Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Computational Intelligence, Project 1st :

Neural Networks

Εκπονήθηκε από τον:

Καπογιάννη Κωσταντίνο Α.Μ. 1072521

GitHub Link:

https://github.com/kwnkapog/Computational-Intelligence-Project

Πάτρα 2024

A1. 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:

```
Original shape of Dataset: (2802, 11)

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2802 entries, 0 to 2801
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              2802 non-null   int64
 1   text            2802 non-null   object
 2   metadata        2802 non-null   object
 3   region_main_id  2802 non-null   int64
 4   region_main     2802 non-null   object
 5   region_sub_id   2802 non-null   int64
 6   region_sub      2802 non-null   object
 7   date_str        2802 non-null   object
 8   date_min        2802 non-null   float64
 9   date_max        2802 non-null   float64
 10  date_circa      2802 non-null   float64
dtypes: float64(3), int64(3), object(5)
memory usage: 240.9+ KB
```

```
Number of NULL values per column:
id                0
text              0
metadata          0
region_main_id    0
region_main       0
region_sub_id     0
region_sub        0
date_str          0
date_min          0
date_max          0
date_circa        0
dtype: int64

Number of unique values per column:
id                2802
text              2795
metadata          2530
region_main_id      14
region_main         14
region_sub_id       73
region_sub          71
date_str          1384
date_min           505
date_max           509
date_circa           2
dtype: int64
```

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)**. Η 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:

$$Tf(\mathbf{t},\mathbf{d}) = \left[\frac{number\ of\ occurances\ of\ the\ term\ \mathbf{t}\ in\ document\ \mathbf{d}}{number\ of\ terms\ present\ in\ document\ \mathbf{d})}\right]$$

2. **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:

$$Idf(\mathbf{t}) = log\left[\frac{number\ of\ documents}{number\ of\ documents\ that\ have\ \mathbf{t}}\right]$$

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:

```
stopwords = nltk.corpus.stopwords.words('greek')
vectorizer = TfidfVectorizer(stop_words=stopwords, max_features=8000)
index_matrix = vectorizer.fit_transform(df['text'].to_list())
```

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:

shape	tuple	2	(2802, 1000)

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:

326	3.1577020354
879	3.2447134124
696	3.4453841078
690	3.5003662422
33	3.5724695361
312	3.6351406431
491	3.6552418224
920	3.7344388641
922	3.7344388641
891	3.7622958186

39	8.2452983706
88	8.2452983706
89	8.2452983706
238	8.2452983706
348	8.2452983706
429	8.2452983706
466	8.2452983706
481	8.2452983706
511	8.2452983706
520	8.2452983706

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:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

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:

```
scaler = MinMaxScaler()
X = scaler.fit_transform(texts)
y = scaler.fit_transform(dates)
```

,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:

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.378886...	0	0	0	0

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

Δεδομένα εξόδου:

0	0.229893...
1	0.432726...
2	0.255184...
3	0.103439...
4	0.204602...
5	0.229893...
6	0.382144...
7	0.305766...
8	0.141375...
9	0.432726...
10	0.464087...
11	0.424127...
12	0.463581...
13	0.230146...

, 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:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
```

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. 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 , καθώς και την αξιολόγησή του για κάθε εποχή από δεδομένα με τα οποία δεν έχει εκπαιδευτεί (διαχωρισμένα από το cross validation).

Χρησιμοποιείται ο **optimizer Adam** για την ελαχιστοποίηση της συνάρτησης σφάλματος, καθώς είναι αρκετά καλύτερος από άλλες 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:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- n is the number of observations

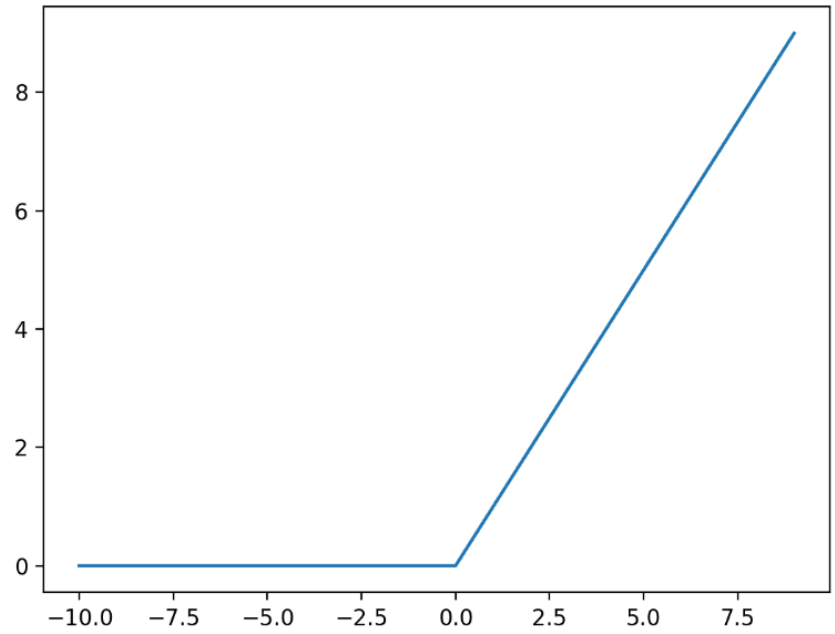
Η υλοποίηση της συνάρτησης φαίνεται παρακάτω:

```
def custom_RMSE(y_true, y_pred):

    return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

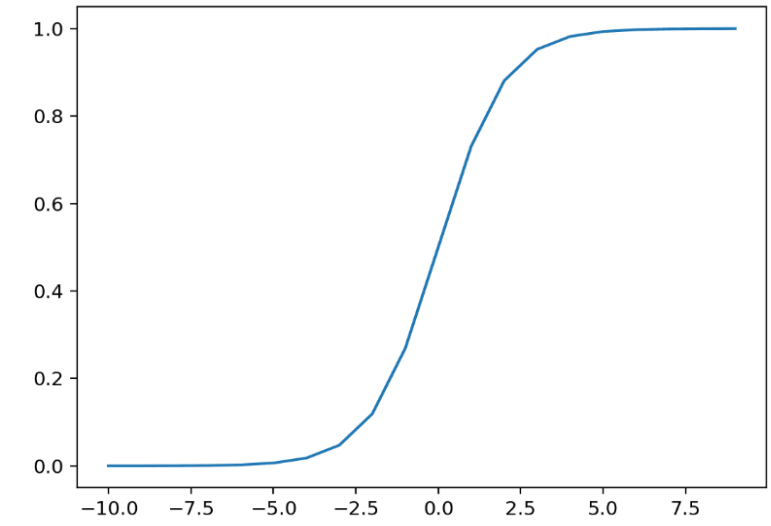
β) Η συνάρτηση ενεργοποίησης ενός νευρωνικού δικτύου καθορίζει πώς ο γραμμικός συνδυασμός των βαρών και της εισόδου μετασχηματίζεται στην έξοδο ενός νευρώνα ή ενός επιπέδου γενικότερα. Η επιλογή συνάρτησης ενεργοποίησης έχει μεγάλη επίπτωση στη ικανότητα και την επίδοση του νευρωνικού δικτύου και γι' 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.



Plot of Inputs vs. Outputs for the ReLU Activation Function.

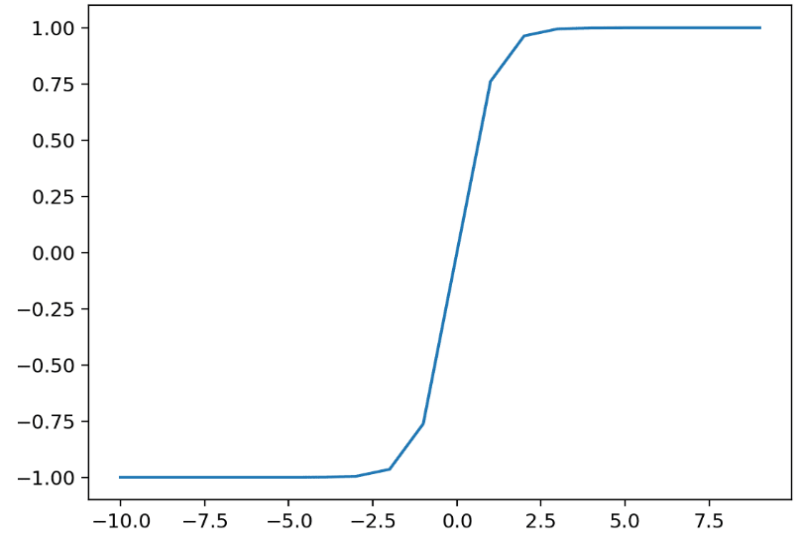
2. **Sigmoid (Logistic) Activation Function:** Η συγκεκριμένη συνάρτηση λαμβάνει μία οποιαδήποτε πραγματική τιμή και σαν αποτέλεσμα επιστρέφει μία τιμή μεταξύ 0 και 1. Όσο μεγαλύτερη είναι η είσοδος, τόσο πιο κοντά είναι η τιμή στο 1 και αντίστροφα. Όταν χρησιμοποιείται η συγκεκριμένη συνάρτηση ενεργοποίησης, 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:



Plot of Inputs vs. Outputs for the Sigmoid Activation Function.

3. **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:

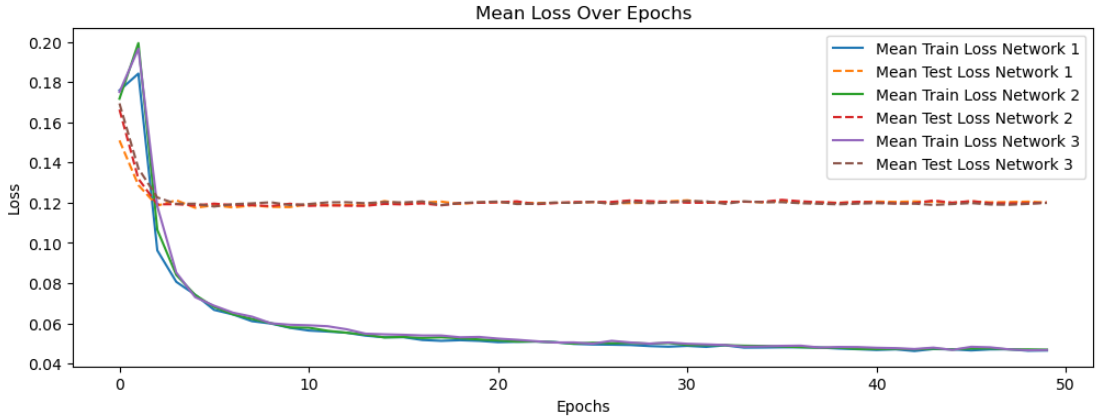


Plot of Inputs vs. Outputs for the Tanh Activation Function.

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 για κάθε εποχή γίνεται το validation με το testing υποσύνολο του συνόλου δεδομένων, για κάθε ένα fold. Παρακάτω, φαίνεται το διάγραμμα της μέσου σφάλματος εκπαίδευσης και ελέγχου ανά εποχή, για 50 εποχές και για νευρωνικά με ένα κρυφό επίπεδο των 250, 500 και 750 κόμβων αντίστοιχα:



Διακρίνουμε ότι το σφάλμα ελέγχου κατά μέσο όρο γενικά, είναι αρκετά μεγαλύτερο του μέσου σφάλματος εκπαίδευσης, ωστόσο είναι μέσα στα επιτρεπτά όρια, δεδομένου ότι ταα δεδομένα του συνόλου ελέγχου είναι άγνωστα. Η σχεδόν ευθεία γραμμή που σχηματίζουν τα δεδομένα ελέγχου για κάθε δίκτυο, μας δηλώνει ότι τα δίκτυα γενικεύονται καλά και η εκπαίδευση τους είναι ανεξάρτητη από την κατανομή των δεδομένων, δηλαδή με λίγα λόγια δεν υπάρχει overfitting. Γενικά διακρίνουμε ότι το νευρωνικό με τους λιγότερους κόμβους στο κρυφό επίπεδο συμπεριφέρεται ελάχιστα καλύτερα, δηλαδή συγκλίνει λίγο πιο γρήγορα με ανεπαίσθητα μικρότερο σφάλμα εκπαίδευσης και ελέγχου. Αυτό γενικά, 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 δηλαδή τις στατιστικές τους ιδιότητες 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, θεωρείται ότι το πρώτο νευρωνικό επίπεδο παρουσιάζει ικανοποιητικά αποτελέσματα, αφού όχι μόνο συγκλίνει στο 0.05 περίπου μέσο σφάλμα μετά από 30 εποχές, αλλά παρουσιάζει και το μικρότερο σφάλμα στα δεδομένα ελέγχου που δηλώνει καλό σχετικά generalization στα δεδομένα.

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:

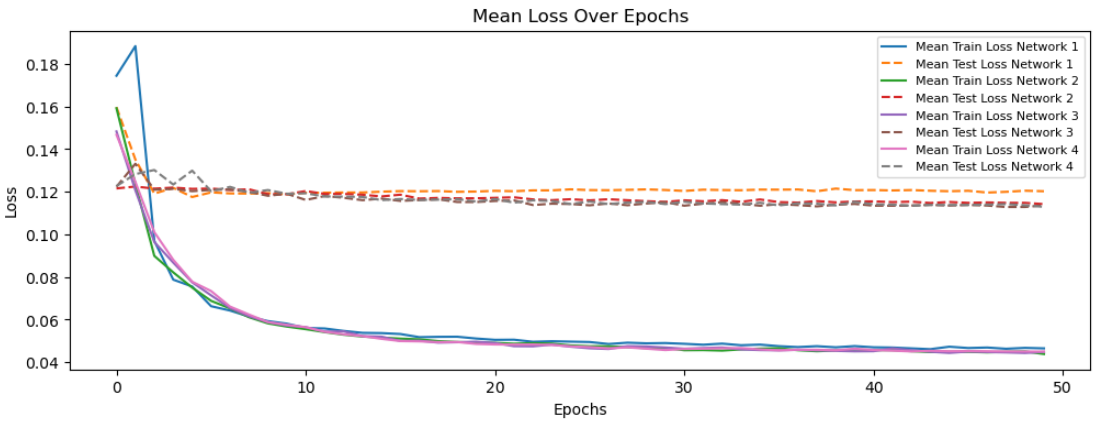
1. **Downscaling**: Η πιο συνηθισμένη τακτική είναι η διαδοχική μείωση των κόμβων σε κάθε κρυφό επίπεδο καθώς μετακινούμαστε πιο «βαθιά» στο νευρωνικό δίκτυο. Η συγκεκριμένη τεχνική είναι γνωστή και ως downscaling και στόχος της είναι να δημιουργήσει ένα funnel effect, όπου το νευρωνικό εστιάζει σταδιακά 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:

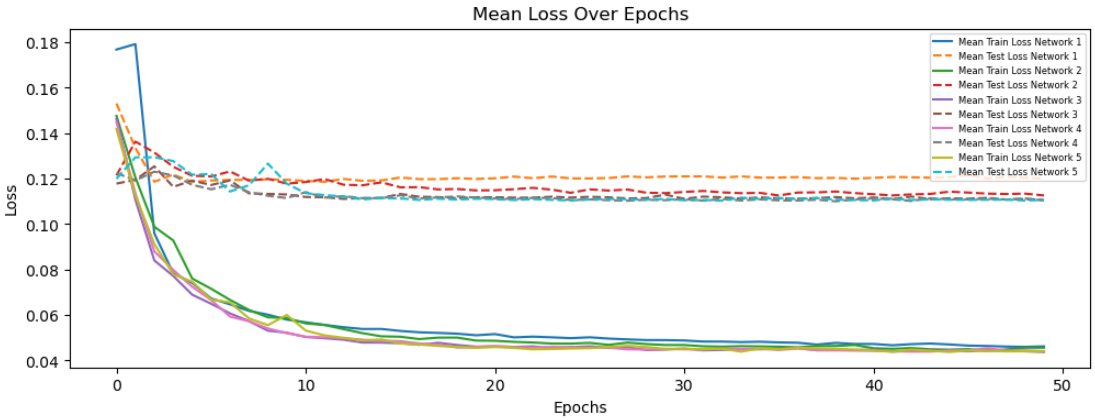
1. **Trial and Error:** Η πιο συνηθισμένη τακτική κυρίως για μικρά προβλήματα που επιλύονται με την χρήση νευρωνικών είναι η συνεχής δοκιμή διαφόρων τιμών των υπερπαραμέτρων των μοντέλων και αξιολόγηση τους έως ότου τα αποτελέσματα που εξάγονται είναι ικανοποιητικά. Μολονότι η συγκεκριμένη τεχνική μπορεί 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.

Εκμεταλλευόμενοι ευρετικές όπως το downscaling με μέγεθος μεταξύ του επιπέδου εισόδου και εξόδου, δηλαδή μεταξύ 8000 και 1, με trial and error για διάφορες τιμές για τους κόμβους του κάθε επιπέδου καταλήξαμε ότι τα παρακάτω νευρωνικά με αρχιτεκτονικές 8000\250\50\1, 8000\400\200\1, και 8000\500\300\1 παρήγαγαν τα καλύτερα μέσα σφάλματα κατά την εκπαίδευσή τους, με το κοινό διάγραμμα του μέσου σφάλματος τους ανά εποχή για 50 εποχές να φαίνεται παρακάτω και το πρώτο νευρωνικό να είναι αυτό με την καλύτερη απόδοση με ένα κρυφό επίπεδο:



Παρατηρούμε ότι σε σχέση με τα αντίστοιχα νευρωνικά ενός επιπέδου, η σύγκλιση απέρχεται αργότερα, ωστόσο για τα νευρωνικά επίπεδα 2 επιπέδων, το μέσο σφαλμα εκπαίδευσης μειώνεται ελάχιστα κάτω από το το 0.05, πίο χαμηλά σε σχέση από τα αντίστοιχα ενός επιπέδου. Η καθυστέρηση της σύγκλισης για τα νευρωνικά δύο επιπέδων οφείλεται κυρίως στο πλήθος των νευρώνων άρα και στον παραμέτρων που πρέπει να ρυθμιστούν, καθιστώντας το νευρωνικό δίκτυο πίο αργό στην μάθηση των μοτίβων των δεδομένων. Ωστόσο τα νευρωνικά με περισσότερα του ενός επίπεδα, έχουν την δυνατότητα να ανιχνεύουν ακόμα πίο πολύπλοκα μοτίβα, συνεπώς το σφάλμα τους κατά την εκπαίδευση είναι πίο μικρό συγκριτικά με νευρωνικά δίκτυα ενός κρυφού επιπέδου. Η βελτιωμένη απόδοση των νευρωνικών δύο κρυφών επιπέδων, φαίνεται κυρίως στο υποσύνολο ελέγχου, όπου το σφάλμα ελέγχου μικραίνει πολύ περισσότερο από τα αντίστοιχα ενός κρυφού επιπέδου όσο περνάνε οι εποχές.

Έπειτα, εκπαιδευτήκαν νευρωνικά των 3 κρυφών επιπέδων, με αρχιτεκτονικές των 8000\400\200\100\1, 8000\500\300\100\1, 8000\800\400\200\1 αντίστοιχα. Παρακάτω, φαίνονται τα μέσα σφάλματα εκπαίδευσης ανά εποχή, για 50 εποχές, με το πρώτο δίκτυο να είναι αυτό με την καλύτερη απόδοση με ένα κρυφό επίπεδο και το δεύτερο να είναι αυτό με την καλύτερη απόδοση για 2 κρυφά επίπεδα:



Βλέπουμε ότι το σφάλμα μειώθηκε στην εκπαίδευση, αλλά βελτίωση φαίνεται περισσότερο στην αξιολόγηση. Βλέπουμε ότι παρόλο που τα δίκτυα έχουν γίνει πολύπλοκα, το σφάλμα ελέγχου παραμένει σταθερό για μεγάλες εποχές, κάτι που μας επιβεβαιώνει ότι το νευρωνικά δίκτυα δεν παρουσιάζουν Overfitting. Από τα δίκτυα 3 κρυφών επιπέδων, διακρίνουμε ότι το δίκτυο 4 (8000\500\300\100\1) είχε την καλύτερη απόδοση γενικά.

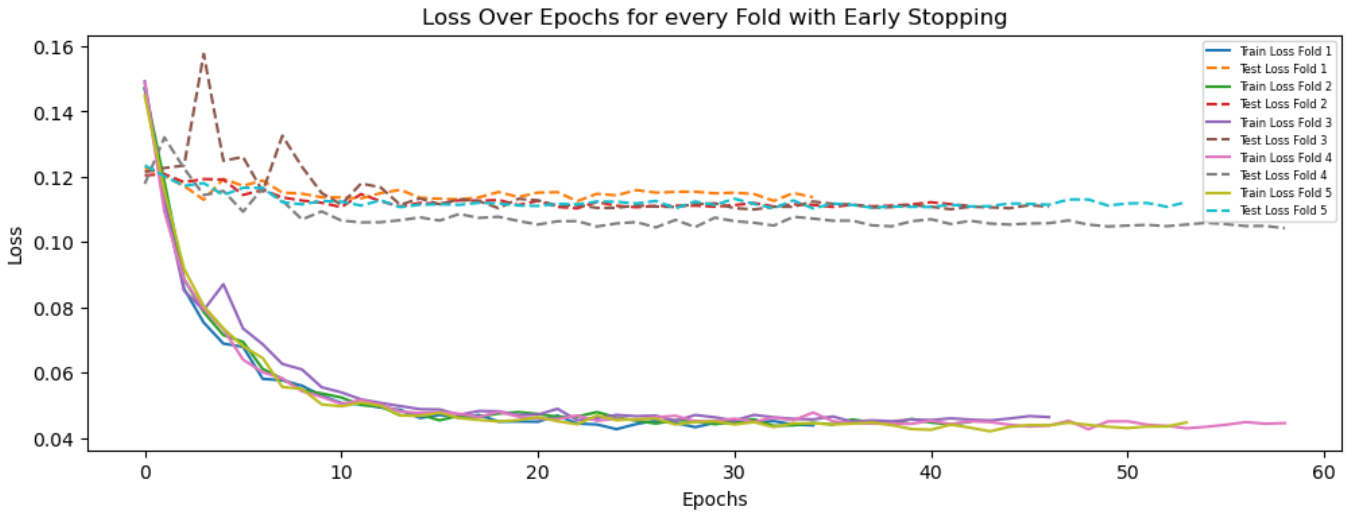
στ) Γενικά, το πιο ευρέως χρησιμοποιήσιμο κριτήριο τερματισμού, είναι το early stopping και οι εναλλακτικές του υλοποιήσεις. Το early stopping, προτάθηκε σαν μέθοδος για την μείωση του overfitting, όντας πιο λειτουργική λύση από τη θεώρηση του αριθμού των εποχών ως υπερπαράμετρο, πράγμα το οποίο οδηγεί στην πολλαπλή εκτέλεση της εκπαίδευσης του εκάστοτε νευρωνικού δικτύου. Κεντρική ιδέα της μεθόδου αυτής είναι η εκπαίδευση του μοντέλου για ένα μεγάλο αριθμό εποχών, αξιολογώντας το μετά από κάθε εποχή με ένα ξεχωριστό υποσύνολο αξιολόγησης, το οποίο δεν μετέχει στην εκπαίδευση του νευρωνικού. Όταν η απόδοση του μοντέλου στο υποσύνολο αξιολόγησης αρχίσει να μειώνεται, δηλαδή το σφάλμα αρχίζει να αυξάνεται για το δικό μας πρόβλημα, η εκπαίδευση σταματάει. Γενικά δύο είναι τα στοιχεία που συνεπάγονται με την χρήση του early stopping:

1. **Παρακολούθηση απόδοσης μοντέλου**, χρησιμοποιώντας μία μετρική όπως το σφάλμα αξιολόγησης του ανεξάρτητου υποσυνόλου αξιολόγησης μετά από κάθε εποχή.

2. **Early Stopping Trigger:** Αποφασίζεται πότε θα σταματήσει η εκπαίδευση ανάλογα με την μετρική που παρακολουθείται. Στην απλή περίπτωση, η εκπαίδευση σταματάει μόλις η απόδοση στο τρέχον υποσύνολο αξιολόγησης μειωθεί παρακάτω από την τιμή του προηγούμενου υποσυνόλου αξιολόγησης. Ωστόσο, λόγω θορύβου, αυτό το συμβάν δεν σηματοδοτεί με βεβαιότητα την πλήρη εκπαίδευση του μοντέλου, όπως εξάλλου φαίνεται και από τα παραπάνω διαγράμματα απόδοσης των νευρωνικών δικτύων. Πιο σωστές αποφάσεις τερματισμού ωστόσο, για το πρόβλημα που επιλύεται εδώ, είναι οι ακόλουθες:
    a. Μη μεταβολή της μετρικής για συγκεκριμένο αριθμό εποχών.
    b. Αλλαγή της απόλυτης τιμής της μετρικής.
    c. Μείωση της απόδοσης που παρατηρείται για ένα συγκεκριμένο αριθμό εποχών.
    d. Μέση μείωση της απόδοσης για συγκεκριμένο αριθμό μετοχών.

Χρησιμοποιούμε το νευρωνικό με την καλύτερη απόδοση για να εφαρμόσουμε την τεχνική του early stopping. Η συγκεκριμένη τεχνική απαιτεί την διαρκή παρακολούθηση του υποσυνόλου αξιολόγησης, το ρόλο του οποίου θα παίξει το σύνολο ελέγχου για κάθε fold. Ωστόσο, γενικά μέθοδοι όπως cross validation μαζί με early stopping δεν συνηθίζεται να εφαρμόζονται μαζί, καθώς η κάθε μία ακολουθά την δικιά της μεθοδολογία για την αποφυγή του overfitting. Ο κυριότερος λόγος γι' αυτή την αντισυμβατικότητα, είναι ότι για κάθε fold τα δεδομένα ελέγχου αλλάζουν, συνεπώς σχεδόν σίγουρα θα αλλάξουν και οι εποχές για τις οποίες εκπαιδεύεται κάθε φορά το νευρωνικό. Έτσι για κάθε fold προκύπτει ότι η εκπαίδευση σχεδόν σίγουρα θα σταματήσει σε διαφορετική εποχή και το να πάρει κανείς την μέση τιμή των εποχών για να σταματήσει εκεί, δεν είναι σίγουρο ότι θα αποδώσει τα καλύτερα αποτελέσματα. Μία λύση η οποία θα μπορούσε να εφαρμοστεί είναι η αξιολόγηση του μοντέλου για ένα fixed υποσύνολο αξιολόγησης, το οποίο δεν προέρχεται από την διαδικασία του cross validation. Ωστόσο, δεδομένου ότι η εκπαίδευση του νευρωνικού μας γίνεται καθαρά για εκπαιδευτικούς σκοπούς, θα παρατεθούν τα γραφήματα για κάθε ένα fold ξεχωριστά.

Επιλέγεται να χρησιμοποιηθεί patience = 10, που ορίζει τον αριθμό των εποχών που θα περιμένει το μοντέλο πριν σταματήσει την εκπαίδευση για την αποφυγή πλατό, ωστόσο δεν ορίζουμε min_delta που ορίζει το πόσο μικρή ορίζεται η μεταβολή του σφάλματος επικύρωσης από εποχή σε εποχή πριν σταματήσει η εκπαίδευση. Προκειμένου η υλοποίηση να είναι εύκολη προς ανάγνωση, υλοποιήθηκε δεύτερη συνάρτηση για το fit του μοντέλου με το callback για το early stopping. Παράλληλα τροποποιήθηκε και η συνάρτηση train_model ώστε να επιλέγεται με το Boolean όρισμα have_callback, αν θέλουμε να εκπαιδεύσουμε το μοντέλο με χρήση της μεθόδου του early stopping. Σε περίπτωση που θέλουμε να μην χρησιμοποιήσουμε early stopping, η συνάρτηση μας επιστρέφει τα μέσα σφάλματα εκπαίδευσης και ελέγχου για κάθε εποχή (αναλυτικά το functionality φαίνεται στο documentation του κώδικα). Παρακάτω, φαίνεται το διάγραμμα των σφαλμάτων εκπαίδευσης και ελέγχου για κάθε fold ξεχωριστά ανά εποχή:



Όπως διακρίνουμε, για κάθε fold η εποχή που σταματάει την εκπαίδευση του το νευρωνικό μας δίκτυο ποικίλει, ωστόσο με αυτή τη τεχνική βλέπουμε ότι η εκπαίδευση σταματάει πριν το σφάλμα αρχίσει να αυξάνεται ή όταν παραμένει οσοδήποτε σταθερό.

Α3. Μεταβολές στον ρυθμό εκπαίδευσης και σταθερής ορμής:

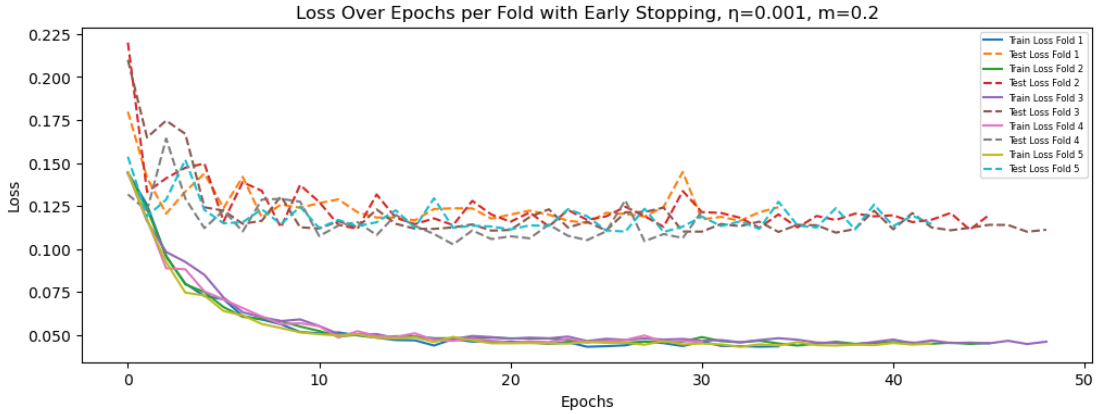
α) Ο ρυθμός μάθησης είναι μία παράμετρος ενός νευρωνικού δικτύου που καθορίζει πόσο μεταβάλλονται τα βάρη και τα biases του δικτύου για κάθε βήμα της εκπαίδευσης όταν ενημερώνονται με βάση τα δεδομένα και την απώλεια που προκύπτει. Όταν έχουμε υψηλό ρυθμό μάθησης, σημαίνει ότι τα βάρη επίσης μεταβάλλονται γρήγορα, ωστόσο αυτό πιθανώς να οδηγήσει σε αυξημένη αστάθεια του δικτύου, καθώς το δίκτυο μπορεί να υπερπηδήσει τις βέλτιστες τιμές των βαρών. Ωστόσο ένας πολύ μικρός ρυθμός μάθησης, οδηγεί σε πολύ μικρές μεταβολές των βαρών, κάνοντας την εκπαίδευσή πολύ αργή με την πιθανότητα η απώλεια να συγκλίνει σε υποβέλτιστες λύσεις (παγίδευση σε τοπικά ελάχιστα).

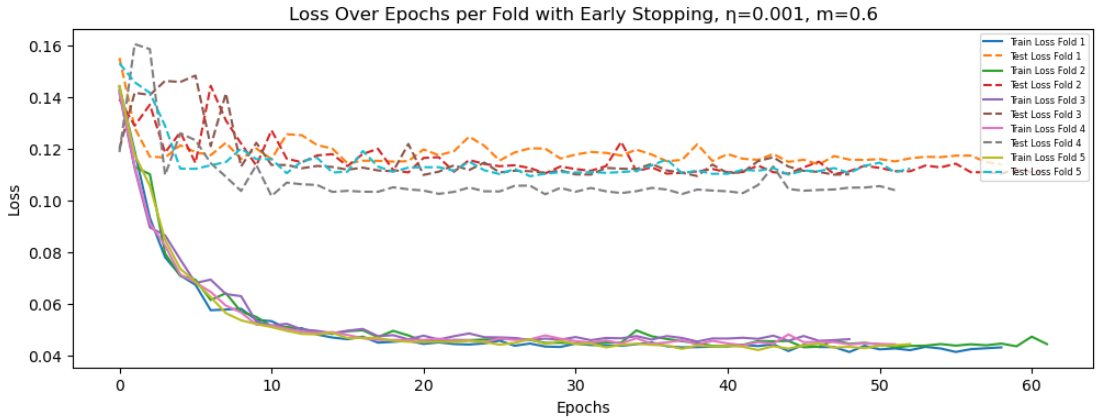
Η σταθερά ορμής (momentum), είναι παράμετρος που χρησιμοποιείται προκειμένου να βοηθήσει στην σταθερότητα και την στην ταχύτητα της εκπαίδευσης. Πιο συγκεκριμένα, στα νευρωνικά δίκτυα χρησιμοποιείται για την ενίσχυση των ενημερώσεων των βαρών, με βάση προηγούμενες ενημερώσεις, με αποτέλεσμα να μειώνεται η αστάθεια και να ομαλοποιείται η πορεία της εκπαίδευσης. Όταν έχουμε υψηλή σταθερά ορμής, τόσο μεγαλύτερη είναι η επιρροή των προηγούμενων ενημερώσεων των βαρών στην τωρινή ενημέρωση και αντίστροφα.

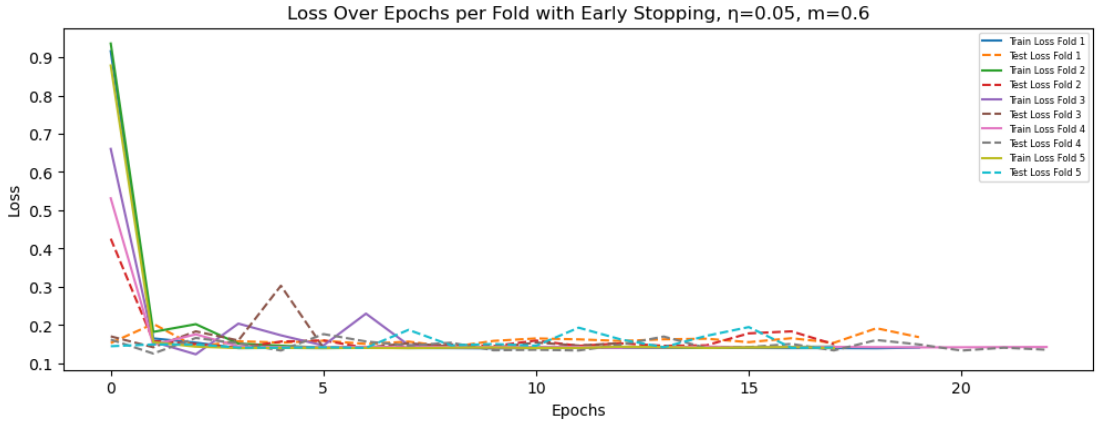
Παρακάτω, διακρίνουμε τον πίνακα με την μέσο RMSE για διάφορες τιμές του ρυθμού μάθησης και της σταθεράς ορμής:

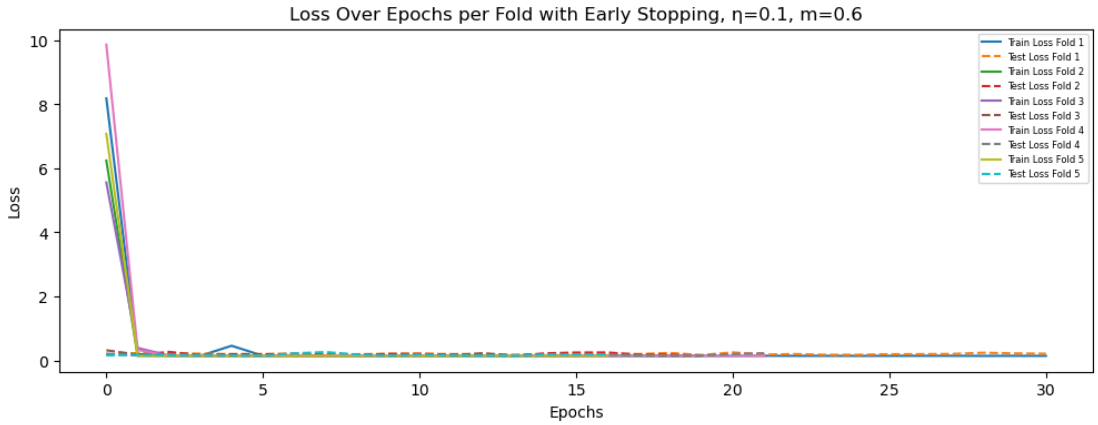
η	**m**	**RMSE**
0.001	0.2	0.05573
0.001	0.6	0.05232
0.05	0.6	0.18271
0.1	0.6	0.49737

Παρακάτω, φαίνονται τα διαγράμματα σφάλματος εκπαίδευσης και ελέγχου για τις παραπάνω τιμές του ρυθμού μάθησης και της σταθεράς ορμής, για κάθε fold και για κάθε εποχή:


Loss Over Epochs per Fold with Early Stopping, η=0.001, m=0.2
Loss
Epochs
Train Loss Fold 1
Test Loss Fold 1
Train Loss Fold 2
Test Loss Fold 2
Train Loss Fold 3
Test Loss Fold 3
Train Loss Fold 4
Test Loss Fold 4
Train Loss Fold 5
Test Loss Fold 5


Loss Over Epochs per Fold with Early Stopping, η=0.001, m=0.6
Loss
Epochs
Train Loss Fold 1
Test Loss Fold 1
Train Loss Fold 2
Test Loss Fold 2
Train Loss Fold 3
Test Loss Fold 3
Train Loss Fold 4
Test Loss Fold 4
Train Loss Fold 5
Test Loss Fold 5


Loss Over Epochs per Fold with Early Stopping, η=0.05, m=0.6
Loss
Epochs
Train Loss Fold 1
Test Loss Fold 1
Train Loss Fold 2
Test Loss Fold 2
Train Loss Fold 3
Test Loss Fold 3
Train Loss Fold 4
Test Loss Fold 4
Train Loss Fold 5
Test Loss Fold 5


Loss Over Epochs per Fold with Early Stopping, η=0.1, m=0.6
Loss
Epochs
Train Loss Fold 1
Test Loss Fold 1
Train Loss Fold 2
Test Loss Fold 2
Train Loss Fold 3
Test Loss Fold 3
Train Loss Fold 4
Test Loss Fold 4
Train Loss Fold 5
Test Loss Fold 5

Παρατηρούμε ότι όταν η σταθερά ορμής είναι μικρή, δηλαδή όταν οι παλαιότερες ενημερώσεις των βαρών δεν παίζουν σημαντικό ρόλο στο καθορισμό των νέων, το σφάλμα δείχνει να «ταλαντώνεται», καθώς το δίκτυο αποσταθεροποιείται, λόγω των μεγάλων αλλαγών στα βάρη. Παρατηρούμε ακόμη ότι όσο η σταθερά ορμής αυξάνεται τόσο περισσότερο σταθεροποιείται το σφάλμα και κατά την εκπαίδευση και κατά τον έλεγχο του δικτύου, ενώ φαίνεται να συγκλίνει ελαφρώς πιο γρήγορα. Είναι προφανές ότι η σταθερά ορμής πρέπει να είναι κάτω της μονάδας. Αν είναι ίση η ελαφρά μεγαλύτερη της μονάδας, οι προηγούμενες ενημερώσεις των βαρών θα παίξουν κυριαρχικό ρόλο στην κατεύθυνση της συνάρτησης και μπορεί να οδηγήσουν σε σύγχυση του νευρωνικού ή σε μεγάλες ταλαντώσεις. Αν το gradient descent προσομοιάζει την κατάβαση σε μία κατηφόρα, η σταθερά ορμής προσομοιάζει την ορμή που κατεβαίνει κανείς την κατηφόρα αυτή. Για μεγάλες τιμές της σταθεράς, καταλαβαίνουμε ότι αντί να κατεβαίνει κανείς την ανηφόρα, θα την ανέβαινε, εφόσον όσο μεγαλύτερη η σταθερά, τόσο μεγαλύτερο ρόλο παίζουν οι παρελθοντικές ενημερώσεις των βαρών στην τωρινή.

Παρατηρούμε ακόμα ότι όσο αυξάνουμε τον ρυθμό μάθησης ναι μεν το δίκτυο συγκλίνει γρήγορα σε κάποιο ελάχιστο το οποίο σχεδόν σίγουρα είναι τοπικό , ωστόσο εγκλωβίζεται εκεί, καθώς δεν μπορεί να πραγματοποιηθούν μικρότερες αλλαγές στα βάρη προκειμένου να προσεγγιστεί μία πιο βέλτιστη λύση.

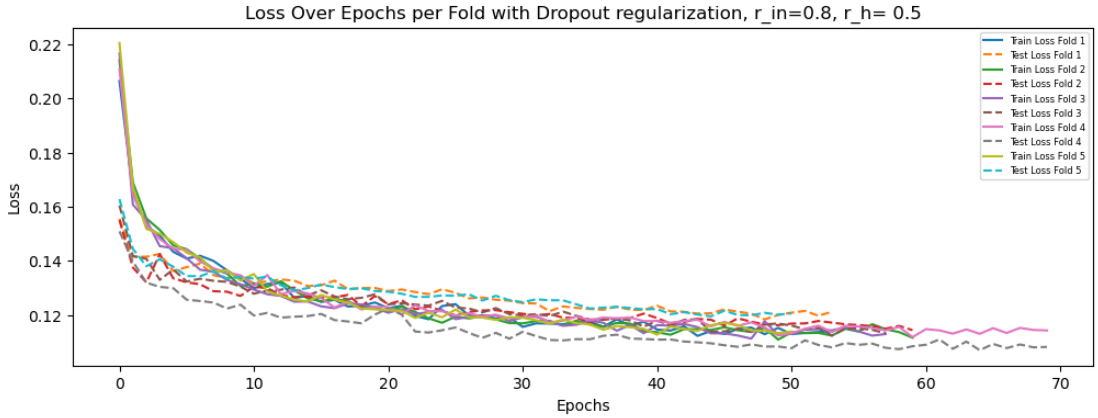
Α4. Ομαλοποίηση:

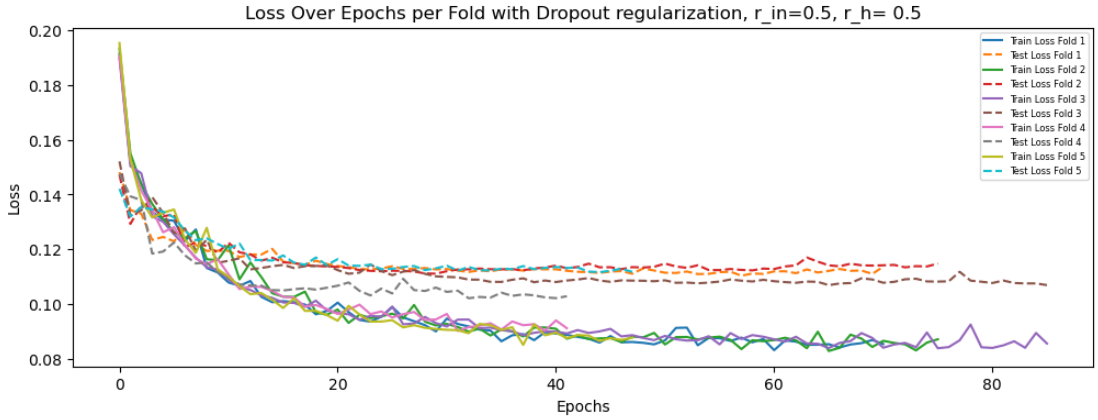
Γι’ αυτό το ερώτημα προστέθηκε μία καινούργια συνάρτηση, η set_model_with_dropout, η οποία προσθέτει dropout layers για κάθε ένα από τα layers του νευρωνικού εκτός από το κρυφό επίπεδο. Η συνάρτηση παίρνει ξεχωριστά dropout rates για το επίπεδο εισόδου και για τα κρυφά επίπεδα. Τέλος, τροποποιήθηκε η συνάρτηση train_model με πρόσθεση ορισμάτων, ώστε να μπορεί κανείς να ορίσει αν θέλει dropout ή όχι και να ορίσει ξεχωριστά τα rates. Προσοχή, ακόμα και σε περίπτωση που δεν θέλει κανείς dropout (dropout = False), τα rates πρέπει να οριστούν, αλλά σε 0.

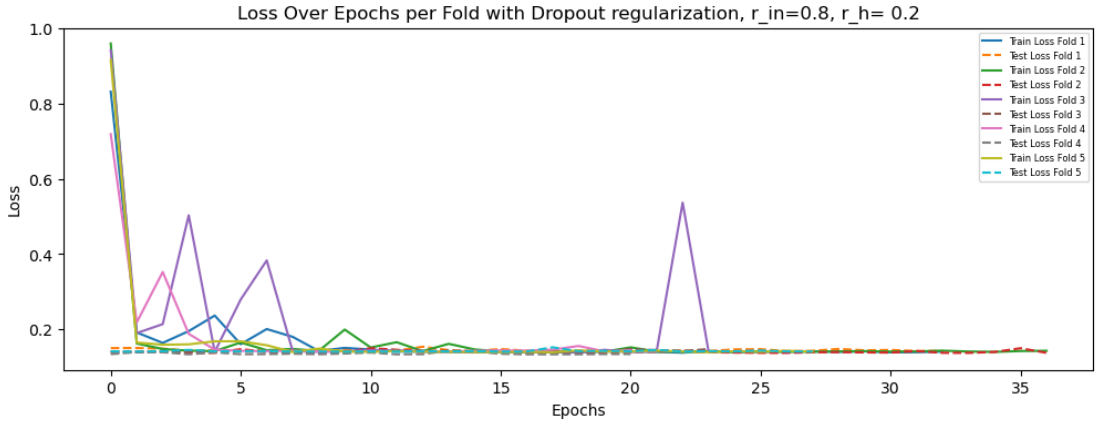
Γενικά, το dropout είναι μία ακόμα τεχνική για αποφυγή overfitting, κατά την οποία, μερικοί νευρώνες επιλέγονται και απενεργοποιούνται τυχαία. Αυτό έχει σαν σκοπό να μάθει το νευρωνικό να δουλεύει με διάφορες αρχιτεκτονικές και να μην στηρίζεται τόσο σε συγκεκριμένους νευρώνες ή features, εξαναγκάζοντάς το να γενικοποιείται καλύτερα για καινούργια δεδομένα.

Πιθανότητες διατήρησης	**RMSE**
r_{in} =0.8 r_h = 0.5	0.121723
r_{in} =0.5 r_h = 0.5	0.124239
r_{in} =0.8 r_h = 0.2	0.190311

Τα διαγράμματα για τις διάφορες τιμές των r φαίνονται παρακάτω:







Όσο αυξάνεται το dropout rate, τόσο μειώνεται ο αριθμός των νευρώνων που συμμετέχουν στην εκμάθηση του μοντέλου για κάθε fold. Συνεπώς αυτό έχει ως αποτέλεσμα την αύξηση του σφάλματος τόσο κατά την εκπαίδευση όσο και κατά την αξιολόγηση του μοντέλου. Γενικά παρατηρούμε ότι στα 0.5 και στα δύο, η απόδοση προσεγγίζει αυτή που είχαμε πριν εισάγουμε την μέθοδο του dropout.