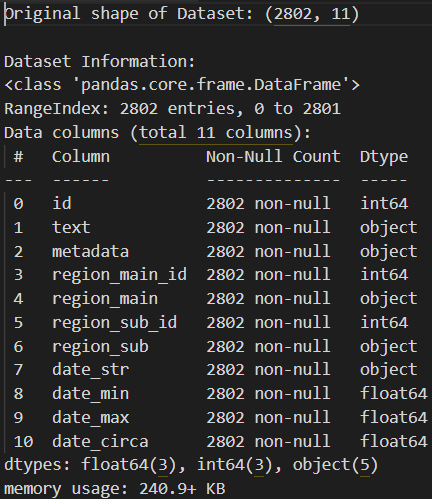
**Α1. Προεπεξεργασία και Προετοιμασία δεδομένων:**

**α)** Η διαδικασία της προεπεξεργασίας των δεδομένων μπορεί να βρεθεί στο αρχείο **preprocessing.py**. Αρχικά, φορτώνουμε το σύνολο δεδομένων σε ένα pandas dataframe, χρησιμοποιώντας ως delimiter το tab. Έπειτα, πραγματοποιούμε κάποιους ελέγχους στο σύνολο δεδομένων, εκμεταλλευόμενοι τις δυνατότητες της βιβλιοθήκης pandas. Διακρίνουμε το σχήμα του dataframe, καθώς και τις διαφορετικές του στήλες. Έπειτα ελέγχουμε για τυχόν μη-υπάρχουσες ή μη ορισμένες τιμές σε κάθε στήλη στο dataset. Αυτός ο έλεγχος γίνεται προκειμένου να εξασφαλίσουμε την ακεραιότητα του συνόλου δεδομένων, καθώς απούσες ή μη καλά ορισμένες τιμές στα γεγονότα, μπορεί να οδηγήσουν σε μεγάλα σφάλματα κατά την εκπαίδευση του νευρωνικού μας δικτύου. Έπειτα, βλέπουμε το σύνολο διαφορετικών τιμών για κάθε στήλη του dataset. Έτσι μπορούμε να καταλάβουμε περισσότερα πράγματα για το σύνολο δεδομένων με το οποίο εργαζόμαστε. Τα αποτελέσματα της εντολής φαίνονται παρακάτω:



Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κατάλογος, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Σειρά έχει η υλοποίηση του **Bag-of-Words (BoW)** Μοντέλου. Το μοντέλο αυτό είναι ιδανικό για την διαχείριση text-based συνόλου δεδομένων, δημιουργεί ένα λεξικό από όλες τις διαφορετικές λέξεις που υπάρχουν σε αυτό. Εδώ οι λέξεις του λεξικού, τα λεγόμενα **tokens**, είναι **unigrams**, δηλαδή κάθε token είναι μία μοναδική λέξη. Έτσι στην περίπτωση μας, το μοντέλο δεν καταμετρά την θέση, την γραμματική ή την δομή των προτάσεων που απαρτίζουν το σύνολο δεδομένων μας, παρά καταμετρά μόνο συχνότητες εμφανίσεων λέξεων, χρησιμοποιώντας μία συγκεκριμένη τεχνική που ονομάζεται **tf-idf vectorization (Term frequency — Inverse document frequency)**. H τεχνική αυτή μας επιτρέπει να ταυτοποιήσουμε τις πιο σημαντικές λέξεις στο κείμενο. Αποτελείται από δύο ποσότητες:

1. **Term Frequency (Tf):** Μας δίνει το πλήθος των εμφανίσεων ενός token που περιλαμβάνεται σε ένα document και υπολογίζεται ως εξής:

**Εικόνα που περιέχει κείμενο, γραμματοσειρά, γραμμή, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα**

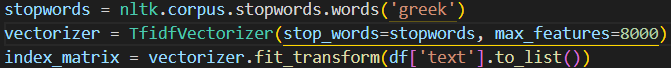
1. **Inverse Document Frequency (Idf):** Μας δίνει την μοναδικότητα ενός token, δεδομένου του συνόλου των documents. Η κεντρική ιδέα με το συγκεκριμένο index είναι ότι μία λέξη η οποία εμφανίζεται αρκετά συχνά στα περισσότερα documents του συνόλου δεδομένων, δεν αποδίδει ιδιαίτερη πληροφορία σχετικά με το εκάστοτε document. Η συγκεκριμένη ποσότητα υπολογίζεται ως εξής:

**Εικόνα που περιέχει γραμματοσειρά, κείμενο, γραμμή, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα**

Η τελική βαθμολογία tf-idf προκύπτει πολλαπλασιάζοντας τις δύο παραπάνω ποσότητες, προσδίδοντας μία τιμή σε κάθε token του λεξικού, ευνοώντας τα πιο σπάνια tokens.

Για την υλοποίηση του συγκεκριμένου μοντέλου χρησιμοποιήθηκε η βιβλιοθήκη **scikit-learn** και συγκεκριμένα η κλάση **TfidfVectorizer** της. Προτιμήθηκε να χρησιμοποιηθεί λίστα με συγκεκριμένα **stopwords**, προκειμένου να φιλτραριστούν από το λεξικό λέξεις οι οποίες είναι συχνά επαναλαμβανόμενες, ωστόσο δεν προσδίδουν κάποια χρήσιμη πληροφορία για το εκάστοτε κείμενο, όπως οι συνδετικές λέξεις. Συγκεκριμένα με τις παρακάτω εντολές, δημιουργούμε ένα αντικείμενο της κλάσης TfidfVectorizer και βρίσκουμε το tfidf μητρώο του λεξικού από τα κείμενα το συνόλου δεδομένων:



Το αντικείμενο της κλάσης TfidfVectorizer, παίρνει ως ορίσματα:

1. **Stop\_words = stopwords:** Μία λίστα από stopwords, την οποία πήραμε από την βιβλιοθήκη **nltk.** Η λίστα αυτή έχει 256 διαφορετικές λέξεις, πολλές από τις οποίες είναι αρχαίες ελληνικές, πράγμα το οποίο είναι αρκετά χρήσιμο για το συγκεκριμένο σύνολο δεδομένων.
2. **Max\_features = 8000:** Καθορίζουμε τον αριθμό των unigrams που θέλουμε να έχει το λεξικό μας. Οι λέξεις καθορίζονται με βάση το tf (term frequency), δηλαδή την συχνότητα εμφάνισης τους σε κάθε κείμενο. Έτσι, έχοντας αποβάλλει λέξεις άχρηστες για την ανάλυση που θέλουμε να πραγματοποιήσουμε, μας μένουν λέξεις οι οποίες είναι συχνά εμφανιζόμενες, ωστόσο έχουν χρήσιμη πληροφορία σχετικά με το εκάστοτε κείμενο στο οποίο βρίσκονται.

Έτσι προκύπτει ένα csr() μητρώο, του οποίου οι γραμμές είναι τα διάφορα documents του συνόλου δεδομένων και οι στήλες οι λέξεις του λεξικού. Στην i-οστή γραμμή αντιστοιχεί Η i-οστή επιγραφή του συνόλου δεδομένων, ενώ στην i-οστή στήλη αντιστοιχεί η i-οστή λέξη του λεξικού. Τιμή στο αντίστοιχο κελί υπάρχει όταν η λέξη σε εκείνη την στήλη βρίσκεται στην αντίστοιχη επιγραφή και η τιμή της αντιστοιχεί στην tf-idf τιμή της λέξης αυτής. Πράγματι, κρίνοντας από το σχήμα του μητρώου, βλέπουμε ότι τα αποτελέσματά μας είναι ακριβή:



Έπειτα, βρίσκουμε τις idf τιμές καθενός από τις λέξεις που υπάρχουν στο λεξικό μας, καθώς αυτές οι τιμές μας δίνουν μία εικόνα για την υπόθεση που κάναμε παραπάνω. Δεν θέλουμε ούτε η τιμή να είναι υψηλή, συνεπώς η λέξη να συναντάται σπάνια στο σύνολο των επιγραφών, ούτε πάρα πολύ χαμηλή, πράγμα που θα δήλωνε ότι συναντάται σε όλα τα κείμενα, οπότε δεν θα μας έδινε καμία χρήσιμη πληροφορία για την εκάστοτε περιγραφή. Παρακάτω, φαίνονται οι πρώτες και οι τελευταίες 10 τιμές του πίνακα idf\_values:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Έτσι καταφέρνουμε να βρούμε λέξεις με σπουδαιότητα, χωρίς να είναι ούτε πολύ σπάνιες, ούτε πάρα πολύ συχνές.

**β)** Παρ’ ότι οι τιμές των vectors βρίσκονται σχετικά κοντά μεταξύ τους, επιλέγουμε να κάνουμε κανονικοποίηση στο εύρος [0,1], τόσο στα vectors, δηλαδή τα δεδομένα εισόδου του μετέπειτα νευρωνικού που θα αναπτύξουμε, όσο και στα δεδομένα εξόδου, δηλαδή στις τιμές της στήλης mean\_date του dataframe που δημιουργήσαμε στην αρχή το τμήματος της προεπεξεργασίας. Η κανονικοποίηση επιλέγεται να εφαρμοστεί για πολλούς διαφορετικούς λόγους, κάποιοι εκ των οποίων είναι:

1. Αποφεύγεται η κυριαρχία κάποιων τιμών, έναντι κάποιων άλλων, επηρεάζοντας έτσι την διαδικασία της μάθησης και οδηγώντας σε biased αποτελέσματα.
2. Οι μικρές τιμές των δεδομένων εισόδου, επιταχύνουν την σύγκλιση της διαδικασίας εισόδου, οδηγώντας στην κατασκευή ενός σταθερού νευρωνικού δικτύου.
3. Τα δεδομένα οπτικοποιούνται ευκολότερα, καθώς οι μεταξύ τους αποστάσεις μειώνονται.

Στην παρούσα εργασία, χρησιμοποιείται, όπως προειπώθηκε, min-max scaling στο πεδίο [0,1] τόσο για τα δεδομένα εισόδου, όσο και εξόδου. Η φόρμουλα η οποία χρησιμοποιείται είναι η εξής:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, λευκό, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Χρησιμοποιείται ακόμα μία φορά η βιβλιοθήκη **sklearn,** από την οποία χρησιμοποιούμε την κλάση **MinMaxScaler,** για την κανονικοποίηση των δεδομένων μας. Αυτή η διαδικασία γίνεται με τις παρακάτω εντολές:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, γραφικά

Περιγραφή που δημιουργήθηκε αυτόματα

,όπου texts τα δεδομένα εισόδου του νευρωνικού και dates τα δεδομένα εξόδου του. Τα αποτελέσματα που προκύπτουν μετά την κανονικοποίηση (ενδεικτικά), φαίνονται παρακάτω:

**Δεδομένα εισόδου:**

Εικόνα που περιέχει στιγμιότυπο οθόνης, πληκτρολόγιο, υπολογιστής

Περιγραφή που δημιουργήθηκε αυτόματα

,το οποίο όπως εξηγήθηκε και παραπάνω είναι ένα sparse μητρώο με τις κανονικοποιημένες tf-idf τιμές των λέξεων που επιλέχθηκαν.

**Δεδομένα εξόδου:**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα

, όπου παρατηρούμε ότι οι τιμές είναι καταρχάς όλες θετικές και πολύ πιο κοντά μεταξύ τους από τις αρχικές, που βρίσκονταν στο εύρος [-720, 1453].

**γ)** Το cross validation, και συγκεκριμένα το **k-fold cross validation**, είναι μία τεχνική που αξιολογεί την συμπεριφορά ενός νευρωνικού δικτύου σε ένα ανεξάρτητο σύνολο δεδομένων. Βασική ιδέα της συγκεκριμένης τεχνικής, είναι ο διαχωρισμός του συνόλου δεδομένων σε k πολλαπλά διακριτά υποσύνολα (folds), ένα εκ των οποίων ορίζεται ως σύνολο επικύρωσης (validation set). Η διαδικασία αυτή επαναλαμβάνεται k φορές, με διαφορετικό σύνολο επικύρωσης κάθε φορά και η τελική απόδοση του μοντέλου εκτιμάται ως ο μέσος όρος των αποδόσεων των k επαναλήψεων. Η συγκεκριμένη τεχνική χρησιμοποιείται ευρέως, καθώς εξασφαλίζει ότι το σύνολο δεδομένων με το οποίο εκπαιδεύεται και αξιολογείται αλλάζει διαρκώς, αποφεύγοντας έτσι προβλήματα υπερπροσαρμογής του μοντέλου στα δεδομένα. Χρησιμοποιείται για ακόμα μία φορά η βιβλιοθήκη **sci-kit learn** Και συγκεκριμένα η **KFold** κλάση της, καθώς ζητάται να γίνει **5-fold Cross Validation**. Η αρχικοποίηση και παραμετροποίηση φαίνεται από τη παρακάτω εντολή:



Χρησιμοποιώντας την εντολή kf.split(X) δημιουργούμε τα 5 folds και έπειτα για κάθε ένα fold παίρνουμε τους δείκτες που χωρίζουν τα δεδομένα σε train και test sets. Συνεπώς στο τέλος δημιουργούμε μία λίστα από λεξικά, όπου το κάθε λεξικό περιέχει πληροφορία για το ποιο fold είναι, τα training και test sets των εισόδων και των targets, καθώς και τις μέσες τιμές κάθε γραμμής των y\_train και y\_test (των δύο ημερομηνιών). Έτσι έχουμε έτοιμα τα δεδομένα που θα εισάγουμε στο νευρωνικό δίκτυο για κάθε fold.

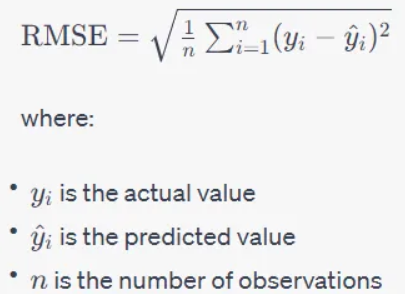
**Α2. Επιλογή Αρχιτεκτονικής**

Για την κατασκευή του νευρωνικού δικτύου χρησιμοποιήθηκε η βιβλιοθήκη **tensorflow** και συγκεκριμένα το API της, το **Keras**. Θεωρήθηκε η καλύτερη επιλογή, δεδομένου ότι προσφέρει ικανοποιητική λειτουργικότητα, χωρίς να χρειάζεται πολύπλοκη υλοποίηση. Υλοποιήθηκαν custom συναρτήσεις για:

1. Την παραμετροποίηση του νευρωνικού δικτύου, όπως τον καθορισμό του αριθμού των κρυφών επιπέδων, του αριθμού των κόμβων ανά κρυφό επίπεδο και του ρυθμού μάθησης.
2. Την μέτρηση σφάλματος μεταξύ των προβλέψεων και των targets στην έξοδο του νευρωνικού δικτύου, με την τεχνική του RMSE (Root Mean Square Error).
3. Την εκπαίδευση του νευρωνικού δικτύου, με τον καθορισμό του αριθμού των batches και των εποχών της εκπαίδευσης.
4. Τον έλεγχο της μάθησης του νευρωνικού δικτύου, μέσω της ερμηνείας του σφάλματος του σε ξεχωριστό κομμάτι των δεδομένων.

Χρησιμοποιείται o **optimizer Adam** για την ελαχιστοποίηση της συνάρτησης σφάλματος, καθώς είναι αρκετά καλύτερος από άλλε εναλλακτικές, λόγω της γρήγορης σύγκλισής του ιδιαίτερα σε υψηλής διαστατικότητας διανυσματικούς χώρους. Ο Adam συνδυάζει τις τεχνικές του Momentum και του RMSprop(Root Mean Square Propagation), προκειμένου να υπολογίσει προσαρμοστικά το ρυθμό μάθησης για κάθε μία παράμετρο βασιζόμενος στη κλίμακα των προηγούμενων gradients.

**a)** Θεωρήσαμε την μέση τιμή των δύο ημερομηνιών ως το target value της εξόδου του νευρωνικού δικτύου, και υλοποιήσαμε την συνάρτηση **custom\_RMSE,** η οποία υπολογίζει το root mean square error των δύο διανυσμάτων, του y\_pred, δηλαδή της εξόδου του νευρωνικού και του y\_true δηλαδή των target values, σύμφωνα με τον τύπο:



Η υλοποίηση της συνάρτησης φαίνεται παρακάτω:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

**β)** Η συνάρτηση ενεργοποίησης ενός νευρωνικού δικτύου καθορίζει πώς ο γραμμικός συνδυασμός των βαρών και της εισόδου μετασχηματίζεται στην έξοδο ενός νευρώνα ή ενός επιπέδου γενικότερα. Η επιλογή συνάρτησης ενεργοποίησης έχει μεγάλη επίπτωση στη ικανότητα και την επίδοση του νευρωνικού δικτύου και γι’ αυτό το λόγο πολλές φορές χρησιμοποιούνται διαφορετικές συναρτήσεις σε διάφορα σημεία του δικτύου. Ιδιαίτερα για τα κρυφά επίπεδα, συνηθίζεται να χρησιμοποιείται η ίδια συνάρτηση ενεργοποίησης για κάθε επίπεδο, η οποία είναι βασικό να είναι διαφορίσιμη και μη γραμμική, προκειμένου να μπορεί να υπολογιστεί η πρώτη παράγωγος που χρησιμοποιείται στην διαδικασία του back propagation για την ανανέωση των τιμών των λαθών, καθώς και να μπορέσει το νευρωνικό δίκτυο να μάθει πιο πολύπλοκες σχέσεις μεταξύ των δεδομένων. Οι πιο συνήθεις συναρτήσεις ενεργοποίησης είναι:

1. **Rectified Linear Activation Function (ReLU):** Είναι η πιο συνηθισμένη συνάρτηση ενεργοποίησης, καθώς είναι και εύκολα υλοποιήσιμη και αποδοτικότερη από τις άλλες προηγουμένως δημοφιλείς συναρτήσεις ενεργοποίησης όπως η σιγμοειδής και η υπερβολική εφαπτομένη. Συγκεκριμένα, είναι λιγότερο επιρρεπής σε προβλήματα όπως το vanishing gradients, κατά το οποίο αλλαγές της κλίσης κοντά στην έξοδο του νευρωνικού, αδυνατούν να διαδοθούν προς τα πίσω, στα επίπεδα κοντά στην είσοδο του νευρωνικού. Παρ’ όλο που η reLU δεν είναι αυστηρά διαφορίσιμη σε όλο το πεδίο ορισμού της λόγω της ξαφνικής της αλλαγής στο 0, η παράγωγός της σε αυτό το σημείο μπορεί να θεωρηθεί 0, ενώ για όλα τα υπόλοιπα σημεία, ο υπολογισμός της είναι αρκετά εύκολος. Ωστόσο η reLU πάσχει από άλλα προβλήματα όπως η νέκρωση των κόμβων, το οποίο είναι ένα φαινόμενο που συμβαίνει όταν η αθροισμένη είσοδος σε έναν νευρώνα είναι μονίμως αρνητική, άρα και η έξοδος του ίση με μηδέν. Ωστόσο παραλλαγές της reLU έχουν σχεδιαστεί προκειμένου να αντιμετωπίσουν το συγκεκριμένο πρόβλημα, όπως η Parametric reLU, όπου η συνάρτηση στο αρνητικό κομμάτι του πεδίου ορισμού, έχει κλίση η οποία καθορίζεται από μία παράμετρο, προκειμένου να αντιμετωπιστεί το πρόβλημα της νέκρωσης των κόμβων.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

1. **Sigmoid (Logistic) Activation Function:** Η συγκεκριμένη συνάρτηση λαμβάνει μία οποιαδήποτε πραγματική τιμή και σαν αποτέλεσμα επιστρέφει μία τιμή μεταξύ 0 και 1. Όσο μεγαλύτερη είναι η είσοδος, τόσο πιο κοντά είναι η τιμή στο 1 και αντίστροφα. ‘Όταν χρησιμοποιείται η συγκεκριμένη συνάρτηση ενεργοποίησης, καλό είναι η αρχικοποίηση των βαρών να γίνεται χρησιμοποιώντας την Xavier Normal και να πραγματοποιείται κανονικοποίηση των δεδομένων εισόδου στο εύρος [0,1]. Η γραφική παράσταση της σιγμοειδούς συνάρτησης φαίνεται παρακάτω:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

1. **Tanh (hyperbolic tangent) Activation Function:** Η συγκεκριμένη συνάρτηση ενεργοποίησης είναι αρκετά παρόμοια με την σιγμοειδή, με την μόνη διαφορά τους να είναι ότι το σύνολο τιμών της συνάρτησης κυμαίνονται στο [-1,1]. Εδώ προτιμάται η κανονικοποίηση των δεδομένων εισόδου στο διάστημα [-1,1]. Η γραφική παράσταση της συνάρτησης φαίνεται παρακάτω:Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, γραμμή

   Περιγραφή που δημιουργήθηκε αυτόματα

Γενικά λόγω προβλημάτων όπως το vanishing gradients, πλέον συναρτήσεις ενεργοποίησης όπως η tanh και η σιγμοειδής έχουν ξεπεραστεί και χρησιμοποιούνται μόνο στα RNNs, όπως στα μοντέλα LSTM. Ιδιαίτερα για τον τύπο δικτύου της συγκεκριμένης εργασίας (MLP), χρησιμοποιείται η reLU, λόγω της απλότητας και των καλών αποδόσεων που έχει και που αναλύθηκαν εκτενώς παραπάνω.

**γ)** Δεδομένου ότι το πρόβλημα που επιλύουμε με το νευρωνικό δίκτυο είναι πρόβλημα γραμμικής παλινδρόμησης, το επίπεδο εξόδου του νευρωνικού μας δικτύου θα έχει την γραμμική συνάρτηση ενεργοποίησης, καθότι το νευρωνικό θα πρέπει να είναι σε θέση να παράξει οποιαδήποτε τιμή ανήκει στο διάστημα [0,1]. Η γραμμική συνάρτηση ενεργοποίησης διατηρεί το εύρος των πραγματικών τιμών, επιτρέποντας στο νευρωνικό δίκτυο να προβλέψει ένα ευρύ φάσμα συνεχών εξόδων, συνεπώς είναι η κατάλληλη επιλογή, ενώ δεν χρειάζεται να προβεί σε κανένα μετασχηματισμό, λόγω της κανονικοποίησης των target\_values.

**δ)** Υπενθυμίζεται σε αυτό το σημείο, ότι το 5-fold cross validation, έχει εφαρμοστεί σε όλο το σύνολο δεδομένων. Έτσι, η εκπαίδευση του δικτύου για κάθε fold γίνεται με το training υποσύνολο για τον αριθμό των εποχών που ορίζεται και μετέπειτα γίνεται το testing με το testing υποσύνολο του συνόλου δεδομένων. Παρακάτω, φαίνεται το διάγραμμα της μέσου σφάλματος εκπαίδευσης ανά εποχή, για 50 εποχές και για νευρωνικά με ένα κρυφό επίπεδο των 1000, 750, 500 και 250 κόμβων αντίστοιχα:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Από το παραπάνω διάγραμμα βλέπουμε ότι το νευρωνικό δίκτυο 4, αυτό με τους 250 κόμβους στο κρυφό επίπεδο, είχε το μικρότερο μέσο σφάλμα ανά εποχή αλλά με μέσο σφάλμα ελέγχου 0,07, σε σχέση με τα 0,06933 και 0,0688των νευρωνικών δικτύων 3, 1 και 2 αντίστοιχα. Παρακάτω βλέπουμε και το διάγραμμα του σφάλματος ελέγχου για κάθε fold και για κάθε νευρωνικό δίκτυο:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εδώ διακρίνουμε ότι το νευρωνικό δίκτυο με το χαμηλότερο σφάλμα ανά εποχή κατά την εκπαίδευση, παρουσιάζει φαινομενικά υψηλότερο σφάλμα ανά fold, πράγμα το οποίο θα μπορούσε να σημαίνει ότι υπόκεινται σε underfitting, δηλαδή ότι το μοντέλο μας είναι αρκετά απλοϊκό για να μπορέσει να μάθει τα μοτίβα των δεδομένων του συνόλου εκπαίδευσης κάθε φορά, οδηγώντας σε κακή γενίκευση στο σύνολο ελέγχου.

Διακρίνουμε ακόμα ότι το σφάλμα ελέγχου κατά μέσο όρο είναι αρκετά μεγαλύτερο του μέσου σφάλματος εκπαίδευσης ακόμα και μετά την σύγκλιση του σφάλματος των μοντέλων μετά από 30 περίπου εποχές, πράγμα το οποίο θα μπορούσε να σημαίνει μία ελαφριά υπερπροσαρμογή, πάνω στα δεδομένα εκπαίδευσης. Ο λόγος που το νευρωνικό με τους λιγότερους κόμβους στο κρυφό επίπεδο συμπεριφέρεται καλύτερα, δηλαδή συγκλίνει πιο γρήγορα με μικρότερο σφάλμα εκπαίδευσης και ελέγχου, μπορεί να οφείλεται σε αρκετούς λόγους, μερικοί από τους οποίους είναι οι ακόλουθοι:

1. Τα μεγαλύτερα νευρωνικά δίκτυα με περισσότερους κόμβους στο κρυφό επίπεδο ενδέχεται να εμφανίζουν φαινόμενα υπερπροσαρμογής. Γενικά η υπερπροσαρμογή συμβαίνει όταν το μοντέλο μαθαίνει υπερβολικά καλά τα δεδομένα εκπαίδευσης και δεν γενικεύεται καλά για καινούργια δεδομένα.
2. Τα μεγαλύτερα δίκτυα, έχουν περισσότερες παραμέτρους να βελτιστοποιήσουν, οδηγώντας έτσι σε μεγαλύτερη πολυπλοκότητα, κάτι το οποίο εξηγεί την πιο αργή σύγκλιση των μεγαλύτερων μοντέλων.
3. Το μικρότερο δίκτυο, λόγω των λιγότερων παραμέτρων που πρέπει να βελτιστοποιηθούν, μαθαίνει καλύτερα γενικευμένα μοτίβα, ωστόσο όπως προ είπαμε υπάρχει ο κίνδυνος να μην μπορεί να μάθει αρκετά εξελιγμένα μοτίβα προκειμένου να μπορεί να κάνει ικανοποιητικές προβλέψεις μετέπειτα.

Με αυτά κατά νου, και έπειτα από την εκπαίδευση με περισσότερους κόμβους στο κρυφό επίπεδο (2.000-3.000), θεωρείται ότι το πρώτο νευρωνικό επίπεδο παρουσιάζει ικανοποιητικά αποτελέσματα, αφού όχι μόνο συγκλίνει στο 0.046 περίπου μέσο σφάλμα μετά από 30 εποχές, αλλά παρουσιάζει και το μικρότερο σφάλμα στα δεδομένα ελέγχου που δηλώνει καλό σχετικά generalization στα δεδομένα. Βεβαίως, για να διαπιστωθεί ενδελεχώς η επίδοση του μοντέλου, πρέπει να ελεγχθεί με υποσύνολο δεδομένων που δεν έχει πρότινος εκπαιδευτεί, κάτι που ωστόσο δεν γίνεται καθώς το σύνολο δεδομένων είναι σχετικά μικρό για τον διαχωρισμό του σε ξεχωριστά σύνολα ελέγχου και εκπαίδευσης.

**ε)** Γενικά προσθέτοντας επίπεδα σε ένα νευρωνικό δίκτυο, του επιτρέπεται να συλλάβει πιο περίπλοκα μοτίβα και σχέσεις μεταξύ των δεδομένων με τα οποία εκπαιδεύεται. Ωστόσο αυτή η πρακτική δεν είναι πάντα βέλτιστη, καθώς τέτοια δίκτυα για απλά ή περιορισμένης έκτασης σύνολα δεδομένων, μπορεί να οδηγηθούν σε υπερπροσαρμογή και άρα σε κακή απόδοση. Οι πρακτικές που ακολουθούνται για την επιλογή αριθμού πολλαπλών κρυφών επιπέδων καθώς και αριθμού κόμβων ανά επίπεδο είναι οι ακόλουθες:

1. **Downscaling**: Η πιο συνηθισμένη τακτική είναι η διαδοχική μείωση των κόμβων σε κάθε κρυφό επίπεδο καθώς μετακινούμαστε πιο «βαθιά» στο νευρωνικό δίκτυο. Η συγκεκριμένη τεχνική είναι γνωστή και ως downscaling και στόχος της είναι να δημιουργήσει ένα funnel effect, όπου το νευρωνικό εστιάζει σταδιακά σε όλο και πιο αφηρημένα μοτίβα πάνω στα δεδομένα.
2. **Upscaling**: Μία άλλη πρακτική, που είναι ωστόσο πιο σπάνια, είναι η σταδιακή επέκταση των κρυφών επιπέδων, γνωστή και ως upscaling. Ωστόσο η συγκεκριμένη είναι ακριβή υπολογιστικά και κοστίζει και σε μνήμη.
3. **Constant Size**: Τελευταία πρακτική που μπορεί να εφαρμοστεί συνήθως είναι η διατήρηση του μεγέθους των κρυφών επιπέδων, όπου είναι η πιο εύκολη πρακτική και δουλεύει καλά για κάποια σενάρια, ωστόσο δεν κάνει πάντα καλή γενίκευση στα δεδομένα εισόδου.

Μετά από μελέτη της βιβλιογραφίας, θεωρείται ότι η καλύτερη πρακτική να εφαρμοστεί είναι το downscaling, προκειμένου να επιτρέψουμε στο νευρωνικό δίκτυο να μάθει πιο αφηρημένα μοτίβα στα δεδομένα. Γενικά δεν υπάρχει κάποιος ενδεικτικός κανόνας για τον ακριβή καθορισμό κρυφών επιπέδων ή κόμβων ανά κρυφό επίπεδο, ωστόσο οι πιο συνήθεις πρακτικές είναι οι ακόλουθες:

1. **Trial and Error:** Η πιο συνηθισμένη τακτική κυρίως για μικρά προβλήματα που επιλύονται με την χρήση νευρωνικών είναι η συνεχής δοκιμή διαφόρων τιμών των υπερπαραμέτρων των μοντέλων και αξιολόγηση τους έως ότου τα αποτελέσματα που εξάγονται είναι ικανοποιητικά. Μολονότι η συγκεκριμένη τεχνική μπορεί να μην εξάγει το καλύτερο δίκτυο για το εκάστοτε πρόβλημα, για προβλήματα όπως το δικό μας, θεωρείται αρκετά αποδεκτή. Ωστόσο για χάριν σαφήνειας παρουσιάζονται και άλλες εναλλακτικές που θα μπορούσαν να είχαν ακολουθηθεί.
2. **Heuristic Search:** Είναι η εύρεση της καταλληλότερης τοπολογίας με βάση γνώση που αποκτήθηκε από προηγούμενα πειράματα που έχουν σχεδόν βέλτιστη τοπολογία για το πρόβλημα που επιλύουν. Τέτοιες ευρετικές συνήθως μας δίνουν ένα σημείο εκκίνησης προκειμένου μετέπειτα με συνεχείς δοκιμές (trial and error) να μπορέσουμε να προσεγγίσουμε την βέλτιστη τοπολογία για το εκάστοτε πρόβλημα.
3. **Exhaustive Search:** Ίσως η πιο μη-βιώσιμη από όλες τις μεθόδους που παρουσιάζονται, όχι από την άποψη του υπολογιστικού κόστους που μία τέτοια έρευνα συνεπάγεται, αλλά για τον χρόνο που απαιτείται για την αξιολόγηση του καθενός μοντέλου ξεχωριστά. Ακόμη, σχεδόν σίγουρα δεν απαιτείται η μία μόνο εκτέλεση των μοντέλων για να βρεθεί το βέλτιστο.
4. **Pruning and Constructive Algorithms:** Τέτοιοι αλγόριθμοι στοχεύουν στην κατασκευή κατάλληλων δικτύων, μειώνοντας ή αυξάνοντας συνάψεις μεταξύ κόμβων, αναλόγως αν το δίκτυο έχει πληθώρα συνάψεων ή έλλειψη αντίστοιχα.
5. **Genetic Algorithms and Neural Networks:** Το πρόβλημα της εύρεσης της βέλτιστης αρχιτεκτονικής ενός νευρωνικού δικτύου για την επίλυση ενός προβλήματος μοιάζει πάρα πολύ με αντίστοιχα προβλήματα που επιλύουν οι γενετικοί αλγόριθμοι. Τ επίπεδα και οι αριθμοί κόμβων του καθενός επιπέδου κωδικοποιούνται κατάλληλα σε χρωμοσώματα, εκπαιδεύονται σε κάποιο task και έπειτα επιλέγονται οι δυνατότεροι υποψήφιοι. Αυτοί διασταυρώνονται μεταλλάσσονται και στο τέλος μας δίνεται η καταλληλότερη αρχιτεκτονική για το τύπο προβλήματος που έχουμε να επιλύσουμε.