GITHUB Link: https://github.com/AlexKanakis5/Keras_AGTexts/tree/main

Η εργασία υλοποιήθηκε με τη χρήση python και με τις βιβλιοθήκες scikit-learn & keras.

A1 - a) Για το tokenization και vectorization των λέξεων του κειμένου χρησιμοποιήθηκε η tf-idf τεχνική

```
tfidf_vectorizer = TfidfVectorizer(min_df=2, max_df=0.8)
```

Με αυτόν τον τρόπο όχι μόνο επιτυγχάνεται ο μετασχηματισμός των δεδομένων σε τιμές που μπορούν να δοθούν ως είσοδος στο μοντέλο αλλά επιτυγχάνονται και άλλα πλεονεκτήματα όπως το οτι βάζει διαφορετικά βάρη σε λέξεις ανάλογα με τη συχνότητα εμφάνισής τους και την εξάλιψη stopwords. Αυτό σημαίνει ότι λέξεις που είναι κοινές στα περισσότερα κείμενα της συλλογής δεν θα είναι τόσο σημαντικές για την πρόβλεψη των ημερομηνιών ενώ λέξεις που εμφανίζονται πολύ συχνά σε κάποια κείμενα αλλα όχι συνολικά στη συλλογή θα έχουν μεγαλύτερο βάρος.

Επίσης χρησιμοποιήθηκε ένα επιπλέον φιλτράρισμα των λέξεων προκειμένου να μειωθει το μέγεθος του λεξιλογίου και να γίνει γρηγορότερα η εκπαιδευση, περιορίζοντας τις λέξεις που εμφανίζοντα πάρα πολύ συχνά ή μόνο 1-2 φορές σε όλη τη συλλογή, άρα και δεν θα ήταν χρήσιμες για τις προβλέψεις.

Without min_df: 20896 tokens
With min_df=5: 1460 tokens
With min_df = 2: 5625 tokens
max_df: 0.8 δεν παρατήρησα κάποια διαφορά στο σύνολο των tokens καθώς δεν υπάρχουν λέξεις που εμφανίζονται >80% των κειμένων

A1 - b)

Δεν χρησιμοποιήθηκε κάποιο scaling στην είσοδο καθώς οι τιμές είναι ήδη scaled από την tf-idf κωδικοποίηση. Για τα δεδομένα της εξόδου πάλι δεν χρησιμοποιήθηκε scaling καθώς το mse όταν το μοντέλο πρέβλεπε τις ημερομηνίες φαινόταν πολύ μικρό ενώ στην πραγματικότητα μια μικρή απόκλιση θα ήταν πολύ μεγαλύτερη μετά το scaling.

A2 – α)

Προτιμήθηκε η πρόβλεψη της μέσης τιμής των άκρων του διαστήματος των ημερομηνιών

A2 – β)

Η συνάρτηση ενεργοποίησης είναι η relu. Είναι η πιο συχνή συνάρτηση ενεργοποίησης και προσφέρει διάφορα πλεονεκτήματα όπως μεγαλύτερη αποδοτικότητα στην εκπαίδευεση σε σχέση με την tahn και την sigmoid, απογεύγει το Vanishing Gradient Problem

A2 – γ)

Η συνάρτηση είναι linear καθώς δίνει ως έξοδο ένα εύρος τιμών όπως το απαιτούμενο, δηλαδή τις ημερομηνίες, πράγμα που άλλες συναρτήσεις όπως η softmax δεν μπορούν να κάνουν.

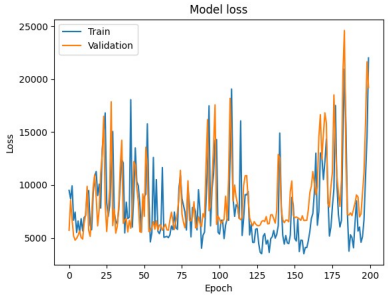
A2 – δ)

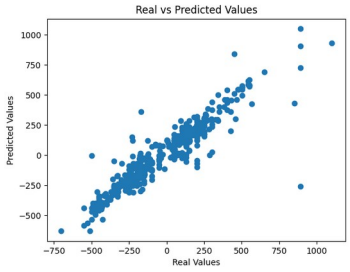
Το RMSE είναι το μέσο RMSE στα 5 folds που δημιουργήθηκαν. Η μέτρηση έγινε αφού το μοντέλο εκπαιδεύτηκε για 200 epochs.

Αριθμός νευρώνων στο κρυφό επίπεδο	RMSE
H1 = 128	29145.829296875
H1 = 64	28638.330078125
H1 = 256	22551.623828125
H1 = 512	23746.85283203125

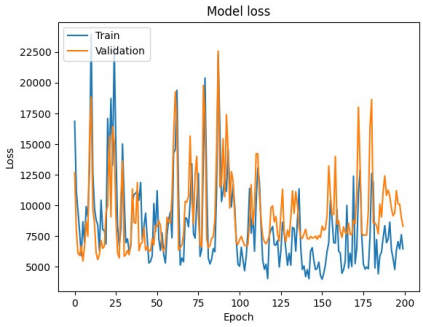
Οι παρακάτω παραστάσεις δείχουν την μεταβολή του loss σε σχέση με το epoch και την σχέση μεταξύ των πραγματικών και τον προβλεπόμενων τιμών στο τελευταίο fold, για κάθε τιμή του H1.

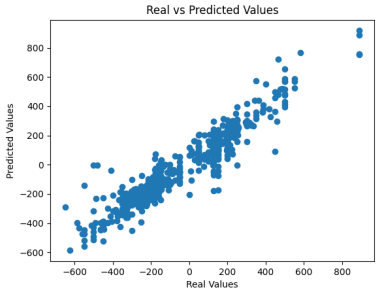
128:



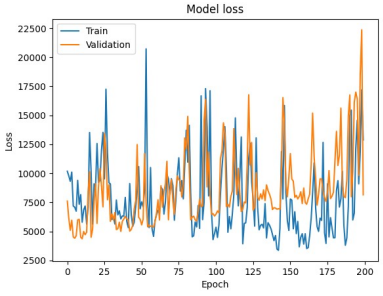


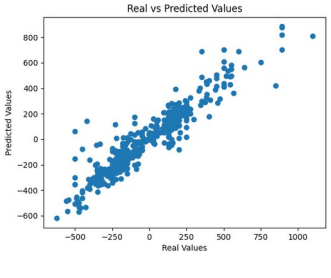
64:



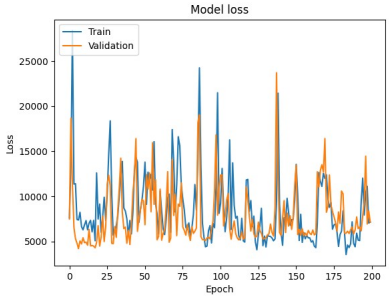


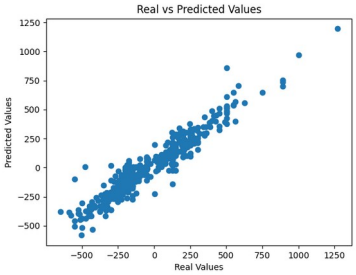
256:





512:

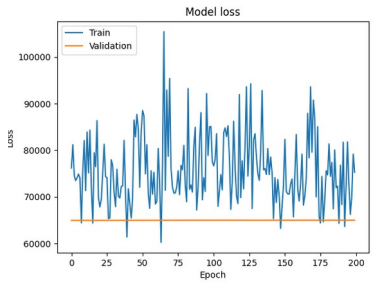


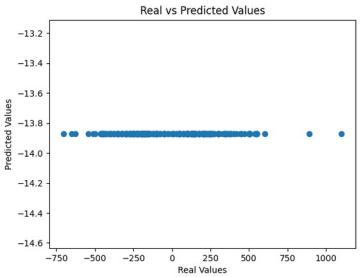


Τα μοντέλα με τον μεγαλύτερο αριθμό κόμβων έχουν μικρότερο error και οι τιμές που προβλέπουν φαίνεται να είναι πιο κοντά στις πραγματικές τιμές. Στο παραπάνω τύπο γραφήματος το ιδανικό θα ήταν όλες να βρίσκονται πάνω στη διαγώνιο πράγμα που συμβαίνει περισσότερο στο μοντέλο των 512 κόμβων. Αυτό μπορεί να οφείλεται στο γεγονός ότι τα δεδομένα της εισόδου είναι μεγάλων διαστάσεων και ένα μοντέλο με περισσότερους κόμβους μπορεί να ερμηνεύσει αυτή την πολυπλοκότητα.

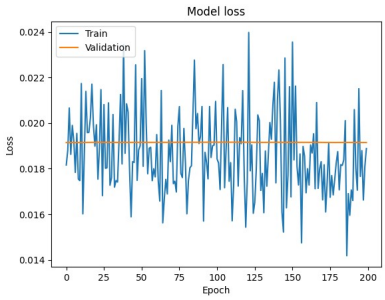
Α2 – ε) Δεν ισχυει πλεον. Η λύση στο πρόβλημα που αντιμετωποίησα ήταν να αλλάξω το optimizer από SGD σε Adam. Ωστόσο αφήνω αυτά που έγραψα εδώ για να φανεί η πορεία που ακολούθησα

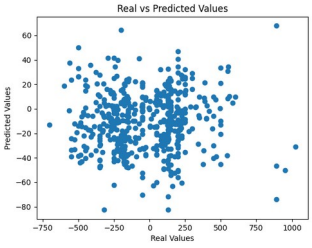
Με τη χρήση ενός ακόμη hidden layer η τιμή που προβλέπεται παραμένει πάντα η ίδια. Δεν μπορώ να εξηγήσω γιατί, υποψιάζομαι ότι έχει να κάνει με το activation function αλλά ό,τι δοκίμασα δεν άλλαξε το αποτέλεσμα.



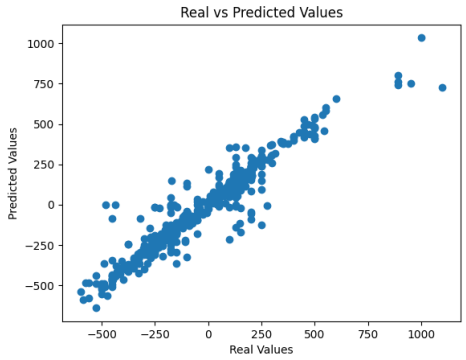


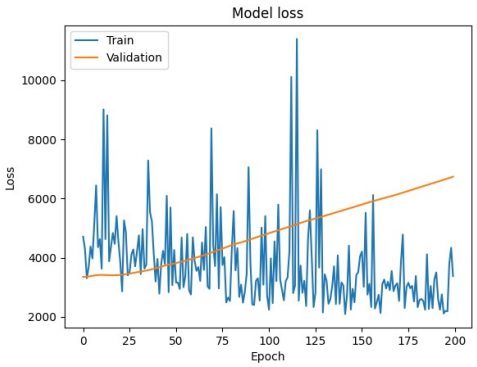
Αυτό δεν συμβαίνει όταν κάνω min-max scaling στην y μεταβλητή. Ωστόσο οι προβλέψεις του μοντέλου είναι πολύ κακές





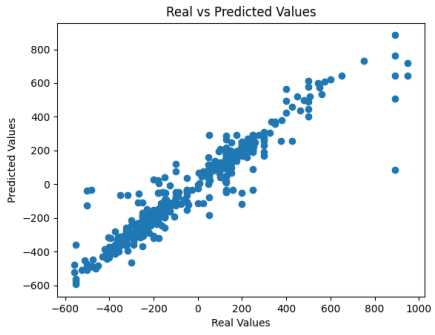
Α2 – ε)
Παρακάτω είναι οι γραφικές παραστάσεις μετά τη χρήση ενός ακόμη hidden layer. Το πρώτο layer έχει 128 κόμβους και το δεύτερο 64

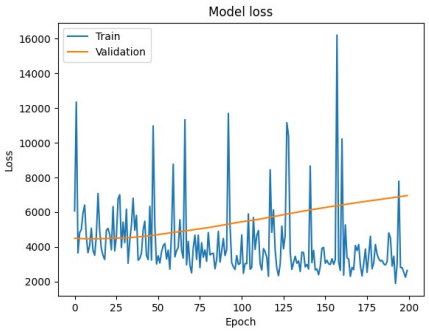




RMSE:16303

Το πρώτο layer έχει 64 κόμβους και το δεύτερο 128





RMSE:16237

Δεν υπάρχει κάποιος κανόνας που να ορίζει ποια στρατηγική είναι αυστηρά καλύτερη. Γενικά τα πλεονεκτήματα που προσφέρει ο αυξανόμενος αριθμός κόμβων είναι ότι το μοντέλο μπορεί να μάθει πιο σύνθετα μοτίβα στα δεδομένα ωστόσο μπορεί να οδηγίσει σε overfitting. Το αντίθετο μπορεί να συμβεί όταν έχουμε μειούμενο αριθμό κόμβων. Στην δική μου περίπτωση δεν φαίνεται να αλλάζει σημαντικά κάτι επομένως χρησιμοποιώ μειόυμενο αριθμό κόμβων που είναι η πιο συχνή τακτική.

Όσον αφορά τη σύγκριση με το ένα κρυφό επίπεδο, το RMSE φαίνεται να μειώνεται σημαντικά

Α2 – στ)

Μία συνθήκη πρόωρου τερματισμού θα μπορούσε να είναι όταν το RMSE αρχίζει να αυξάνεται σε καθε fold. Στην δική μου περίπτωση ωστόσο, μπορεί να αυξηθεί πριν αρχίσει να πέφτει και εφόσον έχω μόνο 5 folds αποφάσισα να μην την χρησιμοποιήσω.

Fold : 0 RMSE: 37573.06640625

17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
Fold : 1 RMSE: 40191.65625
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Fold : 2 RMSE: 10456.595703125
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

A3) Δεν ισχυει πλεον. Η λύση στο πρόβλημα που αντιμετωπίσα ήταν να αλλάξω το optimizer από SGD σε Adam. Ωστόσο αφήνω αυτά που έγραψα εδώ για να φανεί η πορεία που ακολούθησα

Εκπαίδευση για 200 epochs, RMSE → mean RMSE over 5 folds.

n	m	RMSE
0.001	0.2	29186.38740234375
0.001	0.6	19714.94091796875
0.05	0.6	Η predicted τιμή είναι σταθερή
0.1	0.6	Η predicted τιμή είναι σταθερή

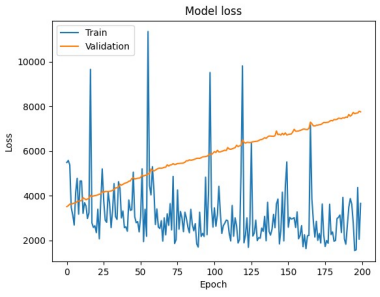
Ειλικρινά, δεν καταλαβαίνω γιατί όταν αλλάζω αυτές τις παραμέτρους παίρνω πάντα μια σταθερή predicted τιμή (π.χ. [-300.5] -28.81951904296875
[-250.5] -28.81951904296875
όπου αριστερά οι πραγματικές τιμές και δεξιά οι predited)

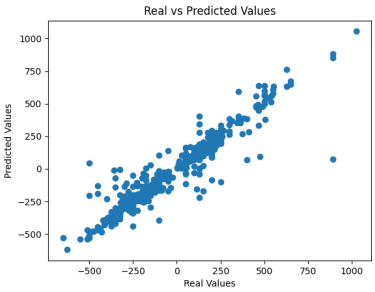
A3)

Εκπαίδευση για 200 epochs, RMSE → mean RMSE over 5 folds.

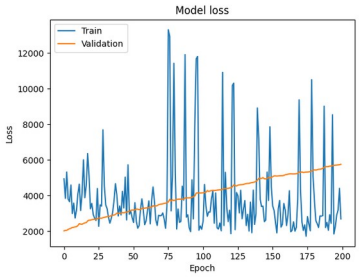
n	m	RMSE
0.001	0.2	14786
0.001	0.6	15778
0.05	0.6	12782
0.1	0.6	11533

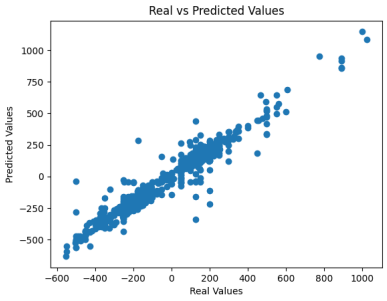
0.001/0.2



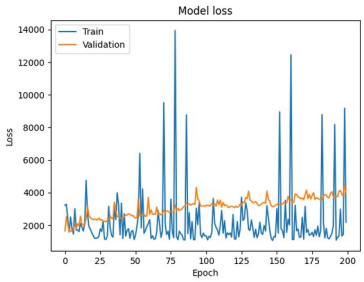


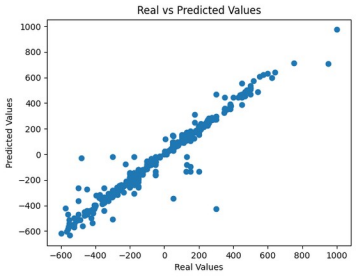
0.001/0.6



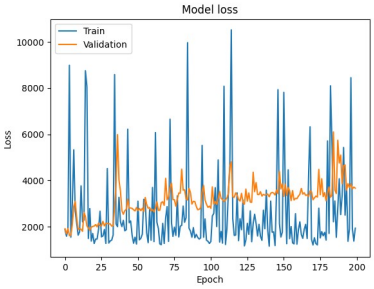


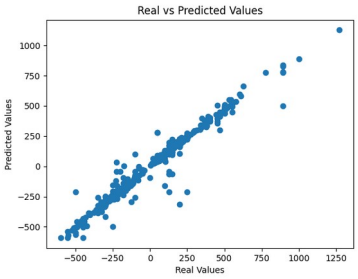
0.05/0.6





0.1/0.6:





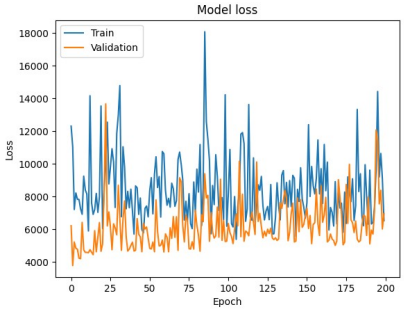
Από τα παραπάνω φαίνεται ότι η αύξηση του learning rate οδηγεί σε γρηγορότερη σύγκλιση, άρα και σε μειούμενο RMSE και η απόκλιση των πραγματικών τιμών από των προβλεπόμενων είναι μικρότερη. Αυτό ενδεχομένως να σημαίνει ότι σε μικρότερο learning rate θα βοηθούσε η εκπαίδευση να γίνεται για περισσότερα βήματα.

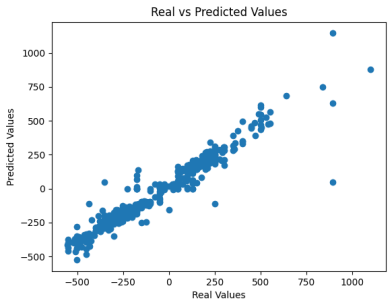
Το momentum λαμβάνει τη μορφή ενός εκθετικού κινητικού μέσου, όπου η τρέχουσα ενημέρωση των παραμέτρων συμβάλλει με ένα βάρος β ένω η προηγούμενη ενημέρωση συμβάλλει με ένα βάρος (1−β). Όταν είναι μικρότερο του 1 τότε η ενημέρωση των παραμέτρων λαμβάνει υπόψη την προηγούμενη κατεύθυνση. Επομένως όταν το momentum είμαι μεγαλύτερο του 1 τότε το 1-β θα ήταν αρνητικό το οποίο δεν βγάζει νόημα.

A4)

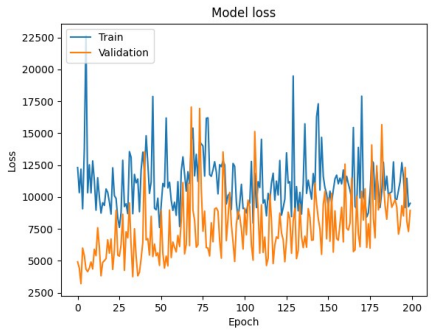
Πιθανότητες διαρήρησης	RMSE
rin =0.8 rh = 0.5	16661
rin =0.5 rh = 0.5	15175
rin =0.8 rh = 0.2	21147

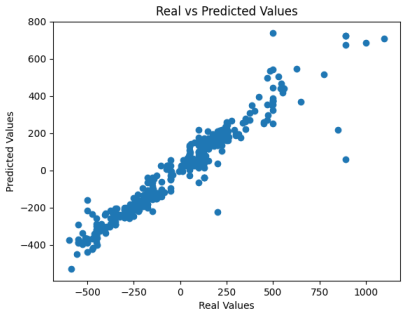
0.8/0.5



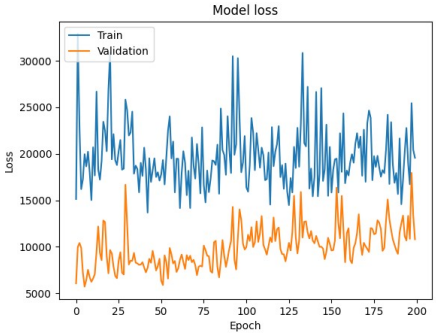


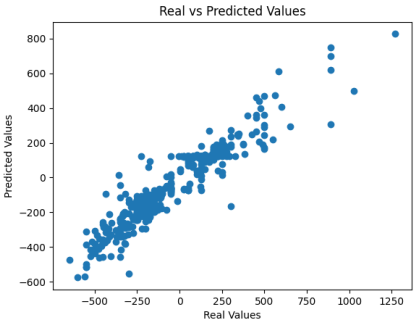
0.5/0.5





0.8/0.2





Θεωρητικά η εφαρμογή του dropout θα βοηθούσε στην αποφηγή του φαινομένου overfitting. Επίσης, στην προκειμένη περίπτωση αυξήθηκε το RMSE καθώς και η απόκλειση των πραγματικών και προβλεπόμενων τιμών.