Project Υπολογιστικής Νοημοσύνης
Μέρος Α

Κωνσταντίνος Βασιλαδιώτης 1088094

Α1). Προεπεξεργασία και Προετοιμασία των δεδομένων:

α. Αρχικά χρησιμοποίησα ως είσοδο στο δίκτυο μου τα εξής πεδία id, text, date_min, date_max, date_circa και region_sub_id από τα οποία μόνο τα δεδομένα του πεδίου text χρειάζονται μετατροπή και ανάλυση στα δομίκα τους στοιχεία. Στην συνέχεια ακολουθεί η διαδικασία vectorization για τα δεδομένα του πεδίου text όπου μετατρέπουμε τα δείγματα του κειμένου σε αριθμητικά διανύσματα τα οποία μπορούν αργότερα να επεξεργαστούν από ένα μοντέλο μηχανικής μάθησης.

```
# Μετατροπή του πεδίου κειμένου σε διανυσματική μορφή TF-IDF
vectorizer = TfidfVectorizer(max_features=1000)
X_text_tfidf = vectorizer.fit_transform(df['text'])
```

Στην συγκεκριμένη άσκηση έγινε χρήση TF-IDF για την κωδικοποίηση των tokens. Επίσης χρησιμοποίησα έναν περιορισμένο αριθμό tokens από όλα που προκύπτουν από το dataset (max_features=1000).

Έπειτα επειδή στο αρχείο arff που δημιούργησα κάθε γραμμή δεδομένων όσον αφορά το πεδίο text είχε πολλές τιμές tf-idf (διότι κάθε κείμενο επιγραφής έχει προφανώς πάνω από μια λέξεις) το αποτέλεσμα ήταν η μορφή του arff αρχείου να μην είναι ορθή και έτσι να μην μπορώ να το εισάγω στο Weka. Για να λύσω αυτό

το πρόβλημα σε κάθε γραμμή του πεδίου text κράτησα μόνο τον μέγιστο tf-idf αριθμό από κάθε επιγραφή.

```
# Εύρεση μέγιστου tf-idf αριθμού για κάθε γραμμή
max_tfidf = X_text_tfidf.max(axis=1).toarray()

# Ενημέρωση του X_text_tfidf με τον μέγιστο tf-idf αριθμό
X_text_tfidf = csr_matrix(max_tfidf)
```

Έτσι η διαδικασία κωδικοποίησης και επεξεργασίας των δεδομένων έχει ολοκληρωθεί καθώς έχει δημιουργηθεί ένα αρχείο arff όπου κάθε attribute είναι της μορφής numeric (τα υπόλοιπα πεδία εκτός του text ήταν ήδη).

Το αρχείο που δημιουργείται έχει αυτή την μορφή:

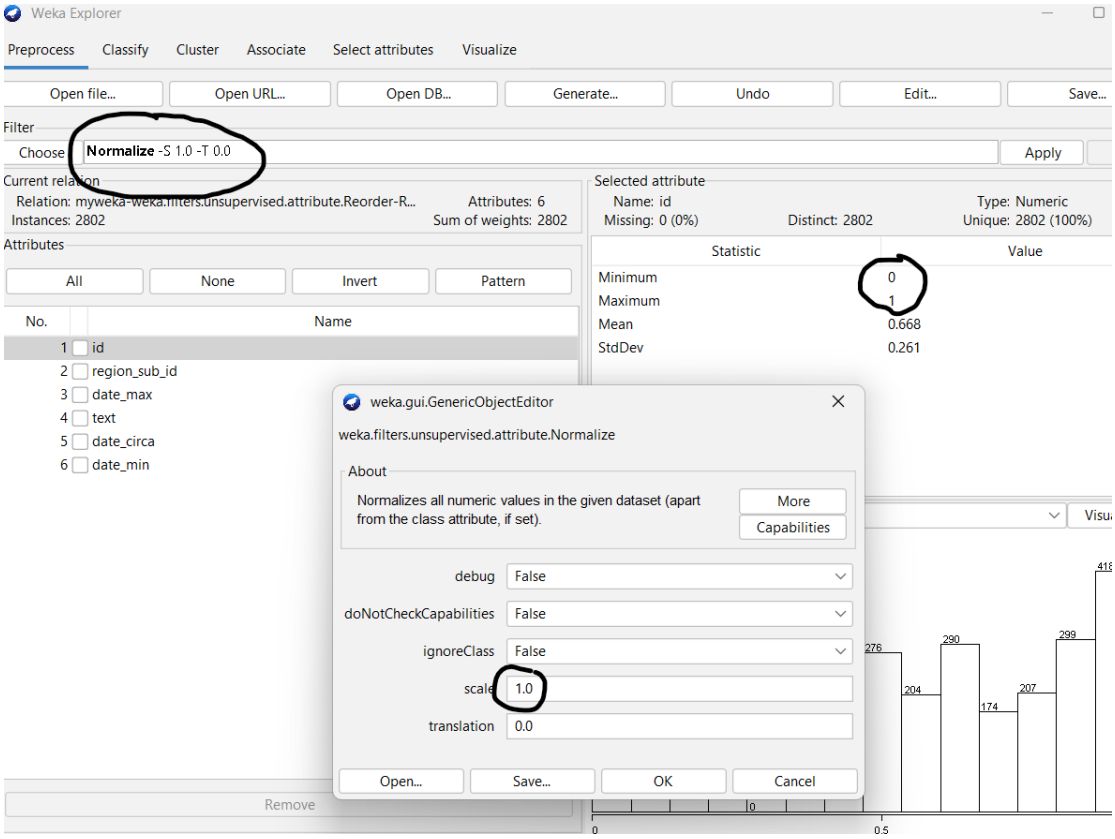
```
@relation 'myweka'
@attribute id NUMERIC
@attribute region_sub_id NUMERIC
@attribute date_min NUMERIC
@attribute date_max NUMERIC
@attribute text NUMERIC
@attribute date_circa NUMERIC

@data
315181.0,1691.0,-275.0,-226.0,0.0,0.0
201686.0,474.0,1.0,300.0,0.0,0.0
153178.0,1485.0,-300.0,-101.0,0.0,0.0
28582.0,1643.0,-600.0,-401.0,1.0,0.0
333620.0,1689.0,-350.0,-251.0,0.7526462932313842,0.0
186989.0,1646.0,-300.0,-201.0,0.41977258494790654,0.0
186751.0,1646.0,1.0,100.0,0.7145981575973436,0.0
```

Και πλέον έχει την κατάλληλη δομή για να εισαχθεί στο Weka.

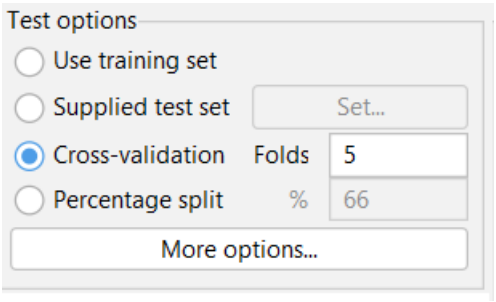
Β. Όσον αφορά την κανονικοποίηση των δεδομένων αυτή θα γίνει μέσω της χρήσης του Weka όπου διαλέγουμε filter -> unsupervised -> attribute -> Normalize και ορίζουμε κλίμακα [0,1]. Αρχικά η κανονικοποίηση εφαρμόζεται μόνο στις εισόδους που πλέον έχουν min = 0, max = 1 όμως παρόλαυτα η έξοδος (που έχουμε θέσει το date_min) παραμένει χωρίς με αποτέλεσμα να έχει εύρος τιμών [-720 , 1100]. Αυτό σημαίνει ότι το αποτέλεσμα (απόκλιση) που θα δούμε αργότερα RMSE θα μετριέται σε χρόνια σε σχέση με το date_min.

Από την άλλη πλευρά έχουμε την επιλογή να κανονικοποιήσουμε και την έξοδο με χρήση κλίμακας [0,1] όπου αυτή την φορά το αποτέλεσμα (απόκλιση) θα είναι ένα ποσοστό του λάθους σε σχέση με την αρχική κλίμακα των δεδομένων.



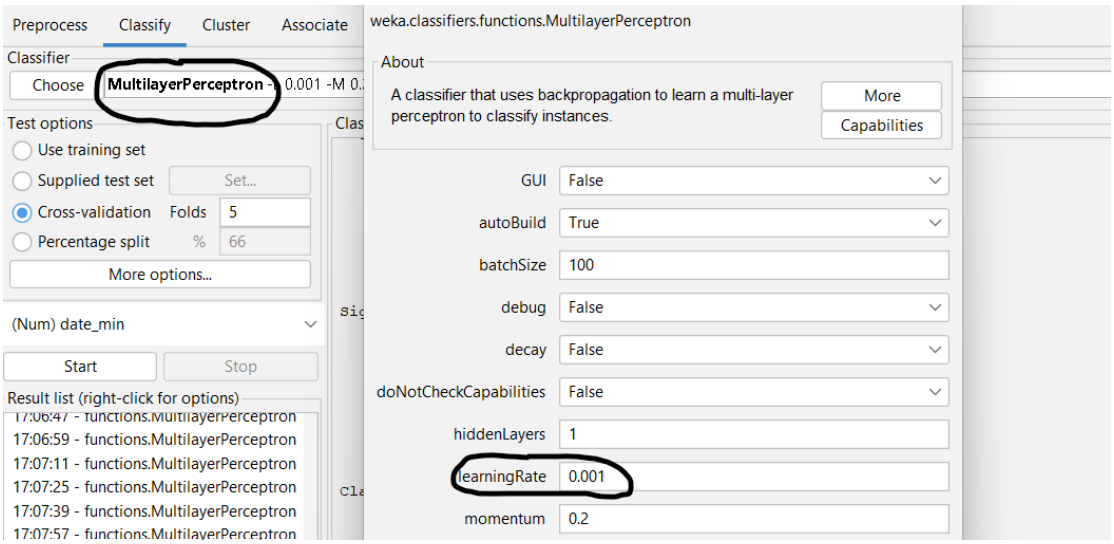
Γ. Η διασταυρωμένη επικύρωση 5-τμημάτων διαιρεί τυχαία το σύνολο των δεδομένων σε 5 υποσύνολα από τα οποία χρησιμοποιεί τα 4 για εκπαίδευση και το τελευταίο 1 για έλεγχο και υπολογίζει το σφάλμα γενίκευσης. Εφόσον λοιπόν η διαδικασία διαιρεί αυτόματα τα δεδομένα σε ένα σύνολο εκπαίδευσης και ένα ελέγχου αυτό σημαίνει ότι εμείς εισάγουμε στο Weka μόνο ένα αρχείο arff με όλα τα data μέσα και την υπόλοιπη διαδικασία την κάνει το Weka. (Είχα φτιάξει άλλον ένα ίδιο κώδικα ο οποίος δημιουργούσε δύο ξεχωριστά αρχεία arff ένα για κάθε σύνολο(εκπαίδευση&έλεγχος) που αθροιστικά είχαν όλα τα δεδομένα μέσα αλλά αν έβαζα ως είσοδο στο Weka το train_arff_file η διαδικασία 5-fold cv θα έκανε την διαίρεση σε ένα ήδη διαιρεμένο αρχείο).

Ακολουθεί η επιλογή της μεθόδου 5-fold CV:



Α2). Επιλογή Αρχιτεκτονικής:

Για την εκπαίδευση χρησιμοποιούμε τον Αλγόριθμο πίσω διάδοσης του λάθους(back-propagation) , ένα κρυφό επίπεδο και αρχικό ρυθμό μάθησης η = 0.001. Αυτό σημαίνει ότι στο Weka επιλέγουμε στον classifier το Multilayer Perceptron και στις ρυθμίσεις του βάζουμε learning rate = 0.001.



Α. Εφόσον έχω επιλέξει ως έξοδο το πεδίο date_min η αξιολόγηση των αποτελεσμάτων θα γίνει με βάση αυτό το πεδίο και με χρήση της RMSE. Αυτό σημαίνει ότι κάθε φορά το RMSE υποδηλώνει την απόκλιση σε χρόνια από το άκρο date_min δεδομένου των εισόδων id, text, date_max, date_circa, region_sub_id. (Εννοείται δεν έχω κανονικοποιήσει την έξοδο ακόμα).

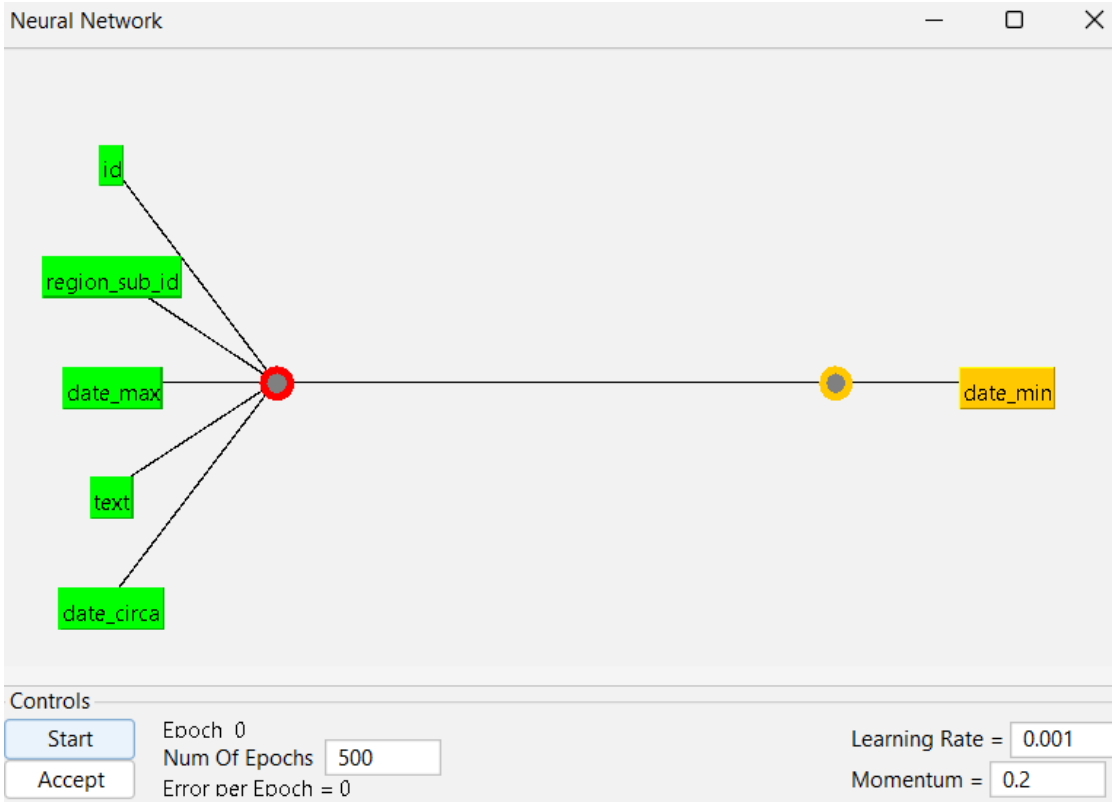
Εδώ ξεκινάμε μία αξιολόγηση με ρυθμό μάθησης η = 0.001 και έναν κρυφό κόμβο:

```
Time taken to build model: 0.22 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8679
Mean absolute error                     89.4234
Root mean squared error                129.0173
Relative absolute error                 41.7917 %
Root relative squared error             49.6657 %
Total Number of Instances             2802
```

Όπου βλέπουμε ότι το σφάλμα είναι περίπου στα 129 χρόνια συγκριτικά με το date_min. Το νευρωνικό δίκτυο αυτή την στιγμή μοιάζει κάπως ετσι:



Με κανονικοποιημένη έξοδο αυτή την φορά έχουμε τα εξής αποτελέσματα:

```
Time taken to build model: 0.18 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8679
Mean absolute error                      0.0491
Root mean squared error                  0.0709
Relative absolute error                 41.7917 %
Root relative squared error             49.6657 %
Total Number of Instances             2802
```

Β. Ως κατάλληλη συνάρτηση ενεργοποίησης των κρυφών κόμβων η αλήθεια είναι ότι δεν είχα επιλογή διότι το Weka σε χρήση Multilayer Perceptron εφαρμόζει αυτόματα σε όλους τους κόμβους μόνο την σιγμοειδή συνάρτηση. Παρόλαυτα η σιγμοειδής συνάρτηση είναι μια πολύ καλή και συνήθης επιλογή για τους κρυφούς κόμβους διότι η έξοδος της κυμαίνεται μεταξύ 0 και 1, επομένως αυτό βοηθά στην κανονικοποίηση, στην κλίμακα δεδομένων καθώς και στην επίτευξη μη-γραμμικότητας του συνόλου δεδομένων. Πιο συγκεκριμένα η σιγμοειδής επειδή είναι μια μη-γραμμική συνάρτηση που επιτρέπει στο νευρωνικό δίκτυο να μάθει πιο πολύπλοκες συσχετίσεις μεταξύ εισόδων και εξόδων. Αυτό είναι σημαντικό διότι η πλειονότητα των πραγματικών προβλημάτων δεν μπορούν να αναλυθούν με απλές γραμμικές συσχετίσεις. Επίσης η συνάρτηση αυτή είναι παραγωγήσιμη που σημαίνει ότι μπορούμε να βρούμε την τοπική κλίση δ κάθε κρυφού νευρώνα j της καμπύλης σε οποιαδήποτε σημεία κάτι που βοηθά τον αλγόριθμο πίσω διάδοσης του λάθους στο πίσω πέρασμα αλλά και στην προσαρμογή των βαρών. Ο υπολογισμός του δ είναι μια βασική απαίτηση του αλγορίθμου back-propagation. Το πρόβλημα που δημιουργείται με αυτήν την συνάρτηση είναι το πρόβλημα του κορεσμού. Όταν η έξοδος ενός νευρώνα j είναι στα όρια, τότε η τοπική κλίση δ βρίσκεται κοντά στο 0 κάτι που έχει ως αποτέλεσμα ο νευρώνας j να μην συνειφέρει στο πίσω πέρασμα του αλγορίθμου.

Γ. Όσον αφορά τον κόμβο εξόδου χρησιμοποίησα την σιγμοειδή πάλι λόγω του Weka αυτόματα, ωστόσο θα μπορούσαμε να διαλέξουμε την tanh(υπερβολική εφαπτομένη) για την πρόβλεψη χρονολογιών ως εξής: Παρέχει ένα εύρος εξόδου [-1,1] άρα η έξοδος του δικτύου θα μπορούσε να ερμηνευτεί ως ένα ποσοστό απόκλισης από το date_min (παρόμοια χρήση με sigmoid) όμως αυτή

την φορά οι αρνητικές τιμές θα υποδεικνύουν απόκλιση πριν από την πραγματική χρονολογία ενώ οι θετικές τιμές απόκλιση μετά την πραγματική χρονολογία. Επίσης όπως και η σιγμοειδής βοηθά στην κανονικοποίηση και στην μεταφορά του εύρους τιμών σε νέα κλίμακα. Επίσης με κατάλληλες τιμές dj μπορούμε να αποφύγουμε το πρόβλημα του κορεσμού. Θα χρησιμοποιηθεί a = 1.7159 και b = 2/3.

Δ. Σε αυτό το ερώτημα πειραματιζόμαστε με τον αριθμό κρυφών κόμβων στο πρώτο επίπεδο.

Αριθμός κρυφών νευρώνων στο κρυφό επίπεδο:	RMSE:
H1 = 1	129.0173
H1 = 3	127.9093
H1 = 7	133.4629

Οι τιμές του RMSE είναι μετρημένες σε χρόνια. Παρατηρούμε ότι το καλύτερο δυνατό αποτέλεσμα με ένα κρυφό επίπεδο είναι με 3 κρυφούς νευρώνες. Αυτό που επίσης παρατήρησα είναι ότι μετά τους 7 κρυφούς νευρώνες όσο προσθέτουμε το σφάλμα σαφώς αυξάνεται μόνο που η αύξηση είναι σχετικά μικρή δηλαδή με 20 κρυφούς νευρώνες το RMSE δεν ξεπερνά τα 140 χρόνια.

Μια άλλη παρατήρηση είναι ότι το λάθος ανά εποχή επηρεάζεται ελάχιστα από την αλλαγή του αριθμού των κρυφών κόμβων.

Αριθμός κρυφών νευρώνων στο κρυφό επίπεδο:	Error per epoch:
H1 = 1	0.0198369
H1 = 3	0.0193999
H1 = 7	0.0197923

(Όσον αφορά τις γραφικές παραστάσεις σύγκλισης δεν βρήκα τέτοια δυνατότητα στο weka.)

Ε. Σε αυτό το ερώτημα πρόσθεσα ένα έως και δυο κρυφά επίπεδα ακόμα, ωστόσο το αποτέλεσμα δεν ήταν ακριβώς αυτό που περίμενα. Πειραματίστηκα με αρκετούς συνδιασμούς κρυφών νευρώνων είτε σε ένα είτε δυο είτε σε τρία κρυφά επίπεδα παρόλαυτα σε κάθε περίπτωση και συνδιασμό το RMSE ήταν σχεδόν ίδιο και έδειχνε να έχει σταθεροποιηθεί σε μια τιμή συγκεκριμένη η οποία ήταν αρκετά χειρότερη από όλες τις τιμές που είχαμε με ένα μόνο κρυφό επίπεδο. Παρακάτω παραθέτω έναν πίνακα με κάποιους από τους συνδιασμούς κρυφών επιπέδων και τα αντίστοιχα RMSE.

Κρυφά επίπεδα νευρώνων:	RMSE:
H1= 1 , H2 = 1	259.828
H1 = 2, H2 = 1	259.8293
H1 = 4, H2 = 2	259.7846
H1 = 1, H2 = 2, H3 = 1	259.8329
H1 = 2, H2 = 2, H3 = 2	259.8504

Πιστεύω πως ίσως η πολυπλοκότητα του προβλήματος να μην χρειάζεται παραπάνω κρυφά επίπεδα, διότι είναι πιθανό παρά την αύξηση της εκπαιδευτικής ικανότητας του μοντέλου(λόγω έξτρα κρυφών επιπέδων), το πρόβλημα να έχει ήδη λυθεί επαρκώς με ένα μόνο επίπεδο και η προσθήκη περισσότερων να μην οδηγεί σε βελτίωση. Επίσης αφού χρησιμοποιούμε τον αλγόριθμο back-propagation ίσως λόγω περισσότερων επιπέδων απαιτείται περισσότερος χρόνος για την σύγκλιση αφού η διαδικασία ενημέρωσης των βαρών γίνεται πιο πολύπλοκη, κάτι που πιθανό να επηρεάζει και το αποτέλεσμά μας.

Παρότι στα συγκεκριμένα πειράματα δεν υπάρχει διαφορά στους διάφορους συνδιασμούς κρυφών layers και nodes προσωπικά η δομή που θα επέλεγα είναι ο αριθμός κόμβων σε κάθε κρυφό επίπεδο να μειώνεται όσο προχωράμε προς την έξοδο. Αυτό γιατί έτσι πλησιάζει όλο και περισσότερο στην εξαγωγή μοτίβων και χαρακτηριστικών ώστε να μπορέσει να αναγνωρίσει πιο εύκολα τον στόχο. Βέβαια η δομή των κρυφών κόμβων και layers δεν είναι συγκεκριμένη κάθε φορά δηλαδή συνήθως εξαρτάται από το πρόβλημα που αντιμετωπίζουμε.

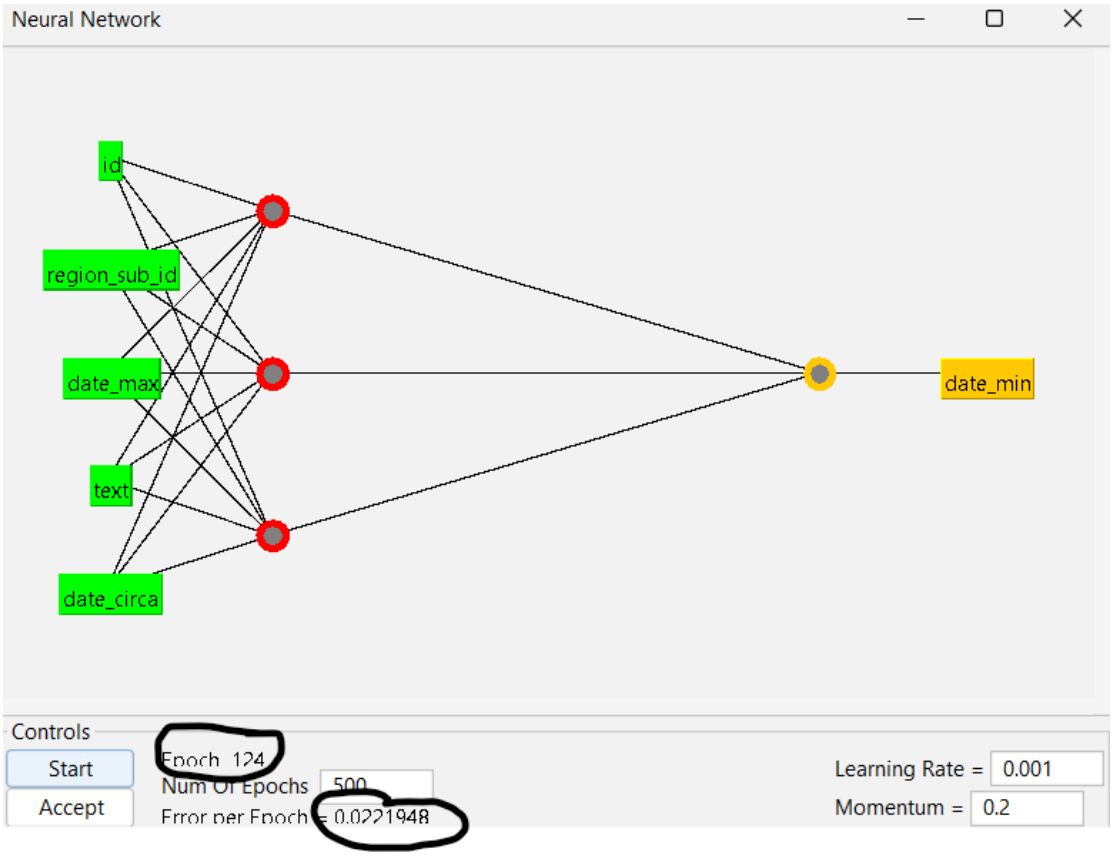
Γενικά ως προς την εύρεση συγκεκριμένου αριθμού κρυφών κόμβων για όλα τα κρυφά layers οι περισσότερες πηγές καταλήγουν στα εξής δυο συμπεράσματα:

1) Ο αριθμός των κρυφών νευρώνων πρέπει να βρίσκεται ανάμεσα στο μέγεθος του input layer και του output layer.
2) Μία κατάλληλη προσέγγιση αριθμού είναι η sqrt(input layer nodes * output layer nodes).

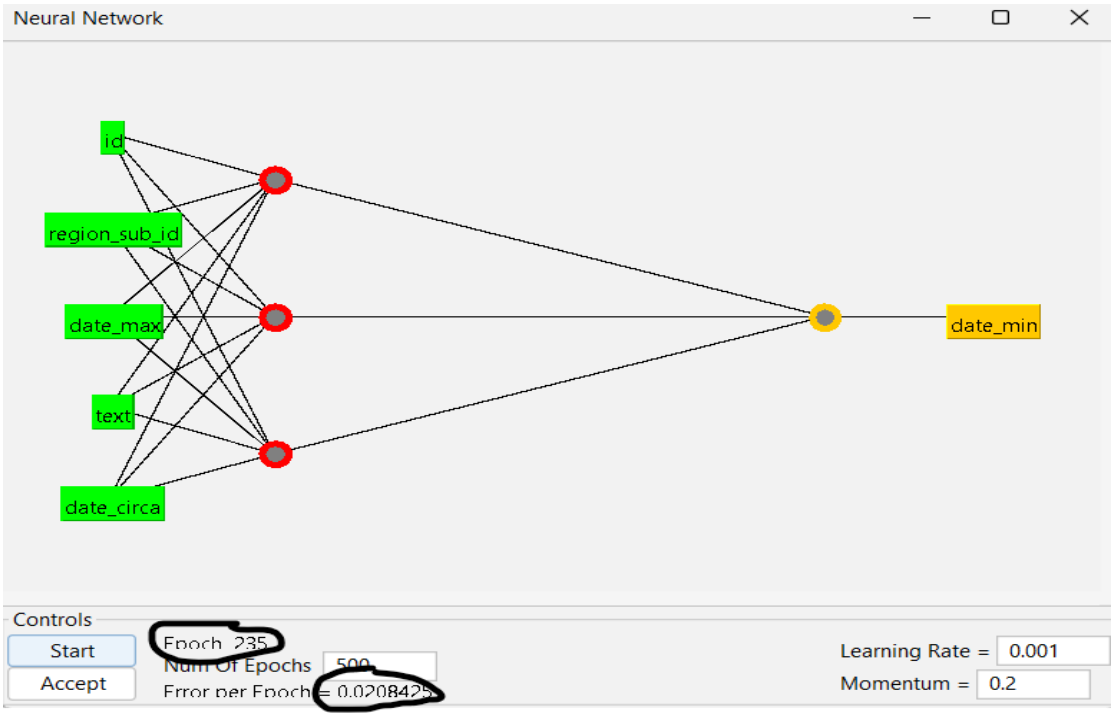
ΣΤ. Γενικά τα κριτήρια τερματισμού του αλγορίθμου back-propagation δεν αποτελούν ξεκάθαρη βάση για την λήξη του αλγορίθμου. Παρολαύτα με βάση τις παρατηρήσεις μου στα πειράματα της συγκεκριμένης εργασίας επιλέγω το κριτήριο της ικανότητας γενίκευσης του μοντέλου. Πιο συγκεκριμένα το μοντέλο έχει πιάσει την βέλτιστη απόδοσή του, αυτή είναι επαρκής, σταθερή (η σχεδόν αμετάβλητη) σε διαφορετικά σύνολα δεδομένων ελέγχου και έτσι σταματά. Έβγαλα το συμπέρασμα της σταθεροποίησης της απόδοσης του μοντέλου μου ως εξής: Όπως είχαμε προαναφέρει η διασταυρωμένη επικύρωση n-τμημάτων διαιρεί τυχαία το σύνολο των δεδομένων σε n υποσύνολα από τα οποία χρησιμοποιεί τα n-1 για εκπαίδευση και το τελευταίο 1 για έλεγχο και υπολογίζει το σφάλμα γενίκευσης. Επομένως για διαφορετικό n κάθε φορά έχουμε και διαφορετικό (σε μέγεθος) σύνολο δεδομένων ελέγχου. Για παράδειγμα άμα n = 2, όλα τα δεδομένα διαιρούνται σε 2 μέρη και το σύνολο δεδομένων ελέγχου είναι πολύ μεγάλο. Έτσι λοιπόν πήρα την καλύτερη περίπτωση μου (ένα κρυφό layer , 3 κρυφοί νευρώνες) και παρατήρησα το εξής: για 2, 3, 4 folds η RMSE δεν ξεπερνά τα 131 χρόνια και από 5 folds και μετά όσο και να διαιρέσω τα δεδομένα μου και να αλλάξω το σύνολο δεδομένων ελέγχου το RMSE παραμένει σταθερό στα 127 χρόνια που είναι και το βέλτιστο. Άρα η απόδοση του μοντέλου είναι πλέον σταθερή και βέλτιστη, κάτι που μας επιτρέπει να λήξουμε τον αλγόριθμο.

Όσον αφορά την τεχνική πρόωρου σταματήματος (early stopping) την χρησιμοποίησα ως εξής: Πήρα πάλι την βέλτιστη σε δομή εκδοχή του δικτύου μου (1 hidden layer, 3 hidden nodes) και αφού είχα ορίσει για την εκπαίδευση συνολικά 500 εποχές, σταμάταγα χειροκίνητα τον αλγόριθμο σε τυχαίες εποχές και παρατηρούσα κάθε φορά το error per epoch. Τα συμπεράσματα είναι τα ακόλουθα:

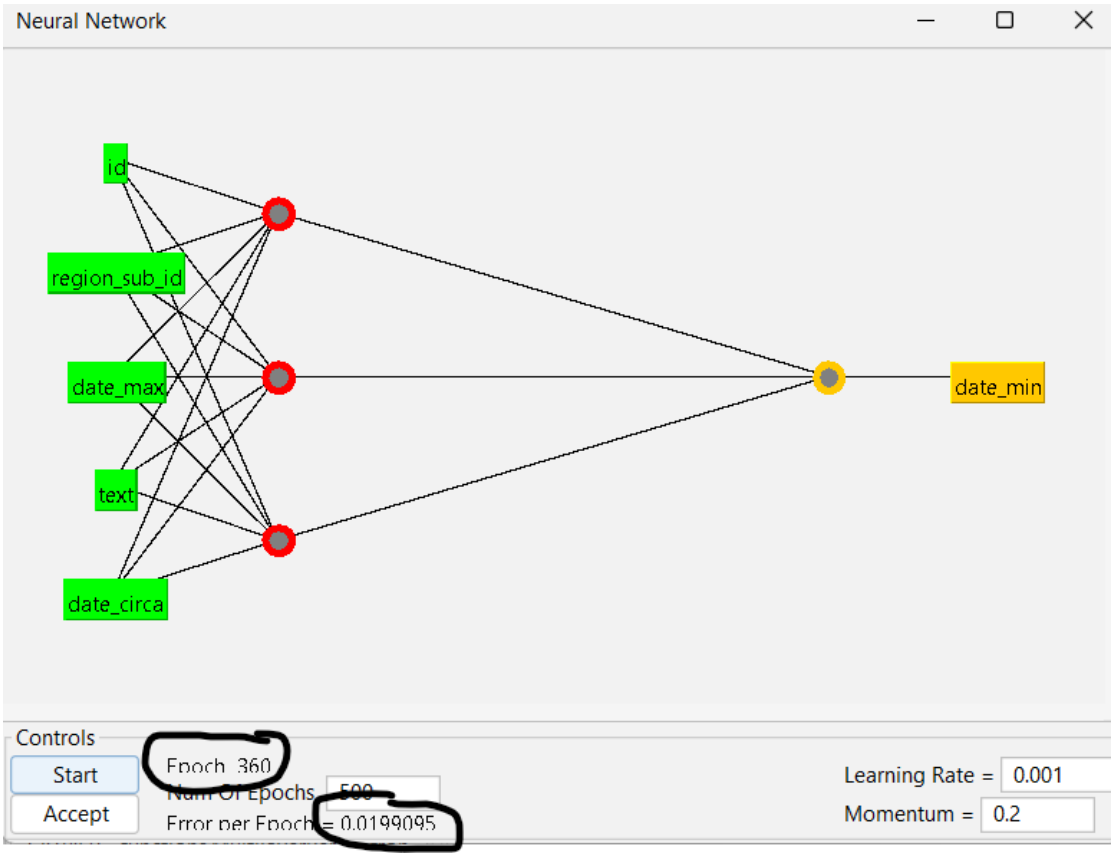
Epoch = 124, error = 0.0221948



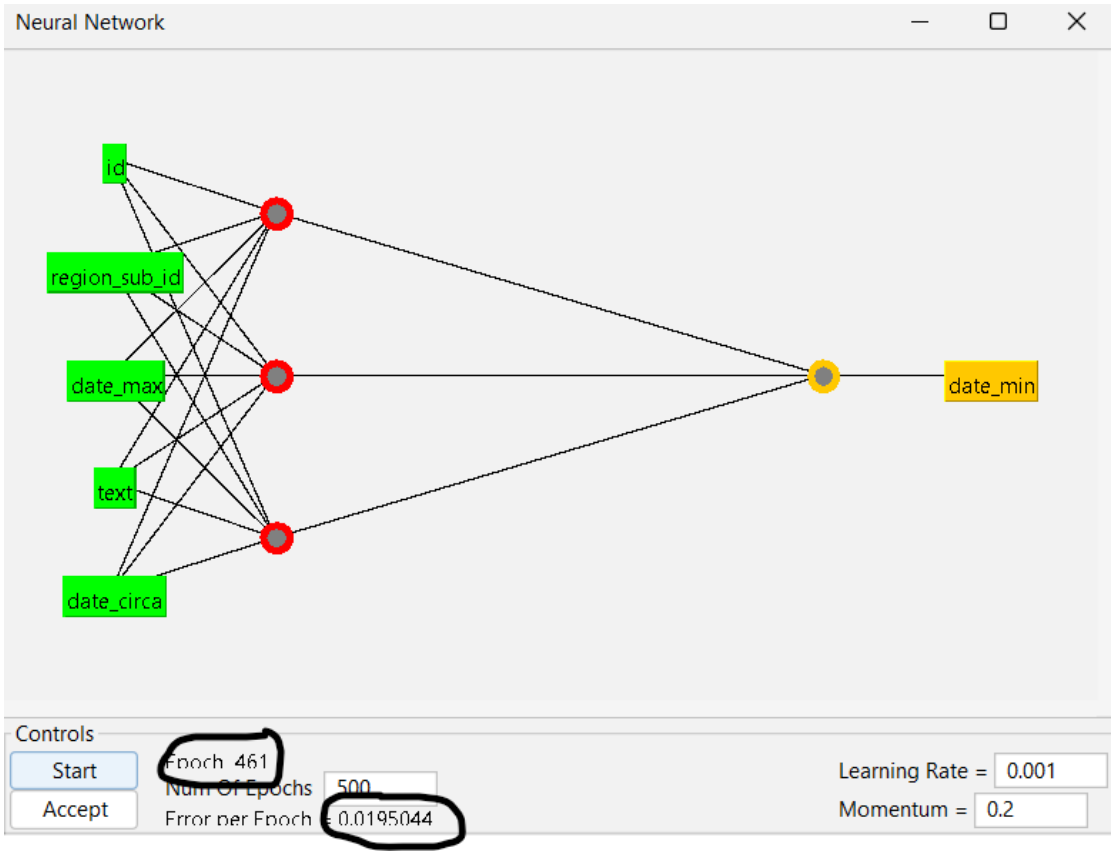
Epoch = 235, error = 0.0208425



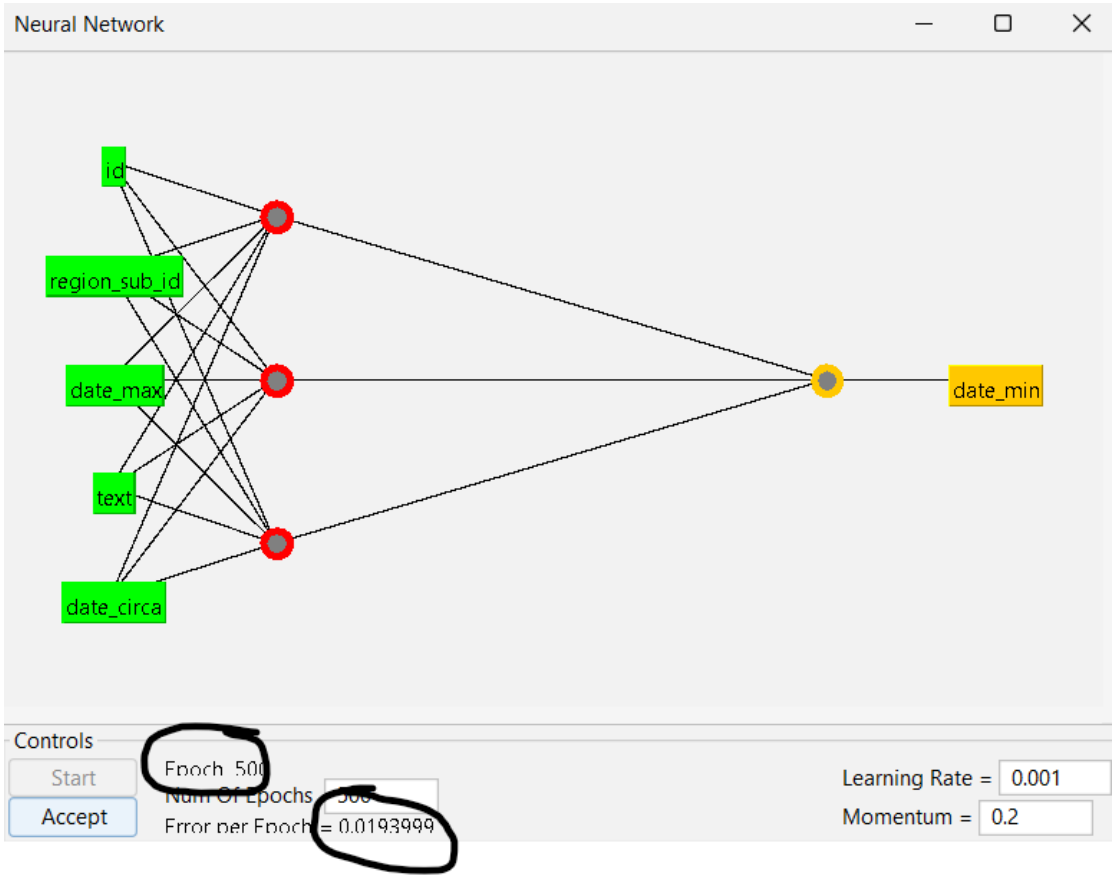
Epoch = 360 , error = 0.0199095



Epoch = 461, error = 0.0195044



Epoch = 500, 0.0193999



Αυτό που βλέπουμε είναι ότι περίπου από την εποχή 361 εώς και 500(τελική) το σφάλμα έχει γίνει σχεδόν σταθερό χωρίς περαιτέρω βελτίωση. (Η βελτίωση είναι υπερβολικά μικρή). Επομένως η σταθεροποίηση του error per epoch σε μια βέλτιστη τιμή , θα μπορούσε να αποτελέσει κριτήριο τερματισμού του αλγορίθμου, όμως παρόλαυτα η τεχνική του πρόωρου σταματήματος στο WEKA με αυτόν τον τρόπο μου φαίνεται πως δεν έχει μεγάλη ακρίβεια και επίσης η σταθεροποίηση γίνεται με βάση το λάθος ανά εποχή και όχι με βάση τις μεταβολές στο RMSE η στο μέσο τετραγωνικό σφάλμα.

Α3).Μεταβολές στον ρυθμό εκπαίδευσης και σταθερά ορμής:

Η τοπολογία που δίνει το καλύτερο αποτέλεσμα στο RMSE στο δίκτυο μας είναι με ένα κρυφό layer και τρεις κρυφούς κόμβους. Επίσης μέχρι τώρα χρησιμοποιούσαμε ρυθμό εκπαίδευσης η = 0.001. Ακολουθούν τα 4 πειράματα με αλλαγή στις δύο υπερπαραμέτρους:

η:	m:	RMSE:
0.001	0.2	127.9093
0.001	0.6	126.3005
0.05	0.6	131.9996
0.1	0.6	132.3874

Γενικά με μικρό η επιτυγχάνουμε μια ομοιόμορφη τροχιά , ακρίβεια αλλά το αρνητικό είναι ότι έχουμε αργή σύγκλιση. Άμα αυξήσουμε το η προκαλούμε επιτάχυνση δηλαδή γρήγορη σύγκλιση όμως αστάθεια. Γιαυτό προσθέτουμε τον παράγοντα ορμή. Στα παραπάνω πειράματα βλέπουμε ότι το καλύτερο δυνατό αποτέλεσμα είναι με μια σχετικά υψηλή ορμή (0.6) όμως το η είναι ακόμα χαμηλό όπως στην αρχή (0.001). Δηλαδή η αύξηση του η στην πορεία όχι μόνο δεν προκάλεσε βελτίωση, αλλά χειροτέρεψε και το αποτέλεσμα. Στο συγκεκριμένο πρόβλημα φαίνεται η αύξηση του η από 0.001 σε 0.05, 0.1 προκαλεί υπερβολική αλλαγή των βαρών κάτι που δυσκολεύει την σύγκλιση στην βέλτιστη λύση και καθιστά δύσκολη την εύρεση του τοπικού ελαχίστου. Επομένως δεν έχουμε το περιθώριο να αυξήσουμε το η τόσο πολύ. Αντίθετα παρατηρούμε ότι η αύξηση

του m από 0.2 σε 0.6 δίνει βελτίωση διότι στο υψηλότερο momentum οι ενημερώσεις των βαρών είναι περισσότερο επηρεασμένες από τις προηγούμενες ενημερώσεις. Αυτό οδηγεί σε μεγαλύτερη ταχύτητα σύγκλισης και καλύτερη ευστάθεια. Τέλος η αύξηση του momentum μειώνει την «ευαισθησία» που αποκτά το η κατά την κατάβαση και αποτρέπει το να κολλήσουν οι παράμετροι στα τοπικά ελάχιστα.

Γιατί m<1;

Σύμφωνα με τον γενικευμένο κανόνα δ ,Δw(n), αν βάλουμε m>1 τα βάρη σε προηγούμενα παραδείγματα training θα αυξηθούν εκθετικά. Αυτό αυξάνει τα βάρη σε πολύ μεγάλο βαθμό ακόμα και αν m=1,01 δηλαδή η εκθετική αύξηση σε μόλις λίγες επαναλήψεις είναι μεγάλη. Κάτι τέτοιο δεν το θέλουμε διότι οι παλαιότερες μεταβολές πρέπει να βαρύνουν λιγότερο. Γιαυτόν τον λόγο το momentum κυμαίνεται από 0 εώς 1 που σημαίνει ότι στον εκθετικό τύπο του κανόνα τα βάρη μειώνονται.

Παρακάτω παραθέτω το path για τον κώδικά μου python ο οποίος περιέχει την ανάγνωση του αρχείου csv, τις μετατροπές του text σε tf-idf μορφή και την δημιουργία του αρχείου arff που χρησιμοποίησα στην πορεία:

https://github.com/kostisvass/NEURAL_NETWORKS_PART_A

Να σημειωθεί ότι το αρχείο NEURAL.py είναι ο κώδικας python και το iphi2802.csv το αρχείο με τα data.