ΕΠΕΞΕΡΓΑΣΙΑΣ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Πρόβλεψη Αναφορών - Citation Link Prediction

Εαρινό Εξάμηνο 2024-2025

Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Πληροφορίες Ομάδας

Όνομα Ομάδας: Link Lords

Όνομα Μέλους 1: Καϊλίδης Κύριλλος ΑΜ: 4680

Όνομα Μέλους 2: Κερασοβίτης Ηλίας ΑΜ: 4699

1. Εισαγωγή

Στο πλαίσιο του παρόντος project , στόχος μας είναι η ανάπτυξη ενός προγράμματος σε python όπου θα πρέπει να προβλέπει εάν ένα άρθρο παραπέμπει σε κάποιο άλλο. Μας παρέχεται ένα σύνολο δεδομένων που περιλαμβάνει υπάρχουσες παραπομπές μεταξύ άρθρων , καθώς και τα ίδια τα άρθρα μαζί με πληροφορίες για τους αρθρογράφους τους .

Αξιοποιώντας τεχνικές επεξεργασίας φυσικής γλώσσας (NLP) , μπορούμε να εξάγουμε σημασιολογικά χαρακτηριστικά από το περιεχόμενο των άρθρων. Παράλληλα μέσω των υπάρχουσων παραπομπών μπορούμε να κατασκευάσουμε ένα γράφο από τον οποίο θα πάρουμε επιπλέων χαρακτηριστικά χρήσιμα για την πρόβλεψη.

Για το τελικό μοντέλο πρόβλεψης θα χρησιμοποιηθεί μια stacking ensemble προσέγγιση η οποία θα συνδυάζει τα αποτελέσματα 3 μοντέλων μηχανικής μάθησης ενισχύοντας την ακρίβεια και την αξιοπιστία των προβλέψεων.

2. Προεπεξεργασία-Preprocessing

Φόρτωμα και επεξεργασία αρχείων

Αρχικά φορτώνουμε όλα τα αρχεία (abstract.txt , authors.txt , edgelist.txt) και τα μετατρέπουμε σε Dataframe με χρήση της βιβλιοθήκης pandas .

Κατά την διαδικασία αυτή με κατάλληλο χειρισμό των delimiters διαχωρίζουμε τα πεδία και τοποθετούμε τις πληροφορίες στις αντίστοιχες στήλες.

Δημιουργία γράφου

Για την κατασκευή του γράφου παραπομπών (citation graph), αρχικά συγκεντρώνουμε όλα τα ζεύγη άρθρων που συνδέονται μέσω αναφορών (edges). Στη συνέχεια, εφαρμόζουμε τη συνάρτηση train_test_split με αναλογία 80/20, ώστε να διαχωρίσουμε τα edges σε σύνολα εκπαίδευσης και επικύρωσης (training και validation).

Αφού πραγματοποιηθεί ο διαχωρισμός των παραπομπών (edges) σε training και validation σύνολα, εντοπίζονται όλοι οι κόμβοι (nodes) που συμμετέχουν σε αυτά. Στη συνέχεια, κατασκευάζουμε τον γράφο χρησιμοποιώντας μόνο τις ακμές του training set, διατηρώντας ωστόσο όλους τους κόμβους, δηλαδή και αυτούς που συμμετέχουν αποκλειστικά στο validation set. Με αυτόν τον τρόπο, αποφεύγεται η ενσωμάτωση των validation ακμών στον γράφο, γεγονός που αποτρέπει τη διαρροή πληροφορίας (data leakage) κατά την εξαγωγή χαρακτηριστικών από τη δομή του γράφου.

Η προσέγγιση αυτή μας επιτρέπει να υπολογίσουμε χαρακτηριστικά (features) βασισμένα στη δομή του γράφου, χωρίς να εισάγεται εκ των προτέρων γνώση από το validation set. Έτσι διασφαλίζεται ότι οι μετρικές απόδοσης που υπολογίζονται στο validation set είναι όσο το δυνατόν πιο αντιπροσωπευτικές της γενικευσιμότητας του μοντέλου, προσομοιώνοντας πιο ρεαλιστικά τις συνθήκες πρόβλεψης του διαγωνισμού στο Kaggle.

Δημιουργία θετικών και αρνητικών δειγμάτων

Για να μπορεί να εκπαιδευτεί το μοντέλο, απαιτείται η ύπαρξη τόσο θετικών όσο και αρνητικών δειγμάτων. Τα θετικά δείγματα αντιστοιχούν στα υπάρχοντα edges του συνόλου εκπαίδευσης και επικύρωσης (train_edges, val_edges), δηλαδή σε περιπτώσεις όπου υφίσταται πράγματι παραπομπή μεταξύ δύο άρθρων.

Για τη δημιουργία αρνητικών δειγμάτων (negative pairs), χρησιμοποιούμε την συνάρτηση generate_negative_pairs, η οποία δημιουργεί τυχαία ζεύγη άρθρων που δεν έχουν μεταξύ τους αναφορά. Κατά την αρνητική δειγματοληψία φροντίζουμε:

- να μην υπάρχουν ζεύγη όπου συμμετέχει το ίδιο άρθρο δύο φορές,
- να μην περιλαμβάνονται ζεύγη που ήδη βρίσκονται στα θετικά δείγματα.

Ο αριθμός των αρνητικών pairs επιλέγεται έτσι ώστε να είναι ίσος με τον αριθμό των θετικών, διατηρώντας ισορροπία μεταξύ των δύο τάξεων (class balance) τόσο στο training όσο και στο validation set. Αυτό είναι κρίσιμο για την αποφυγή προκατάληψης του ταξινομητή υπέρ της πλειοψηφούσας κλάσης και για τη διατήρηση σταθερών μετρικών αξιολόγησης.

3. Εξαγωγή Χαρακτηριστικών

Bert embedding

Για την εξαγωγή των ενσωματώσεων (embeddings) από τα abstracts, χρησιμοποιούμε το προεκπαιδευμένο μοντέλο **BERT** (Bidirectional Encoder Representations from Transformers). Το BERT διαβάζει κάθε πρόταση αμφίδρομα (bidirectionally), δηλαδή ταυτόχρονα από την αρχή και το τέλος της, γεγονός που του επιτρέπει να κατανοεί το νόημα των λέξεων με βάση τα συμφραζόμενα τους — σε αντίθεση με απλούστερα μοντέλα που κωδικοποιούν κάθε λέξη ανεξάρτητα από το περιεχόμενο που την περιβάλλει.

Για παράδειγμα, η λέξη “apple” αποκτά διαφορετικό νοήμα όταν αναφέρεται στο φρούτο από ό,τι όταν αναφέρεται στην εταιρεία. Αυτή η δυνατότητα σημασιολογικής διάκρισης είναι ένα από τα βασικά πλεονεκτήματα των BERT embeddings σε σύγκριση με στατικές αναπαραστάσεις.

Στο κομμάτι του κώδικα χρησιμοποιούμε την βιβλιοθήκη torch για να εκμεταλευτούμε την ταχύτητα που προσφέρει η κάρτα γραφικών στην επεξεργασία των abstract . Σε περίπτωση που δεν υπάρχει διαθέσιμη κάρτα γραφικών με cuda cores το μοντέλο τρέχει στον επεξεργαστή.

Για να διατηρούμε σταθερή χρήση μνήμης (VRAM), αξιοποιώντας όσο το δυνατόν μεγαλύτερο μέρος της χωρίς να την υπερφορτώνουμε — και αποφεύγοντας απώλειες ταχύτητας ή προβλήματα που σχετίζονται με ανεπαρκή διαθέσιμη μνήμη — επιλέξαμε (batch size) ίσο με 8. Με τον τρόπο αυτό επιτυγχάνεται ισορροπία ανάμεσα στην ταχύτητα επεξεργασίας και την αξιοποίηση των διαθέσιμων υπολογιστικών πόρων.

Κατά την εκτέλεση του κώδικα, εντός της βασικής επαναληπτικής δομής (loop), χρησιμοποιούμε τη βιβλιοθήκη tqdm για την παρακολούθηση της προόδου επεξεργασίας κάθε δέσμης. Επιπλέον, εφαρμόζουμε συμπλήρωση (padding) και περικοπή (truncation) στις προτάσεις, με μέγιστο μήκος max_length=512, ώστε όλα τα κείμενα να έχουν το ίδιο μήκος και να αποφεύγονται σφάλματα κατά την παρτίδα επεξεργασίας.

Το μοντέλο εκτελείται εντός torch.no_grad() για την απενεργοποίηση της παρακολούθησης γραφών παραγώγων (gradients) — κάτι που μειώνει την κατανάλωση μνήμης και επιταχύνει την επεξεργασία. Επιπρόσθετα, χρησιμοποιείται το torch.amp.autocast(device_type="cuda", dtype=torch.float16) για πρόβλεψη με μεικτή ακρίβεια (mixed-precision inference), μια τεχνική που ενισχύει την ταχύτητα και μειώνει τις απαιτήσεις σε VRAM, χωρίς να προκαλεί ουσιαστική απώλεια ακρίβειας.

Μετά την επεξεργασία κάθε δέσμης abstracts, προκύπτει ένας πίνακας διαστάσεων (8, 768). Πάνω σε αυτόν εφαρμόζουμε μέση συγκέντρωση (mean pooling), προκειμένου να παραχθεί ένα σταθερό διάνυσμα 768 διαστάσεων για κάθε abstract.

Τέλος, εφαρμόζουμε PCA (Κύριες Συνιστώσες – Principal Component Analysis) για μείωση διαστάσεων, διατηρώντας το 95% της συνολικής διακύμανσης. Με τον τρόπο αυτό καταλήγουμε σε ένα διάνυσμα 225 διαστάσεων, το οποίο είναι πολύ πιο αποδοτικό υπολογιστικά, ενώ διατηρεί την κρίσιμη πληροφορία για τα επόμενα στάδια της μοντελοποίησης.

Node2Vec Embeddings (Ενσωματώσεις Κόμβων με Node2Vec)

Για την αναπαράσταση της δομής του γράφου παραπομπών, εφαρμόσαμε τον αλγόριθμο Node2Vec (ενσωματώσεις κόμβων γραφήματος – node embeddings). Συγκεκριμένα, παράγουμε πολλαπλούς τυχαίους περιπάτους (random walks) μήκους 20 κόμβων στον γράφο. Οι ακολουθίες αυτές χρησιμοποιούνται ως «προτάσεις» για την εκπαίδευση ενός μοντέλου τύπου skip-gram με χρήση του Word2Vec (μέγεθος διανύσματος: vector_size=64, παράθυρο: window=10). Μέσω αυτής της διαδικασίας, κάθε κόμβος αποκτά μια πυκνή αναπαράσταση που ενσωματώνει τόσο την τοπική γειτνίαση όσο και τη θέση του στο παγκόσμιο πλαίσιο του γράφου.

Για να διατηρούμε σταθερές τις αναπαραστάσεις, αποθηκεύουμε τα Node2Vec embeddings σε αρχείο .kv και τα επαναχρησιμοποιούμε σε μελλοντικές εκτελέσεις, χωρίς να απαιτείται επανεκπαίδευση.

Τέλος, εφαρμόζουμε PCA (Κύριες Συνιστώσες – Principal Component Analysis) με στόχο να διατηρήσουμε το 95% της συνολικής διακύμανσης, μειώνοντας τις αρχικές 64 διαστάσεις χωρίς ουσιαστική απώλεια πληροφορίας. Το τελικό διάνυσμα αναπαράστασης είναι πιο συμπαγές και υπολογιστικά αποδοτικό για χρήση από τα μοντέλα πρόβλεψης.

Cosine Similarity

Η συνάφεια συνημιτόνου (cosine similarity) μετρά την ομοιότητα δύο διανυσμάτων ως προς την κατεύθυνσή τους, ανεξαρτήτως μεγέθους. Με άλλα λόγια, αξιολογεί το κατά πόσο δύο διανύσματα "κοιτούν προς την ίδια κατεύθυνση", χωρίς να επηρεάζεται από τη «διάρκεια» ή το μέτρο τους.

Στην περίπτωσή μας, υπολογίζουμε τη συνάφεια μεταξύ ενσωματώσεων (embeddings) που έχουν εξαχθεί είτε από τα abstracts μέσω BERT είτε από τον γράφο παραπομπών μέσω Node2Vec. Ο δείκτης cosine similarity παίρνει τιμές από 0 (καμία ομοιότητα) έως 1 (απόλυτη ταύτιση), παρέχοντας έτσι ένα σαφές μέτρο συγγένειας ανάμεσα σε δύο άρθρα.

Αυτό το μέτρο ενσωματώνεται στο διάνυσμα χαρακτηριστικών (feature vector) κάθε ζεύγους άρθρων και βοηθά τα μοντέλα να εκτιμήσουν πόσο σχετικά είναι δύο άρθρα, ενισχύοντας τη δυνατότητα πρόβλεψης ύπαρξης παραπομπής μεταξύ τους.

Pagerank

Το PageRank μετράει πόσο σημαντικός είναι ένας κόμβος μέσα στο γράφο (graph), δίνοντας υψηλό σκορ σε αυτούς που αναφέρονται από άλλους ήδη υψηλού σκορ κόμβους. Ξεκινάμε δίνοντας ίση αρχική τιμή σε όλα τα papers και, επαναλαμβανόμενα, κάθε άρθρο μοιράζει το score του εξίσου σε όλα τα άρθρα που παραπέμπει. Έτσι, δύο άρθρα με υψηλό άθροισμα PageRank scores έχουν μεγαλύτερη πιθανότητα να παραπέμπουν το ένα στο άλλο.

Clustering Coefficient

Ο συντελεστής συνεκτικότητας (clustering coefficient) ενός κόμβου μετρά τον βαθμό στον οποίο οι γείτονές του είναι επίσης συνδεδεμένοι μεταξύ τους, σχηματίζοντας τοπικές τριγωνικές δομές στον γράφο. Όσο υψηλότερη είναι αυτή η τιμή, τόσο πιο έντονη είναι η ένταξη του κόμβου σε πυκνά συνδεδεμένες τοπικές κοινότητες.

Στο πλαίσιο της πρόβλεψης παραπομπών (link prediction), για κάθε ζεύγος άρθρων υπολογίζουμε το άθροισμα των clustering coefficients των δύο κόμβων. Αυτό επιτρέπει την

εκτίμηση της πιθανότητας τα δύο άρθρα να συμμετέχουν σε κοινές τοπικές δομές, όπου είναι αυξημένες οι πιθανότητες αλληλοαναφοράς.

Common Neighbors

Η μετρική common neighbors (κοινοί γείτονες) βασίζεται στην υπόθεση ότι αν δύο κόμβοι έχουν σημαντικό αριθμό κοινών γειτόνων, τότε είναι πιο πιθανό να συνδέονται και απευθείας μεταξύ τους. Ο αριθμός αυτών των κοινών γειτόνων λειτουργεί ως ένδειξη δομικής εγγύτητας και συνεπώς αυξημένης πιθανότητας ύπαρξης παραπομπής μεταξύ των άρθρων.

Jaccard Coefficient

Ο συντελεστής Jaccard μετρά τη σχετική ομοιότητα μεταξύ των συνόλων των γειτόνων δύο κόμβων. Υπολογίζεται ως το πηλίκο του αριθμού των κοινών γειτόνων προς το συνολικό αριθμό των μοναδικών γειτόνων των δύο κόμβων.

Όταν δύο άρθρα μοιράζονται υψηλό ποσοστό κοινού γειτονικού περιβάλλοντος, η τιμή του συντελεστή Jaccard πλησιάζει το 1, υποδεικνύοντας ότι βρίσκονται σε παρόμοιο τοπικό πλαίσιο και άρα είναι πιο πιθανό να συνδεθούν.

Adamic–Adar Index

Ο δείκτης Adamic–Adar εκτιμά την πιθανότητα σύνδεσης μεταξύ δύο κόμβων, δίνοντας μεγαλύτερη βαρύτητα στους σπάνιους κοινούς γείτονες. Συγκεκριμένα, για κάθε κοινό γείτονα u των κόμβων x και y, προστίθεται η ποσότητα 1/log(degree(u)) στο συνολικό σκορ.

Αυτό σημαίνει ότι οι κόμβοι με λίγες συνδέσεις (χαμηλό degree) συνεισφέρουν περισσότερο στο σκορ, ενισχύοντας την πιθανότητα ότι το ζεύγος x,y έχει ισχυρότερη σχέση. Αντίθετα, κοινή σύνδεση με έναν πολύ δημοφιλή κόμβο δεν θεωρείται ιδιαίτερα ενδεικτική και συμβάλλει ελάχιστα στην πρόβλεψη.

Authorship Overlap

Η επικάλυψη συγγραφέων μετρά το ποσοστό κοινών συγγραφέων μεταξύ δύο άρθρων. Όσο μεγαλύτερη είναι αυτή η τιμή, τόσο αυξάνεται η πιθανότητα να υπάρχει παραπομπή (citation) μεταξύ των δύο άρθρων, καθώς η συγγραφική συνύπαρξη είναι συχνά ενδεικτική θεματολογικής ή ερευνητικής συνάφειας.

Παράλληλος Υπολογισμός Χαρακτηριστικών (Parallel Feature Extraction)

Για κάθε σετ δεδομένων (training, validation και test), ξεκινάμε από το αντίστοιχο DataFrame που περιέχει τα ζεύγη άρθρων προς επεξεργασία. Το κάθε σύνολο χωρίζεται ισομερώς σε 4 υποσύνολα, ένα για κάθε διαθέσιμο πυρήνα της CPU, και εφαρμόζουμε παράλληλη επεξεργασία (parallel processing) για να επιταχύνουμε τον υπολογισμό των χαρακτηριστικών.

Μόλις ολοκληρωθούν οι υπολογισμοί, συγχωνεύουμε τα τέσσερα υποσύνολα σε ένα ενιαίο DataFrame, απορρίπτοντας τις βοηθητικές στήλες p1 και p2 (που αντιστοιχούν στα IDs των άρθρων) και κρατώντας μόνο το τελικό σύνολο με τα υπολογισμένα χαρακτηριστικά (features).

Το αποτέλεσμα αποθηκεύεται σε αρχείο τύπου .pkl (Pickle), ώστε να αποφεύγεται ο επανυπολογισμός στις μελλοντικές εκτελέσεις του pipeline, εξοικονομώντας χρόνο και υπολογιστικούς πόρους.

4. Μοντέλα,Βελτιστοποίηση,Σύγκριση

Scaling (Κανονικοποίηση Δεδομένων)

Για να διασφαλίσουμε ότι κανένα χαρακτηριστικό (feature) δεν κυριαρχεί λόγω διαφορετικού εύρους τιμών, εφαρμόζουμε τυποποίηση (standardization) σε όλα τα δεδομένα με χρήση της κλάσης StandardScaler από το scikit-learn. Με αυτόν τον τρόπο, κάθε χαρακτηριστικό αποκτά μέση τιμή 0 και τυπική απόκλιση 1, εξασφαλίζοντας ομοιογένεια στην κλίμακα των εισόδων.

XGBoost και HyperOpt

Το XGBoost (Extreme Gradient Boosting) είναι ένας ισχυρός αλγόριθμος μηχανικής μάθησης, βασισμένος στην τεχνική ενίσχυσης κλίσης (gradient boosting) με χρήση decision trees. Ουσιαστικά, χτίζει διαδοχικά δέντρα, καθένα εκ των οποίων διορθώνει τα σφάλματα του προηγούμενου, μειώνοντας σταδιακά τη συνολική απώλεια (loss).

Για την βελτιστοποίηση υπερπαραμέτρων, χρησιμοποιήσαμε τη βιβλιοθήκη HyperOpt, η οποία εφαρμόζει Bayesian optimization. Σε αντίθεση με brute-force προσεγγίσεις όπως το GridSearch ή το RandomSearch, το HyperOpt μαθαίνει ένα στατιστικό μοντέλο της μορφής P(score|παραμετροι) και ενημερώνει δυναμικά τις προβλέψεις του με βάση προηγούμενα αποτελέσματα. Έτσι, εντοπίζει ταχύτερα τις περιοχές του χώρου παραμέτρων που οδηγούν στη βέλτιστη επίδοση (χαμηλό log loss στο validation set).

Search Space και Τιμές Παραμέτρων:

- n_estimators: 3000 (μέγιστος αριθμός δέντρων)
- early_stopping_rounds: 100 (τερματισμός εκπαίδευσης χωρίς βελτίωση)
- max_depth: 2 έως 20 (μέγιστο βάθος δέντρου)
- gamma: 0 έως 9 (ελάχιστη μείωση loss για split)
- reg_alpha: 0 έως 200 (L1 κανονικοποίηση)
- reg_lambda: 0 έως 1 (L2 κανονικοποίηση)
- colsample_bytree: 0.5 έως 1 (τυχαίο υποσύνολο χαρακτηριστικών)
- min_child_weight: 0 έως 10 (ελάχιστο βάρος δείγματος για split)

- learning_rate: 0.001 έως 0.05 (ρυθμός μάθησης)
- eval_metric: "logloss" (μετρική αξιολόγησης)
- random_state: 7 (για επαναληψιμότητα)

Τέλος, επιλέγουμε τις υπερπαραμέτρους που επέστρεψε η HyperOpt , αυτές με τις οποίες επιτεύχθηκε η χαμηλότερη log-loss στο validation , και τις χρησιμοποιούμε για το τελικό training του XGBoost (XGBoost (Extreme Gradient Boosting)). Αν και τα ακριβή αποτελέσματα της Bayesian optimization μπορεί να διαφέρουν ελαφρώς κάθε φορά (το τελικό log-loss συνήθως κυμαίνεται σε απόκλιση 0.01–0.05), εμείς διατηρήσαμε τις παραμέτρους που οδήγησαν στο τελικό και καλύτερο αποτέλεσμα στο Kaggle.

Random forest

Το Random Forest (Τυχαίο Δάσος) είναι ένας αλγόριθμος μηχανικής μάθησης που χρησιμοποιεί ένα σύνολο (ensemble) από δέντρα αποφάσεων για να κάνει προβλέψεις. Η βασική ιδέα είναι να συνδυάζονται τα αποτελέσματα πολλών δέντρων αποφάσεων, τα οποία εκπαιδεύονται σε διαφορετικά τυχαία υποσύνολα δεδομένων και χαρακτηριστικών, προκειμένου να βελτιωθεί η ακρίβεια και να μειωθεί το overfitting .

Για το Random Forest, πειραματιστήκαμε κυρίως με δύο υπερπαραμέτρους:

n_estimators (αριθμός δέντρων): Δοκιμάσαμε τιμές στο διάστημα 100–500. Μεγαλύτερος αριθμός δέντρων οδηγούσε σε ελαφρώς καλύτερη ακρίβεια, αλλά αύξανε γραμμικά τον χρόνο εκπαίδευσης. Πέρα από αυτό το εύρος, η αύξηση του αριθμού των δέντρων δεν προσέφερε ουσιαστική βελτίωση στο log-loss και καθιστούσε το μοντέλο υπερβολικά αργό.

max_depth (μέγιστο βάθος): Τιμές μεταξύ 3 και 7 έδωσαν τα πιο ισορροπημένα αποτελέσματα. Μικρότερα βάθη παρήγαγαν απλά δέντρα με μειωμένο κίνδυνο υπερπροσαρμογής (overfitting), ενώ μεγαλύτερα βάθη δεν μείωναν περαιτέρω την απώλεια και επιβράδυναν σημαντικά την εκπαίδευση.

Τελικά επιλέξαμε n_estimators = 300 και max_depth = 5, καθώς αυτή η ρύθμιση παρείχε σταθερή μείωση στο log-loss, χωρίς εμφανή σημάδια overfitting και με ικανοποιητική ταχύτητα εκπαίδευσης. Χρησιμοποιήσαμε criterion="log_loss" ώστε να βελτιστοποιείται άμεσα η ταξινομική απώλεια, verbose=2 για αναλυτική ενημέρωση της προόδου και n_jobs=-1 για χρήση όλων των διαθέσιμων πυρήνων της CPU.

Multilayer Perceptron – MLP

Το Multilayer Perceptron (MLP) είναι ένα είδος τεχνητού νευρωνικού δικτύου με πολλαπλά επίπεδα (layers) διασυνδεδεμένων νευρώνων. Περιλαμβάνει επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα (hidden layers) και ένα επίπεδο εξόδου, και είναι κατάλληλο για την εκμάθηση μη γραμμικών συσχετίσεων στα δεδομένα.

Αρχιτεκτονική και Νευρώνες

Δοκιμάσαμε πολλούς συνδυασμούς νευρώνων, από 128-64-32 έως 8-4. Οι μεγαλύτερες αρχιτεκτονικές οδηγούσαν σε overfitting, καθώς προσαρμόζονταν πολύ στενά στο training set. Δοκιμές με 64-32-16, 64-32 και 32-16 έδειξαν σταδιακή βελτίωση, αλλά η διάταξη 16–8–4 προσέφερε το καλύτερο trade-off μεταξύ ακρίβειας και γενίκευσης, χωρίς overfitting. Αντιθέτως, μικρότερες διατάξεις παρουσίασαν underfitting και χαμηλότερη απόδοση σε σχέση με XGBoost και Random Forest.

Dropout (Τυχαία Απενεργοποίηση Νευρώνων)

Πειραματιστήκαμε με dropout rates 0.5, 0.7 και 0.9 στο πρώτο επίπεδο, 0.3–0.6 στο δεύτερο και 0.1–0.3 στο τρίτο. Το σχήμα (0.7, 0.5, 0.2) αποδείχθηκε το πιο αποτελεσματικό, περιορίζοντας σημαντικά το overfitting. Σε μικρότερα dropout rates, παρατηρήθηκε overfitting, πιθανώς λόγω της μικρής διαστατικότητας των χαρακτηριστικών, με αποτέλεσμα το μοντέλο να απομνημονεύει patterns από το training set και να αποτυγχάνει να τα γενικεύσει — χαρακτηριστικό που φαινόταν από το πολύ χαμηλό training loss (~0) και το υψηλό validation loss (>0.4).

L2 Regularization (Κανονικοποίηση τύπου L2)

Εφαρμόσαμε L2 regularization με συντελεστή 0.1 σε όλα τα επίπεδα του MLP. Η τεχνική αυτή προσθέτει ποινή στις μεγάλες τιμές των βαρών κατά τη διάρκεια της εκπαίδευσης, ενθαρρύνοντας το μοντέλο να διατηρεί τα βάρη του μικρά και ομαλοποιημένα. Αυτό οδηγεί σε απλούστερες μαθηματικά αναπαραστάσεις, περιορίζει την τάση του μοντέλου να "θυμάται" το training set και βελτιώνει τη γενίκευση σε άγνωστα δεδομένα. Το αποτέλεσμα ήταν σταθερή μείωση της val_loss, με ταυτόχρονη αποφυγή υπερπροσαρμογής (overfitting).

Learning Rate (Ρυθμός Μάθησης)

Ο ρυθμός μάθησης (learning rate) καθορίζει το μέγεθος του βήματος με το οποίο ενημερώνονται τα βάρη σε κάθε εποχή. Δοκιμάσαμε τιμές 0.1, 0.01 και 0.001, και καταλήξαμε ότι το καλύτερο αποτέλεσμα επιτυγχάνεται όταν ξεκινάμε με learning_rate = 0.1. Για να αποφευχθεί στασιμότητα, χρησιμοποιήσαμε τον μηχανισμό ReduceLROnPlateau, ο οποίος μειώνει προοδευτικά το learning rate (στο 50% της τρέχουσας τιμής) όταν η απόδοση στο validation set δεν παρουσιάζει βελτίωση για 3 διαδοχικές εποχές.

Η στρατηγική αυτή επιτρέπει γρήγορη αρχική σύγκλιση, ακολουθούμενη από λεπτότερες προσαρμογές όσο πλησιάζουμε στο βέλτιστο, οδηγώντας σε καλύτερη σταθερότητα και τελική ακρίβεια.

Epochs & EarlyStopping (Πρόωρη Διακοπή Εκπαίδευσης)

Ορίσαμε το μέγιστο αριθμό εποχών (epochs) στις 50, αλλά ενεργοποιήσαμε τη τεχνική EarlyStopping με patience = 10. Ο μηχανισμός αυτός παρακολουθεί τη val_loss και διακόπτει αυτόματα την εκπαίδευση όταν δεν υπάρχει βελτίωση για 10 συνεχόμενες εποχές.

Αυτός ο μηχανισμός βοήθησε, στη μείωση του υπολογιστικού κόστους (αποφεύγοντας περιττές εποχές), και στην πρόληψη υπερπροσαρμογής (overfitting), καθώς σταματά τη μάθηση πριν το μοντέλο προσαρμοστεί υπερβολικά στο training set.

Batch Size

Δοκιμάσαμε μικρά batch sizes (64–128), που οδήγησαν σε overfitting και χαμηλή ταχύτητα, καθώς και πολύ μεγάλα (>2048), όπου εμφανίστηκε underfitting. Το 512 αποδείχθηκε η βέλτιστη τιμή, προσφέροντας σταθερή σύγκλιση και καλή ισορροπία ταχύτητας και απόδοσης.

Stacking Ensemble (Στοίβαξη Μοντέλων)

Για να αξιοποιήσουμε τα διαφορετικά πλεονεκτήματα κάθε βασικού μοντέλου (Random Forest, Neural Network, XGBoost), κατασκευάσαμε ένα stacking ensemble – μια τεχνική συνδυασμού μοντέλων όπου η τελική πρόβλεψη βασίζεται σε ένα meta-model που μαθαίνει από τις προβλέψεις άλλων μοντέλων.

Η διαδικασία έχει ως εξής:

Αρχικά, για κάθε δείγμα του validation set, συλλέγουμε τις προβλεπόμενες πιθανότητες (probabilities) από κάθε βασικό μοντέλο:

- rf_val από το Random Forest
- nn_val_probs από το Neural Network
- xgb_val από το XGBoost

Συνδυάζουμε αυτές τις τρεις προβλέψεις σε έναν νέο πίνακα χαρακτηριστικών (X_stack_val) — κάθε γραμμή περιέχει τις τρεις πιθανότητες για ένα ζεύγος άρθρων.

Στη συνέχεια, εκπαιδεύουμε έναν meta-learner, ο οποίος σε αυτή την περίπτωση είναι ένα απλό λογιστικό μοντέλο (Logistic Regression). Ο meta-ταξινομητής μαθαίνει να συνδυάζει και να ζυγίζει τις προβλέψεις των τριών μοντέλων, αξιοποιώντας την αξιοπιστία και την απόδοση καθενός.

Αφού ολοκληρωθεί η εκπαίδευση, χρησιμοποιούμε το τελικό stacking μοντέλο για να παράγουμε τις τελικές προβλεπόμενες πιθανότητες (stack_val_probs) για το validation set.

Τέλος, αξιολογούμε το stacking ως προς:

- Log Loss (λογαριθμική απώλεια),
- Accuracy (ακρίβεια ταξινόμησης),
- AUC (Area Under Curve – εμβαδόν κάτω από την καμπύλη ROC).

Τα αποτελέσματα έδειξαν ότι το stacking πέτυχε χαμηλότερο log loss και υψηλότερη AUC σε σύγκριση με οποιοδήποτε από τα μεμονωμένα μοντέλα. Αυτό καταδεικνύει την αξία της ensemble προσέγγισης, ειδικά όταν τα επιμέρους μοντέλα κάνουν σφάλματα σε διαφορετικές περιπτώσεις.

Πίνακας σύγκρισης μοντέλων:

Model	Log Loss	Accuracy	AUC
Random Forest	0.1932	0.9283	0.9803
XGBoost	0.201	0.9286	0.9805
Neural Network	0.16	0.9455	0.9819
Stacked Ensemble	0.1505	0.9481	0.9848
Kaggle score	0.13260 (private) 0.13533 (public)	–	–

Συμπεράσματα

Η συνολική ανάλυση και ο πειραματισμός με διαφορετικά μοντέλα κατέδειξαν τη σημαντική συμβολή κάθε μεθόδου στην τελική απόδοση, αλλά και την προστιθέμενη αξία του συνδυασμού τους μέσω stacking. Όπως προκύπτει από τον πίνακα αξιολόγησης:

Το νευρωνικό δίκτυο υπερισχύει τόσο του Random Forest όσο και του XGBoost σε όλες τις μετρικές (log loss, accuracy, AUC), επιτυγχάνοντας log loss = 0.16 και AUC = 0.9819.

Το Stacking Ensemble, το οποίο ενσωματώνει και τα τρία μοντέλα, πέτυχε την καλύτερη συνολική απόδοση, με log loss = 0.1505, accuracy = 0.9481 και AUC = 0.9848. Αυτό επιβεβαιώνει ότι τα επιμέρους μοντέλα έχουν συμπληρωματικά σφάλματα και ότι η συνδυαστική πρόβλεψη οδηγεί σε πιο ισχυρά και γενικευμένα αποτελέσματα.

Η τελική Kaggle υποβολή κατέγραψε log loss = 0.1326 στο private leaderboard, δείχνοντας ότι το μοντέλο γενίκευσε ικανοποιητικά και σε τελείως άγνωστα δεδομένα.

Συνολικά, η εργασία ανέδειξε τη σημασία της εξαγωγής ισχυρών χαρακτηριστικών, της συστηματικής βελτιστοποίησης υπερπαραμέτρων και της στρατηγικής συνδυασμού μοντέλων για την επιτυχή επίλυση του προβλήματος πρόβλεψης αναφορών.