accuracy', direction='max')
Trial summary
Hyperparameters:
units: 256
activations: elu
kernel_initializer: glorot_uniform
kernel_regularizers: l2
layers: 1
units0: 1024
learning_rate: 0.0001
```

Εικόνα 7.11: Καλύτεροι υπερπαραμέτροι PCA-DNN.

Το ποσοστό ακρίβειας στα δεδομένα εκπαίδευσης ήταν 99.8% καθώς και στα δεδομένα δοκιμής ήταν 96.6%.

```
print(accuracy_score(y_test,y_pred_pca_dnn))  # accuracy = correct/total
print(f1_score(y_test,y_pred_pca_dnn,average="micro"))
```

```
0.9660953800298062
0.9660953800298062
```

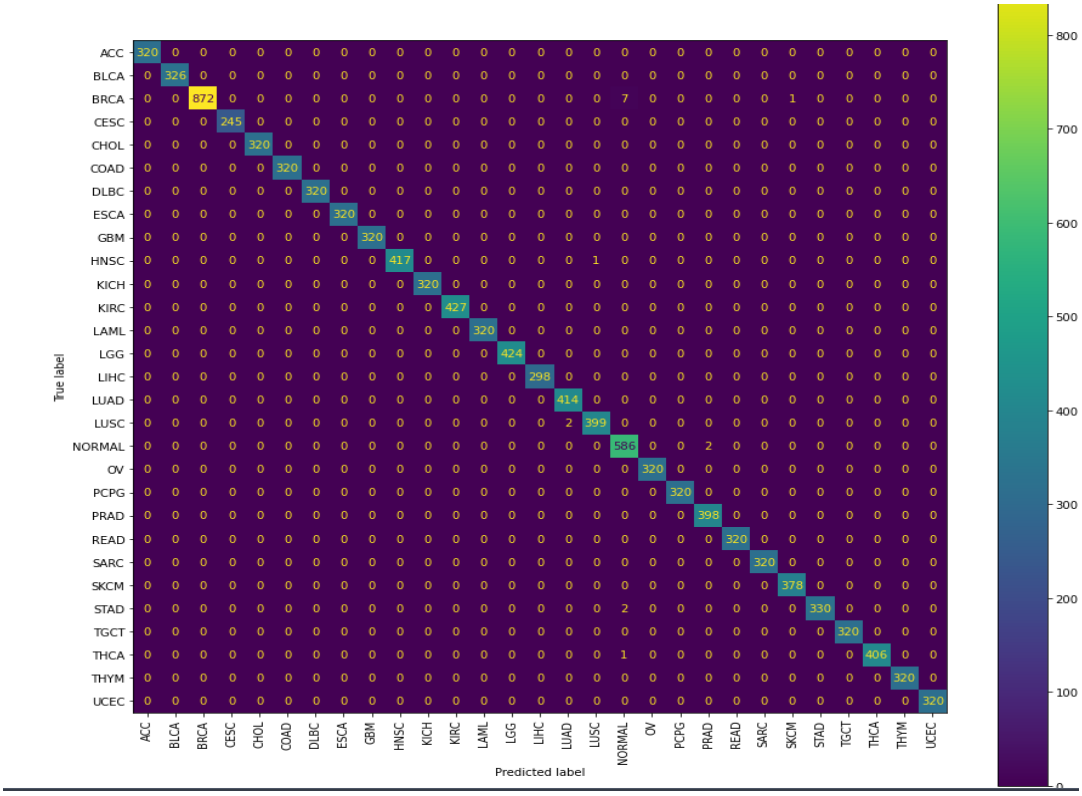
Εικόνα 7.12: Accuracy PCA-DNN στα δεδομένα δοκιμής

```
print(accuracy_score(y_train,y_pred_train_pca_dnn))  # accuracy = correct/total
print(f1_score(y_train,y_pred_train_pca_dnn,average="micro"))
```

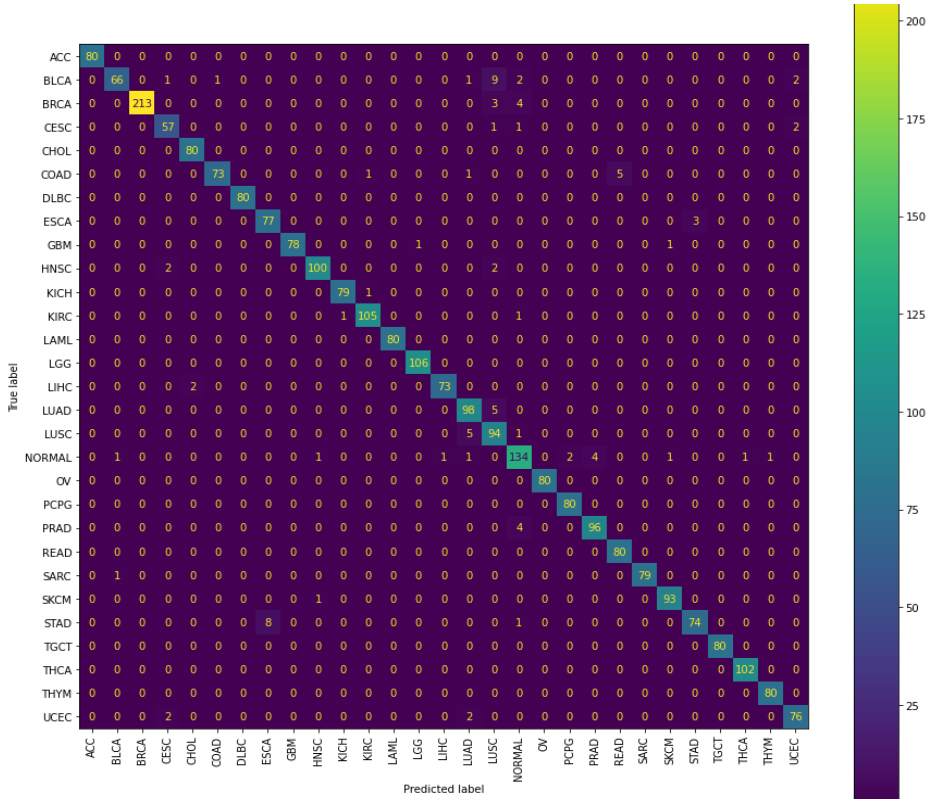
```
0.9985096870342772
0.9985096870342772
```

Εικόνα 7.13: Accuracy PCA-DNN στα δεδομένα εκπαίδευσης

Τέλος, οι πίνακες «σύγχυσης» που παράχθηκαν φαίνονται παρακάτω στα δεδομένα εκπαίδευσης και δοκιμής.



Εικόνα 7.14: Πίνακας «σύγχυσης» PCA-DNN στα δεδομένα εκπαίδευσης



Εικόνα 7.15: Πίνακας «σύγχυσης» PCA-DNN στα δεδομένα δοκιμής

7.2.2 Δεύτερη Υβριδική Ανάλυση – Autoencoders => ML Algorithms

7.2.2.1 Autoencoder

Χρησιμοποιήθηκαν τα δεδομένα που παράχθηκαν από το **Autoencoder** στην ταξινόμηση τους με την χρήση των αλγορίθμων μηχανικής μάθησης. Ο αλγόριθμος με τα καλύτερα αποτελέσματα ήταν ο **SVC** με πυρήνα την μέθοδο **Poly** (Polynomial – Πολυωνυμική).

```
best_model_auto = autoencoder_tuner.get_best_models(1)[0]
best_model_auto
```

```
SVC(kernel='poly', random_state=1)
```

Εικόνα 7.16: Εικόνα υπερπαραμέτρων μοντέλου SVC με δεδομένα από τον Autoencoder.

Το ποσοστό ακρίβειας στα δεδομένα εκπαίδευσης και δοκιμής ήταν 99.4% και 96.3% αντίστοιχα.

```
print(accuracy_score(y_test,y_pred_autoencoder))
print(f1_score(y_test,y_pred_autoencoder,average="micro"))
```

```
0.9631147540983607
0.9631147540983607
```

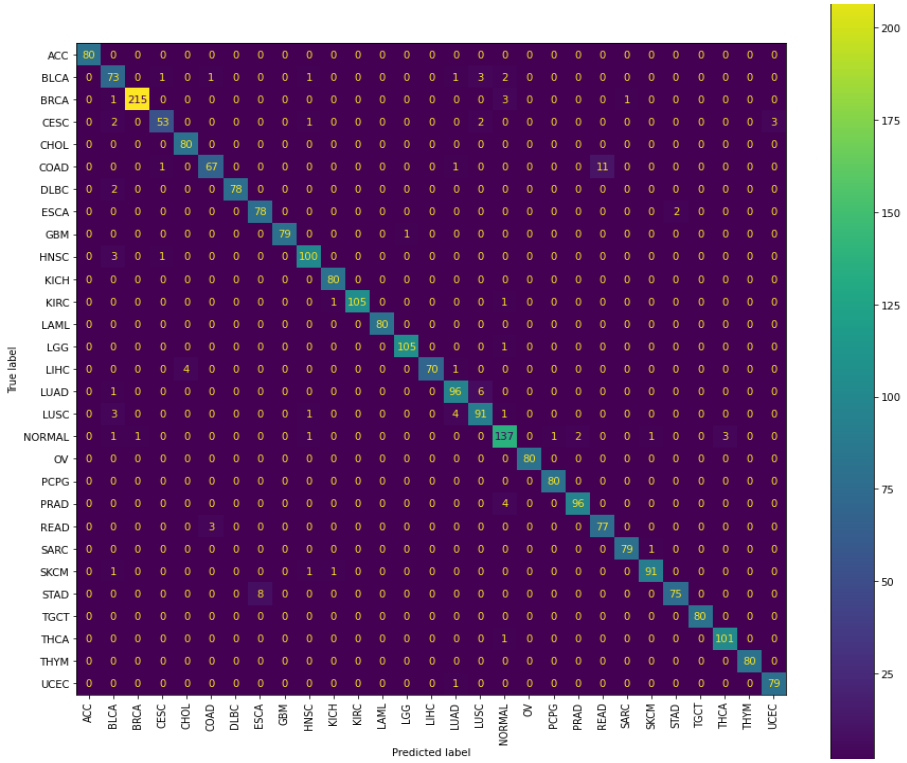
Εικόνα 7.17: Accuracy Autoencoder-SVC στα δεδομένα δοκιμής

```
print(accuracy_score(y_train,y_pred_train_autoencoder))
print(f1_score(y_train,y_pred_train_autoencoder,average="micro"))
```

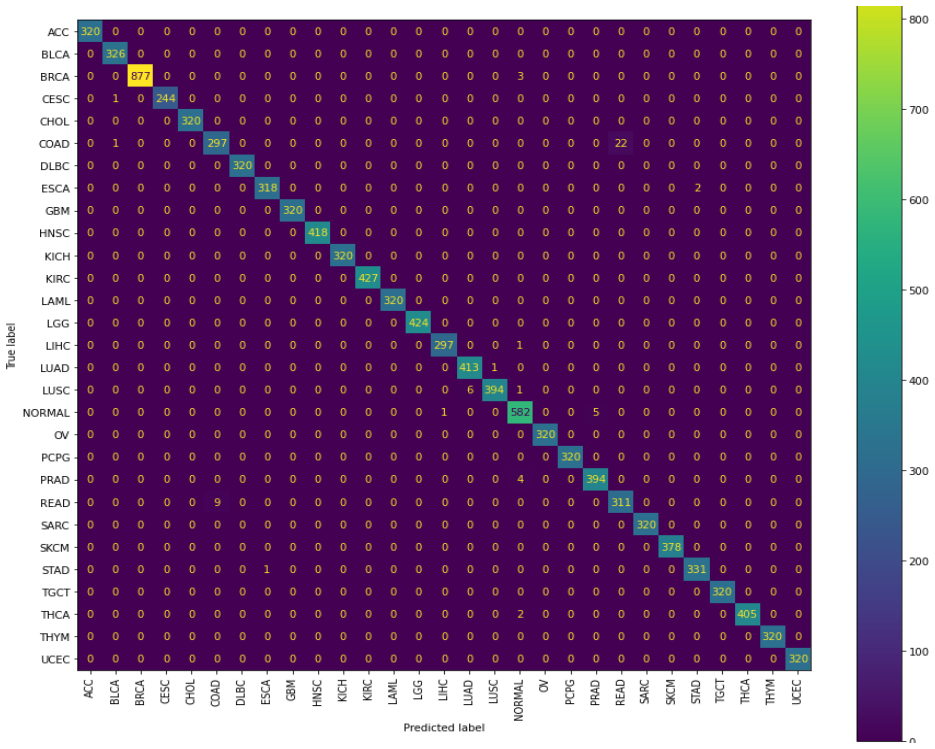
```
0.9944113263785395
0.9944113263785396
```

Εικόνα 7.18: Accuracy Autoencoder-SVC στα δεδομένα εκπαίδευσης

Παρακάτω επισυνάπτονται οι πίνακες «σύγχυσης» για τα δεδομένα δοκιμής και εκπαίδευσης.



Εικόνα 7.19: Πίνακας «σύγχυσης» Autoencoder-SVC στα δεδομένα δοκιμής



Εικόνα 7.20: Πίνακας «σύγχυσης» Autoencoder-SVC στα δεδομένα εκπαίδευσης

7.2.2.2 Denoising Autoencoder

Όπως και παραπάνω τα δεδομένα που παράχθηκαν από αυτό το μοντέλο τροφοδοτήθηκαν σε αλγόριθμους μηχανικής μάθησης για την διαδικασία της ταξινόμησης. Το καλύτερο μοντέλο που επιλέχθηκε ήταν ο αλγόριθμος **SVC.** Για τον πυρήνα είχε επιλεχθεί η μέθοδος **Poly** ενώ η υπερπαράμετρος C είχε τεθεί στην τιμή 0.75.

```
best_model_deno = deno_tuner.get_best_models(1)[0]
best_model_deno
```

```
SVC(C=0.75, kernel='poly', random_state=1)
```

Εικόνα 7.21: Εικόνα υπερπαραμέτρων μοντέλου SVC με δεδομένα από τον Denoising Autoencoder.

Το ποσοστό ακρίβειας στα δεδομένα εκπαίδευσης και δοκιμής ήταν 97.6%, 94.8% αντίστοιχα.

```
print(accuracy_score(y_test,y_pred_denoizing_autoencoder))  # accuracy = correct/total
print(f1_score(y_test,y_pred_denoizing_autoencoder,average="micro"))
```

```
0.948956780923994
0.948956780923994
```

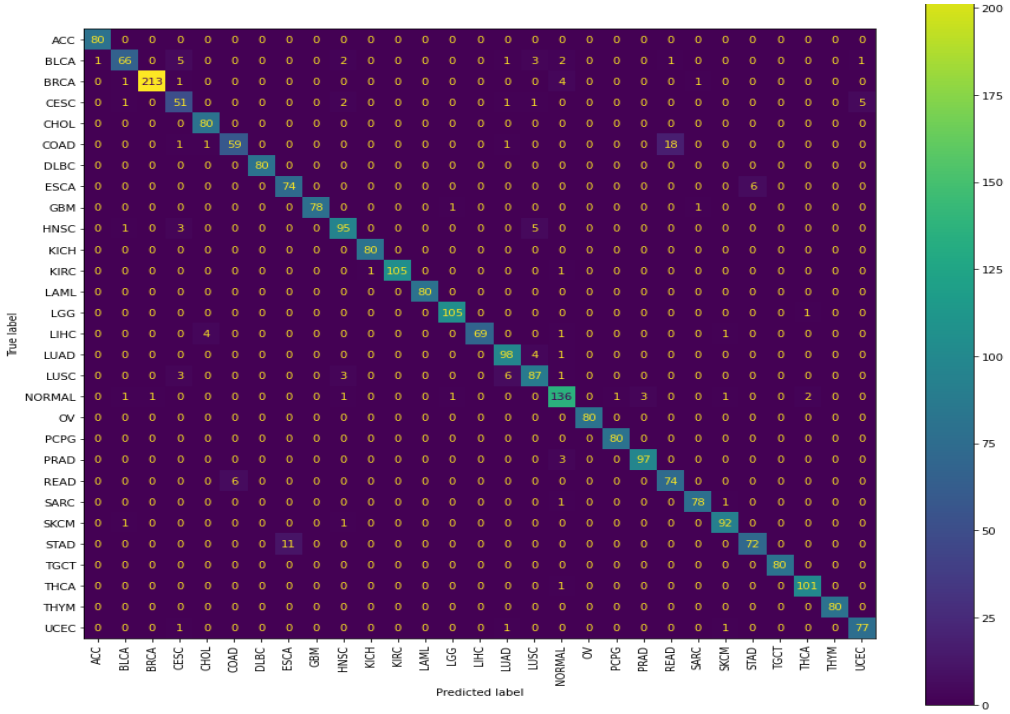
Εικόνα 7.22: Accuracy Denoising Autoencoder-SVC στα δεδομένα δοκιμής

```
print(accuracy_score(y_train,y_pred_train_denoizing_autoencoder))  # accuracy = correct/total
print(f1_score(y_train,y_pred_train_denoizing_autoencoder,average="micro"))
```

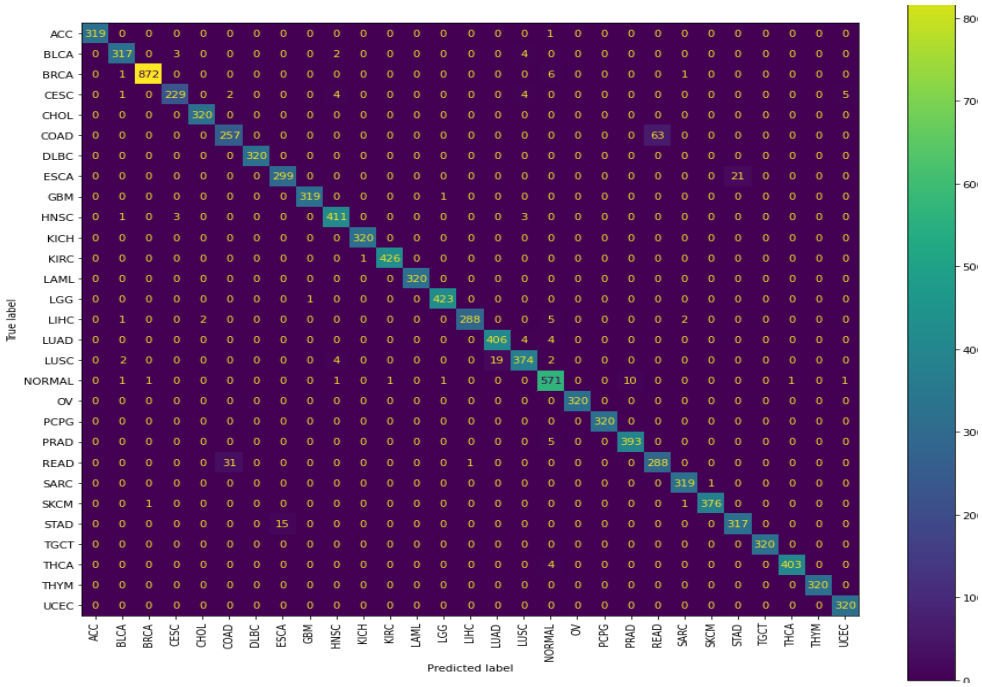
```
0.9768070044709389
0.9768070044709389
```

Εικόνα 7.23: Accuracy Denoising Autoencoder-SVC στα δεδομένα εκπαίδευσης

Παρακάτω επισυνάπτονται οι πίνακες «σύγχυσης» για τα δεδομένα δοκιμής και εκπαίδευσης.



Εικόνα 7.24: Πίνακας «σύγχυσης» Denoising Autoencoder-SVC στα δεδομένα εκπαίδευσης



Εικόνα 7.25: Πίνακας «σύγχυσης» Denoising Autoencoder-SVC στα δεδομένα δοκιμής

7.2.2.3 Variational Autoencoder

Ο καλύτερος αλγόριθμος που επιλέχθηκε με τα δεδομένα από τον Variational Autoencoder ήταν ο SVC με την χρήση γραμμικού πυρήνα (Linear Kernel) και το C επιλεγμένο στο 0.5.

```
best_model_vae = vae_tuner.get_best_models(1)[0]
best_model_vae
```

```
SVC(C=0.5, kernel='linear', random_state=1)
```

Εικόνα 7.26: Εικόνα υπερπαραμέτρων μοντέλου SVC με δεδομένα από τον Variational Autoencoder.

Το ποσοστό ακρίβειας ανήλθε στο 96.1% στα δεδομένα δοκιμής και 97.5% στα δεδομένα εκπαίδευσης.

```
print(accuracy_score(y_test,y_pred_vae))  # accuracy = correct/total
print(f1_score(y_test,y_pred_vae,average="micro"))
```

```
0.9619970193740686
0.9619970193740686
```

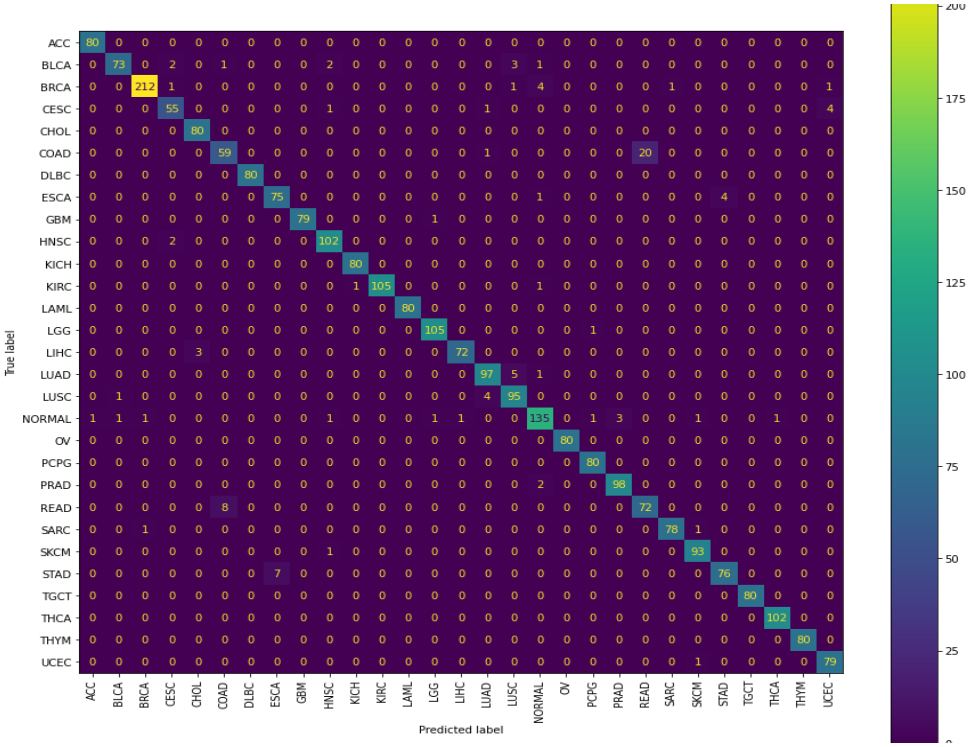
Εικόνα 7.27: Accuracy Variational Autoencoder-SVC στα δεδομένα δοκιμής

```
print(accuracy_score(y_train,y_pred_train_vae))  # accuracy = correct/total
print(f1_score(y_train,y_pred_train_vae,average="micro"))
```

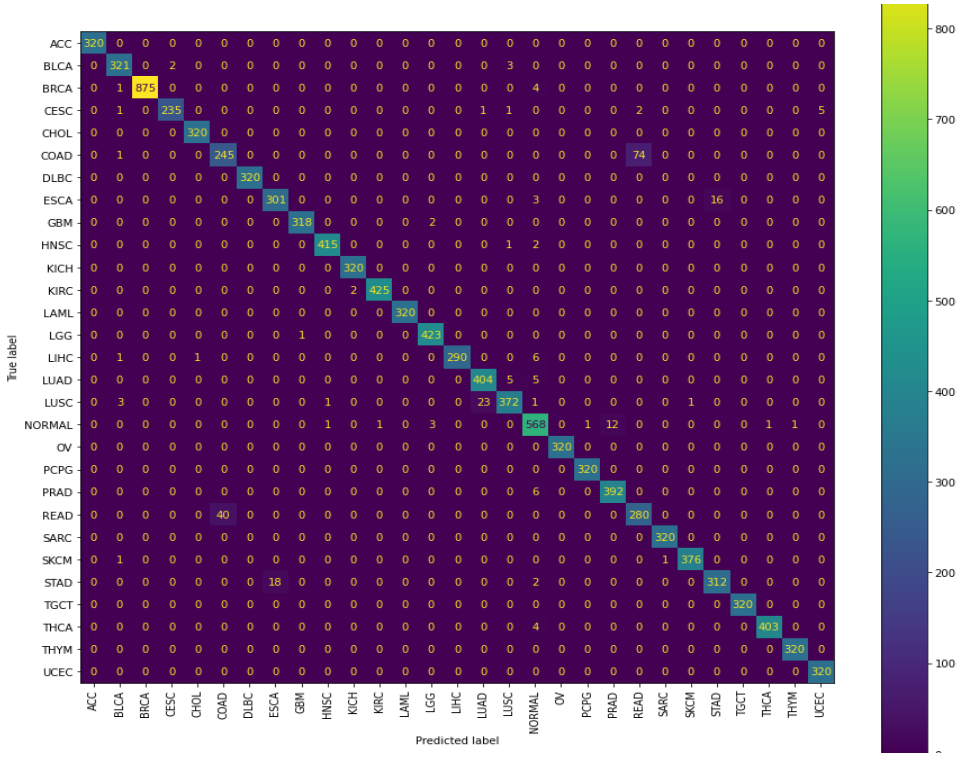
```
0.9756892697466468
0.9756892697466468
```

Εικόνα 7.28: Accuracy Variational Autoencoder-SVC στα δεδομένα εκπαίδευσης

Παρακάτω, επισυνάπτονται οι πίνακες «σύγχυσης» για τα δεδομένα εκπαίδευσης και δοκιμής αντίστοιχα.



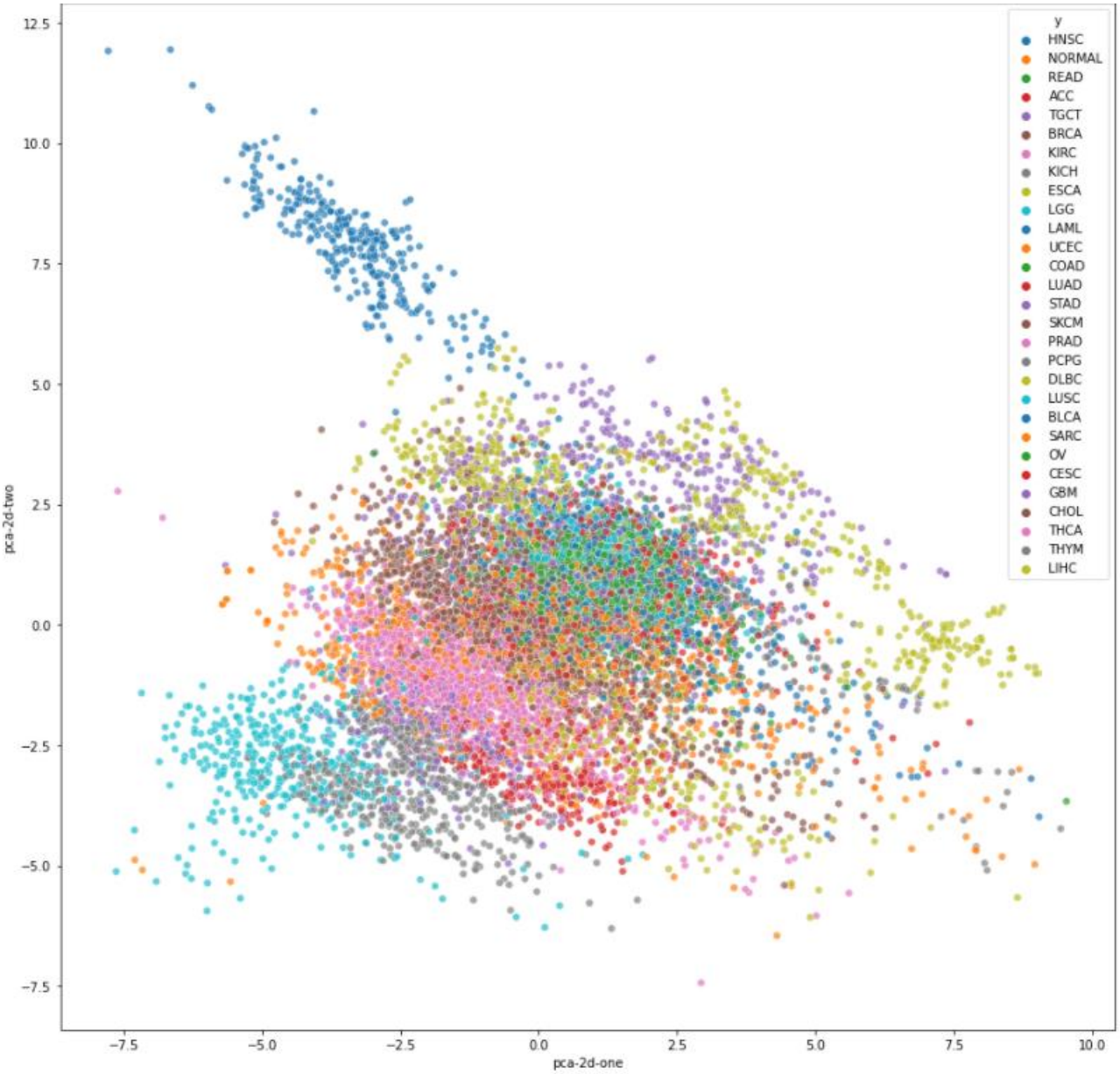
Εικόνα 7.29: Πίνακας «σύγχυσης» Variational Autoencoder-SVC στα δεδομένα εκπαίδευσης



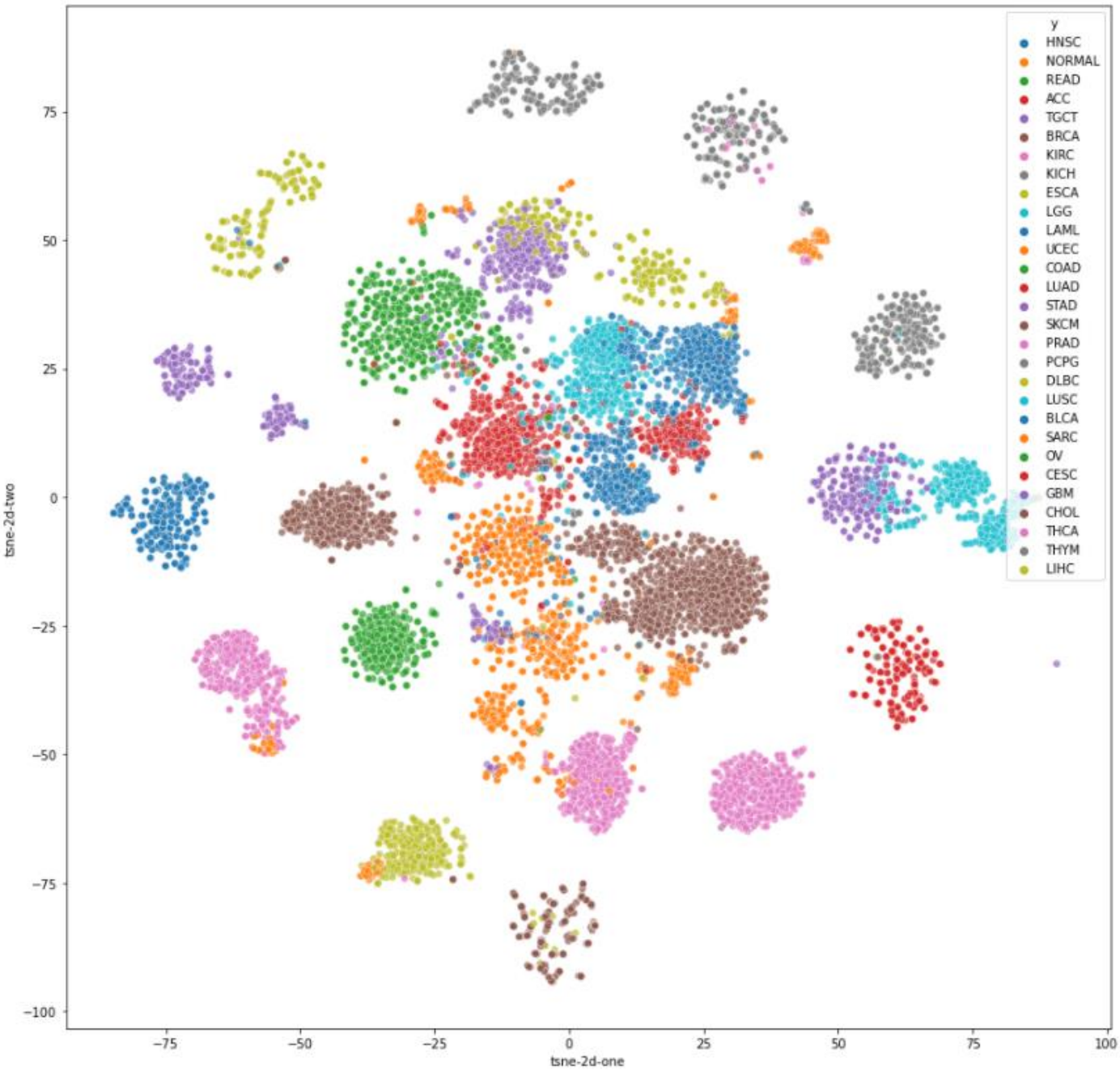
Εικόνα 7.30: Πίνακας «σύγχυσης» Variational Autoencoder-SVC στα δεδομένα δοκιμής

7.3 Ποιοτική σύγκριση αποτελεσμάτων

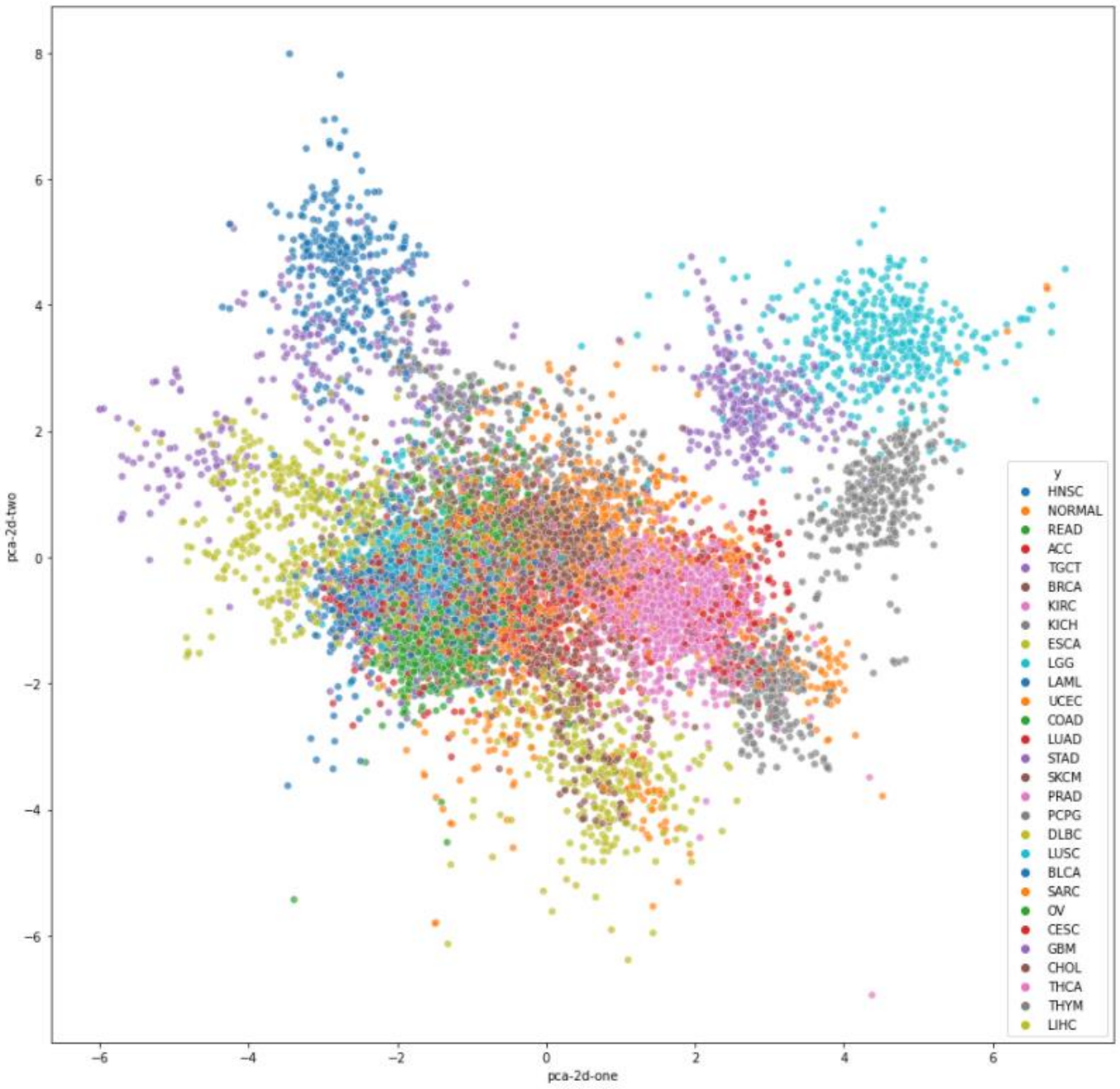
Απ' όλες τις αναλύσεις που διεξήχθησαν ήταν δύσκολη η επιλογή της καλύτερης. Το μεγαλύτερο ποσοστό ακρίβειας στα δεδομένα αξιολόγησης που παρατηρήθηκε ήταν αυτό όπου ο αριθμός των χαρακτηριστικών μειώθηκε με τον αλγόριθμο PCA και για την ταξινόμηση των δεδομένων αυτών χρησιμοποιήθηκε ο αλγόριθμος μηχανικής μάθησης SVC με το ποσοστό της ακρίβειας να φτάνει το 97.4%. Από την άλλη, είχε επιτευχθεί ένα ποσοστό ακρίβειας στο 96% και 95.4% με την χρήση ενός βαθύ νευρωνικού δικτύου στα δεδομένα του Autoencoder και του μοντέλου Deep Cross στα τα δεδομένα του Variational Autoencoder αντίστοιχα. Αυτό που παρατηρείται είναι ότι οι αλγόριθμοι μηχανικής μάθησης είχαν καλύτερο αποτέλεσμα. Αν όμως παρατηρηθεί ο αριθμός των χαρακτηριστικών διαπιστώνεται ότι για τον αλγόριθμο PCA για να υπάρξουν καλά αποτελέσματα έπρεπε να διατηρηθεί τουλάχιστον το 95% της πληροφορίας των αρχικών δεδομένων. Αυτό είχε ως αποτέλεσμα ο μικρότερος αριθμός χαρακτηριστικών που θα μπορούσε να δώσει ο PCA για να διατηρήσει τόση πληροφορία να είναι τα 2,210 χαρακτηριστικά. Σε σχέση με τις αρχιτεκτονικές Autoencoder και Variational Autoencoder που κατάφεραν να έχουν πολύ καλά αποτελέσματα, μειώνοντας τον αριθμό των χαρακτηριστικών στα 100 και 70 αντίστοιχα. Επίσης, παρατηρώντας και τις υβριδικές αναλύσεις που διεξήχθησαν παρατηρούμε ότι με τα δεδομένα από τον αλγόριθμο PCA το βαθύ νευρωνικό δίκτυο μπόρεσε να προβλέψει τα δεδομένα με 96.6% επιτυχία σε σχέση με τα δεδομένα που προήλθαν από το μοντέλο Autoencoder και την χρήση του αλγορίθμου SVC για την ταξινόμηση με ποσοστό ακρίβειας στο 96.3%. Από τις παραπάνω αναλύσεις, ίσως το καλύτερο αποτέλεσμα είναι η μείωση των χαρακτηριστικών με την χρήση ενός δικτύου Autoencoder και στην συνέχεια η ταξινόμηση τους με την χρήση του αλγορίθμου SVC. Παρακάτω παρουσιάζεται η οπτικοποίηση των δεδομένων εκπαίδευσης με την χρήση του αλγορίθμου PCA και του αλγορίθμου t-SNE, καθώς και των δεδομένων εκπαίδευσης που μειώθηκαν με το δίκτυο Variational Autoencoder και στην συνέχεια χρησιμοποιήθηκαν οι αλγόριθμοι PCA και t-SNE στα δεδομένα αυτά.



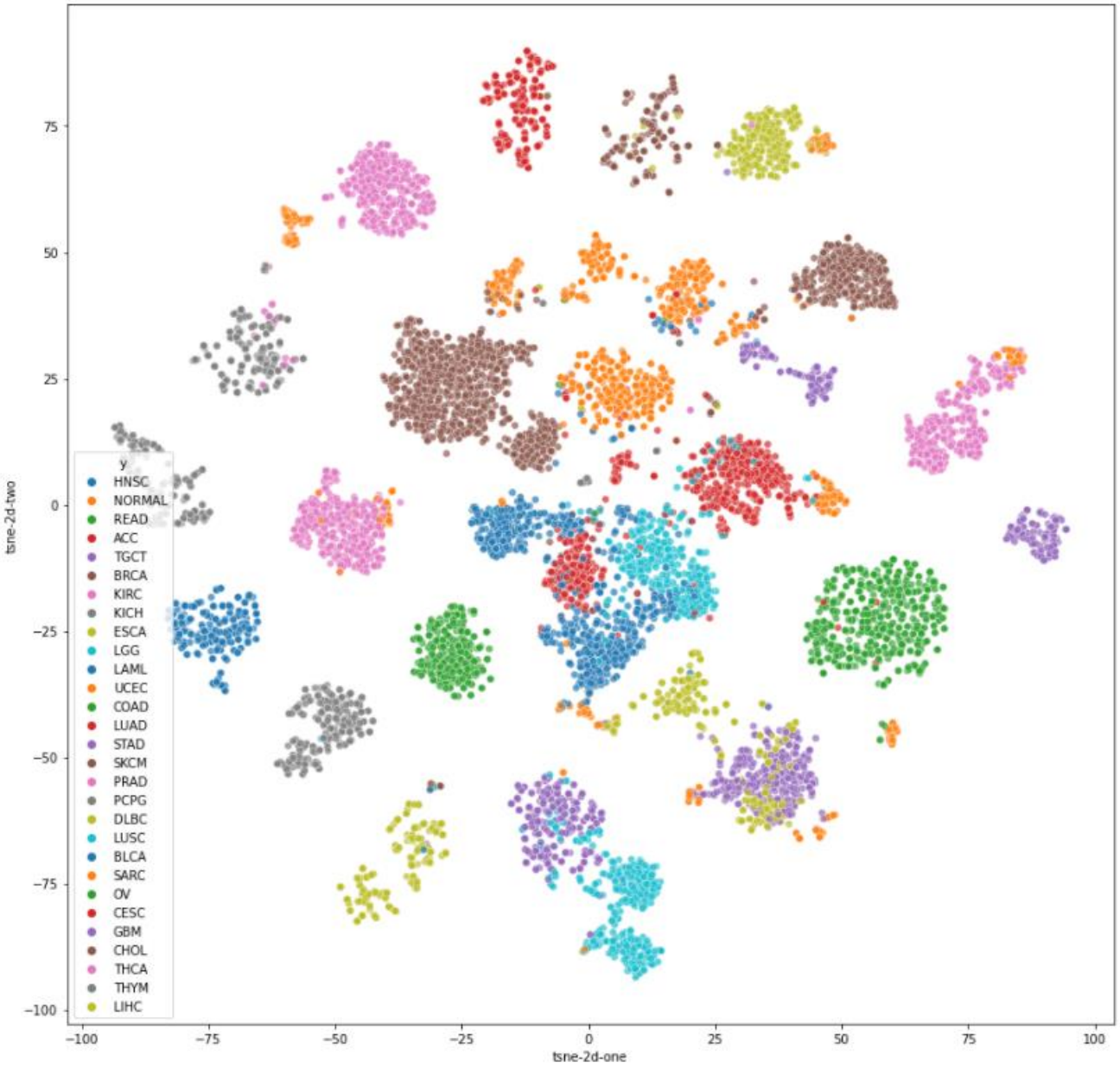
Εικόνα 7.31: Οπτικοποίηση δεδομένων με τον αλγόριθμο PCA



Εικόνα 7.32: Οπτικοποίηση δεδομένων με τον αλγόριθμο t-SNE



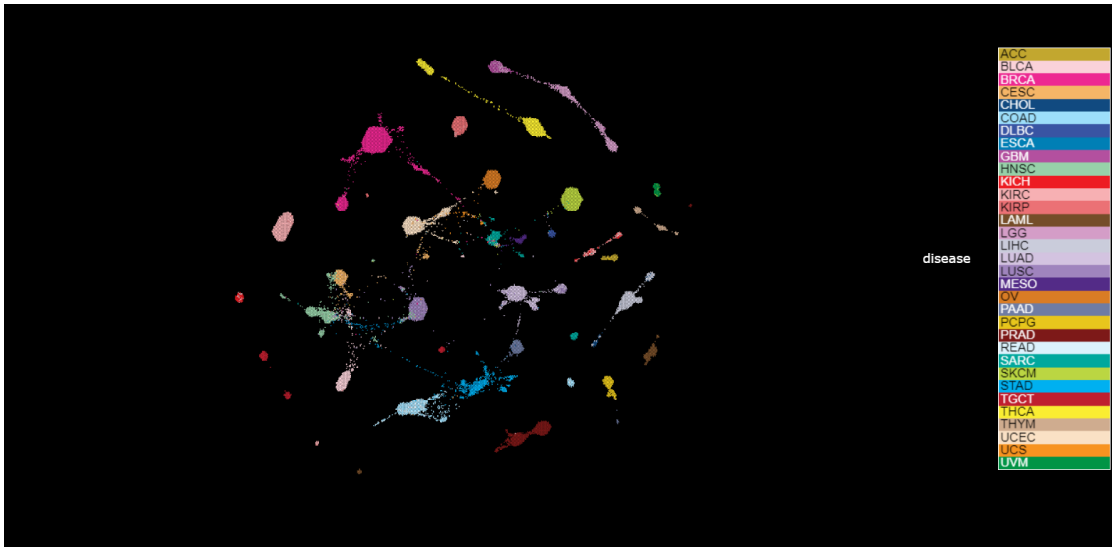
Εικόνα 7.33: Οπτικοποίηση δεδομένων με την Χρήση Variational Autoencoder και PCA



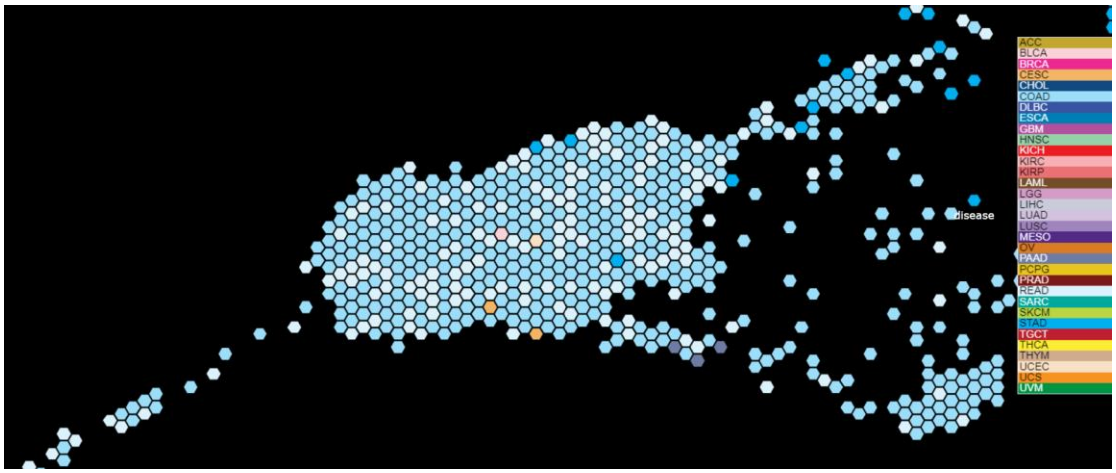
Εικόνα 7.34: Οπτικοποίηση δεδομένων με την Χρήση Variational Autoencoder και t-SNE

Από τις οπτικοποιήσεις παρατηρείται ότι η καλύτερη είναι η τελευταία, σύμφωνα με την οποία τα δεδομένα μειώθηκαν με την χρήση του μοντέλου Variational Autoencoder και του αλγορίθμου t-SNE. Αυτό συμβαίνει επειδή τα νευρωνικά δίκτυα και ο αλγόριθμος t-SNE είναι σε θέση να αιχμαλωτίζει γραμμικές και μη συσχετίσεις στα δεδομένα. Τέλος, απ' όλες τις ταξινομήσεις παρατηρήθηκε ότι σχεδόν όλα τα μοντέλα είχαν ένα μικρό ποσοστό λάθους στην διαδικασία της ταξινόμησης μεταξύ διαφορετικών τύπων καρκίνου. Πιο συγκεκριμένα, πολλά μοντέλα ταξινόμησαν γονιδιακές εκφράσεις ως τον τύπο READ (Rectum Adenocarcinoma) ενώ στην πραγματικότητα ήταν COAD (Colon Adenocarcinoma) καθώς και το αντίθετο,

πρόβλεψαν COAD ενώ ήταν READ. Άλλη μια λάθος ταξινόμηση που προέκυψε, ήταν η ταξινόμηση εκφράσεων ως ESCA (Esophageal Carcinoma) ενώ ήταν STAD (Stomach Adenocarcinoma) και το αντίθετο. Αυτό συμβαίνει επειδή οι γονιδιακές εκφράσεις αυτών των τύπων έχουν πολλές ομοιότητες μεταξύ τους. Για την επιβεβαίωση ότι αυτοί οι τύποι έχουν όντως ομοιότητες, παρατηρήθηκαν οι παραπάνω τύποι στον χάρτη όγκων που έχει δημιουργήσει το UCSC (University of California – Santa Cruz) τα αποτελέσματα παρουσιάζονται παρακάτω[113][114].



Εικόνα 7.35: Οπτικοποίηση ειδών καρκίνου από το Tumor Map. Πηγή: University of California Santa Cruz, https://tumormap.ucsc.edu/



Εικόνα 7.35: Οπτικοποίηση συσχετίσεων μεταξύ των κλάσεων READ και COAD από το Tumor Map. Πηγή: University of California Santa Cruz, https://tumormap.ucsc.edu/

[113] Yulia Newton, Adam M. Novak, Teresa Swatloski, Duncan C. McColl, Sahil Chopra, Kiley Graim, Alana S. Weinstein, Robert Baertsch, Sofie R. Salama, Kyle Ellrott, Manu Chopra, Theodore C. Goldstein, David Haussler, Olena Morozova, Joshua M. Stuart, " TumorMap: Exploring the Molecular Similarities of Cancer Samples in an Interactive Portal", NCBI, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5751940/ (Ανακτήθηκε 30/3/2021)
[114] University of California Santa Cruz, https://tumormap.ucsc.edu/

Εικόνα 7.36: Οπτικοποίηση συσχετίσεων μεταξύ των κλάσεων ESCA και STAD από το Tumor Map. Πηγή: University of California Santa Cruz, https://tumormap.ucsc.edu/

Τέλος παρουσιάζεται ένας συγκεντρωτικός πίνακας με τα αποτελέσματα όλων των αναλύσεων που διεξήχθη.

Algorithm	Accuracy, F1 score on Test Set	Accuracy, F1 score on Training Set
PCA (reduction to 2210 features with 95% variance) + SVC	**97.4%**	99.2%
PCA (reduction to 2210 features with 95% variance) + DNN	96.6%	**99.8%**
Autoencoder (reduction to 100 features) + SVC	96.3%	99.4%
Autoencoder (reduction to 100 features) + DNN	96%	98%
Autoencoder (reduction to 100 features) + Deep Cross Model	94%	95.8%
Denoising Autoencoder (reduction to 50 features) + SVC	94.8%	97.6%
Denoising Autoencoder(reduction to 50 features) + DNN	94.5%	98.7%
Denoising Autoencoder(reduction to 50 features) + Deep Cross Model	81.2%	83%
Variational Autoencoder (reduction to 70 features) + SVC	96.1%	97.5%
Variational Autoencoder(reduction to 70 features) + DNN	95%	96%
Variational Autoencoder(reduction to 70 features) + Deep Cross Model	95.4%	97%

Πίνακας 7.1: Ποιοτικά ποσοστά ακρίβειας μεταξύ των διαφορετικών αλγόριθμων.

7.4 Ποσοτική σύγκριση αποτελεσμάτων

Στην ποιοτική σύγκριση των αναλύσεων λαμβάνεται υπόψη ο χρόνος διεξαγωγής κάθε ανάλυσης. Αρχικά, κάθε ανάλυση διεξήχθη στο ίδιο σύνολο δεδομένων το οποίο είχε μέγεθος 1.52 Gb. Για την διεξαγωγή όλων των αναλύσεων χρησιμοποιήθηκε υπολογιστική μηχανή που παρείχε η εταιρία Kaggle για την χρήση GPU για την εκπαίδευση των νευρωνικών δικτύων. Παρακάτω, αναφέρονται οι παροχές αυτής της μηχανής.

- Επεξεργαστής: Intel Xeon
- Frequency: 2.00 GHz
- Cache Size: 39 Mb
- GPU Size: 16gb
- Ram Size: 16 Gb
- Disk: 19.6 Gb

Στον πίνακα παρακάτω φαίνεται ο χρόνος εκτέλεσης κάθε διαδικασίας σε λεπτά.

Διαδικασία	**Λεπτά**
PCA – 95% variance	25
PCA – 2 συνιστώσες	12
t-SNE – 2 συνιστώσες	43
ML algos on PCA data – Bayesian Opt	194
DNN on PCA data – Bayesian Opt	3
Autoencoder – Bayesian Opt	29
Denoising Autoencoder -Bayesian Opt	28
Variational Autoencoder – Bayesian Opt	36
ML algos on Autoencoder data -Bayesian Opt	22
ML algos on Denoising Autoencoder data -Bayesian Opt	13
ML algos on Variational Autoencoder data -Bayesian Opt	5
DNN on Autoencoder data- Bayesian Opt	4
DNN on Denoising Autoencoder data- Bayesian Opt	17
DNN on Variational Autoencoder data- Bayesian Opt	3
Deep Cross Model on Autoencoder data- Bayesian Opt	3
Deep Cross Model on Denoising Autoencoder data- Bayesian Opt	2

Deep Cross Model on Variational Autoencoder data- Bayesian Opt	2
PCA on Variational Autoencoder Data	0.1
t-SNE on Variational Autoencoder data	0.5

Πίνακας 7.2: Ποσοτικά ποσοστά χρόνου μεταξύ των διαφορετικών αλγόριθμων.

Από την παρατήρηση των χρόνων φαίνεται ότι όσο πιο πολλά δεδομένα τόσο περισσότερο χρόνο χρειάστηκαν οι αλγόριθμοι μηχανικής μάθησης για να εκτελεστούν. Παραδείγματος χάρη, στα 2,210 χαρακτηριστικά χρειάστηκαν 3 ώρες για να ολοκληρωθεί η Μπεϋζιανή βελτιστοποίηση. Η βιβλιοθήκη Tensorflow παρέχει την δυνατότητα να εκτελεστούν τα νευρωνικά δίκτυα στην GPU αντί για την CPU οπότε αυτόματα η εκπαίδευση των δικτύων ήταν τουλάχιστον $\times 3$ πιο γρήγορη.

8. Μελλοντική έρευνα

Σε επίπεδο βιολογίας η προτεινόμενη μεθοδολογία μπορεί να επεκταθεί στην ανάλυση λειτουργικών μονοπατιών γονιδιακών δικτύων με τη χρήση εκφράσεων γονιδίων όπως ο αλγόριθμος **MinePath** αλλά και η χρήση του σε συνδυασμό με ολοκληρωμένες εφαρμογές υπολογιστικής βιολογίας[115,116]. Στην συνέχεια, για την ανάλυση που πραγματοποιήθηκε τα αποτελέσματα θα μπορούσαν να βελτιωθούν με αρκετούς τρόπους όπως την καλύτερη βελτιστοποίηση των υπερπαραμέτρων. Ένας απλός και ίσως ο πιο σημαντικός τρόπος βελτίωσης των αποτελεσμάτων είναι η συλλογή περισσοτέρων δειγμάτων. Επιπρόσθετα, ένας τρόπος θα ήταν να αυξηθεί το εύρος των τιμών που είχαν οριστεί για ορισμένες υπερπαραμέτρους. Για παράδειγμα, στην υπερπαράμετρο του **ποσοστού εκμάθησης** (**Learning Rate**) οι τιμές που είχαν οριστεί ήταν 0.001,0.0001,0.00001, ενώ θα μπορούσε να είχε οριστεί διαφορετικό εύρος τιμών από 0.1 – 0.00001. Σαφώς μεγαλύτερο εύρος τιμών σημαίνει και περισσότερη υπολογιστική πολυπλοκότητα. Ταυτόχρονα με την ένταξη μεγαλύτερου εύρους υπερπαραμέτρων θα μπορούσε να αυξηθεί και ο αριθμός των **μέγιστων**

[115] Lefteris Koumakis, Alexandros Kanterakis, Evgenia Kartsaki, Maria Chatzimina, Michalis Zervakis, Manolis Tsiknakis, Despoina Vassou, Dimitris Kafetzopoulos, Kostas Marias, Vassilis Moustakis, George Potamias, "MinePath: Mining for Phenotype Differential Sub-paths in Molecular Pathways", PLOS COMPUTATIONAL BIOLOGY, 10 November 2016, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005187 (Ανακτήθηκε 3/4/2021)

[116] Lefteris Koumakis, Panos Roussos, George Potamias, "minepath.org: a free interactive pathway analysis web server", Oxford Academic, 3 July 2017, https://academic.oup.com/nar/article/45/W1/W116/3744540 (Ανακτήθηκε 3/4/2021)

δοκιμών (**Max Trials**) στον αλγόριθμο Μπεϋζιανής βελτιστοποίησης, ώστε να υπάρξει η εύρεση μιας καλύτερης επιλογής υπερπαραμέτρων. Ακόμη, θα μπορούσαν να χρησιμοποιηθούν διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων για την διαδικασία της ταξινόμησης σε σχέση με τις δύο διαφορετικές αρχιτεκτονικές που χρησιμοποιήθηκαν, δηλαδή, την αρχιτεκτονική **Deep-Cross Networks** και ενός απλού **βαθύ νευρωνικού δικτύου** (**DNN**). Φυσικά, τα τεχνολογικά επιτεύγματα χρόνο με το χρόνο εξελίσσονται και ένα ιδιαίτερο είδος μηχανικής μάθησης που θα αποτελέσει μεγάλο παράγοντα ανάπτυξης είναι η **ενισχυτική διδασκαλία** (**Reinforcement Learning**). Σε αυτή, ένας πράκτορας μαθαίνει να καταλαβαίνει το περιβάλλον του μέσω της διαδικασίας της τιμωρίας και της επιβράβευσης σε συνδυασμό με την χρήση νευρωνικών δικτύων. Ένα ακόμα είδος που έχει αρχίσει να αποκτά ενδιαφέρον είναι η **αυτό-επιβλεπόμενη μάθηση** (**Self-Supervised Learning**), όπου η μηχανή δεν χρειάζεται τον ανθρώπινο παράγοντα ώστε να παρέχει δεδομένα με ετικέτες αλλά οι μηχανές αναλύουν, ταξινομούν και παράγουν ετικέτες μόνες τους.

9.Βιβλιογραφία

Ξενόγλωσση:

- Andriy Burkov, "The hundred page – Machine learning", January 2019
- Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow", O'Reilly Media, July 2019
- Chris Albon, "Python Machine Learning Cookbook", O'Reilly Media, April 2018
- Fabio Nelli, "Python Data Analytics- with Pandas, NumPy and Matplotlib", Second Edition, Apress, 2018
- Francois Chollet, "Deep Learning with Python", Manning Publications, 2018
- Gavin Hackeling, "Mastering Machine Learning with Scikit Learn", Second Edition, Packt, July 2017
- Ian Goodfellow, Yoshua Bengio, Aaron Courvilla, "Deep Learning", 2015
- Jonathan Pevsner, "Bioinformatics and functional genomics", Wiley Blackwell, Third Edition, Singapore 2015
- Neil C. Jones, Pavel A. Pevzner, "An introduction to Bioinformatics algorithms", Massachusetts Institute of technology, August 2004
- Prateek Joshi, "Artificial Intelligence with python", Packt Publishing, January 2017

Διαδικτυακές πηγές:

- Abhishek Sharma, "Decision Tree vs. Random Forest – Which Algorithm Should you Use", Analytics Vidhya, 12 May 2020, https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/ (Ανακτήθηκε 21/3/2021)
- Adrian Rosebrock, "Denoising autoencoders with Keras, Tensorflow, and Deep Learning",14 February 2020, https://www.pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/ (Ανακτήθηκε 16/3/2021)
- Alfredo Canziani, "Week 14- Practicum: Overfitting and Regularization, and Bayesian Neural Nets", YouTube, 25 September 2020, https://www.youtube.com/watch?v=DL7iew823c0 (Ανακτήθηκε 29/3/2021)
- Ali Hassan Sial, Syed Yahya, Shah Rashdi, Dr.Hafeez Khan, "Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python", February 2021 https://bit.ly/3vucEv6 (Ανακτήθηκε 16/3/2021)
- Amar Budhiraja, "Dropout in (Deep) Machine Learning", 15 December 2016, https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (Ανακτήθηκε 3/7/2021)
- Andre Violante, "An Introduction to t-SNE with Python Example", Towards Data Science,29 August 2018, https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1 (Ανακτήθηκε 21/3/2021)
- ANIRUDDHA BHANDARI," Feature Scaling for Machine Learning: Understanding the Difference Between the Normalization vs. Standardization", April 3,2020, https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/ (Ανακτήθηκε 3/7/2021)
- Behrooz Azarkhalili, Ali Saberi, Hamidreza Chitsaz, Ali Sharifi-Zarchi, "DeePathology: Deep Multi-Task Learning for Inferring Molecular Pathology from Cancer Transcriptome" **,** https://www.nature.com/articles/s41598-019-52937-5#Sec15 **(** Ανακτήθηκε 23/2/2021 **)**

- BioinformaticsHome, https://bioinformaticshome.com/tools/rna-seq/descriptions/RSEM.html (Ανακτήθηκε 3/7/2021)
- Bio-rad, «What is gene expression analysis», https://www.bio-rad.com/en-gr/applications-technologies/what-gene-expression-analysis?ID=LUSNINKSY (Ανακτήθηκε 11/2/2021)
- Bo Li & Colin N Dewey, "RSEM: accurate transcript quanitification from RNA-Seq data with or without a reference genome", August 4 2011 https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-323 (Ανακτήθηκε 21/1/2021)
- Broad Institute, Firehose http://gdac.broadinstitute.org/ (Ανακτήθηκε 16/1/2021)
- Cory Mitchell, "Bioinformatics", Investopedia, December 10, 2020, https://www.investopedia.com/terms/b/bioinformatics.asp (ανακτήθηκε: 15/1/2021)
- D.Urda, J.Montes-Torres, F.Moreno, L.Franco, J.M.Jerez, "Deep Learning to Analyze RNA-Seq Gene Expression Data", https://link.springer.com/chapter/10.1007/978-3-319-59147-6_5, 18 May 2017 (Ανακτήθηκε 15/2/2021)
- Dabal Pedamonti, "Comparison of non-linear activations functions for deep neural networks on MNIST classification Task", https://arxiv.org/pdf/1804.02763.pdf (Ανακτήθηκε 29/3/2021)
- Deepai, «Neural Network», https://deepai.org/machine-learning-glossary-and-terms/neural-network (Ανακτήθηκε 29/1/2021)
- Diedrik P. Kingma,Danilo J. Rezende,Shakir Mohamed,Max Welling, "Semi supervised learning with deep generative models" http://llcao.net/cu-deeplearning15/presentation/Semi-supervised%20Learning%20with%20Deep%20Generative%20Models.pdf (Ανακτήθηκε 22/1/2021)
- Gokul Elumalai, "Pros and Cons of common Machine Learning Algorithms", Medium, 19 November 2019, https://medium.com/@gokul.elumalai05/pros-and-cons-of-common-machine-learning-algorithms-45e05423264f (Ανακτήθηκε 21/3/2021)
- Google Developers," Training and Test Sets: Splitting Data", https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data (Ανακτήθηκε 7/3/2021)
- Great Learning Team, "What is Ridge Regression", Great Learning, 15 October 2020, https://www.mygreatlearning.com/blog/what-is-ridge-regression/ (Ανακτήθηκε 21/3/2021)
- https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/ (Ανακτήθηκε 21/1/2021)
- https://keras-team.github.io/keras-tuner/ (Ανακτήθηκε 14/3/2021)
- https://www.illumina.com/documents/products/datasheets/datasheet_hiseq2000.pdf
- Irhum Shafkat, "Intuively Understanding Variational Autoencoder", Towards Data Science, 4 February 2018, https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf (Ανακτήθηκε 20/3/2021)
- James Dellinger, "Weight Initialization in Neural Networks: A journey From the Basics to Kaiming", Towards Data Science, 3 April 2019, https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79 (Ανακτήθηκε 29/3/2021)
- Jason Brownlee, "A gentle Introduction to Batch Normalization for Deep Neural Networks", Machine Learning Mastery, 16 January 2019,

https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/ (Ανακτήθηκε 29/3/2021)

- Jason Brownlee, "A gentle Introduction to Dropout for Regularizing Deep Neural Networks", Machine Learning Mastery, 3 December 2018, https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (Ανακτήθηκε 28/3/3021)
- Jason Brownlee, "Extreme Gradient Boosting (XGBoost) Ensemble in Python", Machine Learning Mastery, 23 November 2020, https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/ (Ανακτήθηκε 21/3/2021)
- Jason Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning", Machine Learning Mastery, 3 July 2017, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (Ανακτήθηκε 22/3/2021)
- Jason Brownlee, "How to Calculate the SVD from Scratch with Python", Machine Learning Mastery, 26 February 2018, https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/ (Ανακτήθηκε 21/3/2021)
- Jason Brownlee, "How to Implement Bayesian Optimization from scratch in Python", Machine Learning Mastery, 9 Octomber 2019, https://machinelearningmastery.com/what-is-bayesian-optimization/ (Ανακτήθηκε 21/3/2021)
- Jason Brownlee, «SMOTE for imbalanced classification with python», January 17,2020 https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ (Ανακτήθηκε 2/3/2021)
- Jo Vandesompele, "Seven tips for bio-statistical analysis of gene expression data",11 December,2013, https://blog.qbaseplus.com/seven-tips-for-bio-statistical-analysis-of-gene-expression-data (Ανακτήθηκε 14/3/2021)
- Joseph Rocca «Understanding Variational Autoencoders (VAEs) », towards data science, 24 September 2019, https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 (Ανακτήθηκε 23/2/2021)
- Joshua Gould, Broad Institute, "Preprocess Dataset", https://genepattern.org/modules/docs/PreprocessDataset/5 (Ανακτήθηκε 14/3/2021)
- Julie Prost, "Hands on hyperparameters tuning with Keras Tuner", https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner (Ανακτήθηκε 14/3/2021)
- Khalid Salama, "Structured data learning with Wide, Deep and Cross Networks", 31/12/2020, https://keras.io/examples/structured_data/wide_deep_cross_networks/ (Ανακτήθηκε 19/3/2021)
- Lefteris Koumakis, Alexandros Kanterakis, Evgenia Kartsaki, Maria Chatzimina, Michalis Zervakis, Manolis Tsiknakis, Despoina Vassou, Dimitris Kafetzopoulos, Kostas Marias, Vassilis Moustakis, George Potamias, "MinePath: Mining for Phenotype Differential Sub-paths in Molecular Pathways", PLOS COMPUTATIONAL BIOLOGY, 10 November 2016, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005187 (Ανακτήθηκε 3/4/2021)
- Lefteris Koumakis, "Deep Learning models in genomics; are we there yet?", ScienceDirect,https://www.sciencedirect.com/science/article/pii/S2001037020303068 (Ανακτήθηκε 4/1/2021)

- Lefteris Koumakis, Panos Roussos, George Potamias, “minepath.org: a free interactive pathway analysis web server”, Oxford Academic, 3 July 2017, https://academic.oup.com/nar/article/45/W1/W116/3744540 (Ανακτήθηκε 3/4/2021)
- Lefteris Koumakis, Vassilis Moustakis, Michalis Zervakis, Dimitris Kafetzopoulos, George Potamias, “Coupling Regulatory Networks and Microarrays: Revealing Molecular Regulations of Breast Cancer Treatment Responses”, Springer Link, https://link.springer.com/chapter/10.1007/978-3-642-30448-4_30 (Ανακτήθηκε 3/4/2021)
- Mathworks, “What Deep Learning?”, “3 things you need to know”, https://www.mathworks.com/discovery/deep-learning.html (Ανακτήθηκε 11/2/2021)
- Mustafa Murat ARAT, “Weight Initialization Schemes – Xavier (Glorot) and He”, Mustafa Murat ARAT, 5 February 2019, https://mmuratarat.github.io/2019-02-25/xavier-glorot-he-weight-init (Ανακτήθηκε 29/3/2021)
- National Human Genome Research Institute, “A brief guide to genomics”, https://www.genome.gov/about-genomics/fact-sheets/A-Brief-Guide-to-Genomics (Ανακτήθηκε 21/1/2021)
- National Human Genome Research Institute, “Gene Expression”, https://www.genome.gov/genetics-glossary/Gene-Expression (Ανακτήθηκε 22/1/2021)
- Neetika Khandelwal, “A Brief Introduction to XGBoost”, Towards Data Science,7 July 2020, https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6 (Ανακτήθηκε 21/3/2021)
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Rusian Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf (Ανακτήθηκε 22/3/2021)
- Peter I. Frazier, “A tutorial on Bayesian Optimization”, Arxiv, 10 July 2018, https://arxiv.org/pdf/1807.02811.pdf (Ανακτήθηκε 21/3/2021)
- Pierre Baldi, Peter Sadowski, ”Understanding Dropout”, https://bit.ly/2PFaUP5 (Ανακτήθηκε 28/3/2021)
- Ritambhara Singh, Jack Lanchantin, Gabriel Robins, Yanjun Qi, “DeepChrome: deep-learning for predicting gene expression from histone modifications”, 29 August 2016, https://bit.ly/38R3Vtb (Ανακτήθηκε 17/3/2021)
- Rohith Gandhi, “Naïve Bayes Classifier”, Towards Data Science, 5 May 2018, https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (Ανακτήθηκε 21/3/2021)
- Rohith Gandhi, “Support Vector Machine – Introduction to Machine Learning Algorithms”, Towards Data Science, 7 June 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Ανακτήθηκε 21/3/2021)
- Ruairi J MacKenzie, “RNA-seq: Basics, Applications and Protocol”, Apr 06, 2018 https://www.technologynetworks.com/genomics/articles/rna-seq-basics-applications-and-protocol-299461 (Ανακτήθηκε στις 16/1/2021)
- Ruoxi Wang, Bin Fu, Gang Fu, Mingliang Wang, “Deep & Cross Network for Ad Click Predictions”, Arxiv, 17 August 2017, https://arxiv.org/pdf/1708.05123.pdf (Ανακτήθηκε 22/3/2021)
- Scikit Learn, “ Stochastic Gradient Descent ” (Ανακτήθηκε 21/3/2021)

- Scikit Learn, “Bernoulli Naïve Bayes”, https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html (Ανακτήθηκε 21/3/2021)
- Scikit Learn, “Naïve Bayes”, https://scikit-learn.org/stable/modules/naive_bayes.html (Ανακτήθηκε 22/3/2021)
- Scikit Learn, “Stratified K-Fold ”, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (Ανακτήθηκε 23/3/2021)
- Scikit Learn, ”Ridge Classifier”, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html (Ανακτήθηκε 21/3/2021)
- Scikit Learn, ”t-SNE ”, https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html (Ανακτήθηκε 21/3/2021)
- Sebai Dorsaf, “Comprehensive synthesis of the main activation functions pros and cons”, Medium, 2 April 2020, https://medium.com/analytics-vidhya/comprehensive-synthesis-of-the-main-activation-functions-pros-and-cons-dab105fe4b3b (Ανακτήθηκε 29/3/2021)
- SevenBridges, “The Cancer Genome Atlas” https://www.sevenbridges.com/tcga/ (Ανακτήθηκε στις 16/1/2021)
- Shan Suthaharan, “Support Vector Machine”, Springer Link, https://link.springer.com/chapter/10.1007/978-1-4899-7641-3_9 (Ανακτήθηκε 21/3/2021)
- Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry, “How does Batch Normalization Help Optimization”, https://arxiv.org/pdf/1805.11604.pdf (Ανακτήθηκε 29/3/2021)
- Shwetta Bhatt, “5 things you need to know about Reinforcemt learning”, https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html (Ανακτήθηκε 21/1/2021)
- Statquest with Josh Starmer ,”StatQuest:DESeq2, part1, Library Normalization”, 27 March 2017 https://www.youtube.com/watch?v=UFB993xufUU&list=PLblh5JKOoLUJo2Q6xK4tZElbIvAACEykp&index=13 (Ανακτήθηκε 14/3/2021)
- Sunil Ray, “6 Easy Steps To Learn Naïve Bayes Algorithm with codes in Python and R”, Analytics Vidhya, 11 September 2017, https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ (Ανακτήθηκε 21/3/2021)
- Sunpark, “It’s Deep Learning Times: A New Frontier of Data”, December 2019, https://towardsdatascience.com/its-deep-learning-times-a-new-frontier-of-data-a1e9ef9fe9a8 (Ανακτήθηκε 11/2/2021)
- TCGA, ”HiSeq 2000 System User Guide”,https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga/using-tcga/technology/illumina-hiseq2000-data-sheet (Ανακτήθηκε 7/3/2021)
- University of California Santa Cruz, https://tumormap.ucsc.edu/
- Will Koehrsen, “A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning”, Towards Data Science, 24 June 2018, https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-

hyperparameter-optimization-for-machine-learning-b8172278050f (Ανακτήθηκε 21/3/2021)

- YourGenome, “What is gene expression”, https://www.yourgenome.org/facts/what-is-gene-expression (Ανακτήθηκε 11/2/2021)
- Yulia Newton, Adam M. Novak, Teresa Swatloski, Duncan C. McColl, Sahil Chopra, Kiley Graim, Alana S. Weinstein, Robert Baertsch, Sofie R. Salama, Kyle Ellrott, Manu Chopra, Theodore C. Goldstein, David Haussler, Olena Morozova, Joshua M. Stuart, “ TumorMap: Exploring the Molecular Similarities of Cancer Samples in an Interactive Portal”, NCBI, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5751940/ (Ανακτήθηκε 30/3/2021)
- Τριανταφυλλίδης Α, ”Ειδικά θέματα γενετικής” https://opencourses.auth.gr/modules/document/file.php/OCRS146/%CE%A0%CE%B1%CF%81%CE%BF%CF%85%CF%83%CE%B9%CE%AC%CF%83%CE%B5%CE%B9%CF%82/%CE%95%CE%BD%CF%8C%CF%84%CE%B7%CF%84%CE%B1%2001%3A%20%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE%20%CF%83%CF%84%CE%B7%20%CE%B3%CE%BF%CE%BD%CE%B9%CE%B4%CE%B9%CF%89%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AE.pdf (Ανακτήθηκε 21/1/2021)