Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

TransferMind: Εφαρμογή Ιστού για την Στατιστική Ανάλυση και Εξόρυξη Δεδομένων Απόδοσης στο Ποδόσφαιρο

Φοιτητές:
Κωνσταντίνος Πάντενας - Κωνσταντίνος Περιστεράκης
Αριθμός Μητρώου: 113773 - 175120

Επιβλέπων:
Στέφανος Ουγιάρογλου

31 May 2026

Title of Dissertation TransferMind: A Web Application for Statistical Analysis and Performance Data Mining in Football
Code of Dissertation 25252
Student's full name Konstantinos Pantenas - Konstantinos Peristerakis
Supervisor's full name Stefanos Ougiaroglou
Date of undertaking 01-04-2025
Date of completion 31-05-2026

We hereby affirm the authorship of this paper as well as the acknowledgement and credit of whichever assistance We received in its composition. We have, furthermore, noted the various sources from which We extracted data, ideas, visual or written material, in paraphrase or exact quotation. Moreover, we affirm the exclusive composition of this paper by myself only, for the purpose of it being a dissertation, in the Department of Information and Electronic Engineering of the I.H.U.

The approval of this dissertation by the Department of Information and Electronic Engineering of the International Hellenic University, does not necessarily entail the adoption of the author's views, on behalf of the Department.

Ευχαριστίες

Αφιερώνουμε την παρούσα πτυχιακή εργασία στον καθηγητή μας, κ. Ουγιάρογλου, για την καθοδήγηση, τη στήριξη και την εμπιστοσύνη που μας έδειξε καθ' όλη τη διάρκεια της εκπόνησής της. Οι συμβουλές του και η επιστημονική του επίβλεψη συνέβαλαν ουσιαστικά στην ολοκλήρωση της προσπάθειάς μας. Παράλληλα, αφιερώνουμε την εργασία αυτή στις οικογένειές μας, οι οποίες στάθηκαν δίπλα μας με υπομονή, κατανόηση και συνεχή ενθάρρυνση σε όλη τη διάρκεια των σπουδών μας. Η παρουσία τους αποτέλεσε σημαντικό στήριγμα στις απαιτητικές στιγμές και μας έδωσε τη δύναμη να συνεχίσουμε. Τέλος, αφιερώνουμε την εργασία στους απανταχού ποδοσφαιρόφιλους, των οποίων η αγάπη για το ποδόσφαιρο, η ανάγκη για καλύτερη κατανόηση του παιχνιδιού και το ενδιαφέρον για τα δεδομένα αποτέλεσαν πηγή έμπνευσης για την ιδέα και την ανάπτυξη της εφαρμογής TransferMind.

Πρόλογος

Το ποδόσφαιρο αποτελεί ένα από τα πιο δημοφιλή αθλήματα παγκοσμίως και συγκεντρώνει το ενδιαφέρον εκατομμυρίων ανθρώπων. Για πολλούς φιλάθλους δεν είναι απλώς ένα άθλημα, αλλά ένα πεδίο συζήτησης, ανάλυσης και συνεχούς αναζήτησης απαντήσεων σχετικά με την απόδοση ομάδων και παικτών. Τα τελευταία χρόνια, η εξέλιξη της τεχνολογίας και η αυξανόμενη διαθεσιμότητα στατιστικών δεδομένων έχουν επηρεάσει σημαντικά τον τρόπο με τον οποίο προσεγγίζεται το παιχνίδι. Η παρατήρηση ενός αγώνα μπορεί πλέον να συνδυαστεί με αριθμητικά δεδομένα, συγκρίσεις και οπτικοποιήσεις, προσφέροντας μια πιο ολοκληρωμένη εικόνα της αγωνιστικής πραγματικότητας.

Η επιλογή του θέματος της παρούσας πτυχιακής εργασίας προέκυψε από το ενδιαφέρον μας για τη σύνδεση της πληροφορικής με τον χώρο του ποδοσφαίρου. Θελήσαμε να μελετήσουμε πώς τα διαθέσιμα ποδοσφαιρικά δεδομένα μπορούν να παρουσιαστούν με τρόπο πιο οργανωμένο και κατανοητό, ώστε να μην παραμένουν απλώς αριθμοί σε πίνακες, αλλά να αποκτούν πρακτική αξία για τον χρήστη. Μέσα από αυτή τη διαδικασία, η εργασία αποτέλεσε για εμάς μια ευκαιρία να εφαρμόσουμε στην πράξη γνώσεις που αποκτήθηκαν κατά τη διάρκεια των σπουδών μας.

Η ανάπτυξη του TransferMind ανέδειξε τη σημασία της σωστής συλλογής, οργάνωσης και παρουσίασης της πληροφορίας. Σε ένα πεδίο όπως το ποδόσφαιρο, όπου τα δεδομένα μπορούν να ερμηνευθούν με πολλούς τρόπους, η κατάλληλη επεξεργασία τους είναι απαραίτητη ώστε ο χρήστης να μπορεί να οδηγηθεί σε πιο ουσιαστικά συμπεράσματα. Για τον λόγο αυτό, η εφαρμογή σχεδιάστηκε ως ένα ολοκληρωμένο σύστημα που συνδέει την τεχνολογική υλοποίηση με την ανάγκη για απλούστερη και πιο κατανοητή ανάλυση.

Στα κεφάλαια που ακολουθούν παρουσιάζονται οι βασικές τεχνολογίες που χρησιμοποιήθηκαν, το θεωρητικό υπόβαθρο των αλγορίθμων ανάλυσης δεδομένων, η αρχιτεκτονική και η υλοποίηση της εφαρμογής, καθώς και η τελική παρουσίαση του TransferMind. Η εργασία αυτή φιλοδοξεί να αναδείξει τον τρόπο με τον οποίο η πληροφορική μπορεί να συμβάλει στην καλύτερη κατανόηση του σύγχρονου ποδοσφαίρου.

Abstract

This thesis concerns the design and development of TransferMind, a web application for the organization, analysis, and presentation of football data. The main objective of the thesis is to create a system that is not limited to the simple display of statistical information, but supports a better understanding of team and player performance through comparisons, visualizations, and data analysis techniques.
To achieve this objective, a practical development approach was followed, combining web technologies, a database, the collection and processing of football statistics, as well as data mining and machine learning algorithms. More specifically, the application utilizes techniques such as K-Means Clustering, hierarchical clustering, and the Apriori algorithm, aiming to reveal similarities, relationships, and patterns that are not always immediately evident through the simple reading of numerical data.
The thesis demonstrates that football statistics can be transformed into more functional and usable information when they are organized and presented through an appropriate interactive environment. TransferMind is addressed to different categories of users, such as football fans, students, researchers, and smaller teams, offering a more accessible way to explore and understand football analytics.

Περίληψη

Η παρούσα πτυχιακή εργασία αφορά τη σχεδίαση και ανάπτυξη της εφαρμογής TransferMind, μιας διαδικτυακής εφαρμογής για την οργάνωση, ανάλυση και παρουσίαση ποδοσφαιρικών δεδομένων. Κύριος στόχος της εργασίας είναι η δημιουργία ενός συστήματος που δεν περιορίζεται στην απλή εμφάνιση στατιστικών στοιχείων, αλλά υποστηρίζει την καλύτερη κατανόηση της απόδοσης ομάδων και παικτών μέσα από συγκρίσεις, οπτικοποιήσεις και τεχνικές ανάλυσης δεδομένων.

Για την επίτευξη του στόχου αυτού ακολουθήθηκε μια πρακτική προσέγγιση ανάπτυξης, η οποία συνδύασε τεχνολογίες διαδικτύου, βάση δεδομένων, συλλογή και επεξεργασία ποδοσφαιρικών στατιστικών, καθώς και αλγορίθμους εξόρυξης δεδομένων και μηχανικής μάθησης. Συγκεκριμένα, στην εφαρμογή αξιοποιούνται τεχνικές όπως το K-Means Clustering, η ιεραρχική συσταδοποίηση και ο αλγόριθμος Apriori, με σκοπό την ανάδειξη ομοιοτήτων, σχέσεων και μοτίβων που δεν είναι πάντα άμεσα εμφανή μέσα από την απλή ανάγνωση αριθμητικών δεδομένων.

Η εργασία αναδεικνύει ότι τα ποδοσφαιρικά στατιστικά μπορούν να μετατραπούν σε πιο λειτουργική και αξιοποιήσιμη πληροφορία όταν οργανώνονται και παρουσιάζονται μέσα από ένα κατάλληλο διαδραστικό περιβάλλον. Το TransferMind απευθύνεται σε διαφορετικές κατηγορίες χρηστών, όπως φίλαθλους, φοιτητές, ερευνητές και μικρότερες ομάδες, προσφέροντας έναν πιο προσιτό τρόπο διερεύνησης και κατανόησης της ποδοσφαιρικής ανάλυσης.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Ποδοσφαιρικοί Αγώνες και Στατιστικά

Το ποδόσφαιρο αποτελεί ένα από τα πιο δημοφιλή και αναγνωρίσιμα αθλήματα παγκοσμίως, προσελκύοντας το ενδιαφέρον εκατομμυρίων φιλάθλων, αθλητών, προπονητών και επαγγελματιών του χώρου. Πέρα όμως από την ψυχαγωγική και αγωνιστική του διάσταση, αποτελεί και ένα ιδιαίτερα σύνθετο πεδίο ανάλυσης. Η απόδοση μιας ομάδας ή ενός ποδοσφαιριστή δεν εξαρτάται από έναν μόνο παράγοντα, αλλά διαμορφώνεται μέσα από τον συνδυασμό τεχνικών, τακτικών, φυσικών και ψυχολογικών χαρακτηριστικών. Για τον λόγο αυτό, η ανάγκη καλύτερης κατανόησης του παιχνιδιού οδήγησε σταδιακά στην ανάπτυξη της ανάλυσης ποδοσφαιρικών αγώνων, γνωστής και ως match analysis.

Η ανάλυση ποδοσφαιρικών αγώνων αφορά τη συλλογή, καταγραφή και επεξεργασία δεδομένων που προκύπτουν κατά τη διάρκεια ενός αγώνα, με σκοπό την εξαγωγή χρήσιμων συμπερασμάτων για την απόδοση ομάδων και παικτών. Η ανάλυση απόδοσης στο ποδόσφαιρο έχει εξελιχθεί σε χρήσιμο εργαλείο για προπονητές και αναλυτές, καθώς προσφέρει πιο αντικειμενική εικόνα για όσα συμβαίνουν μέσα στο γήπεδο [1]. Παράλληλα, η σύγχρονη ανάλυση δεν περιορίζεται μόνο σε απλά στατιστικά, αλλά εξετάζει τεχνικά, τακτικά και φυσικά στοιχεία του παιχνιδιού [2].

Με την εξέλιξη της τεχνολογίας, το football analytics έχει αλλάξει σημαντικά. Στο παρελθόν, η αξιολόγηση βασιζόταν κυρίως στην παρατήρηση των προπονητών και σε βασικά στατιστικά, όπως τα γκολ, οι πάσες ή τα σουτ. Σήμερα, η χρήση μεγάλου όγκου δεδομένων επιτρέπει πιο αναλυτική μελέτη της απόδοσης. Η εισαγωγή των tracking data, δηλαδή δεδομένων που καταγράφουν την κίνηση παικτών και μπάλας, έδωσε τη δυνατότητα για βαθύτερη κατανόηση της τακτικής συμπεριφοράς, των αποστάσεων, των ταχυτήτων και των θέσεων των παικτών στο γήπεδο. Τα μεγάλα δεδομένα στο ποδόσφαιρο δημιουργούν νέες δυνατότητες για την ανάλυση της τακτικής, αλλά και νέες προκλήσεις ως προς τη σωστή ερμηνεία τους [3].

Ένα βασικό μέρος της ποδοσφαιρικής ανάλυσης αφορά τα δεδομένα ομάδων. Αυτά περιγράφουν τη συνολική εικόνα μιας ομάδας σε έναν αγώνα ή σε μια ολόκληρη αγωνιστική περίοδο. Σε αυτή την κατηγορία περιλαμβάνονται μεταβλητές όπως η κατοχή μπάλας, οι τελικές προ-

σπάθειες, οι επιτυχημένες πάσες, τα γκολ, οι ανακτήσεις κατοχής και τα expected goals. Τα στοιχεία αυτά βοηθούν στη σύγκριση ομάδων, στην αναγνώριση διαφορετικών στυλ παιχνιδιού και στην αξιολόγηση της επιθετικής ή αμυντικής λειτουργίας τους.

Αντίστοιχα, τα δεδομένα παικτών προσφέρουν πιο λεπτομερή εικόνα για την ατομική απόδοση. Εκτός από βασικά τεχνικά στοιχεία, όπως πάσες, σουτ, ντρίμπλες και αμυντικές ενέργειες, μπορούν να καταγραφούν και φυσικά χαρακτηριστικά, όπως η απόσταση που διανύθηκε, η ταχύτητα και οι επιταχύνσεις. Νεότεροι δείκτες, όπως τα expected assists και τα possession value μοντέλα, προσπαθούν να αποτυπώσουν την αξία κάθε ενέργειας μέσα στον αγώνα. Τέτοιες προσεγγίσεις μπορούν να δείξουν καλύτερα τη συνεισφορά ενός παίκτη, ακόμη και όταν η ενέργειά του δεν οδηγεί άμεσα σε γκολ [4].

Η ανάλυση παικτών συνδέεται άμεσα και με το scouting. Παλαιότερα, η αξιολόγηση ενός ποδοσφαιριστή βασιζόταν κυρίως στην παρακολούθηση αγώνων από scouts και προπονητές. Σήμερα, τα δεδομένα χρησιμοποιούνται για τον εντοπισμό παικτών με συγκεκριμένα χαρακτηριστικά, αγωνιστικό προφίλ ή πιθανότητα εξέλιξης. Μέθοδοι όπως το clustering μπορούν να ομαδοποιήσουν παίκτες με παρόμοιο στυλ παιχνιδιού, ενώ τεχνικές πρόβλεψης μπορούν να βοηθήσουν στην εκτίμηση της μελλοντικής τους απόδοσης.

Γενικότερα, τα ποδοσφαιρικά στατιστικά χωρίζονται σε επιθετικά, αμυντικά, στατιστικά κατοχής και φυσικά στατιστικά. Τα επιθετικά περιλαμβάνουν γκολ, σουτ και δημιουργία ευκαιριών, ενώ τα αμυντικά περιλαμβάνουν τάκλιν, interceptions και ανακτήσεις. Τα στατιστικά κατοχής αφορούν την κυκλοφορία της μπάλας, ενώ τα φυσικά στατιστικά σχετίζονται με την κίνηση και την ένταση. Μέσα σε αυτό το πλαίσιο, η Μηχανική Μάθηση χρησιμοποιείται όλο και περισσότερο στο ποδόσφαιρο, με τεχνικές όπως classification, clustering και predictive analytics να βοηθούν στην πρόβλεψη αποτελεσμάτων, την αξιολόγηση απόδοσης και την αναγνώριση αγωνιστικών προτύπων.

1.2 Διαδικτυακοί Τόποι Ανάλυσης Ποδοσφαιρικών Δεδομένων

Οι διαδικτυακοί τόποι ανάλυσης ποδοσφαιρικών δεδομένων αποτελούν πλέον βασικό μέσο πληροφόρησης για φιλάθλους, αναλυτές, δημοσιογράφους και επαγγελματίες του ποδοσφαίρου. Μέσα από αυτές τις πλατφόρμες, τα στατιστικά στοιχεία ομάδων και παικτών παρουσιάζονται με οργανωμένο και πιο κατανοητό τρόπο, ώστε ο χρήστης μπορεί να αποκτήσει καλύτερη εικόνα για έναν αγώνα, έναν ποδοσφαιριστή ή μια ομάδα, χωρίς να βασίζεται μόνο στην προσωπική παρατήρηση. Παράλληλα, η συνεχής ανάπτυξη του football analytics έχει οδηγήσει στη δημιουργία πιο σύνθετων εργαλείων, τα οποία δεν περιορίζονται στην απλή εμφάνιση αριθμών, αλλά υποστηρίζουν πιο βαθιά ανάλυση απόδοσης, scouting και τακτικής συμπεριφοράς.

Η Opta αποτελεί έναν από τους πιο γνωστούς παρόχους ποδοσφαιρικών δεδομένων. Ο βασικός της ρόλος είναι η καταγραφή αγωνιστικών γεγονότων, όπως πάσες, σουτ, μονομαχίες, φάουλ

και αμυντικές ενέργειες. Η αξιοπιστία τέτοιων δεδομένων είναι ιδιαίτερα σημαντική, καθώς χρησιμοποιούνται τόσο από επαγγελματικές ομάδες όσο και από ερευνητές. Η αξιοπιστία των ζωντανών στατιστικών της Opta έχει εξεταστεί ερευνητικά και δείχνει ότι η καταγραφή τους μπορεί να αποτελέσει σταθερή βάση για ανάλυση απόδοσης [5]. Πέρα από τη συλλογή δεδομένων, η Opta, μέσω της Stats Perform, συνδέεται και με πιο προχωρημένες λύσεις ανάλυσης, όπως το Opta ProVision, το οποίο αξιοποιεί τη βάση δεδομένων της Opta για δημιουργία αναφορών, οπτικοποιήσεων και πιο λεπτομερή μελέτη ομάδων, παικτών και αγώνων [6].

Η Wyscout ακολουθεί μια πιο σύνθετη προσέγγιση, καθώς συνδυάζει στατιστικά δεδομένα, βίντεο και πληροφορίες παικτών. Για αυτόν τον λόγο χρησιμοποιείται συχνά στο scouting και στην ανάλυση αντιπάλων. Η δυνατότητα σύνδεσης μιας αγωνιστικής ενέργειας με το αντίστοιχο βίντεο βοηθά τον αναλυτή να εξετάσει όχι μόνο το αποτέλεσμα μιας ενέργειας, αλλά και το αγωνιστικό πλαίσιο μέσα στο οποίο έγινε. Τα δεδομένα της Wyscout έχουν αξιοποιηθεί και σε ερευνητικό επίπεδο, μέσα από δημόσια σύνολα δεδομένων ποδοσφαιρικών γεγονότων [7]. Αυτό δείχνει ότι τέτοιου είδους πλατφόρμες δεν έχουν μόνο πρακτική αξία για ομάδες και scouts, αλλά μπορούν να αποτελέσουν και βάση για επιστημονική μελέτη πάνω στην ανάλυση ποδοσφαιρικών αγώνων.

Το SofaScore απευθύνεται περισσότερο στον απλό χρήστη, καθώς παρουσιάζει live scores, στατιστικά αγώνα, συνθέσεις, συγκρίσεις ομάδων και βαθμολογίες παικτών με άμεσο και φιλικό τρόπο. Το σύστημα αξιολόγησης παικτών δίνει μια γρήγορη εικόνα της απόδοσης, όμως δεν πρέπει να αντιμετωπίζεται ως απόλυτη μέτρηση. Τα player ratings είναι χρήσιμα, αλλά κάθε πλατφόρμα εφαρμόζει διαφορετική μεθοδολογία και δίνει διαφορετικό βάρος στις επιμέρους αγωνιστικές ενέργειες [8]. Επομένως, τέτοιες πλατφόρμες είναι ιδιαίτερα χρήσιμες για την άμεση ενημέρωση του φιλάθλου, αλλά συνήθως δεν παρέχουν στον χρήστη τα ίδια επίπεδα ανάλυσης που έχουν στη διάθεσή τους οι επαγγελματικές ομάδες.

Το Transfermarkt διαφέρει από τις προηγούμενες πλατφόρμες, καθώς εστιάζει κυρίως στην οικονομική και μεταγραφική πλευρά του ποδοσφαίρου. Παρέχει πληροφορίες για παίκτες, ομάδες, μεταγραφές, συμβόλαια, τραυματισμούς και εκτιμώμενες αγοραστικές αξίες. Δεδομένα του Transfermarkt έχουν χρησιμοποιηθεί σε μελέτες σχετικές με την αξία των ποδοσφαιριστών και τη λειτουργία της μεταγραφικής αγοράς [9], [10]. Με αυτόν τον τρόπο, η συγκεκριμένη πλατφόρμα δεν εστιάζει τόσο στην τακτική ή αγωνιστική ανάλυση ενός αγώνα, αλλά περισσότερο στην αποτύπωση της οικονομικής αξίας και της μεταγραφικής εικόνας των παικτών και των συλλόγων.

Πέρα από τις παραπάνω πλατφόρμες, στο επαγγελματικό ποδόσφαιρο χρησιμοποιούνται και πιο εξειδικευμένα εργαλεία ανάλυσης δεδομένων. Παραδείγματα τέτοιων εργαλείων είναι το Hudl StatsBomb, το Hudl Sportscode, το SkillCorner και το Catapult. Οι πλατφόρμες αυτές απευθύνονται κυρίως σε συλλόγους, αναλυτικά τμήματα, scouts, προπονητές και ομοσπονδίες, προσφέροντας δυνατότητες που ξεπερνούν την απλή παρουσίαση στατιστικών. Μέσα από τέτοια συστήματα, μια ομάδα μπορεί να αναλύσει αγώνες, να μελετήσει αντιπάλους, να αξιολογήσει πιθανούς μεταγραφικούς στόχους, να συνδέσει βίντεο με στατιστικά δεδομένα και να δημιουργήσει πιο εξειδικευμένες αναφορές απόδοσης.

Ιδιαίτερο ενδιαφέρον παρουσιάζει το γεγονός ότι αρκετά από αυτά τα εργαλεία αξιοποιούν

advanced metrics, tracking data, computer vision και τεχνικές μηχανικής μάθησης. Για παράδειγμα, πλατφόρμες όπως το StatsBomb περιλαμβάνουν δείκτες όπως τα expected goals, το On-Ball Value και το expected pass, με στόχο την πιο λεπτομερή αξιολόγηση των ενεργειών που πραγματοποιούνται μέσα στον αγώνα [11]. Η επιστημονική βιβλιογραφία έχει επίσης δείξει ότι η αξία μιας ενέργειας δεν μπορεί πάντα να αποτυπωθεί μόνο από βασικά στατιστικά, καθώς είναι σημαντικό να λαμβάνεται υπόψη το πώς μια ενέργεια επηρεάζει την πιθανότητα μιας ομάδας να σκοράρει ή να δεχθεί γκολ [4]. Παράλληλα, η χρήση tracking data και χωροχρονικής ανάλυσης επιτρέπει τη μελέτη της θέσης, της κίνησης και της αλληλεπίδρασης των παικτών μέσα στο γήπεδο [12].

Η εξέλιξη αυτή συνδέεται άμεσα με τη γενικότερη χρήση μεγάλων δεδομένων και μηχανικής μάθησης στο ποδόσφαιρο. Η ανάλυση δεν περιορίζεται πλέον μόνο στο τι συνέβη σε έναν αγώνα, αλλά προσπαθεί να εξηγήσει γιατί συνέβη και ποια μοτίβα μπορούν να εντοπιστούν μέσα από τα δεδομένα. Έρευνες έχουν δείξει ότι τα μεγάλα δεδομένα δημιουργούν νέες δυνατότητες για την τακτική ανάλυση αλλά και νέες προκλήσεις ως προς τη σωστή ερμηνεία τους [3]. Παράλληλα, η μηχανική μάθηση εφαρμόζεται όλο και περισσότερο σε προβλήματα όπως η αξιολόγηση απόδοσης, η πρόβλεψη αποτελεσμάτων, η ανάλυση τεχνικοτακτικών χαρακτηριστικών και η αναγνώριση αγωνιστικών προτύπων [13].

Συνολικά, οι διαδικτυακές πλατφόρμες ποδοσφαιρικών δεδομένων δείχνουν ότι η ανάλυση του ποδοσφαίρου έχει περάσει πλέον σε μια πιο οργανωμένη και τεκμηριωμένη μορφή. Κάθε πλατφόρμα εξυπηρετεί διαφορετικές ανάγκες, από την άμεση ενημέρωση του φιλάθλου μέχρι την επαγγελματική ανάλυση απόδοσης, το scouting και τη μεταγραφική αξιολόγηση. Ωστόσο, η ύπαρξη πολλών διαφορετικών πηγών δεδομένων δείχνει και την ανάγκη για πιο απλές εφαρμογές που μπορούν να παρουσιάζουν την πληροφορία με συγκεντρωμένο και κατανοητό τρόπο.

1.3 Κίνητρο

Το βασικό κίνητρο της εργασίας προέκυψε από την ανάγκη να αξιοποιηθούν τα ποδοσφαιρικά στατιστικά με πιο ουσιαστικό τρόπο. Παρότι σήμερα υπάρχουν πολλές πλατφόρμες που παρουσιάζουν δεδομένα για ομάδες, παίκτες και αγώνες, ο χρήστης συχνά περιορίζεται στην απλή προβολή αριθμών και πινάκων. Η πληροφορία υπάρχει, αλλά δεν είναι πάντα εύκολο να μετατραπεί σε χρήσιμη ανάλυση. Για παράδειγμα, ο χρήστης μπορεί να δει τα στατιστικά μιας ομάδας, αλλά δεν έχει πάντα τη δυνατότητα να συγκρίνει οργανωμένα διαφορετικές ομάδες, να εντοπίσει παρόμοια αγωνιστικά προφίλ ή να αναγνωρίσει μοτίβα μέσα από τεχνικές ανάλυσης δεδομένων.

Στο επαγγελματικό ποδόσφαιρο, οι μεγάλες ομάδες έχουν πλέον πρόσβαση σε εξειδικευμένα εργαλεία ανάλυσης, τα οποία συνδυάζουν βίντεο αγώνων, στατιστικά παικτών, δεδομένα απόδοσης, scouting reports και λειτουργίες σύγκρισης. Μέσα από τέτοια συστήματα, μπορούν να μελετήσουν τον τρόπο παιχνιδιού τους, να αναλύσουν αντιπάλους ή να αξιολογήσουν πιθανούς μεταγραφικούς στόχους. Ωστόσο, η πρόσβαση σε τέτοιες λύσεις δεν είναι ίδια για όλους. Το κόστος, η ανάγκη για εξειδικευμένο προσωπικό και οι τεχνικές απαιτήσεις μπορούν να απο-

τελέσουν εμπόδιο για μικρότερους συλλόγους, φοιτητές, ερευνητές ή απλούς φιλάθλους. Έτσι δημιουργείται ένα κενό ανάμεσα στα προηγμένα εργαλεία των επαγγελματικών οργανισμών και στις πιο απλές πλατφόρμες που είναι διαθέσιμες στο ευρύ κοινό.

Το κενό αυτό αποτέλεσε την κεντρική ιδέα για την σχεδίαση και ανάπτυξη του TransferMind. Η εφαρμογή σχεδιάστηκε με σκοπό να συνδυάσει την παρουσίαση ποδοσφαιρικών στατιστικών με τεχνικές εξόρυξης δεδομένων και μηχανικής μάθησης, μέσα από ένα πιο απλό και κατανοητό περιβάλλον. Με αυτόν τον τρόπο, δεν περιορίζεται μόνο στην εμφάνιση αριθμών, αλλά προσπαθεί να δώσει στον χρήστη μια πιο ουσιαστική εικόνα για την απόδοση ομάδων και παικτών. Η χρήση αλγορίθμων όπως το K-Means Clustering, η ιεραρχική συσταδοποίηση και ο Apriori δίνει τη δυνατότητα να αναδειχθούν σχέσεις, ομοιότητες και πρότυπα που δεν είναι πάντα άμεσα εμφανή μέσα από τα απλά στατιστικά.

Ένα ακόμη κίνητρο ήταν η δημιουργία μιας εφαρμογής που να μπορεί να χρησιμοποιηθεί από διαφορετικές κατηγορίες χρηστών, χωρίς να απαιτείται εξειδικευμένη τεχνική γνώση. Ένας φίλαθλος μπορεί να κατανοήσει καλύτερα την εικόνα μιας ομάδας ή ενός παίκτη, ενώ ένας φοιτητής ή ερευνητής μπορεί να δει πώς εφαρμόζονται αλγόριθμοι ανάλυσης δεδομένων σε ένα πραγματικό πεδίο ενδιαφέροντος. Επίσης, μια μικρότερη ομάδα ή ένας ερασιτεχνικός σύλλογος μπορεί να αντιληφθεί πώς τα διαθέσιμα στατιστικά μπορούν να αξιοποιηθούν πιο οργανωμένα, ακόμη και χωρίς σύνθετες υποδομές. Αντίστοιχα, ένας αναλυτής μπορεί να αξιοποιήσει τις συγκρίσεις, τις ομαδοποιήσεις και τις οπτικοποιήσεις για να εξάγει πιο γρήγορα συμπεράσματα.

Παράλληλα, η επιλογή του συγκεκριμένου θέματος συνδέεται και με το ενδιαφέρον για την ανάπτυξη μιας ολοκληρωμένης διαδικτυακής εφαρμογής. Το TransferMind δεν περιορίζεται σε μια θεωρητική παρουσίαση της ποδοσφαιρικής ανάλυσης, αλλά αποτελεί ένα πραγματικό σύστημα που ενώνει πολλά στάδια της διαδικασίας: τη συλλογή δεδομένων, την αποθήκευσή τους, την επεξεργασία τους με αλγορίθμους και την παρουσίαση των αποτελεσμάτων στον τελικό χρήστη. Έτσι, μέσα από την υλοποίηση της εφαρμογής, δόθηκε η δυνατότητα να εφαρμοστούν στην πράξη τεχνολογίες web ανάπτυξης, βάσεις δεδομένων, οπτικοποιήσεις και τεχνικές ανάλυσης δεδομένων σε ένα ενιαίο έργο.

Συνεπώς, το κίνητρο της εργασίας δεν περιορίζεται μόνο στην παρουσίαση ποδοσφαιρικών δεδομένων, αλλά αφορά την προσπάθεια μετατροπής τους σε πιο χρήσιμη και κατανοητή πληροφορία. Το TransferMind αναπτύχθηκε ώστε να προσφέρει έναν τρόπο εξερεύνησης, σύγκρισης και ανάλυσης ομάδων και παικτών, αξιοποιώντας διαδραστικές λειτουργίες και τεχνικές μηχανικής μάθησης. Με αυτόν τον τρόπο, η ανάλυση δεν αντιμετωπίζεται ως κάτι που αφορά μόνο μεγάλες επαγγελματικές ομάδες, αλλά ως μια διαδικασία που μπορεί να γίνει πιο κατανοητή και προσβάσιμη προς τον απλό χρήστη.

1.4 Συνεισφορά

Ο σχεδίασμος και η υλοποίηση της παρούσας εργασίας έρχεται να καλύψει το κενό που αναφέρθηκε στην προηγούμενη ενότητα, προσφέροντας ένα προσιτό περιβάλλον για την εξερεύ-

νηση και την ανάλυση ποδοσφαιρικών δεδομένων. Η βασική συνεισφορά της εφαρμογής είναι ότι δεν περιορίζεται στην απλή παρουσίαση στατιστικών στοιχείων, αλλά ενσωματώνει τεχνικές εξόρυξης δεδομένων και μηχανικής μάθησης με τρόπο κατανοητό για τον τελικό χρήστη. Μέσα από λειτουργίες όπως η σύγκριση ομάδων, η παρουσίαση προφίλ ομάδων και παικτών, η ομαδοποίηση με K-Means Clustering, η ιεραρχική συσταδοποίηση και η εξαγωγή συσχετίσεων με τον αλγόριθμο Apriori, ο χρήστης μπορεί να δει τα δεδομένα από διαφορετικές οπτικές. Έτσι, η εφαρμογή δεν εμφανίζει απλώς αριθμούς, αλλά προσπαθεί να αναδείξει σχέσεις, ομοιότητες και μοτίβα που μπορεί να μην είναι εύκολα αντιληπτά μέσα από μια απλή ανάγνωση στατιστικών.

Μια ακόμη σημαντική συνεισφορά αφορά την προσβασιμότητα της ανάλυσης. Το TransferMind σχεδιάστηκε ώστε να μπορεί να χρησιμοποιηθεί χωρίς να απαιτείται εξειδικευμένη γνώση προγραμματισμού, στατιστικής ή μηχανικής μάθησης. Αυτό είναι ιδιαίτερα σημαντικό, καθώς αρκετά επαγγελματικά εργαλεία ανάλυσης απευθύνονται σε αναλυτές, scouts ή τεχνικά τμήματα ομάδων. Αντίθετα, η παρούσα εφαρμογή επιχειρεί να παρουσιάσει σύνθετες αναλυτικές διαδικασίες μέσα από ένα πιο απλό και διαδραστικό περιβάλλον. Με αυτόν τον τρόπο, ένας φίλαθλος μπορεί να κατανοήσει καλύτερα την εικόνα μιας ομάδας ή ενός παίκτη, ενώ ένας φοιτητής μπορεί να δει πώς εφαρμόζονται αλγόριθμοι μηχανικής μάθησης σε ένα πραγματικό σύνολο δεδομένων.

Η εφαρμογή μπορεί επίσης να έχει πρακτική αξία για μικρότερες ομάδες ή ερασιτεχνικούς συλλόγους, οι οποίοι δεν έχουν πάντα τη δυνατότητα να χρησιμοποιήσουν ακριβές επαγγελματικές πλατφόρμες. Παρότι το TransferMind δεν αντικαθιστά εξειδικευμένα συστήματα που χρησιμοποιούνται από επαγγελματικούς συλλόγους υψηλού επιπέδου, μπορεί να λειτουργήσει ως ένα πιο απλό εργαλείο κατανόησης και διερεύνησης δεδομένων. Για παράδειγμα, μια μικρότερη ομάδα θα μπορούσε να αξιοποιήσει αντίστοιχες λειτουργίες για να παρατηρήσει αγωνιστικά χαρακτηριστικά, να συγκρίνει ομάδες ή να εντοπίσει κοινά μοτίβα απόδοσης. Με αυτή την έννοια, η συνεισφορά της εφαρμογής δεν βρίσκεται μόνο στην τεχνική της υλοποίηση, αλλά και στην προσπάθεια να κάνει την ανάλυση δεδομένων πιο προσιτή σε χρήστες που συνήθως δεν έχουν πρόσβαση σε επαγγελματικά εργαλεία.

Παράλληλα, η εργασία παρουσιάζει και τεχνική συνεισφορά, καθώς η εφαρμογή δεν αποτελεί απλώς μια θεωρητική προσέγγιση της ποδοσφαιρικής ανάλυσης, αλλά ένα ολοκληρωμένο διαδικτυακό σύστημα. Το TransferMind ανακτά τα διαθέσιμα δεδομένα, τα οργανώνει σε βάση δεδομένων, τα επεξεργάζεται μέσω αλγορίθμων μηχανικής μάθησης και τα παρουσιάζει στον χρήστη μέσα από διαδραστικές λειτουργίες και οπτικοποιήσεις. Με αυτόν τον τρόπο, αναδεικνύεται πώς διαφορετικά τεχνολογικά μέρη, όπως το backend, η βάση δεδομένων, οι αλγόριθμοι ανάλυσης και η διεπαφή χρήστη, μπορούν να συνεργαστούν σε μια ενιαία εφαρμογή που μετατρέπει τα ποδοσφαιρικά στατιστικά σε πιο λειτουργική και κατανοητή πληροφορία.

Επιπλέον, η εφαρμογή συνεισφέρει εκπαιδευτικά, καθώς μπορεί να χρησιμοποιηθεί ως παράδειγμα εφαρμογής αλγορίθμων μηχανικής μάθησης σε ένα πεδίο που είναι οικείο και ενδιαφέρον για πολλούς χρήστες. Οι αλγόριθμοι δεν παρουσιάζονται μόνο θεωρητικά, αλλά εφαρμόζονται σε πραγματικό σενάριο, με οπτικοποιήσεις και αποτελέσματα που βοηθούν στην κατανόηση της λειτουργίας τους. Αυτό μπορεί να βοηθήσει τον χρήστη να αντιληφθεί καλύτερα έννοιες όπως η συσταδοποίηση, η σύγκριση χαρακτηριστικών και η αναγνώριση συσχετίσεων.

Τέλος, η συνεισφορά της εφαρμογής θα αξιολογηθεί και μέσα από τη μελέτη της ευχρηστίας της. Για τον σκοπό αυτό προβλέπεται η χρήση του ερωτηματολογίου System Usability Scale (SUS), το οποίο αποτελεί μια καθιερωμένη μέθοδο αξιολόγησης της αντιλαμβανόμενης ευχρηστίας ενός συστήματος [14]. Μέσα από την αξιολόγηση αυτή θα εξεταστεί κατά πόσο οι χρήστες μπορούν να αλληλεπιδράσουν εύκολα με την εφαρμογή, να κατανοήσουν τις λειτουργίες της και να αξιοποιήσουν τα αποτελέσματα που παρουσιάζει. Μετά την ολοκλήρωση της αξιολόγησης, τα αποτελέσματα του SUS θα ενσωματωθούν στην εργασία, ώστε η συνεισφορά της εφαρμογής να μην αποτυπωθεί μόνο σε θεωρητικό και τεχνικό επίπεδο, αλλά και μέσα από την εμπειρία των χρηστών.

1.5 Οργάνωση της εργασίας

Η παρούσα εργασία οργανώνεται σε οκτώ κεφάλαια, τα οποία παρουσιάζουν σταδιακά το θεωρητικό υπόβαθρο, τις τεχνολογίες που χρησιμοποιήθηκαν, τους αλγορίθμους ανάλυσης δεδομένων, τη διαδικασία υλοποίησης της εφαρμογής TransferMind, καθώς και την αξιολόγηση και τα τελικά συμπεράσματα.

Στο πρώτο κεφάλαιο παρουσιάζεται η εισαγωγή της εργασίας. Αρχικά γίνεται αναφορά στον ρόλο των ποδοσφαιρικών στατιστικών και στη σημασία τους για την ανάλυση αγώνων, ομάδων και παικτών. Στη συνέχεια εξετάζονται διαδικτυακοί τόποι και πλατφόρμες ανάλυσης ποδοσφαιρικών δεδομένων, όπως η Opta, η Wyscout, το SofaScore και το Transfermarkt, καθώς και πιο εξειδικευμένα εργαλεία που χρησιμοποιούνται στο επαγγελματικό ποδόσφαιρο. Έπειτα παρουσιάζεται το κίνητρο που οδήγησε στην ανάπτυξη της εφαρμογής, καθώς και η συνεισφορά της εργασίας σε πρακτικό, τεχνικό και εκπαιδευτικό επίπεδο.

Στο δεύτερο κεφάλαιο αναλύονται οι βασικές γλώσσες, τεχνολογίες και βιβλιοθήκες που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής. Συγκεκριμένα, παρουσιάζονται το Node.js, η Python, η PostgreSQL σε συνδυασμό με το Supabase, καθώς και το React με TypeScript. Η ενότητα αυτή έχει ως στόχο να δώσει το απαραίτητο τεχνολογικό υπόβαθρο, ώστε να γίνει κατανοητός ο ρόλος κάθε εργαλείου μέσα στο συνολικό σύστημα.

Στο τρίτο κεφάλαιο παρουσιάζονται οι αλγόριθμοι εξόρυξης δεδομένων και συσταδοποίησης που χρησιμοποιούνται στην εφαρμογή. Αρχικά γίνεται εισαγωγή στην έννοια της cluster analysis και στη συνέχεια αναλύεται ο αλγόριθμος K-Means, μαζί με τη μέθοδο elbow για την επιλογή κατάλληλου αριθμού συστάδων. Ακολουθεί η παρουσίαση της ιεραρχικής συσταδοποίησης, και συγκεκριμένα της agglomerative προσέγγισης, ενώ στη συνέχεια εξετάζεται η εξόρυξη κανόνων συσχέτισης μέσω του αλγορίθμου Apriori. Για κάθε αλγόριθμο παρουσιάζονται παραδείγματα, ώστε να γίνει πιο σαφής ο τρόπος λειτουργίας του και η σύνδεσή του με την ποδοσφαιρική ανάλυση.

Στο τέταρτο κεφάλαιο περιγράφεται η διαδικασία αξιοποίησης του SofaScore API και η δημιουργία του συνόλου δεδομένων που χρησιμοποιείται από την εφαρμογή. Παρουσιάζεται ο τρόπος συλλογής, οργάνωσης και αποθήκευσης των δεδομένων, καθώς και η σύνδεσή τους με τη βάση δεδομένων. Επιπλέον, παρατίθενται ενδεικτικά αποσπάσματα κώδικα, ώστε να

αποτυπωθούν βασικά σημεία της διαδικασίας άντλησης και διαχείρισης των δεδομένων.

Στο πέμπτο κεφάλαιο αναλύεται η σχεδίαση και η υλοποίηση της εφαρμογής TransferMind. Αρχικά παρουσιάζεται η ανάλυση απαιτήσεων, μέσα από λειτουργικές ανάγκες και ενδεικτικά user stories. Στη συνέχεια περιγράφεται η αρχιτεκτονική του συστήματος, με τη βοήθεια διαγραμμάτων ροής και block diagrams. Ακολουθεί η υλοποίηση του backend, η ανάπτυξη των Python modules για την ανάλυση συστάδων, καθώς και η υλοποίηση του frontend. Τέλος, γίνεται αναφορά στο GitHub repository, στη διαδικασία deployment, στη χρήση GitHub Actions και στον τρόπο με τον οποίο το έργο οργανώθηκε ως ανοιχτό λογισμικό.

Στο έκτο κεφάλαιο παρουσιάζεται η εφαρμογή TransferMind από την πλευρά του τελικού χρήστη. Μέσα από στιγμιότυπα οθόνης και επεξηγηματικό κείμενο, αναλύονται οι βασικές λειτουργίες της εφαρμογής, όπως η αρχική σελίδα, τα προφίλ ομάδων και παικτών, η σύγκριση ομάδων, η ανάλυση συστάδων, η εφαρμογή του K-Means, η ιεραρχική συσταδοποίηση, ο Apriori και οι διαθέσιμες οπτικοποιήσεις. Στόχος του κεφαλαίου είναι να παρουσιαστεί με πρακτικό τρόπο ο τρόπος χρήσης της εφαρμογής και η πληροφορία που μπορεί να αντλήσει ο χρήστης από αυτή.

Στο έβδομο κεφάλαιο εξετάζεται η εμπειρία χρήσης της εφαρμογής. Η αξιολόγηση βασίζεται στο ερωτηματολόγιο System Usability Scale (SUS), το οποίο χρησιμοποιείται για την αποτίμηση της ευχρηστίας ενός συστήματος. Αρχικά παρουσιάζεται ο τρόπος υπολογισμού του SUS score και στη συνέχεια αναλύονται τα αποτελέσματα που θα προκύψουν από τη συμμετοχή των χρηστών. Μέσα από την ενότητα αυτή επιχειρείται να αξιολογηθεί κατά πόσο η εφαρμογή είναι κατανοητή, εύχρηστη και κατάλληλη για το κοινό στο οποίο απευθύνεται.

Τέλος, στο όγδοο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και οι πιθανές μελλοντικές επεκτάσεις της εφαρμογής. Συνοψίζονται τα βασικά σημεία που προέκυψαν από την ανάπτυξη και τη χρήση του TransferMind, ενώ παράλληλα προτείνονται ιδέες για περαιτέρω βελτίωση, όπως η ενσωμάτωση περισσότερων δεδομένων, η υποστήριξη επιπλέον πρωταθλημάτων, η προσθήκη νέων αλγορίθμων και η επέκταση των διαθέσιμων οπτικοποιήσεων.

Κεφάλαιο 2

Γλώσσες και Τεχνολογίες

Στο παρόν κεφάλαιο παρουσιάζονται οι γλώσσες προγραμματισμού, οι τεχνολογίες, οι βιβλιοθήκες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής TransferMind. Η εφαρμογή αποτελείται από διαφορετικά μέρη, τα οποία συνεργάζονται μεταξύ τους με σκοπό τη συλλογή, αποθήκευση, επεξεργασία, ανάλυση και οπτικοποίηση ποδοσφαιρικών δεδομένων. Η επιλογή των τεχνολογιών βασίστηκε στις ανάγκες του συστήματος και στον ρόλο που έπρεπε να καλύπτει κάθε επίπεδο της εφαρμογής.

Πιο συγκεκριμένα, οι βασικές τεχνολογίες που χρησιμοποιήθηκαν στην εφαρμογή είναι οι εξής:

- Node.js και Express, για την ανάπτυξη του backend και τη δημιουργία του API της εφαρμογής, την ανάκτηση, επεξεργασία και αποθήκευση ποδοσφαιρικών δεδομένων στη βάση δεδομένων από εξωτερικό API.
- Python, για την υλοποίηση των αλγορίθμων ανάλυσης και εξόρυξης δεδομένων, όπως η συσταδοποίηση ομάδων και η εξαγωγή κανόνων συσχέτισης.
- PostgreSQL και Supabase, για την αποθήκευση και διαχείριση των δεδομένων της εφαρμογής.
- React και TypeScript, για την ανάπτυξη του frontend και τη δημιουργία ενός διαδραστικού περιβάλλοντος χρήστη.

2.1 Node.js και Express

Το Node.js αποτελεί ένα περιβάλλον εκτέλεσης που επιτρέπει τη χρήση της JavaScript στην πλευρά του διακομιστή. Στην παρούσα εργασία αξιοποιήθηκε για την ανάπτυξη του backend της εφαρμογής, καθώς παρέχει ένα κατάλληλο περιβάλλον για τη δημιουργία διαδικτυακών υπηρεσιών, τη διαχείριση αιτημάτων και την επικοινωνία με άλλα μέρη του συστήματος. Η

χρήση της JavaScript στην πλευρά του διακομιστή έχει συνδεθεί με την ανάπτυξη δικτυακών εφαρμογών που βασίζονται σε ασύγχρονη και οδηγούμενη από γεγονότα εκτέλεση, χαρακτηριστικά που συναντώνται συχνά σε εφαρμογές Node.js [15]. Στο TransferMind, το Node.js λειτούργησε ως το βασικό επίπεδο του backend, το οποίο συντονίζει την επικοινωνία με το frontend, τη βάση δεδομένων και τα επιμέρους modules ανάλυσης.

2.1.1 Node.js Runtime και ασύγχρονη εκτέλεση

Ένα βασικό χαρακτηριστικό του Node.js είναι ο τρόπος με τον οποίο διαχειρίζεται λειτουργίες εισόδου και εξόδου. Σε μία εφαρμογή backend, αρκετές λειτουργίες δεν ολοκληρώνονται άμεσα, όπως η επικοινωνία με μία βάση δεδομένων, η αποστολή αιτημάτων σε εξωτερικές υπηρεσίες ή η εκτέλεση βοηθητικών διεργασιών. Αν το σύστημα περίμενε κάθε τέτοια λειτουργία να ολοκληρωθεί πριν συνεχίσει, η απόκριση της εφαρμογής θα μπορούσε να επιβαρυνθεί σημαντικά. Το Node.js ακολουθεί ένα ασύγχρονο και event-driven μοντέλο, το οποίο επιτρέπει στην εφαρμογή να συνεχίζει την εκτέλεσή της και να χειρίζεται άλλα αιτήματα όσο αναμένονται αποτελέσματα από λειτουργίες εισόδου και εξόδου [15].

Η λογική αυτή είναι χρήσιμη σε εφαρμογές που δέχονται πολλά αιτήματα ή χρειάζεται να επικοινωνούν συχνά με εξωτερικές υπηρεσίες. Στην περίπτωση του TransferMind, το χαρακτηριστικό αυτό ταιριάζει με τις ανάγκες του συστήματος, καθώς το backend διαχειρίζεται αιτήματα από το frontend, επικοινωνεί με τη βάση δεδομένων, ανακτά ποδοσφαιρικά δεδομένα από εξωτερικό API και εκτελεί διαδικασίες που σχετίζονται με τα Python modules. Έτσι, το Node.js δεν χρησιμοποιείται μόνο για τη δημιουργία ενός απλού διακομιστή, αλλά ως περιβάλλον που διευκολύνει τον συντονισμό διαφορετικών λειτουργιών της εφαρμογής.

2.1.2 Express Framework

Για την ανάπτυξη του backend χρησιμοποιήθηκε το Express, ένα framework που λειτουργεί πάνω από το Node.js και χρησιμοποιείται για τη δημιουργία διαδικτυακών εφαρμογών και APIs. Σε σχέση με τη χρήση του Node.js χωρίς κάποιο framework, το Express προσφέρει έναν πιο οργανωμένο τρόπο διαχείρισης των αιτημάτων, των απαντήσεων και των διαδρομών της εφαρμογής. Η χρήση του σε συνδυασμό με το Node.js εμφανίζεται συχνά σε στοίβες ανάπτυξης διαδικτυακών υπηρεσιών, όπως η MEAN στοίβα, όπου το Express.js και το Node.js χρησιμοποιούνται για την υλοποίηση RESTful υπηρεσιών [16].

Κεντρικό χαρακτηριστικό του Express είναι η δρομολόγηση των αιτημάτων. Μέσω των routes, ο προγραμματιστής μπορεί να ορίσει ποια λειτουργία θα εκτελείται όταν ο διακομιστής λαμβάνει ένα συγκεκριμένο αίτημα. Για παράδειγμα, ένα αίτημα μπορεί να αφορά την ανάκτηση δεδομένων, την αποστολή νέων δεδομένων ή την εκτέλεση μιας λειτουργίας ανάλυσης. Παράλληλα, το Express υποστηρίζει τη χρήση ενδιάμεσων συναρτήσεων, γνωστών ως middleware, οι οποίες παρεμβάλλονται στη ροή επεξεργασίας ενός αιτήματος. Τέτοιες συναρτήσεις μπορούν να χρησιμοποιηθούν για ελέγχους, επεξεργασία δεδομένων, διαχείριση σφαλμάτων ή άλλες βοηθητικές λειτουργίες του backend. Σε σχετική μελέτη σύγκρισης τεχνολογιών για REST APIs, το

Express.js εξετάζεται ως framework σχεδιασμένο για τη γλώσσα JavaScript και το περιβάλλον Node.js [17]. Στο πλαίσιο του TransferMind, το Express αξιοποιήθηκε ως το framework πάνω στο οποίο οργανώθηκε το backend API της εφαρμογής.

2.1.3 REST API

Η επικοινωνία ανάμεσα στο frontend και το backend οργανώνεται με βάση τη λογική ενός REST API. Το REST δεν είναι συγκεκριμένη τεχνολογία ή βιβλιοθήκη, αλλά αρχιτεκτονικό στυλ για δικτυακά συστήματα. Στη διδακτορική διατριβή του Roy Fielding, το REST παρουσιάζεται ως τρόπος οργάνωσης κατανεμημένων συστημάτων υπερμέσων, με βασικούς περιορισμούς που επηρεάζουν τον τρόπο επικοινωνίας μεταξύ των μερών ενός συστήματος [18]. Ένα από τα κύρια χαρακτηριστικά του είναι ο διαχωρισμός των ρόλων ανάμεσα στον client και τον server, ώστε το κάθε μέρος να έχει ξεκάθαρη ευθύνη.

Στην πράξη, ένα REST API οργανώνει την πρόσβαση στα δεδομένα μέσα από πόρους. Ένας πόρος μπορεί να αντιστοιχεί, για παράδειγμα, σε ομάδες, παίκτες, στατιστικά ή αποτελέσματα ανάλυσης. Η πρόσβαση σε αυτούς τους πόρους γίνεται μέσω συγκεκριμένων endpoints, ενώ η ενέργεια που ζητείται δηλώνεται συνήθως με μεθόδους του HTTP, όπως GET, POST, PUT ή DELETE. Η βιβλιογραφία αντιμετωπίζει τα RESTful APIs ως διαδεδομένο τρόπο παροχής υπηρεσιών μέσω του ιστού, ιδιαίτερα σε συστήματα όπου διαφορετικά μέρη της εφαρμογής χρειάζεται να επικοινωνούν μέσω δικτύου [19], [20]. Στο TransferMind, αυτή η προσέγγιση επιτρέπει στο frontend να επικοινωνεί με το backend με προβλέψιμο τρόπο, λαμβάνοντας δομημένες απαντήσεις που μπορούν να αξιοποιηθούν από το περιβάλλον χρήστη.

2.1.4 Επικοινωνία με εξωτερικά APIs και Python Modules

Ένα ακόμα σημείο στο οποίο αξιοποιείται το Node.js είναι η επικοινωνία με εξωτερικές υπηρεσίες και APIs. Σε μία εφαρμογή που βασίζεται σε δεδομένα, το backend συχνά χρειάζεται να στέλνει αιτήματα σε τρίτα συστήματα, να λαμβάνει απαντήσεις, να επεξεργάζεται τα δεδομένα και να τα αποθηκεύει στη βάση. Το ασύγχρονο μοντέλο του Node.js είναι κατάλληλο για τέτοιου είδους λειτουργίες, επειδή επιτρέπει τη διαχείριση λειτουργιών δικτύου χωρίς η εφαρμογή να εξαρτά την εκτέλεσή της από την άμεση ολοκλήρωση κάθε αιτήματος [15]. Στο TransferMind, αυτή η δυνατότητα αξιοποιείται για την ανάκτηση ποδοσφαιρικών δεδομένων από εξωτερικό API τρίτου παρόχου, συγκεκριμένα από το Sofascore μέσω RapidAPI. Για την πραγματοποίηση των HTTP αιτημάτων χρησιμοποιείται το Axios, ενώ τα δεδομένα που ανακτώνται επεξεργάζονται από το backend και αποθηκεύονται στη βάση δεδομένων.

Το Node.js χρησιμοποιήθηκε επίσης ως ενδιάμεσο επίπεδο ανάμεσα στο backend και τα Python modules της εφαρμογής. Παρόλο που το backend έχει αναπτυχθεί σε JavaScript, οι αλγόριθμοι ανάλυσης δεδομένων υλοποιούνται σε Python, καθώς η γλώσσα αυτή διαθέτει ώριμο οικοσύστημα βιβλιοθηκών για επεξεργασία δεδομένων, συσταδοποίηση και εξαγωγή κανόνων συσχέτισης. Για τη συνεργασία των δύο μερών, το backend εκκινεί ξεχωριστές διεργασίες που εκτελούν τα αντίστοιχα Python scripts. Μέσω αυτής της διαδικασίας, αποστέλλονται τα απα-

ραίτητα δεδομένα στα modules ανάλυσης, λαμβάνονται τα αποτελέσματα και στη συνέχεια επιστρέφονται στο frontend σε οργανωμένη μορφή. Με αυτόν τον τρόπο, το σύστημα αξιοποιεί το Node.js για την επικοινωνία και τον συντονισμό, ενώ χρησιμοποιεί την Python για τους υπολογιστικά εξειδικευμένους αλγορίθμους.

Συνολικά, το Node.js αποτέλεσε κατάλληλη επιλογή για το backend της εφαρμογής, καθώς επέτρεψε την οργάνωση του REST API, την ασύγχρονη επικοινωνία με εξωτερικές υπηρεσίες και τη σύνδεση με τα Python modules. Έτσι, λειτούργησε ως το βασικό επίπεδο συντονισμού ανάμεσα στο frontend, τη βάση δεδομένων, των εξωτερικών ποδοσφαιρικών δεδομένων και των αλγορίθμων ανάλυσης.

2.2 Python

Στην εφαρμογή TransferMind, η Python χρησιμοποιείται ως εξειδικευμένο υπολογιστικό επίπεδο για τις λειτουργίες ανάλυσης και εξόρυξης δεδομένων. Παρόλο που το κύριο backend της εφαρμογής βασίζεται στο Node.js και στο Express, οι πιο απαιτητικές αναλυτικές διαδικασίες εκτελούνται μέσω ξεχωριστών Python scripts, τα οποία καλούνται από το backend με τον μηχανισμό που περιγράφηκε στην προηγούμενη ενότητα. Με αυτόν τον τρόπο, η εφαρμογή συνδυάζει την ευελιξία ενός web backend με το επιστημονικό οικοσύστημα της Python, το οποίο περιλαμβάνει βιβλιοθήκες για αριθμητικούς υπολογισμούς, επεξεργασία δεδομένων, μηχανική μάθηση και οπτικοποίηση. Η Python έχει καθιερωθεί ιδιαίτερα στον χώρο της επιστήμης δεδομένων και της μηχανικής μάθησης, καθώς υποστηρίζεται από βιβλιοθήκες που επιτρέπουν την ανάπτυξη αναλυτικών εφαρμογών με σχετικά ενιαίο και πρακτικό τρόπο [21].

2.2.1 Επεξεργασία Δεδομένων

Ένας από τους βασικούς λόγους για τους οποίους η Python χρησιμοποιείται συχνά σε εφαρμογές ανάλυσης δεδομένων είναι το πλούσιο οικοσύστημα βιβλιοθηκών που διαθέτει για την οργάνωση, τον μετασχηματισμό και την αριθμητική επεξεργασία δεδομένων. Στο πλαίσιο τέτοιων εφαρμογών, τα αρχικά δεδομένα σπάνια βρίσκονται αμέσως στη μορφή που απαιτεί ένας αλγόριθμος. Συνήθως χρειάζεται να καθαριστούν, να φιλτραριστούν, να ελεγχθούν ως προς την εγκυρότητά τους και να μετατραπούν σε κατάλληλες δομές, όπως πίνακες ή διανύσματα χαρακτηριστικών.

Για τέτοιες διαδικασίες, μία από τις πιο διαδεδομένες βιβλιοθήκες είναι η pandas. Η pandas παρέχει τη δομή DataFrame, η οποία επιτρέπει την αναπαράσταση δεδομένων σε μορφή πίνακα με γραμμές και στήλες. Αυτή η μορφή είναι ιδιαίτερα κοντά στον τρόπο με τον οποίο αποθηκεύονται και παρουσιάζονται πολλά πραγματικά δεδομένα, όπως στατιστικά στοιχεία, εγγραφές βάσεων δεδομένων ή αρχεία CSV. Μέσα από μία τέτοια δομή, ο προγραμματιστής μπορεί να οργανώσει τα δεδομένα με βάση συγκεκριμένες στήλες, να επιλέξει υποσύνολα εγγραφών, να αντιμετωπίσει ελλιπείς τιμές και να εφαρμόσει μετασχηματισμούς πριν από την εκτέλεση ενός αλγορίθμου. Ο McKinney παρουσιάζει την pandas ως βιβλιοθήκη που αναπτύχθηκε για

να διευκολύνει την πρακτική επεξεργασία δομημένων δεδομένων στην Python, ιδιαίτερα σε περιβάλλοντα στατιστικής ανάλυσης και επιστημονικού υπολογισμού [22].

Παράλληλα, σημαντικό ρόλο έχει και η NumPy, η οποία αποτελεί βασικό θεμέλιο του επιστημονικού οικοσυστήματος της Python. Η NumPy επιτρέπει την αποδοτική αναπαράσταση και επεξεργασία αριθμητικών πινάκων, κάτι που είναι απαραίτητο όταν τα δεδομένα πρέπει να χρησιμοποιηθούν σε μαθηματικούς ή στατιστικούς υπολογισμούς. Σε αντίθεση με τις απλές λίστες της Python, οι πίνακες της NumPy είναι σχεδιασμένοι για αριθμητικές πράξεις μεγάλης κλίμακας και επιτρέπουν πράξεις πάνω σε ολόκληρους πίνακες ή διανύσματα. Αυτό είναι ιδιαίτερα χρήσιμο στην ανάλυση δεδομένων, επειδή πολλοί αλγόριθμοι βασίζονται σε αριθμητικά χαρακτηριστικά, αποστάσεις, πίνακες και διανύσματα. Οι Harris et al. περιγράφουν τη NumPy ως βασική βιβλιοθήκη για array programming στην Python και ως κεντρικό στοιχείο πάνω στο οποίο στηρίζεται μεγάλο μέρος του επιστημονικού οικοσυστήματος της γλώσσας [23].

Η pandas και η NumPy χρησιμοποιούνται συχνά συνδυαστικά στην ανάλυση δεδομένων. Η pandas προσφέρει δομές υψηλού επιπέδου, όπως το DataFrame, οι οποίες είναι κατάλληλες για την οργάνωση δεδομένων σε γραμμές και στήλες. Αντίστοιχα, η NumPy παρέχει αποδοτικούς αριθμητικούς πίνακες, οι οποίοι είναι χρήσιμοι σε μαθηματικούς υπολογισμούς και σε αλγορίθμους μηχανικής μάθησης. Στο πλαίσιο της εφαρμογής, ο συνδυασμός τους υποστηρίζει τη μετάβαση από τα αρχικά στατιστικά δεδομένα σε αριθμητικές μορφές κατάλληλες για συσταδοποίηση και εξόρυξη κανόνων.

Στο γενικό πλαίσιο της μηχανικής μάθησης, η επεξεργασία δεδομένων προηγείται σχεδόν πάντα της εφαρμογής του ίδιου του αλγορίθμου. Η ποιότητα και η μορφή των δεδομένων επηρεάζουν άμεσα το αποτέλεσμα της ανάλυσης. Για παράδειγμα, ένας αλγόριθμος συσταδοποίησης απαιτεί συνήθως αριθμητικά χαρακτηριστικά οργανωμένα σε πίνακα, ενώ ένας αλγόριθμος εξόρυξης κανόνων συσχέτισης απαιτεί δεδομένα σε μορφή συναλλαγών ή συνόλων αντικειμένων. Επομένως, οι βιβλιοθήκες επεξεργασίας δεδομένων δεν λειτουργούν απλώς βοηθητικά, αλλά αποτελούν βασικό μέρος της συνολικής αναλυτικής διαδικασίας.

2.2.2 Βιβλιοθήκες Μηχανικής Μάθησης

Η Python χρησιμοποιείται ευρέως σε εφαρμογές μηχανικής μάθησης, επειδή διαθέτει βιβλιοθήκες που παρέχουν έτοιμες και δοκιμασμένες υλοποιήσεις αλγορίθμων. Αντί ο προγραμματιστής να υλοποιεί από την αρχή πολύπλοκες μαθηματικές μεθόδους, μπορεί να αξιοποιήσει βιβλιοθήκες που έχουν αναπτυχθεί και ελεγχθεί από την επιστημονική και προγραμματιστική κοινότητα. Αυτό είναι ιδιαίτερα χρήσιμο σε εφαρμογές ανάλυσης δεδομένων, όπου η αξιοπιστία των αποτελεσμάτων εξαρτάται τόσο από την ποιότητα των δεδομένων όσο και από τη σωστή εφαρμογή των αλγορίθμων.

Μία από τις βασικότερες βιβλιοθήκες μηχανικής μάθησης στην Python είναι η scikit-learn. Η βιβλιοθήκη αυτή παρέχει ένα ενιαίο περιβάλλον για πολλούς αλγορίθμους εποπτευόμενης και μη εποπτευόμενης μάθησης, όπως ταξινόμηση, παλινδρόμηση, μείωση διαστατικότητας και συσταδοποίηση. Ένα από τα βασικά της πλεονεκτήματα είναι ότι προσφέρει κοινή λογική

χρήσης ανάμεσα στους αλγορίθμους, γεγονός που διευκολύνει τη μετάβαση από μία μέθοδο σε μία άλλη και μειώνει την πολυπλοκότητα της υλοποίησης. Σύμφωνα με τους Pedregosa et al., η scikit-learn σχεδιάστηκε ως βιβλιοθήκη μηχανικής μάθησης ανοιχτού κώδικα, με έμφαση στην ευκολία χρήσης, στην αποδοτικότητα και στην ενσωμάτωση με το ευρύτερο επιστημονικό οικοσύστημα της Python [24]. Στο πλαίσιο της συσταδοποίησης, ένας από τους πιο γνωστούς αλγορίθμους που παρέχει η scikit-learn είναι ο K-Means, ο οποίος χρησιμοποιείται στην εφαρμογή και θα αναλυθεί περαιτέρω σε ακόλουθο κεφάλαιο.

Εκτός από τη scikit-learn, σημαντική θέση στο επιστημονικό οικοσύστημα της Python έχει και η SciPy. Η SciPy παρέχει συλλογή από αλγορίθμους και εργαλεία για επιστημονικούς και αριθμητικούς υπολογισμούς, όπως γραμμική άλγεβρα, βελτιστοποίηση, στατιστική επεξεργασία και συσταδοποίηση. Οι Virtanen et al. παρουσιάζουν τη SciPy ως βιβλιοθήκη που συγκεντρώνει βασικούς αλγορίθμους επιστημονικού υπολογισμού στην Python και επισημαίνουν ότι το εύρος των λειτουργιών της καλύπτει, μεταξύ άλλων, στατιστική, γραμμική άλγεβρα, επεξεργασία σημάτων και clustering [25].

Σε σχέση με την ιεραρχική συσταδοποίηση, η SciPy χρησιμοποιείται επειδή παρέχει εργαλεία που επιτρέπουν την κατασκευή ιεραρχικών σχέσεων μεταξύ των παρατηρήσεων. Σε αυτή την προσέγγιση, τα δεδομένα δεν χωρίζονται απλώς σε έναν τελικό αριθμό ομάδων, αλλά οργανώνονται σε μια ακολουθία συγχωνεύσεων, η οποία δείχνει πώς τα πιο κοντινά στοιχεία ή ομάδες ενώνονται σταδιακά. Η θεωρία της ιεραρχικής συσταδοποίησης βασίζεται στην ιδέα ότι οι παρατηρήσεις μπορούν να οργανωθούν σε δενδρική δομή, ανάλογα με τις αποστάσεις ή τις ομοιότητές τους. Οι Murtagh και Contreras παρουσιάζουν μια επισκόπηση των αλγορίθμων ιεραρχικής συσταδοποίησης και αναλύουν τον ρόλο των agglomerative μεθόδων και των διαφορετικών τρόπων σύνδεσης συστάδων [26]. Η ύπαρξη διαφορετικών μεθόδων σύνδεσης, όπως ward, complete, average και single linkage, επιτρέπει διαφορετικούς τρόπους μέτρησης της απόστασης ανάμεσα σε συστάδες. Ειδικά για τη μέθοδο Ward, οι Murtagh και Legendre εξετάζουν τον τρόπο με τον οποίο υλοποιείται το κριτήριο της συγκεκριμένης μεθόδου σε αλγορίθμους ιεραρχικής συσταδοποίησης[27]. Έτσι, η SciPy δεν προσφέρει μόνο έναν τελικό διαχωρισμό των δεδομένων, αλλά και τη δυνατότητα ανάλυσης της ιεραρχικής σχέσης μεταξύ τους.

Για την εξόρυξη κανόνων συσχέτισης χρησιμοποιείται η βιβλιοθήκη mlxtend. Η mlxtend παρέχει επεκτάσεις και βοηθητικά εργαλεία για εργασίες μηχανικής μάθησης και επιστήμης δεδομένων, συμπληρώνοντας το οικοσύστημα της Python με επιπλέον λειτουργίες που δεν περιλαμβάνονται πάντα στις βασικές βιβλιοθήκες. Ο Raschka παρουσιάζει τη mlxtend ως βιβλιοθήκη που παρέχει utilities και επεκτάσεις για το επιστημονικό οικοσύστημα της Python, με λειτουργίες που σχετίζονται, μεταξύ άλλων, με machine learning, data science και association rule mining [28]. Στην παρούσα εργασία, η mlxtend αξιοποιείται για την εφαρμογή του Apriori και τη δημιουργία κανόνων συσχέτισης.

Ο Apriori είναι ένας κλασικός αλγόριθμος για την εύρεση συχνών συνόλων αντικειμένων και συνδέεται άμεσα με την εξόρυξη κανόνων συσχέτισης. Οι Agrawal και Srikant παρουσίασαν αποδοτικούς αλγορίθμους για την εξόρυξη κανόνων συσχέτισης σε μεγάλες βάσεις δεδομένων, με στόχο την εύρεση μοτίβων μέσα σε δεδομένα συναλλαγών [29]. Η βασική λογική είναι ότι, αν ένα σύνολο αντικειμένων εμφανίζεται συχνά, τότε τα υποσύνολά του είναι επίσης υποψήφια να εμφανίζονται συχνά. Αυτή η ιδιότητα βοηθά στη μείωση του χώρου αναζήτησης και

επιτρέπει στον αλγόριθμο να εντοπίζει συχνά μοτίβα πιο αποδοτικά. Με βάση τα συχνά σύνολα αντικειμένων, μπορούν να δημιουργηθούν κανόνες συσχέτισης που περιγράφουν σχέσεις της μορφής «αν εμφανίζεται το Α, τότε είναι πιθανό να εμφανίζεται και το Β». Η αρχική εργασία των Agrawal, Imieliński και Swami έθεσε το πρόβλημα της εξόρυξης κανόνων συσχέτισης σε μεγάλες βάσεις συναλλαγών και συνέδεσε τέτοιους κανόνες με την ανακάλυψη σχέσεων ανάμεσα σε σύνολα αντικειμένων [30]. Ο τρόπος λειτουργίας του Apriori και οι μετρικές αξιολόγησης των κανόνων, όπως support, confidence και lift, θα εξεταστούν αναλυτικότερα σε επόμενο κεφάλαιο.

Συνολικά, οι βιβλιοθήκες scikit-learn, SciPy και mlxtend καλύπτουν διαφορετικές ανάγκες της μηχανικής μάθησης και της εξόρυξης δεδομένων. Η scikit-learn προσφέρει ένα γενικό και συνεκτικό πλαίσιο για αλγορίθμους μηχανικής μάθησης, η SciPy παρέχει ισχυρά εργαλεία για επιστημονικούς υπολογισμούς και ιεραρχική συσταδοποίηση, ενώ η mlxtend επεκτείνει τις δυνατότητες της Python με εργαλεία όπως ο Apriori και οι κανόνες συσχέτισης. Ο συνδυασμός τους καθιστά την Python κατάλληλη για εφαρμογές που απαιτούν τόσο αριθμητική επεξεργασία όσο και πιο σύνθετες αναλυτικές μεθόδους.

2.2.3 Βιβλιοθήκες Οπτικοποίησης

Η οπτικοποίηση δεδομένων αποτελεί σημαντικό μέρος της ανάλυσης, καθώς επιτρέπει την καλύτερη κατανόηση μοτίβων, σχέσεων και διαφορών που μπορεί να μην είναι άμεσα εμφανείς μέσα από αριθμητικούς πίνακες. Σε εφαρμογές ανάλυσης δεδομένων και μηχανικής μάθησης, τα γραφήματα δεν χρησιμοποιούνται μόνο για την παρουσίαση αποτελεσμάτων, αλλά και για την ερμηνεία τους. Για παράδειγμα, ένα γράφημα μπορεί να βοηθήσει στην κατανόηση της κατανομής των δεδομένων, στη σύγκριση ομάδων ή στην παρουσίαση της δομής που δημιουργείται από έναν αλγόριθμο συσταδοποίησης.

Στο οικοσύστημα της Python, μία από τις πιο γνωστές βιβλιοθήκες για οπτικοποίηση είναι η Matplotlib. Η Matplotlib παρέχει εργαλεία για τη δημιουργία δισδιάστατων γραφημάτων, όπως γραμμικά διαγράμματα, ραβδογράμματα, διαγράμματα διασποράς και άλλες μορφές απεικόνισης δεδομένων. Επιτρέπει επίσης έλεγχο πάνω σε στοιχεία όπως οι άξονες, οι ετικέτες, οι τίτλοι και η μορφή εξαγωγής του γραφήματος. Γι' αυτόν τον λόγο χρησιμοποιείται συχνά τόσο σε ερευνητικά περιβάλλοντα όσο και σε εφαρμογές που χρειάζονται στατικές ή δημοσιεύσιμες οπτικοποιήσεις. Ο Hunter περιγράφει τη Matplotlib ως περιβάλλον παραγωγής δισδιάστατων γραφικών για την Python, το οποίο μπορεί να χρησιμοποιηθεί για ανάπτυξη εφαρμογών, διαδραστική χρήση και δημιουργία γραφημάτων κατάλληλων για δημοσίευση [31].

Στην περίπτωση της ιεραρχικής συσταδοποίησης, η οπτικοποίηση συνδέεται συχνά με το δενδρόγραμμα. Το δενδρόγραμμα είναι ένα διάγραμμα που παρουσιάζει τη διαδικασία συγχώνευσης των παρατηρήσεων ή των συστάδων σε έναν ιεραρχικό αλγόριθμο. Μέσα από αυτό, ο αναγνώστης μπορεί να δει ποια στοιχεία ενώνονται πρώτα, ποιες ομάδες σχηματίζονται στη συνέχεια και σε ποιο επίπεδο απόστασης πραγματοποιούνται οι συγχωνεύσεις. Η χρήση δενδρογραμμάτων συνδέεται άμεσα με την ιεραρχική συσταδοποίηση, καθώς η ίδια η μέθοδος παράγει μια δενδρική αναπαράσταση των σχέσεων ανάμεσα στις παρατηρήσεις [26].

Η συνεργασία της SciPy με τη Matplotlib είναι σημαντική σε αυτό το σημείο. Η SciPy παρέχει τα αποτελέσματα των επιστημονικών και αλγοριθμικών υπολογισμών, ενώ η Matplotlib αναλαμβάνει την απεικόνισή τους. Αυτό δείχνει ένα γενικό χαρακτηριστικό του οικοσυστήματος της Python: οι βιβλιοθήκες δεν λειτουργούν απομονωμένα, αλλά συνδυάζονται μεταξύ τους. Μία βιβλιοθήκη μπορεί να είναι υπεύθυνη για τον υπολογισμό ενός αποτελέσματος, ενώ μία άλλη για την παρουσίασή του σε κατανοητή οπτική μορφή.

Συνολικά, η Matplotlib χρησιμοποιήθηκε ως βασικό εργαλείο για την παραγωγή στατικών γραφημάτων, όπως το δενδρόγραμμα της ιεραρχικής συσταδοποίησης. Σε συνδυασμό με τη SciPy, επιτρέπει τα αποτελέσματα των αλγορίθμων να παρουσιάζονται με πιο κατανοητό και ερμηνεύσιμο τρόπο. Με αυτόν τον τρόπο, η οπτικοποίηση δεν αποτελεί απλώς αισθητικό στοιχείο, αλλά μέρος της ίδιας της διαδικασίας ανάλυσης και κατανόησης των δεδομένων.

2.3 PostgreSQL / Supabase

Για την υλοποίηση της εφαρμογής επιλέχθηκε ο συνδυασμός PostgreSQL και Supabase, επειδή καλύπτει τις βασικές ανάγκες μιας εφαρμογής που βασίζεται σε οργανωμένα, συσχετισμένα και επαναχρησιμοποιήσιμα δεδομένα. Η PostgreSQL χρησιμοποιείται ως σχεσιακή βάση δεδομένων, δηλαδή ως το επίπεδο στο οποίο αποθηκεύονται οι βασικές οντότητες της εφαρμογής, οι σχέσεις μεταξύ τους και τα στατιστικά στοιχεία που αξιοποιούνται από το υπόλοιπο σύστημα. Το Supabase λειτουργεί συμπληρωματικά, προσφέροντας ένα διαχειριζόμενο περιβάλλον γύρω από την PostgreSQL, ώστε η βάση να μπορεί να συνδεθεί πιο εύκολα με την εφαρμογή και να υποστηρίξει λειτουργίες που σχετίζονται με πρόσβαση, ασφάλεια και ανάπτυξη backend υπηρεσιών. Με αυτόν τον τρόπο, η PostgreSQL αναλαμβάνει την οργάνωση και ακεραιότητα των δεδομένων, ενώ το Supabase διευκολύνει την αξιοποίησή τους μέσα στο συνολικό πληροφοριακό σύστημα.

2.3.1 PostgreSQL

Η PostgreSQL αποτελεί σύγχρονη εξέλιξη του POSTGRES, ενός συστήματος βάσεων δεδομένων που αναπτύχθηκε στο Πανεπιστήμιο της Καλιφόρνιας στο Berkeley. Οι Stonebraker και Kemnitz παρουσιάζουν το POSTGRES ως ένα σύστημα επόμενης γενιάς, το οποίο επεκτείνει το κλασικό σχεσιακό μοντέλο με δυνατότητες που σχετίζονται με πιο σύνθετα δεδομένα, κανόνες, επεκτασιμότητα και πιο πλούσια μοντελοποίηση πληροφοριών [32]. Η PostgreSQL ακολουθεί αυτή τη λογική και χρησιμοποιείται ευρέως ως αντικειμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων, κατάλληλο για εφαρμογές που απαιτούν σταθερή αποθήκευση, οργανωμένη ανάκτηση και διατήρηση της συνέπειας των δεδομένων.

Η βασική λογική μιας σχεσιακής βάσης δεδομένων στηρίζεται στην οργάνωση των δεδομένων σε πίνακες. Κάθε πίνακας αντιπροσωπεύει μια συγκεκριμένη κατηγορία δεδομένων, ενώ κάθε γραμμή αντιστοιχεί σε μία εγγραφή και κάθε στήλη σε ένα χαρακτηριστικό αυτής της εγγραφής. Αυτή η μορφή είναι ιδιαίτερα χρήσιμη όταν τα δεδομένα μπορούν να χωριστούν σε

καθαρές οντότητες και να συνδεθούν μεταξύ τους με σαφείς σχέσεις. Στο πλαίσιο μιας εφαρμογής ανάλυσης ποδοσφαιρικών δεδομένων, τέτοιες οντότητες μπορεί να είναι οι ομάδες, οι παίκτες, οι διοργανώσεις, οι σεζόν και τα στατιστικά στοιχεία. Η χρήση του σχεσιακού μοντέλου επιτρέπει σε αυτές τις πληροφορίες να αποθηκεύονται ξεχωριστά, αλλά να συνδέονται όταν απαιτείται από την εφαρμογή.

Η σύνδεση των πινάκων πραγματοποιείται μέσω κλειδιών και σχέσεων. Ένα πρωτεύον κλειδί προσδιορίζει μοναδικά κάθε εγγραφή ενός πίνακα, ενώ ένα ξένο κλειδί χρησιμοποιείται για να δηλώσει σχέση με εγγραφή άλλου πίνακα. Με αυτόν τον τρόπο περιορίζεται η άσκοπη επανάληψη των ίδιων δεδομένων και διευκολύνεται η διατήρηση της συνέπειάς τους. Η αξία αυτής της λογικής δεν περιορίζεται μόνο στην αποθήκευση. Επηρεάζει και τον τρόπο με τον οποίο η εφαρμογή ανακτά δεδομένα, επειδή επιτρέπει τη σύνθεση πληροφοριών από διαφορετικούς πίνακες μέσω ερωτημάτων. Σε ερευνητικό πλαίσιο, οι Yatsenko, Walker και Tolias αναφέρουν ότι το σχεσιακό μοντέλο παρέχει αυστηρότητα και σαφήνεια στον ορισμό δομών δεδομένων και στη διατύπωση ερωτημάτων πάνω σε πολύπλοκα δεδομένα [33].

Σημαντικό ρόλο στη σχεδίαση μιας βάσης δεδομένων έχει και η έννοια του schema. Το schema μπορεί να θεωρηθεί ως το οργανωτικό πλαίσιο μέσα στο οποίο ορίζονται τα αντικείμενα της βάσης, όπως πίνακες, σχέσεις, περιορισμοί και βοηθητικές δομές. Η ύπαρξη ενός καθαρού schema βοηθά ώστε η βάση να παραμένει κατανοητή και επεκτάσιμη, ειδικά όταν η εφαρμογή μεγαλώνει και προστίθενται νέες λειτουργίες. Στην παρούσα εργασία, η ύπαρξη οργανωμένου schema είναι απαραίτητη, επειδή τα δεδομένα που χρησιμοποιούνται δεν είναι απομονωμένες εγγραφές, αλλά συνδέονται μεταξύ τους. Για παράδειγμα, ένας παίκτης σχετίζεται με ομάδα, μία ομάδα με διοργάνωση και σεζόν, ενώ τα στατιστικά στοιχεία εξαρτώνται από συγκεκριμένο πλαίσιο αγώνων ή περιόδου.

Η PostgreSQL είναι κατάλληλη για τέτοιες περιπτώσεις, επειδή επιτρέπει τη διαχείριση δομημένων δεδομένων με συνέπεια και παρέχει μηχανισμούς που βοηθούν στη διατήρηση της ακεραιότητας της βάσης. Σε μια εφαρμογή όπως το TransferMind, η βάση δεν λειτουργεί απλώς ως αποθηκευτικός χώρος. Αντίθετα, αποτελεί το σημείο στο οποίο συγκεντρώνονται τα δεδομένα που ανακτώνται από εξωτερικές πηγές, οργανώνονται με βάση το μοντέλο της εφαρμογής και στη συνέχεια αξιοποιούνται από το backend, το frontend και τα modules ανάλυσης δεδομένων. Επομένως, η επιλογή μιας σχεσιακής βάσης δεδομένων ταιριάζει με τη φύση του προβλήματος, καθώς τα ποδοσφαιρικά δεδομένα περιλαμβάνουν πολλές οντότητες και σχέσεις που πρέπει να διατηρούνται με σαφή και αξιόπιστο τρόπο.

2.3.2 Supabase

Το Supabase χρησιμοποιήθηκε ως πλατφόρμα που βασίζεται στην PostgreSQL και παρέχει έτοιμες backend υπηρεσίες γύρω από αυτήν. Σε αντίθεση με την απλή εγκατάσταση μιας βάσης δεδομένων, μια πλατφόρμα backend-as-a-service προσπαθεί να μειώσει μέρος της υποδομής που θα έπρεπε να υλοποιηθεί χειροκίνητα, όπως η διαχείριση σύνδεσης με τη βάση, η πρόσβαση στα δεδομένα και η υποστήριξη βοηθητικών υπηρεσιών. Η βιβλιογραφία για serverless και Backend-as-a-Service αρχιτεκτονικές αναφέρει ότι τέτοιες προσεγγίσεις μεταφέρουν μέρος της

ευθύνης υποδομής στην πλατφόρμα, επιτρέποντας στον προγραμματιστή να εστιάσει περισσότερο στη λογική της εφαρμογής και λιγότερο στη διαχείριση χαμηλού επιπέδου υποδομών [34].

Στο πλαίσιο της παρούσας εργασίας, το Supabase αξιοποιήθηκε επειδή συνδυάζει τη σχεσιακή λογική της PostgreSQL με ένα πιο άμεσο περιβάλλον ανάπτυξης εφαρμογών. Η ύπαρξη γραφικού περιβάλλοντος διαχείρισης, η δυνατότητα σύνδεσης με εφαρμογές μέσω API και η υποστήριξη υπηρεσιών που σχετίζονται με αποθήκευση και ασφάλεια το καθιστούν χρήσιμο σε εφαρμογές που χρειάζονται γρήγορη και οργανωμένη πρόσβαση στα δεδομένα. Οι Kolesnikov et al. εξετάζουν τη χρήση σύγχρονων cloud τεχνολογιών και serverless αρχιτεκτονικών για την ανάπτυξη web εφαρμογών και αναφέρουν πρακτική αξιοποίηση της πλατφόρμας Supabase σε πραγματικό σενάριο ανάπτυξης πληροφοριακού συστήματος [35]. Αυτό δείχνει ότι τέτοιες πλατφόρμες μπορούν να χρησιμοποιηθούν ως πρακτική λύση για την ταχύτερη ανάπτυξη εφαρμογών που βασίζονται σε δεδομένα.

Ένα βασικό πλεονέκτημα του Supabase είναι ότι επιτρέπει στην εφαρμογή να αξιοποιεί τη βάση δεδομένων με πιο άμεσο τρόπο, χωρίς να χάνεται η σχεσιακή δομή της PostgreSQL. Οι πίνακες, οι σχέσεις και οι περιορισμοί παραμένουν στο επίπεδο της βάσης, ενώ η εφαρμογή μπορεί να επικοινωνεί με τα δεδομένα μέσα από οργανωμένο backend κώδικα και API. Αυτό είναι σημαντικό, επειδή η βάση δεν αντιμετωπίζεται ως απλή αποθήκη αρχείων, αλλά ως κεντρικό σημείο οργάνωσης της πληροφορίας. Για την εφαρμογή TransferMind, αυτό σημαίνει ότι τα δεδομένα που σχετίζονται με ομάδες, παίκτες, διοργανώσεις και στατιστικά μπορούν να διατηρούνται σε συνεκτική μορφή και να χρησιμοποιούνται από διαφορετικά μέρη του συστήματος.

Η ασφάλεια και ο έλεγχος πρόσβασης αποτελούν επίσης σημαντικό μέρος μιας εφαρμογής που βασίζεται σε βάση δεδομένων. Σε συστήματα όπου διαφορετικοί χρήστες ή διαφορετικά επίπεδα της εφαρμογής έχουν πρόσβαση σε δεδομένα, είναι απαραίτητο να υπάρχουν μηχανισμοί που περιορίζουν τι μπορεί να ανακτηθεί ή να τροποποιηθεί. Στο πεδίο των βάσεων δεδομένων, οι μηχανισμοί ελέγχου πρόσβασης χρησιμοποιούνται για να διατηρείται η προστασία των δεδομένων και να αποφεύγεται η μη εξουσιοδοτημένη χρήση τους. Οι Mohamed et al. επισημαίνουν ότι οι απαιτήσεις για ασφάλεια και έλεγχο πρόσβασης σε βάσεις δεδομένων έχουν αυξηθεί λόγω της σημασίας της προστασίας δεδομένων, των νομικών απαιτήσεων και της χρήσης βάσεων σε κρίσιμα πληροφοριακά συστήματα [36].

Σε επίπεδο πιο λεπτομερούς ελέγχου πρόσβασης, η PostgreSQL και το Supabase υποστηρίζουν μηχανισμούς όπως το Row Level Security. Σύμφωνα με τους Dar, Hershcovitch και Morrison, το Row Level Security επιτρέπει τον περιορισμό της πρόσβασης σε συγκεκριμένες εγγραφές ενός πίνακα, ανάλογα με τον χρήστη ή το πλαίσιο πρόσβασης [37]. Παρόλο που η παρούσα εργασία δεν εστιάζει στην ασφάλεια, η ύπαρξη τέτοιων μηχανισμών δείχνει ότι το οικοσύστημα PostgreSQL/Supabase μπορεί να υποστηρίξει εφαρμογές που απαιτούν πιο ελεγχόμενη διαχείριση δεδομένων.

Συνολικά, ο συνδυασμός PostgreSQL και Supabase κρίθηκε κατάλληλος για την εφαρμογή, επειδή προσφέρει τα πλεονεκτήματα μιας σχεσιακής βάσης δεδομένων μαζί με τις ευκολίες μιας σύγχρονης cloud πλατφόρμας. Η PostgreSQL παρέχει τη δομή, τις σχέσεις και τη συνέπεια των δεδομένων, ενώ το Supabase διευκολύνει την ανάπτυξη και τη σύνδεση της βάσης με

την υπόλοιπη εφαρμογή. Για το TransferMind, όπου τα δεδομένα προέρχονται από εξωτερικές πηγές και στη συνέχεια χρησιμοποιούνται σε προφίλ ομάδων, προφίλ παικτών, συγκρίσεις και αλγορίθμους ανάλυσης, αυτή η επιλογή υποστηρίζει τόσο την οργάνωση όσο και την πρακτική αξιοποίηση της πληροφορίας.

2.4 React / TypeScript

Το React και το TypeScript χρησιμοποιήθηκαν στην εφαρμογή TransferMind για την ανάπτυξη του frontend, δηλαδή του τμήματος της εφαρμογής με το οποίο αλληλεπιδρά ο χρήστης. Σε αντίθεση με το backend, το οποίο είναι υπεύθυνο για την επικοινωνία με τη βάση δεδομένων, τα εξωτερικά APIs και τα Python modules, το frontend αναλαμβάνει την παρουσίαση των δεδομένων, τη διαχείριση των επιλογών του χρήστη και την εμφάνιση των αποτελεσμάτων σε διαδραστική μορφή. Ο συνδυασμός React και TypeScript επιλέχθηκε επειδή επιτρέπει την ανάπτυξη διεπαφών που βασίζονται σε επαναχρησιμοποιήσιμα τμήματα κώδικα, ενώ παράλληλα προσφέρει καλύτερο έλεγχο στη μορφή των δεδομένων που χρησιμοποιούνται στην εφαρμογή.

2.4.1 React

Το React είναι βιβλιοθήκη της JavaScript για την ανάπτυξη διεπαφών χρήστη, με βασική ιδέα την οργάνωση της εφαρμογής σε components. Ένα component μπορεί να θεωρηθεί ως ανεξάρτητο τμήμα της διεπαφής, το οποίο δέχεται δεδομένα, διατηρεί κατάσταση όπου χρειάζεται και παράγει το αντίστοιχο οπτικό αποτέλεσμα. Αντί η διεπαφή να υλοποιείται ως ένα ενιαίο και δύσκολα διαχειρίσιμο σύνολο κώδικα, χωρίζεται σε μικρότερα μέρη, όπως πίνακες, φόρμες, κάρτες, γραφήματα και panels αποτελεσμάτων. Οι Madsen, Lhoták και Tip αναλύουν τη σημασιολογία του React και περιγράφουν τις εφαρμογές React ως δομές που αποτελούνται από components οργανωμένα σε δενδρική μορφή, με properties, state και διαδικασίες render που καθορίζουν την εμφάνιση της διεπαφής [38].

Η λογική των components είναι ιδιαίτερα χρήσιμη σε εφαρμογές όπως το TransferMind, όπου το περιβάλλον χρήστη περιλαμβάνει διαφορετικές σελίδες και λειτουργίες. Για παράδειγμα, η αρχική σελίδα, η προβολή βαθμολογιών, το προφίλ ομάδας, η αναζήτηση παικτών και η σύγκριση ομάδων μπορούν να οργανωθούν ως ξεχωριστά τμήματα της εφαρμογής. Αυτή η προσέγγιση διευκολύνει την ανάπτυξη, επειδή κάθε μέρος της διεπαφής μπορεί να σχεδιαστεί, να ελεγχθεί και να επαναχρησιμοποιηθεί ανεξάρτητα από τα υπόλοιπα. Παράλληλα, όταν αλλάζει η κατάσταση της εφαρμογής, το React αναλαμβάνει την ενημέρωση της διεπαφής με τρόπο που συνδέει τα δεδομένα με την οπτική παρουσίασή τους.

Η εφαρμογή λειτουργεί ως single-page application, δηλαδή η πλοήγηση ανάμεσα στις βασικές οθόνες δεν απαιτεί πλήρη επαναφόρτωση της σελίδας. Η λογική αυτή είναι χαρακτηριστική των σύγχρονων web εφαρμογών, στις οποίες η διεπαφή ενημερώνεται δυναμικά με βάση τις επιλογές του χρήστη και τα δεδομένα που λαμβάνονται από τον server. Οι Mesbah και van Deursen περιγράφουν τις single-page AJAX εφαρμογές ως μοντέλο στο οποίο η διεπαφή

αποτελείται από επιμέρους τμήματα που μπορούν να ενημερώνονται ανεξάρτητα, με στόχο καλύτερη διαδραστικότητα και μικρότερη αντιληπτή καθυστέρηση για τον χρήστη [39]. Στο TransferMind, η πλοήγηση ανάμεσα στις σελίδες οργανώνεται με React Router, ώστε συγκεκριμένα paths της εφαρμογής να αντιστοιχούν σε συγκεκριμένες οθόνες.

Ένα ακόμη σημαντικό στοιχείο του React είναι η διαχείριση κατάστασης. Σε μια διαδραστική εφαρμογή, η διεπαφή δεν παραμένει στατική. Αλλάζει ανάλογα με τις επιλογές του χρήστη, τα δεδομένα που ανακτώνται από το backend και τα αποτελέσματα που επιστρέφουν οι αλγόριθμοι ανάλυσης. Στο TransferMind, η κατάσταση χρησιμοποιείται για επιλογές όπως οι ομάδες που συγκρίνονται, τα στατιστικά που επιλέγονται, ο αριθμός των συστάδων, τα αποτελέσματα των αλγορίθμων και οι πληροφορίες που εμφανίζονται σε πίνακες ή γραφήματα. Η έννοια της κατάστασης είναι κεντρική στο React, επειδή η αλλαγή της οδηγεί σε ανανέωση της διεπαφής και επιτρέπει στον χρήστη να βλέπει άμεσα το αποτέλεσμα των ενεργειών του [38].

2.4.2 TypeScript

Το TypeScript χρησιμοποιήθηκε μαζί με το React για να προσθέσει στατικό έλεγχο τύπων στον κώδικα του frontend. Η JavaScript είναι δυναμικά τυποποιημένη γλώσσα, κάτι που προσφέρει ευελιξία, αλλά μπορεί να οδηγήσει σε σφάλματα που εμφανίζονται μόνο κατά την εκτέλεση. Το TypeScript προσθέτει ένα επίπεδο τύπων πάνω από τη JavaScript, επιτρέποντας στον προγραμματιστή να περιγράφει τη μορφή των δεδομένων, των αντικειμένων και των συναρτήσεων πριν από την εκτέλεση του προγράμματος. Οι Bierman, Abadi και Torgersen παρουσιάζουν το TypeScript ως επέκταση της JavaScript που υποστηρίζει, μεταξύ άλλων, classes, interfaces, modules και ένα gradual type system, με στόχο την καλύτερη ανάπτυξη μεγαλύτερων εφαρμογών [40].

Η χρήση TypeScript είναι ιδιαίτερα χρήσιμη σε εφαρμογές που διαχειρίζονται σύνθετες δομές δεδομένων. Στο TransferMind, το frontend δεν χειρίζεται απλές τιμές, αλλά αντικείμενα που περιγράφουν ομάδες, παίκτες, σεζόν, στατιστικά, αποτελέσματα συσταδοποίησης και κανόνες συσχέτισης. Με τη χρήση τύπων και interfaces, η εφαρμογή μπορεί να περιγράφει με μεγαλύτερη σαφήνεια ποια πεδία αναμένονται σε κάθε αντικείμενο και πώς αυτά χρησιμοποιούνται από τα components. Αυτό βοηθά στην καλύτερη οργάνωση του κώδικα και μειώνει την πιθανότητα λαθών που οφείλονται σε λανθασμένη χρήση δεδομένων.

Ο στατικός έλεγχος τύπων δεν σημαίνει ότι όλα τα σφάλματα αποφεύγονται αυτόματα. Ωστόσο, μπορεί να εντοπίσει ένα μέρος των προβλημάτων πριν η εφαρμογή εκτελεστεί στον browser. Οι Gao, Bird και Barr μελέτησαν δημόσια σφάλματα σε JavaScript projects και εξέτασαν σε ποιο βαθμό στατικά συστήματα τύπων, όπως το TypeScript και το Flow, θα μπορούσαν να τα εντοπίσουν πριν από τη διόρθωσή τους. Τα αποτελέσματά τους δείχνουν ότι ο στατικός έλεγχος τύπων μπορεί να εντοπίσει σημαντικό ποσοστό σφαλμάτων, αν και δεν αποτελεί από μόνος του πλήρη λύση για την ποιότητα λογισμικού [41]. Επομένως, στην παρούσα εφαρμογή, η αξία του TypeScript βρίσκεται κυρίως στην καλύτερη περιγραφή των δεδομένων, στη σαφέστερη οργάνωση των components και στον έλεγχο λαθών σε πρώιμο στάδιο ανάπτυξης.

Ιδιαίτερα χρήσιμη είναι η χρήση του TypeScript στην επικοινωνία του frontend με το backend.

Η εφαρμογή διαθέτει κεντρικό επίπεδο API, μέσα από το οποίο γίνονται αιτήματα προς το Express backend. Εκεί ορίζονται συναρτήσεις που καλούν endpoints για ομάδες, παίκτες, βαθμολογίες, δεδομένα σύγκρισης, K-Means clustering, Agglomerative clustering και Association Rules. Η ύπαρξη τύπων για τα δεδομένα που στέλνονται και λαμβάνονται βοηθά ώστε το frontend να γνωρίζει τη μορφή των απαντήσεων και να τις αξιοποιεί σωστά στην παρουσίαση. Η λογική αυτή συνδέεται και με τις αρχές των RESTful υπηρεσιών, όπου η επικοινωνία μεταξύ client και server οργανώνεται γύρω από πόρους και αιτήματα με σαφή δομή [18], [19].

2.4.3 Οπτικοποίηση και διαδραστική παρουσίαση δεδομένων

Στο κομμάτι της ανάλυσης, το React και το TypeScript έχουν σημαντικό ρόλο στην εμφάνιση των αποτελεσμάτων των αλγορίθμων. Το frontend δεν εκτελεί τους αλγορίθμους εξόρυξης δεδομένων, αλλά είναι το επίπεδο στο οποίο τα αποτελέσματά τους μετατρέπονται σε πληροφορία κατανοητή για τον χρήστη. Στη σελίδα σύγκρισης ομάδων, για παράδειγμα, ο χρήστης μπορεί να επιλέξει στατιστικά χαρακτηριστικά και να δει αποτελέσματα από μεθόδους όπως K-Means και Agglomerative clustering. Τα αποτελέσματα αυτά παρουσιάζονται μέσα από πίνακες, περιλήψεις συστάδων και γραφήματα, ώστε η ανάλυση να μην παραμένει μόνο σε αριθμητική μορφή.

Για ορισμένα γραφήματα της εφαρμογής χρησιμοποιείται η βιβλιοθήκη Recharts. Η επιλογή αυτή ταιριάζει με τη λογική του React, επειδή τα γραφήματα ενσωματώνονται στη διεπαφή ως components και μπορούν να ενημερώνονται όταν αλλάζουν τα δεδομένα. Παρόλο που η συγκεκριμένη βιβλιοθήκη αποτελεί τεχνική επιλογή της υλοποίησης, η γενικότερη ιδέα της δυναμικής οπτικοποίησης δεδομένων στο web συνδέεται με την ευρύτερη βιβλιογραφία της διαδικτυακής οπτικοποίησης. Οι Bostock, Ogievetsky και Heer, παρουσιάζοντας το D3, δείχνουν πώς η σύνδεση δεδομένων με στοιχεία του Document Object Model μπορεί να υποστηρίξει δυναμικές και διαδραστικές οπτικοποιήσεις στο περιβάλλον του browser [42]. Με παρόμοια λογική, τα γραφήματα στο TransferMind βοηθούν ώστε τα αριθμητικά αποτελέσματα των αναλυτικών διαδικασιών να παρουσιαστούν με πιο άμεσο και κατανοητό τρόπο.

Η χρήση οπτικών αναπαραστάσεων είναι σημαντική σε εφαρμογές ανάλυσης δεδομένων, επειδή ο χρήστης συχνά χρειάζεται να συγκρίνει ομάδες, να εντοπίσει διαφορές ή να κατανοήσει τη δομή που προκύπτει από έναν αλγόριθμο. Ένας πίνακας αριθμών μπορεί να παρέχει ακριβή πληροφορία, αλλά δεν βοηθά πάντα στην άμεση κατανόηση μοτίβων. Αντίθετα, ένα γράφημα μπορεί να κάνει πιο εμφανείς τις σχέσεις μεταξύ μεταβλητών, τις διαφορές ανάμεσα σε ομάδες ή την κατανομή των αποτελεσμάτων. Για τον λόγο αυτό, το frontend του TransferMind δεν λειτουργεί μόνο ως απλή οθόνη εμφάνισης δεδομένων, αλλά ως βασικό επίπεδο ερμηνείας των αποτελεσμάτων της εφαρμογής.

Συνολικά, ο συνδυασμός React και TypeScript επέτρεψε την ανάπτυξη ενός οργανωμένου και διαδραστικού frontend. Το React υποστήριξε τη δημιουργία επαναχρησιμοποιήσιμων components και δυναμικών οθονών, ενώ το TypeScript βοήθησε στον καλύτερο έλεγχο των δεδομένων που ανταλλάσσονται με το backend. Έτσι, τα αποτελέσματα της εφαρμογής μπορούν να παρουσιάζονται στον χρήστη με πιο σταθερό, σαφή και κατανοητό τρόπο.

Κεφάλαιο 3

Αλγόριθμοι Εξόρυξης Δεδομένων

Στο κεφάλαιο αυτό παρουσιάζονται οι βασικοί αλγόριθμοι εξόρυξης δεδομένων που χρησιμοποιούνται στην εφαρμογή TransferMind. Οι αλγόριθμοι αυτοί αξιοποιούνται με σκοπό την ανάλυση ποδοσφαιρικών δεδομένων, την αναγνώριση ομοιοτήτων ανάμεσα σε ομάδες ή αγωνιστικές περιόδους, καθώς και την εξαγωγή χρήσιμων προτύπων από τα διαθέσιμα στατιστικά στοιχεία.

Αρχικά, αναλύεται η έννοια της ανάλυσης συστάδων, η οποία αποτελεί τη βάση για την ομαδοποίηση δεδομένων με παρόμοια χαρακτηριστικά. Στη συνέχεια, παρουσιάζεται ο αλγόριθμος K-Means, μαζί με τη μέθοδο Elbow, η οποία χρησιμοποιείται για την εκτίμηση ενός κατάλληλου αριθμού συστάδων. Έπειτα, εξετάζεται η ιεραρχική συσταδοποίηση μέσω του Agglomerative Clustering, με αναφορά στο dendrogram και στις διαφορετικές μεθόδους linkage. Τέλος, παρουσιάζεται η εξόρυξη κανόνων συσχέτισης με τον αλγόριθμο Apriori, η οποία χρησιμοποιείται για τον εντοπισμό σχέσεων και συχνών συνδυασμών μέσα στα δεδομένα.

3.1 Ανάλυση Συστάδων – Cluster Analysis

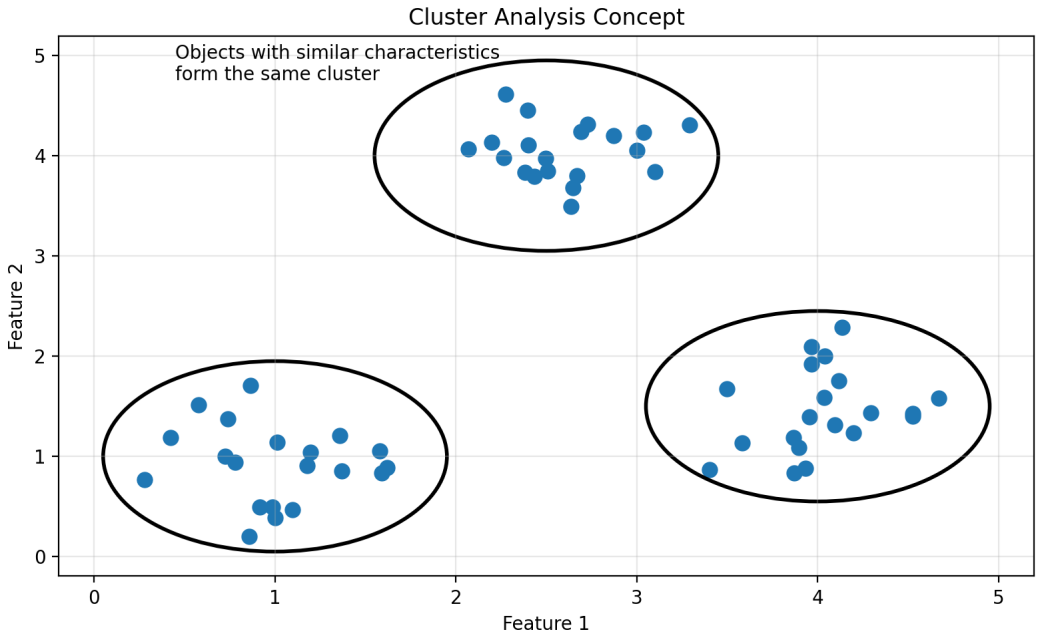
Η ανάλυση συστάδων, γνωστή και ως cluster analysis, αποτελεί μία από τις βασικές μεθόδους της μη επιβλεπόμενης μάθησης. Σε αντίθεση με την επιβλεπόμενη μάθηση, όπου τα δεδομένα συνοδεύονται από γνωστές κατηγορίες ή ετικέτες, η συσταδοποίηση εφαρμόζεται σε περιπτώσεις όπου οι ομάδες δεν είναι προκαθορισμένες. Στόχος της είναι να εντοπίσει φυσικές ομαδοποιήσεις μέσα στα δεδομένα, με βάση την ομοιότητα των αντικειμένων που εξετάζονται. Με άλλα λόγια, τα στοιχεία που ανήκουν στην ίδια συστάδα πρέπει να παρουσιάζουν υψηλό βαθμό ομοιότητας μεταξύ τους, ενώ τα στοιχεία διαφορετικών συστάδων να διαφέρουν όσο το δυνατόν περισσότερο. Για τον λόγο αυτό, η συσταδοποίηση χρησιμοποιείται ευρέως στην εξόρυξη δεδομένων, στην αναγνώριση προτύπων και στη μηχανική μάθηση, καθώς επιτρέπει την κατανόηση μεγάλων συνόλων δεδομένων και την αποκάλυψη σχέσεων που δεν είναι άμεσα εμφανείς [43], [44], [45].

Η σημασία της συσταδοποίησης έχει αναδειχθεί έντονα στη βιβλιογραφία, καθώς αποτελεί βασικό εργαλείο για την ανακάλυψη δομών σε δεδομένα χωρίς προκαθορισμένη ταξινόμηση. Ο Jain τονίζει ότι, παρά την ανάπτυξη πολλών αλγορίθμων συσταδοποίησης, οι βασικές προκλήσεις παραμένουν η επιλογή κατάλληλου μέτρου απόστασης, η ερμηνεία των αποτελεσμάτων και η αξιολόγηση της ποιότητας των συστάδων [43]. Παράλληλα, οι Kaufman και Rousseeuw παρουσιάζουν τη συσταδοποίηση ως μια διαδικασία αναζήτησης ομάδων μέσα σε δεδομένα, δίνοντας έμφαση στη σημασία της απόστασης ή ανομοιότητας μεταξύ των παρατηρήσεων [44].

Επιπλέον, η βιβλιογραφία διακρίνει τη συσταδοποίηση σε διαφορετικές κατηγορίες μεθόδων, όπως διαμεριστικές, ιεραρχικές, πυκνοτικές και μοντελοκεντρικές προσεγγίσεις. Οι διαμεριστικές μέθοδοι, όπως ο K-Means, προσπαθούν να χωρίσουν τα δεδομένα σε έναν προκαθορισμένο αριθμό ομάδων, ενώ οι ιεραρχικές μέθοδοι δημιουργούν μια δενδρική δομή σχέσεων μεταξύ των παρατηρήσεων. Η επιλογή του κατάλληλου αλγορίθμου εξαρτάται από τη μορφή των δεδομένων, το μέγεθος του συνόλου δεδομένων, την ύπαρξη θορύβου και τον βαθμό ερμηνευσιμότητας που απαιτείται από την ανάλυση [43], [46].

Στο πεδίο της ποδοσφαιρικής ανάλυσης, η ανάλυση συστάδων μπορεί να αξιοποιηθεί για την ομαδοποίηση ομάδων, παικτών ή αγωνιστικών περιόδων, με βάση επιλεγμένα στατιστικά χαρακτηριστικά. Για παράδειγμα, δύο ποδοσφαιρικές ομάδες μπορούν να θεωρηθούν παρόμοιες όταν εμφανίζουν κοντινές τιμές σε δείκτες όπως οι νίκες, τα γκολ, οι τελικές προσπάθειες, η κατοχή της μπάλας ή άλλες μετρικές απόδοσης. Αντίθετα, ομάδες με διαφορετική αγωνιστική συμπεριφορά είναι πιθανό να τοποθετηθούν σε διαφορετικές συστάδες. Έτσι, η συσταδοποίηση δεν περιορίζεται στην απλή παρουσίαση στατιστικών τιμών, αλλά προσφέρει έναν πιο οργανωμένο τρόπο ανάλυσης και ερμηνείας των δεδομένων.

Στην εφαρμογή TransferMind, η ανάλυση συστάδων χρησιμοποιείται ως εργαλείο διερευνητικής ανάλυσης ποδοσφαιρικών δεδομένων. Ο χρήστης έχει τη δυνατότητα να επιλέξει ομάδες, αγωνιστικές περιόδους και στατιστικές μεταβλητές, ώστε η εφαρμογή να δημιουργήσει ένα σύνολο δεδομένων κατάλληλο για επεξεργασία από τους αλγορίθμους συσταδοποίησης. Πριν από την εφαρμογή των αλγορίθμων, τα επιλεγμένα χαρακτηριστικά μετατρέπονται σε αριθμητική μορφή και κανονικοποιούνται. Η κανονικοποίηση αποτελεί απαραίτητο στάδιο, επειδή οι μεταβλητές μπορεί να έχουν διαφορετικές μονάδες μέτρησης ή διαφορετικά εύρη τιμών. Αν δεν εφαρμοστεί, μια μεταβλητή με μεγαλύτερες αριθμητικές τιμές μπορεί να επηρεάσει δυσανάλογα το τελικό αποτέλεσμα, οδηγώντας σε λιγότερο αξιόπιστες συστάδες.



Σχήμα 3.1: Εννοιολογική απεικόνιση της ανάλυσης συστάδων, όπου αντικείμενα με υψηλή ομοιότητα ομαδοποιούνται στην ίδια συστάδα. Προσαρμοσμένο από τη θεωρητική περιγραφή της συσταδοποίησης στη βιβλιογραφία [43], [44].

Η εικόνα 3.1 παρουσιάζει τη γενική λογική της συσταδοποίησης με επιστημονικό τρόπο. Εξηγεί τη βασική αρχή ότι οι παρατηρήσεις που βρίσκονται κοντά μεταξύ τους στον χώρο των χαρακτηριστικών σχηματίζουν κοινές ομάδες. Με αυτόν τον τρόπο, η εικόνα βοηθά τον αναγνώστη να κατανοήσει οπτικά τη σχέση ανάμεσα στην έννοια της ομοιότητας και στη δημιουργία συστάδων.

3.2 K-Means Clustering και Elbow Method

3.2.1 K-Means Clustering

Ο K-Means είναι ένας από τους πιο γνωστούς αλγορίθμους συσταδοποίησης και χρησιμοποιείται συχνά σε προβλήματα όπου χρειάζεται να χωριστούν δεδομένα σε ομάδες με παρόμοια χαρακτηριστικά. Ανήκει στις διαμεριστικές μεθόδους, καθώς ο χρήστης ορίζει από πριν τον αριθμό των συστάδων, ο οποίος συμβολίζεται με K. Κάθε συστάδα αντιπροσωπεύεται από ένα κεντροειδές, δηλαδή ένα σημείο που εκφράζει το μέσο προφίλ των παρατηρήσεων που ανήκουν στη συγκεκριμένη ομάδα.

Η βασική ιδέα του K-Means συνδέεται με την εργασία του MacQueen, όπου περιγράφεται η διαδικασία διαχωρισμού πολυδιάστατων παρατηρήσεων σε k ομάδες με βάση τα χαρακτηριστικά τους [47]. Ο αλγόριθμος προσπαθεί να οργανώσει τα δεδομένα έτσι ώστε οι παρατηρήσεις που

ανήκουν στην ίδια συστάδα να είναι όσο γίνεται πιο κοντά μεταξύ τους. Με αυτόν τον τρόπο δημιουργούνται ομάδες που παρουσιάζουν κοινά μοτίβα και μπορούν να ερμηνευτούν ευκολότερα. Παρόμοια, η βιβλιογραφία περιγράφει τον K-Means ως μέθοδο τοπικής βελτιστοποίησης, η οποία επιδιώκει τη μείωση των αποστάσεων των σημείων από τα κέντρα των συστάδων τους [43], [48], [49]. Η απλότητα και η υπολογιστική αποδοτικότητα του αλγορίθμου τον έχουν καταστήσει ιδιαίτερα δημοφιλή, ωστόσο η ποιότητα του αποτελέσματος μπορεί να επηρεαστεί από την αρχικοποίηση των κεντροειδών, την ύπαρξη ακραίων τιμών και την υπόθεση ότι οι συστάδες έχουν σχετικά συμπαγή ή σφαιρική μορφή [43], [49].

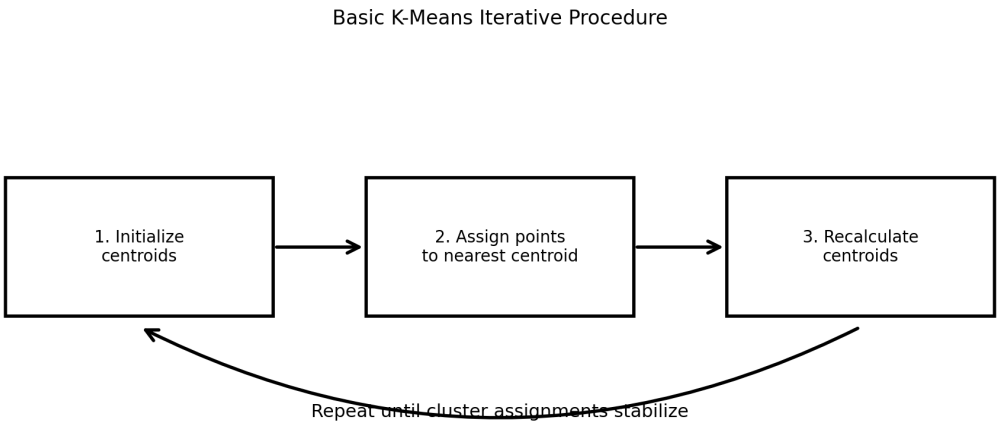
Στην περίπτωση του TransferMind, οι παρατηρήσεις αντιστοιχούν σε ποδοσφαιρικές ομάδες ή σε ομάδες ανά αγωνιστική περίοδο. Τα χαρακτηριστικά τους προέρχονται από στατιστικές μετρικές απόδοσης, όπως γκολ, σουτ, πάσες, clean sheets ή άλλα στοιχεία που είναι διαθέσιμα στην εφαρμογή. Έτσι, ο K-Means χρησιμοποιείται για να εντοπίσει ομάδες που έχουν παρόμοια αγωνιστική συμπεριφορά, όχι με βάση μία μόνο μεταβλητή, αλλά με βάση το συνολικό στατιστικό τους προφίλ.

3.2.2 Βήματα του αλγορίθμου

Η λειτουργία του K-Means βασίζεται σε μια επαναληπτική διαδικασία. Αρχικά επιλέγεται ο αριθμός των συστάδων K και στη συνέχεια ορίζονται τα αρχικά κεντροειδή. Έπειτα, κάθε παρατήρηση ανατίθεται στο κοντινότερο κεντροειδές, συνήθως με βάση την ευκλείδεια απόσταση. Αφού ολοκληρωθεί αυτή η ανάθεση, τα κεντροειδή υπολογίζονται ξανά, παίρνοντας νέα θέση σύμφωνα με τον μέσο όρο των παρατηρήσεων που ανήκουν σε κάθε συστάδα. Η ίδια διαδικασία επαναλαμβάνεται μέχρι να σταθεροποιηθούν οι αναθέσεις ή μέχρι να ολοκληρωθεί ένας συγκεκριμένος αριθμός επαναλήψεων.

Η ποιότητα του αποτελέσματος μπορεί να επηρεαστεί από την αρχική επιλογή των κεντροειδών. Για αυτόν τον λόγο, στη βιβλιογραφία έχει προταθεί η μέθοδος K-Means++, η οποία δίνει μεγαλύτερη προσοχή στην αρχικοποίηση των κέντρων και στοχεύει στη βελτίωση της τελικής συσταδοποίησης [50]. Η μέθοδος αυτή προτάθηκε ώστε να μειώσει την πιθανότητα κακής αρχικοποίησης, η οποία μπορεί να οδηγήσει τον K-Means σε λιγότερο κατάλληλη τοπική λύση.

Η σημασία της αρχικοποίησης είναι κρίσιμη, επειδή ο K-Means δεν εγγυάται πάντα την εύρεση της καθολικά βέλτιστης λύσης. Αντίθετα, συγκλίνει συνήθως σε ένα τοπικό ελάχιστο της αντικειμενικής συνάρτησης. Για τον λόγο αυτό, σε πρακτικές εφαρμογές χρησιμοποιούνται συχνά πολλαπλές αρχικοποιήσεις ή πιο προσεκτικές στρατηγικές επιλογής αρχικών κέντρων, όπως η K-Means++ [49], [50]. Η προσέγγιση αυτή είναι ιδιαίτερα χρήσιμη σε πολυδιάστατα δεδομένα, όπου μικρές αλλαγές στην αρχική θέση των κεντροειδών μπορούν να οδηγήσουν σε διαφορετικές τελικές συστάδες.



Σχήμα 3.2: Σχηματική παρουσίαση των βασικών βημάτων του K-Means: αρχικοποίηση κεντροειδών, ανάθεση σημείων στο κοντινότερο κέντρο και επαναϋπολογισμός των κεντροειδών. Προσαρμοσμένο από τη θεωρητική περιγραφή των MacQueen και Arthur–Vassilvitskii [47], [50].

Στο TransferMind, πριν εφαρμοστεί ο αλγόριθμος, τα δεδομένα περνούν από στάδιο προεπεξεργασίας. Η εφαρμογή δημιουργεί έναν αριθμητικό πίνακα χαρακτηριστικών, όπου κάθε γραμμή αντιστοιχεί σε μία ομάδα ή ομάδα-σεζόν και κάθε στήλη αντιστοιχεί σε ένα επιλεγμένο στατιστικό. Οι τιμές κανονικοποιούνται με min-max normalization, ώστε όλες οι μεταβλητές να βρίσκονται στην ίδια κλίμακα, από 0 έως 1. Αυτό είναι απαραίτητο, επειδή διαφορετικά στατιστικά μπορεί να έχουν διαφορετικά εύρη τιμών και να επηρεάζουν άνισα τον υπολογισμό των αποστάσεων.

3.2.3 Αντικειμενική συνάρτηση

Η αντικειμενική συνάρτηση του K-Means βασίζεται στη μείωση της απόστασης των παρατηρήσεων από τα κεντροειδή των συστάδων τους. Η ποσότητα αυτή είναι γνωστή ως inertia ή WCSS, δηλαδή Within-Cluster Sum of Squares. Όσο μικρότερη είναι η τιμή της, τόσο πιο κοντά βρίσκονται τα σημεία στα κέντρα των συστάδων τους και τόσο πιο συμπαγείς θεωρούνται οι ομάδες.

Η λογική αυτή συνδέεται με την ελαχιστοποίηση των τετραγωνικών αποστάσεων μέσα σε κάθε συστάδα. Οι Arthur και Vassilvitskii περιγράφουν τον K-Means ως μια μέθοδο που επιδιώκει να μειώσει τη μέση τετραγωνική απόσταση ανάμεσα στα σημεία και στα κέντρα των συστάδων τους [50]. Παρ' όλα αυτά, η συνεχής μείωση της inertia δεν σημαίνει πάντα ότι το αποτέλεσμα γίνεται πιο χρήσιμο. Όσο αυξάνεται το K, η inertia συνήθως μειώνεται, όμως μετά από ένα σημείο η προσθήκη νέων συστάδων μπορεί να μη δίνει ουσιαστικά καλύτερη ερμηνεία.

Η αντικειμενική συνάρτηση μπορεί να εκφραστεί ως:

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$

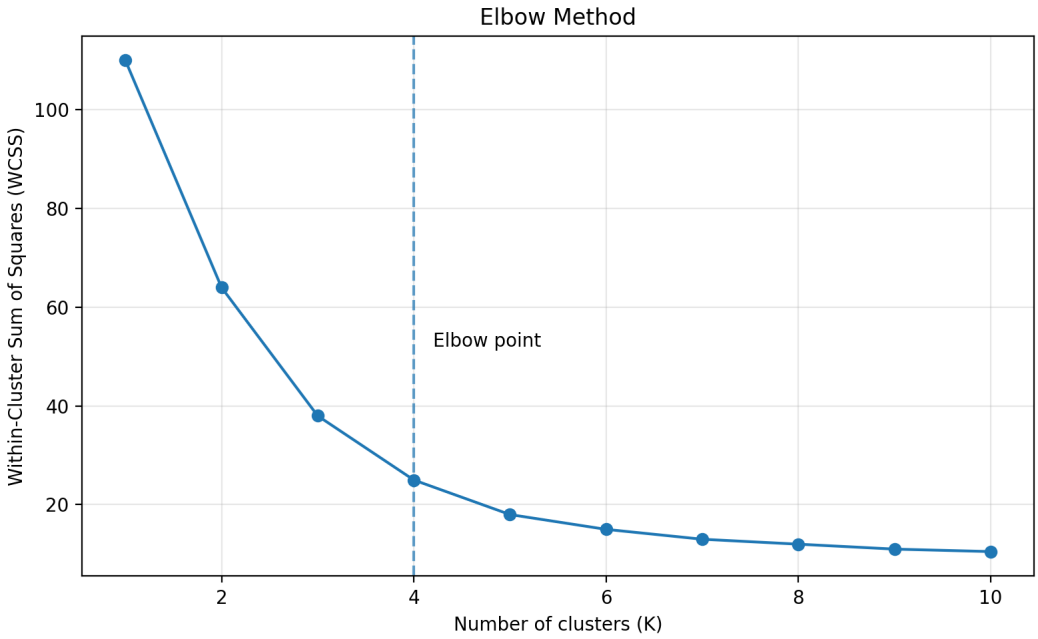
όπου C_i είναι η συστάδα i, x είναι ένα σημείο δεδομένων και μ_i είναι το κεντροειδές της συστάδας. Στο TransferMind, η τιμή της inertia επιστρέφεται από την Python υλοποίηση του K-Means και χρησιμοποιείται τόσο στην τελική συσταδοποίηση όσο και στη μέθοδο Elbow. Έτσι, η θεωρητική λειτουργία του αλγορίθμου συνδέεται άμεσα με την πρακτική λειτουργία της εφαρμογής.

3.2.4 Elbow Method

Ένα βασικό ζήτημα του K-Means είναι ότι ο αριθμός των συστάδων πρέπει να επιλεγεί πριν από την εκτέλεση του αλγορίθμου. Στην πράξη, όμως, ο χρήστης δεν γνωρίζει πάντα ποια τιμή του K είναι κατάλληλη. Για τον λόγο αυτό χρησιμοποιείται η μέθοδος Elbow, η οποία εκτελεί τον K-Means για διαφορετικές τιμές του K και υπολογίζει για κάθε περίπτωση την αντίστοιχη inertia ή WCSS.

Η μέθοδος Elbow δεν αποτελεί αυστηρό μαθηματικό κριτήριο, αλλά περισσότερο μια εμπειρική και οπτική τεχνική αξιολόγησης. Η βασική της χρησιμότητα είναι ότι βοηθά τον αναλυτή να εντοπίσει ένα σημείο ισορροπίας ανάμεσα στην απλότητα του μοντέλου και στη μείωση της εσωτερικής ανομοιογένειας των συστάδων. Παρ' όλα αυτά, σε ορισμένα σύνολα δεδομένων ο «αγκώνας» μπορεί να μην είναι ξεκάθαρος, οπότε είναι χρήσιμο να συνδυάζεται με επιπλέον δείκτες αξιολόγησης, όπως το silhouette coefficient ή άλλες μετρικές εγκυρότητας συστάδων [43], [51].

Σύμφωνα με την τεκμηρίωση του scikit-learn, ο K-Means χωρίζει τα δείγματα σε ομάδες προσπαθώντας να ελαχιστοποιήσει την inertia, δηλαδή το within-cluster sum-of-squares, ενώ ο αριθμός των συστάδων πρέπει να έχει οριστεί εκ των προτέρων [45]. Η μέθοδος Elbow αξιοποιεί ακριβώς αυτή τη συμπεριφορά. Παρατηρεί πώς μειώνεται η inertia όσο αυξάνεται το K και αναζητά το σημείο όπου η μείωση αρχίζει να γίνεται λιγότερο έντονη. Το σημείο αυτό μοιάζει με «αγκώνα» στο διάγραμμα και μπορεί να χρησιμοποιηθεί ως ένδειξη για την επιλογή του αριθμού συστάδων.



Σχήμα 3.3: Ενδεικτικό επιστημονικό διάγραμμα της μεθόδου Elbow, όπου η μείωση του WCSS γίνεται σταδιακά μικρότερη όσο αυξάνεται ο αριθμός των συστάδων. Προσαρμοσμένο από τη βιβλιογραφική περιγραφή του K-Means και της inertia [43], [45].

Η εικόνα 3.3 δείχνει γενικά τη λογική της μεθόδου Elbow. Η καμπύλη παρουσιάζει τη σταδιακή μείωση του WCSS όσο αυξάνεται ο αριθμός των συστάδων, ενώ το σημείο όπου η μείωση γίνεται λιγότερο έντονη μπορεί να χρησιμοποιηθεί ως ένδειξη για την επιλογή μιας κατάλληλης τιμής του K.

3.2.5 Υλοποίηση στο TransferMind

Στην εφαρμογή TransferMind, ο K-Means ενσωματώνεται σε μια ολοκληρωμένη ροή ανάλυσης. Το backend προετοιμάζει τα δεδομένα, δημιουργεί τον κανονικοποιημένο πίνακα χαρακτηριστικών και καλεί Python script για την εκτέλεση του αλγορίθμου. Η Python πλευρά χρησιμοποιεί το scikit-learn, το οποίο παρέχει έτοιμη υλοποίηση του K-Means και επιστρέφει βασικές πληροφορίες, όπως τις αναθέσεις των ομάδων σε συστάδες, τα κεντροειδή, την inertia και τον αριθμό των επαναλήψεων [45].

Μετά την εκτέλεση του αλγορίθμου, η εφαρμογή μπορεί να παρουσιάσει το μέσο προφίλ κάθε συστάδας. Οι τιμές είναι κανονικοποιημένες από 0 έως 1, ώστε τα διαφορετικά στατιστικά να μπορούν να συγκριθούν στην ίδια κλίμακα. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει ποιες συστάδες έχουν υψηλότερες ή χαμηλότερες τιμές σε συγκεκριμένα χαρακτηριστικά και να κατανοήσει καλύτερα τι αντιπροσωπεύει κάθε cluster. Η χρήση τέτοιων οπτικοποιήσεων δεν είναι απλώς αισθητική επιλογή, αλλά βοηθά στη μετάβαση από το αριθμητικό αποτέλεσμα του αλγορίθμου στην ερμηνεία των αγωνιστικών προφίλ.

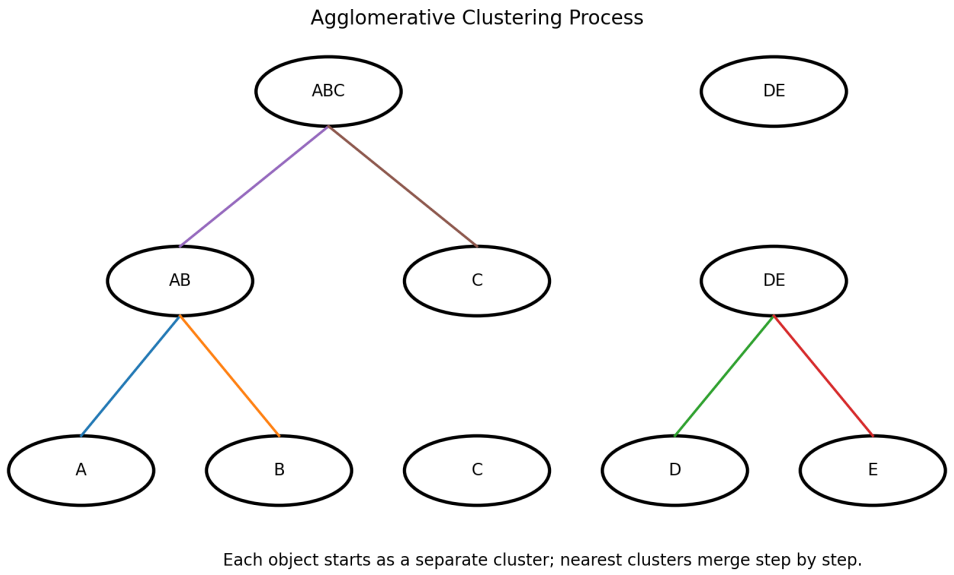
3.3 Agglomerative Clustering

Η agglomerative clustering, ή αλλιώς συσσωρευτική ιεραρχική συσταδοποίηση, αποτελεί μία από τις βασικές μεθόδους της ιεραρχικής ανάλυσης συστάδων. Σε αντίθεση με τον K-Means, όπου ο αριθμός των συστάδων ορίζεται από την αρχή και ο αλγόριθμος προσπαθεί να χωρίσει απευθείας τα δεδομένα σε ομάδες, η agglomerative clustering ακολουθεί μια πιο σταδιακή διαδικασία. Στην αρχή, κάθε παρατήρηση θεωρείται ως ξεχωριστή συστάδα. Στη συνέχεια, σε κάθε βήμα, συγχωνεύονται οι δύο πιο κοντινές συστάδες, μέχρι όλες οι παρατηρήσεις να ενταχθούν σε μία ενιαία ιεραρχική δομή [52], [53].

Η ιεραρχική συσταδοποίηση είναι ιδιαίτερα χρήσιμη όταν ο αναλυτής επιθυμεί να εξετάσει όχι μόνο την τελική ομαδοποίηση, αλλά και τα ενδιάμεσα στάδια με τα οποία σχηματίζονται οι συστάδες. Σε αντίθεση με τον K-Means, δεν απαιτεί απαραίτητα την αρχική επιλογή συγκεκριμένου αριθμού συστάδων, καθώς η ιεραρχία μπορεί να κοπεί σε διαφορετικά επίπεδα ανάλογα με την επιθυμητή ανάλυση. Ωστόσο, έχει μεγαλύτερο υπολογιστικό κόστος σε μεγάλα σύνολα δεδομένων και επηρεάζεται σημαντικά από την επιλογή του μέτρου απόστασης και της μεθόδου linkage [27], [46], [52].

Το βασικό χαρακτηριστικό αυτής της μεθόδου είναι ότι δεν παράγει μόνο μια τελική ομαδοποίηση, αλλά παρουσιάζει ολόκληρη τη διαδικασία σχηματισμού των συστάδων. Αυτό είναι ιδιαίτερα χρήσιμο όταν ο στόχος δεν είναι μόνο να βρεθεί σε ποια ομάδα ανήκει κάθε στοιχείο, αλλά και να γίνει κατανοητή η σχέση μεταξύ των δεδομένων. Στο TransferMind, η agglomerative clustering χρησιμοποιείται για την ανάλυση ποδοσφαιρικών ομάδων με βάση επιλεγμένα στατιστικά χαρακτηριστικά. Με αυτόν τον τρόπο, ο χρήστης μπορεί να παρατηρήσει ποιες ομάδες παρουσιάζουν παρόμοιο αγωνιστικό προφίλ και ποιες διαφέρουν περισσότερο.

Κεντρικό ρόλο στη μέθοδο έχει η έννοια του linkage. Το linkage καθορίζει τον τρόπο με τον οποίο υπολογίζεται η απόσταση ανάμεσα σε δύο συστάδες. Επειδή κάθε συστάδα μπορεί να αποτελείται από περισσότερες από μία παρατηρήσεις, η απόσταση μεταξύ δύο συστάδων δεν είναι πάντα μοναδικά ορισμένη. Για τον λόγο αυτό, υπάρχουν διαφορετικές μέθοδοι linkage, οι οποίες επηρεάζουν τον τρόπο με τον οποίο σχηματίζεται η ιεραρχία. Στην εφαρμογή TransferMind υποστηρίζονται τέσσερις βασικές επιλογές linkage: Ward, Complete, Average και Single [45], [53].



Σχήμα 3.4: Σχηματική απεικόνιση της agglomerative clustering: κάθε παρατήρηση ξεκινά ως ξεχωριστή συστάδα και οι πιο κοντινές συστάδες συγχωνεύονται σταδιακά. Προσαρμοσμένο από τη βιβλιογραφία για την ιεραρχική συσταδοποίηση [52], [53].

3.3.1 Ward

Η μέθοδος Ward βασίζεται στην προσπάθεια ελαχιστοποίησης της αύξησης της διακύμανσης μέσα στις συστάδες κατά τη διαδικασία συγχώνευσης. Σε κάθε βήμα, ο αλγόριθμος επιλέγει να ενώσει τις δύο συστάδες που προκαλούν τη μικρότερη δυνατή αύξηση της εσωτερικής ανομοιογένειας. Με αυτόν τον τρόπο, η Ward linkage τείνει να δημιουργεί πιο συμπαγείς και ισορροπημένες συστάδες [27], [54]. Η μέθοδος αυτή θεωρείται ιδιαίτερα σημαντική στην ιεραρχική συσταδοποίηση, επειδή συνδέει τη διαδικασία συγχώνευσης με ένα σαφές κριτήριο βελτιστοποίησης, δηλαδή τη μείωση του σφάλματος ή της εσωτερικής διασποράς των ομάδων [54].

Στο πλαίσιο της ποδοσφαιρικής ανάλυσης, η Ward linkage είναι χρήσιμη όταν ο στόχος είναι να εντοπιστούν ομάδες με παρόμοιο συνολικό αγωνιστικό προφίλ. Επειδή τα στατιστικά χαρακτηριστικά στο TransferMind κανονικοποιούνται πριν από την εφαρμογή του αλγορίθμου, η συγκεκριμένη μέθοδος αποτελεί κατάλληλη επιλογή για την παρουσίαση του βασικού παραδείγματος, καθώς δίνει πιο καθαρές και ερμηνεύσιμες συστάδες.

3.3.2 Complete

Η Complete linkage υπολογίζει την απόσταση ανάμεσα σε δύο συστάδες με βάση τη μεγαλύτερη απόσταση μεταξύ των παρατηρήσεων που ανήκουν σε αυτές. Αυτό σημαίνει ότι δύο συστάδες συγχωνεύονται μόνο όταν ακόμη και τα πιο απομακρυσμένα σημεία τους δεν έχουν πολύ

μεγάλη απόσταση μεταξύ τους. Η μέθοδος αυτή θεωρείται πιο αυστηρή, επειδή αποφεύγει να ενώσει συστάδες που περιέχουν πολύ διαφορετικές παρατηρήσεις [45], [53].

Στο TransferMind, η Complete linkage μπορεί να χρησιμοποιηθεί όταν ο χρήστης θέλει να εντοπίσει ομάδες που παρουσιάζουν σταθερή ομοιότητα σε ολόκληρο το αγωνιστικό τους προφίλ. Έτσι, δεν αρκεί μόνο μια μεμονωμένη ομοιότητα ανάμεσα σε δύο ομάδες, αλλά εξετάζεται αν οι συστάδες παραμένουν κοντινές ακόμη και στα πιο διαφορετικά σημεία τους.

3.3.3 Average

Η Average linkage υπολογίζει την απόσταση ανάμεσα σε δύο συστάδες με βάση τον μέσο όρο των αποστάσεων όλων των παρατηρήσεων που ανήκουν σε αυτές. Η μέθοδος αυτή βρίσκεται ανάμεσα στη λογική της Complete και της Single linkage, καθώς δεν βασίζεται ούτε στη μεγαλύτερη ούτε στη μικρότερη απόσταση. Αντίθετα, λαμβάνει υπόψη τη συνολική σχέση μεταξύ των δύο συστάδων [53].

Η Average linkage μπορεί να θεωρηθεί μια πιο ισορροπημένη προσέγγιση. Στην εφαρμογή TransferMind, μπορεί να βοηθήσει όταν ο χρήστης θέλει να εξετάσει τη γενική ομοιότητα ανάμεσα σε ομάδες, χωρίς το αποτέλεσμα να επηρεάζεται υπερβολικά από ένα πολύ κοντινό ή ένα πολύ μακρινό ζεύγος παρατηρήσεων. Για αυτόν τον λόγο, αποτελεί χρήσιμη επιλογή σε περιπτώσεις όπου ζητείται μια πιο συνολική εικόνα των δεδομένων.

3.3.4 Single

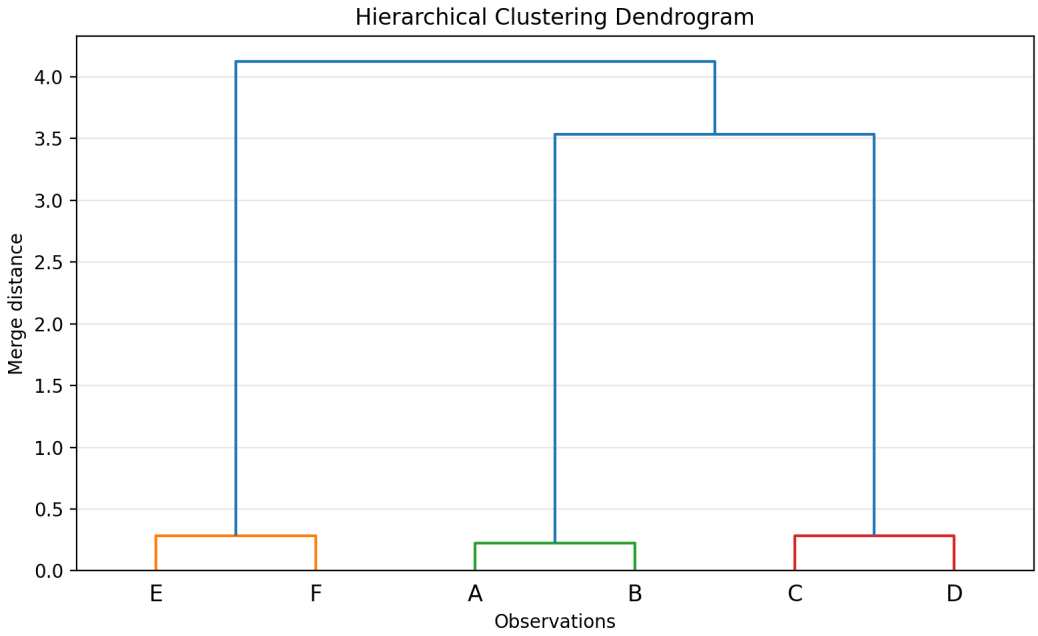
Η Single linkage βασίζεται στη μικρότερη απόσταση ανάμεσα σε δύο παρατηρήσεις που ανήκουν σε διαφορετικές συστάδες. Με άλλα λόγια, δύο συστάδες μπορούν να συγχωνευθούν όταν υπάρχει έστω και ένα ζεύγος παρατηρήσεων που βρίσκεται πολύ κοντά. Η μέθοδος αυτή μπορεί να εντοπίσει πιο επιμήκεις ή μη σφαιρικές δομές, όμως είναι περισσότερο ευαίσθητη σε θόρυβο και ακραίες τιμές [45], [53].

Στο TransferMind, η Single linkage μπορεί να χρησιμοποιηθεί ως εναλλακτική επιλογή διερεύνησης. Ωστόσο, επειδή μπορεί να δημιουργήσει συστάδες που ενώνονται σταδιακά μέσα από μεμονωμένες κοντινές ομάδες, χρειάζεται προσεκτική ερμηνεία. Για τον λόγο αυτό, είναι χρήσιμη κυρίως για σύγκριση με τις υπόλοιπες μεθόδους linkage.

3.3.5 Dendrogram

Το dendrogram αποτελεί τη βασική οπτική αναπαράσταση της ιεραρχικής συσταδοποίησης. Πρόκειται για ένα διάγραμμα που δείχνει τη σειρά με την οποία συγχωνεύονται οι παρατηρήσεις ή οι συστάδες. Στον οριζόντιο άξονα εμφανίζονται οι παρατηρήσεις που αναλύονται, ενώ στον κάθετο άξονα εμφανίζεται η απόσταση συγχώνευσης. Όσο χαμηλότερα ενώνονται δύο παρατηρήσεις ή συστάδες, τόσο μεγαλύτερη ομοιότητα παρουσιάζουν. Αντίθετα, όταν η

συγχώνευση γίνεται σε μεγαλύτερο ύψος, αυτό δείχνει μεγαλύτερη απόσταση μεταξύ των αντίστοιχων συστάδων [52], [55].



Σχήμα 3.5: Επιστημονική απεικόνιση dendrogram για την ιεραρχική συσταδοποίηση. Το ύψος κάθε συγχώνευσης εκφράζει την απόσταση ή ανομοιότητα μεταξύ των συστάδων. Προσαρμοσμένο από τη βιβλιογραφία της ιεραρχικής συσταδοποίησης [52], [55].

Η εικόνα 3.5 έχει θεωρητικό χαρακτήρα και εξηγεί τον τρόπο με τον οποίο διαβάζεται ένα dendrogram. Με αυτόν τον τρόπο, ο αναγνώστης μπορεί να κατανοήσει τη σύνδεση ανάμεσα στο ύψος της συγχώνευσης, την απόσταση μεταξύ συστάδων και την τελική επιλογή του επιπέδου στο οποίο μπορεί να κοπεί η ιεραρχία.

3.4 Association Rules Mining και Apriori

Σε αντίθεση με τους αλγορίθμους συσταδοποίησης, οι οποίοι στοχεύουν στην ομαδοποίηση παρόμοιων παρατηρήσεων, η εξόρυξη κανόνων συσχέτισης επικεντρώνεται στην ανακάλυψη σχέσεων μεταξύ χαρακτηριστικών ή γεγονότων που εμφανίζονται συχνά μαζί. Για τον λόγο αυτό, οι κανόνες συσχέτισης είναι ιδιαίτερα χρήσιμοι σε περιπτώσεις όπου ο στόχος δεν είναι η δημιουργία ομάδων, αλλά η κατανόηση επαναλαμβανόμενων συνδυασμών μέσα στα δεδομένα [29], [30], [56].

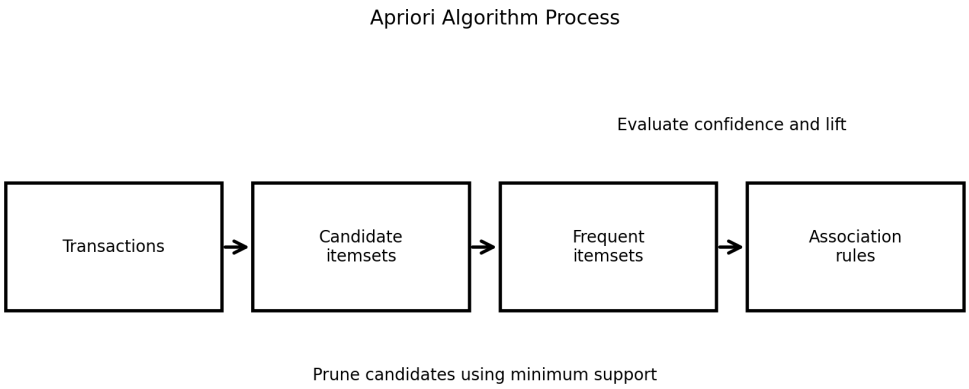
Η εξόρυξη κανόνων συσχέτισης, γνωστή ως association rules mining, αποτελεί τεχνική εξόρυξης δεδομένων που στοχεύει στον εντοπισμό σχέσεων μεταξύ στοιχείων που εμφανίζονται συχνά μαζί μέσα σε ένα σύνολο δεδομένων. Η μέθοδος έγινε ευρέως γνωστή μέσα από την εργασία των Agrawal, Imielinski και Swami, οι οποίοι παρουσίασαν το πρόβλημα της ανακάλυψης

κανόνων συσχέτισης σε μεγάλες βάσεις δεδομένων [30]. Στη συνέχεια, οι Agrawal και Srikant πρότειναν αποδοτικούς αλγορίθμους για την εξόρυξη τέτοιων κανόνων, μεταξύ των οποίων και ο Apriori [29]. Η σχετική βιβλιογραφία τονίζει ότι οι κανόνες συσχέτισης παραμένουν σημαντικοί επειδή προσφέρουν ερμηνεύσιμα αποτελέσματα, τα οποία μπορούν να εκφραστούν με απλή μορφή τύπου "αν ισχύει το X, τότε είναι πιθανό να ισχύει και το Y" [56], [57].

Ο Apriori βασίζεται στην ιδέα ότι, αν ένα σύνολο στοιχείων είναι συχνό, τότε όλα τα υποσύνολά του πρέπει επίσης να είναι συχνά. Η ιδιότητα αυτή επιτρέπει στον αλγόριθμο να μειώσει τον αριθμό των υποψήφιων συνδυασμών που πρέπει να εξεταστούν. Αντί να ελέγχει όλους τους δυνατούς συνδυασμούς, ο Apriori δημιουργεί σταδιακά μεγαλύτερα itemsets ξεκινώντας από τα συχνά μεμονωμένα στοιχεία και απορρίπτοντας όσα δεν ικανοποιούν το ελάχιστο όριο υποστήριξης.

Βασικές μετρικές της μεθόδου είναι το support, το confidence και το lift. Το support δείχνει πόσο συχνά εμφανίζεται ένας συνδυασμός στοιχείων στο σύνολο δεδομένων. Το confidence εκφράζει πόσο συχνά εμφανίζεται το συμπέρασμα ενός κανόνα όταν εμφανίζεται η προϋπόθεσή του. Το lift χρησιμοποιείται για να αξιολογήσει αν η σχέση μεταξύ δύο στοιχείων είναι ισχυρότερη από αυτή που θα αναμενόταν τυχαία. Οι μετρικές αυτές επιτρέπουν στον χρήστη να ξεχωρίσει τους πιο ενδιαφέροντες και χρήσιμους κανόνες από έναν μεγάλο αριθμό πιθανών συσχετίσεων [58], [59].

Στο TransferMind, ο Apriori μπορεί να αξιοποιηθεί για την αναγνώριση συχνών συνδυασμών σε ποδοσφαιρικά δεδομένα. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για να εξεταστεί αν συγκεκριμένα χαρακτηριστικά εμφανίζονται συχνά μαζί, όπως υψηλή επιθετική απόδοση, αυξημένα σουτ, υψηλή κατοχή ή συγκεκριμένα μοτίβα αγωνιστικής συμπεριφοράς. Με αυτόν τον τρόπο, η εφαρμογή δεν περιορίζεται μόνο στη συσταδοποίηση ομάδων, αλλά μπορεί να αναδείξει και σχέσεις μεταξύ στατιστικών χαρακτηριστικών.



Σχήμα 3.6: Σχηματική απεικόνιση της διαδικασίας Apriori: δημιουργία συχνών itemsets, εφαρμογή ελάχιστου support και παραγωγή κανόνων συσχέτισης. Προσαρμοσμένο από τις εργασίες των Agrawal, Imielinski και Swami, καθώς και των Agrawal και Srikant [29], [30].

Η εικόνα 3.6 παρουσιάζει τη βασική λειτουργία του Apriori σε θεωρητικό επίπεδο. Αρχικά, τα δεδομένα εξετάζονται ως συναλλαγές ή σύνολα χαρακτηριστικών. Στη συνέχεια δημιουργούνται υποψήφια itemsets, εφαρμόζεται το ελάχιστο όριο υποστήριξης και τελικά παράγονται κανόνες συσχέτισης που μπορούν να αξιολογηθούν με βάση μετρικές όπως support, confidence και lift.

Κεφάλαιο 4

Sofascore API και Δημιουργία Συνόλου Δεδομένων / Βάσης Δεδομένων

4.1 Ρόλος του Sofascore API στην εργασία

Η εφαρμογή TransferMind βασίζεται σε ποδοσφαιρικά δεδομένα, τα οποία πρέπει να είναι οργανωμένα, επαναχρησιμοποιήσιμα και κατάλληλα για στατιστική ανάλυση. Για τον λόγο αυτό χρησιμοποιήθηκε το Sofascore API μέσω της πλατφόρμας RapidAPI, με σκοπό τη συλλογή δεδομένων που αφορούν διοργανώσεις, σεζόν, βαθμολογίες, ομάδες, παίκτες και αγωνιστικά στατιστικά. Το Sofascore λειτουργεί ως η εξωτερική πηγή δεδομένων, ενώ η βάση Supabase/PostgreSQL χρησιμοποιείται ως το μόνιμο και οργανωμένο σύνολο δεδομένων της εφαρμογής.

Η λογική της εφαρμογής δεν βασίζεται σε απευθείας κλήσεις από το frontend προς το Sofascore API. Αντίθετα, όλες οι κλήσεις πραγματοποιούνται στο backend, μέσα από ειδικά ingestion scripts. Με αυτόν τον τρόπο, η συλλογή των δεδομένων γίνεται ελεγχόμενα, τα δεδομένα μετατρέπονται στη μορφή που χρειάζεται η εφαρμογή και στη συνέχεια αποθηκεύονται στη βάση. Αυτή η προσέγγιση βοηθά στη διατήρηση πιο καθαρής αρχιτεκτονικής, επειδή το frontend δεν χρειάζεται να γνωρίζει λεπτομέρειες για τα εξωτερικά endpoints ή για τη μορφή των αρχικών απαντήσεων του API.

Στο backend υπάρχει κεντρικός HTTP client, ο οποίος χρησιμοποιείται από τα επιμέρους ingestion scripts. Ο client αυτός περιλαμβάνει τη βασική διεύθυνση του API, τα απαραίτητα headers για το RapidAPI και κοινές ρυθμίσεις, όπως το timeout των αιτημάτων. Έτσι, οι κλήσεις προς το Sofascore API γίνονται με ενιαίο τρόπο σε όλο το project.

```
export const client = axios.create({
baseURL: API_BASE,
timeout: INGESTION_REQUEST_TIMEOUT_MS,
headers: {
"X-RapidAPI-Key": RAPIDAPI_KEY,
```

```
"X-RapidAPI-Host": RAPIDAPI_HOST,
Accept: "application/json",
},
});
```

Listing 4.1: Κεντρικοποιημένος Axios client για το Sofascore API

Το παραπάνω απόσπασμα δείχνει ότι η πρόσβαση στο API δεν γίνεται διάσπαρτα μέσα στον κώδικα, αλλά μέσα από έναν κοινό client. Αυτό είναι χρήσιμο, γιατί αν χρειαστεί να αλλάξει η βασική διεύθυνση, το host ή κάποια ρύθμιση των αιτημάτων, η αλλαγή μπορεί να γίνει σε ένα μόνο σημείο. Παράλληλα, τα επιμέρους scripts μπορούν να επικεντρωθούν στη λογική συλλογής και αποθήκευσης των δεδομένων, χωρίς να επαναλαμβάνουν τον τρόπο σύνδεσης με το API.

4.2 Διαδικασία συλλογής δεδομένων

Η διαδικασία συλλογής δεδομένων οργανώνεται σε διαδοχικά βήματα. Η σειρά αυτών των βημάτων έχει σημασία, επειδή ορισμένα δεδομένα εξαρτώνται από άλλα. Για παράδειγμα, πριν συλλεχθούν οι ομάδες, πρέπει πρώτα να έχουν συλλεχθεί οι βαθμολογίες, αφού από αυτές εντοπίζονται οι ομάδες που συμμετέχουν σε κάθε διοργάνωση και σεζόν. Αντίστοιχα, πριν συλλεχθούν οι παίκτες, πρέπει να υπάρχουν ήδη οι ομάδες από τις οποίες θα ανακτηθούν τα squads.

Η αρχική διαδικασία ingestion ξεκινά από τις σεζόν και τις διοργανώσεις, συνεχίζει με τις βαθμολογίες και τις ομάδες, και στη συνέχεια περνά στα στατιστικά ομάδων, στους παίκτες και στα στατιστικά παικτών. Η λογική αυτή αποτυπώνεται στο αρχείο `backend/ingestion/initIngestio` όπου ορίζεται η σειρά εκτέλεσης των βασικών βημάτων.

```
const INIT_STEPS = [
{
name: "seasons",
description: "last 3 seasons fetch",
entityType: ENTITY_TYPES.SEASONS,
entityKey: "seasons:init_last_3",
run: () => runFetchSeasons(),
},
{
name: "tournaments",
description: "tournament metadata",
entityType: ENTITY_TYPES.TOURNAMENTS,
entityKey: "tournaments:init_metadata",
run: () => runFetchTournaments(),
},
{
name: "standings",
```

```
description: "all DB seasons",
entityType: ENTITY_TYPES.STANDINGS,
entityKey: "standings:init_all_db_seasons",
run: () => runFetchStandings({ mode: "init" }),
},
];
```

Listing 4.2: Βήματα της αρχικής διαδικασίας ingestion

Το παραπάνω snippet παρουσιάζει την αρχική μορφή της ροής. Αρχικά συλλέγονται οι σεζόν, στη συνέχεια τα μεταδεδομένα των διοργανώσεων και έπειτα οι βαθμολογίες. Στην πλήρη διαδικασία ακολουθούν και άλλα βήματα, όπως η συλλογή ομάδων, λογότυπων, στατιστικών ομάδων, παικτών και στατιστικών παικτών. Με αυτόν τον τρόπο, το σύστημα δημιουργεί σταδιακά ένα σύνολο δεδομένων που μπορεί να χρησιμοποιηθεί τόσο από το backend όσο και από το frontend της εφαρμογής.

Εκτός από την αρχική συλλογή, υπάρχουν και ξεχωριστά refresh jobs. Αυτά χρησιμοποιούνται για την ανανέωση συγκεκριμένων τμημάτων του dataset, όπως οι βαθμολογίες, τα στατιστικά ομάδων, οι παίκτες ή τα στατιστικά παικτών. Η ύπαρξη ξεχωριστών jobs βοηθά ώστε να μην χρειάζεται κάθε φορά να εκτελείται όλη η αρχική διαδικασία από την αρχή. Έτσι, το σύστημα μπορεί να ενημερώνει μόνο τα δεδομένα που είναι πιθανότερο να έχουν αλλάξει.

4.3 Δεδομένα διοργανώσεων και σεζόν

Οι διοργανώσεις και οι σεζόν αποτελούν τη βάση πάνω στην οποία οργανώνονται όλα τα υπόλοιπα δεδομένα. Στην εφαρμογή, οι διοργανώσεις αντιστοιχούν σε ποδοσφαιρικά πρωταθλήματα ή competitions, ενώ οι σεζόν αντιστοιχούν σε συγκεκριμένες χρονικές περιόδους αυτών των διοργανώσεων. Για παράδειγμα, μια διοργάνωση μπορεί να είναι ένα εθνικό πρωτάθλημα, ενώ μια σεζόν μπορεί να είναι η αγωνιστική περίοδος 2024/2025.

Η συλλογή των σεζόν γίνεται με βάση συγκεκριμένα Sofascore tournament ids. Για κάθε διοργάνωση, το σύστημα καλεί το αντίστοιχο endpoint του API και επιλέγει έναν περιορισμένο αριθμό πρόσφατων σεζόν. Παράλληλα, γίνεται προσπάθεια να εντοπιστεί ποια από αυτές είναι η τρέχουσα σεζόν, ώστε η εφαρμογή να μπορεί να ξεχωρίζει τα πιο πρόσφατα δεδομένα.

```
const selectedSeasons = seasons.slice(0, SEASONS_PER_TOURNAMENT);
const currentSeason = getCurrentSeasonFromList(selectedSeasons);
const currentApiId = currentSeason?.id ?? null;
const rowsForInsert = selectedSeasons.map((s) => ({
api_id: s.id,
name: s.name,
year: s.year,
tournament_id: tournamentId,
is_current: currentApiId !== null && s.id === currentApiId,
```

```
}));
const { error } = await supabase
.from("seasons")
.upsert(rowsForInsert, { onConflict: "api_id" });
```

Listing 4.3: Μετατροπή σεζόν Sofascore σε εγγραφές της βάσης

Στο παραπάνω παράδειγμα φαίνεται η μετατροπή των δεδομένων του Sofascore σε εγγραφές του πίνακα `seasons`. Το εξωτερικό id της σεζόν αποθηκεύεται στο πεδίο `api_id`, ενώ το `tournament_id` συνδέει τη σεζόν με τη διοργάνωση στην οποία ανήκει. Το πεδίο `is_current` χρησιμοποιείται για να δηλώσει αν η συγκεκριμένη σεζόν θεωρείται η τρέχουσα για τη διοργάνωση.

Η αποθήκευση γίνεται με χρήση `upsert`. Αυτό σημαίνει ότι αν μια σεζόν υπάρχει ήδη στη βάση με το ίδιο `api_id`, τότε η υπάρχουσα εγγραφή ενημερώνεται. Αν δεν υπάρχει, δημιουργείται νέα εγγραφή. Η επιλογή αυτή είναι ιδιαίτερα χρήσιμη σε διαδικασίες ingestion, επειδή τα ίδια scripts μπορούν να εκτελεστούν πολλές φορές χωρίς να δημιουργούνται διπλότυπες εγγραφές.

Αφού αποθηκευτούν οι σεζόν, ακολουθεί η συλλογή των μεταδεδομένων των διοργανώσεων. Το Sofascore επιστρέφει πληροφορίες μέσα από το αντικείμενο `uniqueTournament`, οι οποίες μετατρέπονται σε γραμμή του πίνακα `tournaments`.

```
const t = data.uniqueTournament;
const row = {
api_id: t.id,
name: t.name,
country: t.category?.country?.name || null,
logo_md5: t.logo?.md5 || null,
logo_id: t.logo?.id || null,
flag_code: t.category?.flag || null,
};
const { error } = await supabase
.from("tournaments")
.upsert([row], { onConflict: "api_id" });
```

Listing 4.4: Αντιστοίχιση διοργάνωσης Sofascore στον πίνακα tournaments

Με αυτόν τον τρόπο, ο πίνακας `tournaments` αποθηκεύει βασικά στοιχεία για κάθε διοργάνωση, όπως το όνομα, τη χώρα και πληροφορίες που σχετίζονται με το λογότυπο ή τη σημαία. Η ύπαρξη ξεχωριστού πίνακα διοργανώσεων κάνει το dataset πιο καθαρό, επειδή οι ίδιες πληροφορίες δεν χρειάζεται να επαναλαμβάνονται σε κάθε σεζόν ή σε κάθε βαθμολογία.

4.4 Δεδομένα ομάδων και παικτών

Οι ομάδες συλλέγονται κυρίως μέσα από τις βαθμολογίες. Η λογική αυτή είναι πρακτική, επειδή οι βαθμολογίες δείχνουν ποιες ομάδες συμμετέχουν σε μια συγκεκριμένη διοργάνωση και σεζόν. Το σύστημα διαβάζει τα `team_id` που εμφανίζονται στον πίνακα `standings`, ελέγχει ποιες ομάδες λείπουν από τον πίνακα `teams` και στη συνέχεια ζητά λεπτομέρειες μόνο για τις ομάδες που δεν έχουν ήδη αποθηκευτεί.

```
const res = await client.get("/teams/detail", {
params: { teamId: String(teamId) },
});
const data = res.data;
const t = data?.team ?? data;
const row = {
api_id: t.id,
name: t.name ?? null,
tournament_id: t.primaryUniqueTournament?.id ?? null,
city: t.venue?.city?.name ?? null,
stadium: t.venue?.name ?? null,
};
const { error } = await supabase
.from("teams")
.upsert([row], { onConflict: "api_id" });
```

Listing 4.5: Συλλογή και upsert στοιχείων ομάδας

Το συγκεκριμένο απόσπασμα δείχνει τη χαρτογράφηση μιας ομάδας από τη μορφή του API στη μορφή της βάσης. Εκτός από το όνομα και το `api_id`, αποθηκεύονται και επιπλέον πληροφορίες, όπως η πόλη και το γήπεδο, όταν αυτές είναι διαθέσιμες. Με αυτόν τον τρόπο, το προφίλ ομάδας στην εφαρμογή μπορεί να εμφανίζει περισσότερα στοιχεία από μια απλή ονομασία.

Η συλλογή των παικτών γίνεται μέσα από τα squads των ομάδων. Το σύστημα διαβάζει τις ομάδες της τρέχουσας σεζόν και για κάθε ομάδα καλεί το endpoint που επιστρέφει το roster. Επειδή το API μπορεί να επιστρέψει παίκτες σε διαφορετικές ομάδες δεδομένων, όπως `players`, `nationalPlayers` ή `foreignPlayers`, χρησιμοποιείται ένα `Set` για την αποφυγή διπλότυπων ids.

```
const res = await client.get("/teams/get-squad", {
params: { teamId: String(team_api_id) },
});
const data = res.data;
const playerIdSet = new Set();
const groups = [
data.players || [],
data.nationalPlayers || [],
data.foreignPlayers || [],
```

```
];
for (const group of groups) {
for (const item of group) {
const p = item?.player || item;
if (p?.id) {
playerIdSet.add(p.id);
}
}
}
const playerIds = [...playerIdSet];
```

Listing 4.6: Αφαίρεση διπλότυπων παικτών από τα squads ομάδων

Η χρήση του `Set` είναι μια απλή αλλά χρήσιμη τεχνική, επειδή επιτρέπει στο σύστημα να κρατήσει κάθε παίκτη μόνο μία φορά, ακόμη και αν εμφανιστεί σε περισσότερες από μία κατηγορίες της απάντησης. Αφού εντοπιστούν τα ids των παικτών, το σύστημα μπορεί να ζητήσει αναλυτικά στοιχεία για κάθε παίκτη και να τα αποθηκεύσει στον πίνακα `players`.

```
const row = {
api_id: p.id,
name: p.name || null,
team_id: p.team?.id ?? null,
nationality: p.country?.name ?? null,
height: p.height ?? null,
date_of_birth: p.dateOfBirth || null,
foot: p.preferredFoot || null,
jersey_num: p.jerseyNumber ?? null,
contract: p.contractUntilTimestamp || null,
market_value: p.proposedMarketValue ?? null,
market_value_currency: p.proposedMarketValueRaw?.currency || null,
};
const { data, error } = await supabase
.from("players")
.upsert([row], { onConflict: "api_id" })
.select("id");
```

Listing 4.7: Μετατροπή παίκτη Sofascore σε εγγραφή players

Ο πίνακας `players` περιλαμβάνει βασικά στοιχεία ταυτότητας και προφίλ, όπως όνομα, εθνικότητα, ύψος, ημερομηνία γέννησης, προτιμώμενο πόδι, αριθμό φανέλας και χρηματιστηριακή αξία, όταν αυτά είναι διαθέσιμα. Η χρήση του `api_id` ως πεδίου σύγκρουσης στο `upsert` επιτρέπει την αποφυγή διπλότυπων εγγραφών για τον ίδιο παίκτη.

Επιπλέον, οι θέσεις των παικτών οργανώνονται σε ξεχωριστούς πίνακες. Ο πίνακας `positions` αποθηκεύει τις διαθέσιμες θέσεις, ενώ ο πίνακας `player_positions` λειτουργεί ως ενδιάμεσος πίνακας σύνδεσης μεταξύ παικτών και θέσεων. Αυτή η επιλογή είναι πιο ευέλικτη από την αποθήκευση μιας απλής συμβολοσειράς μέσα στον πίνακα `players`, επειδή ένας παίκτης

μπορεί να συνδεθεί με περισσότερες από μία θέσεις.

4.5 Στατιστικά ομάδων και παικτών

Εκτός από τα βασικά στοιχεία διοργανώσεων, ομάδων και παικτών, η εφαρμογή συλλέγει και στατιστικά. Τα στατιστικά αυτά είναι απαραίτητα για τις αναλύσεις που παρουσιάζονται στα επόμενα κεφάλαια, όπως η σύγκριση ομάδων, η ανάλυση συστάδων και η παρουσίαση αγωνιστικών χαρακτηριστικών.

Οι βαθμολογίες αποτελούν ένα από τα πρώτα είδη αγωνιστικών δεδομένων που συλλέγονται. Από αυτές προκύπτουν στοιχεία όπως η θέση της ομάδας, οι αγώνες, οι νίκες, οι ισοπαλίες, οι ήττες, τα γκολ υπέρ, τα γκολ κατά, η διαφορά τερμάτων και οι βαθμοί.

```
const totalStandings = standingsList.filter(
(table) =>
table?.type === "total" &&
Array.isArray(table?.rows) &&
table.rows.length > 0,
);
const rowsToUpsert = totalStandings.flatMap((table) => {
return table.rows.map((r) => {
const gf = r.scoresFor ?? r.scores?.for ?? r.goalsFor ?? 0;
const ga = r.scoresAgainst ?? r.scores?.against ?? r.goalsAgainst ?? 0;
return {
  api_id: r.id,
  team_id: r.team?.id,
  tournament_id: tournamentId,
  season_id: seasonId,
  position: r.position ?? null,
  matches: r.matches ?? 0,
  wins: r.wins ?? 0,
  draws: r.draws ?? 0,
  losses: r.losses ?? 0,
  goals_for: gf,
  goals_against: ga,
  goal_diff: gf - ga,
  points: r.points ?? 0,
};

});
});
```

Listing 4.8: Φιλτράρισμα και χαρτογράφηση βαθμολογίας

Το snippet δείχνει ότι το σύστημα κρατά μόνο τους συνολικούς πίνακες βαθμολογίας που περιέ-

χουν πραγματικές γραμμές. Στη συνέχεια, κάθε γραμμή μετατρέπεται σε εγγραφή για τη βάση. Ειδική προσοχή δίνεται στα γκολ υπέρ και κατά, επειδή το API μπορεί να τα επιστρέψει με διαφορετικές ονομασίες πεδίων. Για τον λόγο αυτό χρησιμοποιούνται εναλλακτικές επιλογές με τον τελεστή `??`, ώστε να αυξηθεί η ανθεκτικότητα της μετατροπής.

Τα στατιστικά ομάδων συλλέγονται με βάση τρεις βασικές παραμέτρους: ομάδα, διοργάνωση και σεζόν. Αυτές οι τρεις τιμές καθορίζουν με ακρίβεια σε ποιο αγωνιστικό πλαίσιο ανήκουν τα στατιστικά. Έτσι, μια ομάδα μπορεί να έχει διαφορετική εγγραφή στατιστικών για διαφορετική σεζόν ή διαφορετική διοργάνωση.

```
const res = await client.get("/teams/get-statistics", {
params: {
teamId: team_id,
tournamentId: tournament_id,
seasonId: season_id,
},
});
let s = res.data?.statistics;
if (!s && res.data && res.data.goalsScored !== undefined) {
s = res.data;
}
const hasStats = !!(s && Object.keys(s).length > 0);
if (!hasStats) {
const skeletonRow = {
team_id,
tournament_id,
season_id,
matches: 0,
has_stats: false,
};
}
```

Listing 4.9: Κλήση στατιστικών ομάδας και skeleton row

Η ύπαρξη του πεδίου `has_stats` βοηθά στη διάκριση ανάμεσα σε δύο διαφορετικές περιπτώσεις. Η πρώτη είναι όταν μια ομάδα έχει πραγματικά μηδενικές τιμές σε κάποια στατιστικά. Η δεύτερη είναι όταν το API δεν επέστρεψε διαθέσιμα στατιστικά. Στη δεύτερη περίπτωση, το σύστημα μπορεί να αποθηκεύσει μια βασική εγγραφή με `has_stats: false`, ώστε να είναι γνωστό ότι έγινε προσπάθεια συλλογής, αλλά δεν υπήρχαν διαθέσιμα δεδομένα.

Η αποθήκευση των στατιστικών ομάδας γίνεται με σύνθετο κλειδί. Το κλειδί αυτό αποτελείται από το `team_id`, το `season_id` και το `tournament_id`. Με αυτόν τον τρόπο διασφαλίζεται ότι υπάρχει μόνο μία εγγραφή στατιστικών για κάθε συνδυασμό ομάδας, σεζόν και διοργάνωσης.

```
const { error } = await supabase.from("team_stats").upsert(statsRow, {
onConflict: "team_id, season_id, tournament_id",
```

```
});
if (error) {
console.error(DB Error (Team ${team_id}):, error.message);
throw error;
} else {
console.log(Saved: Team ${team_id});
}
```

Listing 4.10: Upsert στατιστικών ομάδας με σύνθετο κλειδί

Αντίστοιχη λογική χρησιμοποιείται και για τα στατιστικά παικτών. Η κλήση προς το API γίνεται με βάση το id του παίκτη, της διοργάνωσης και της σεζόν. Αν δεν επιστραφούν διαθέσιμα στατιστικά, αποθηκεύεται και εδώ μια κενή εγγραφή με `has_stats: false`.

```
const res = await client.get("/players/get-statistics", {
params: {
playerId: player.api_id,
tournamentId: tournament_api_id,
seasonId: season_api_id,
},
});
const s = res.data?.statistics;
const hasStats = !!(s && Object.keys(s).length > 0);
if (!hasStats) {
const skeletonRow = {
player_id: player.api_id,
team_id,
tournament_id,
season_id,
has_stats: false,
};
const { error } = await supabase
.from("player_stats")
.upsert(skeletonRow, {
onConflict: "player_id, team_id, season_id, tournament_id",
});
}
```

Listing 4.11: Κλήση στατιστικών παίκτη και αποθήκευση κενών εγγραφών

Η χρήση ξεχωριστών πινάκων για `team_stats` και `player_stats` κάνει το dataset πιο οργανωμένο. Τα βασικά στοιχεία ταυτότητας των ομάδων και των παικτών παραμένουν στους πίνακες `teams` και `players`, ενώ τα αγωνιστικά στατιστικά αποθηκεύονται σε ξεχωριστούς πίνακες που συνδέονται με συγκεκριμένη σεζόν και διοργάνωση.

4.6 Αποθήκευση στη βάση δεδομένων

Η αποθήκευση των δεδομένων γίνεται σε βάση Supabase/PostgreSQL. Η PostgreSQL παρέχει το σχεσιακό μοντέλο, δηλαδή πίνακες, πεδία, περιορισμούς μοναδικότητας και συσχετίσεις. Το Supabase λειτουργεί ως πλατφόρμα που διευκολύνει τη σύνδεση του backend με τη βάση και παρέχει έτοιμο client για την εκτέλεση ερωτημάτων.

Οι βασικοί πίνακες που χρησιμοποιούνται στο συγκεκριμένο κεφάλαιο είναι οι εξής:

- `tournaments`: αποθηκεύει στοιχεία διοργανώσεων.
- `seasons`: αποθηκεύει τις σεζόν κάθε διοργάνωσης.
- `standings`: αποθηκεύει τις βαθμολογίες ομάδων.
- `teams`: αποθηκεύει τα βασικά στοιχεία των ομάδων.
- `team_stats`: αποθηκεύει στατιστικά ομάδων ανά σεζόν και διοργάνωση.
- `players`: αποθηκεύει τα βασικά στοιχεία των παικτών.
- `player_stats`: αποθηκεύει στατιστικά παικτών ανά σεζόν, ομάδα και διοργάνωση.
- `positions`: αποθηκεύει τις θέσεις παικτών.
- `player_positions`: συνδέει παίκτες με θέσεις.

Πέρα από τους βασικούς πίνακες, υπάρχουν και βοηθητικοί πίνακες για την παρακολούθηση της διαδικασίας ingestion. Τέτοιοι πίνακες είναι οι `entity_freshness`, `ingestion_runs`, `ingestion_step_logs` και `api_request_logs`. Οι πίνακες αυτοί δεν περιέχουν αγωνιστικά δεδομένα, αλλά βοηθούν στον έλεγχο της διαδικασίας συλλογής, στην καταγραφή επιτυχημένων ή αποτυχημένων βημάτων και στην παρακολούθηση του πότε ενημερώθηκε τελευταία φορά κάθε κατηγορία δεδομένων.

Επιπλέον, στο σχήμα της βάσης υπάρχουν views που διευκολύνουν τόσο το ingestion όσο και την ανάκτηση δεδομένων από την εφαρμογή. Για παράδειγμα, το view `current_season_teams` βοηθά στον εντοπισμό των ομάδων που ανήκουν στις τρέχουσες σεζόν, ενώ το `standings_with_team_in` συνδυάζει πληροφορίες βαθμολογίας με στοιχεία ομάδας. Με αυτόν τον τρόπο, σύνθετες πληροφορίες μπορούν να ανακτηθούν πιο εύκολα χωρίς να επαναλαμβάνονται πολύπλοκα joins σε διάφορα σημεία του κώδικα.

4.7 Διαχείριση διπλότυπων και λογική upsert

Ένα βασικό ζήτημα στη δημιουργία ενός dataset από εξωτερικό API είναι η αποφυγή διπλότυπων εγγραφών. Τα ingestion scripts μπορεί να εκτελεστούν πολλές φορές, είτε χειροκίνητα

είτε μέσω αυτοματοποιημένων refresh jobs. Αν κάθε εκτέλεση δημιουργούσε νέες εγγραφές χωρίς έλεγχο, η βάση θα γέμιζε γρήγορα με επαναλαμβανόμενα δεδομένα. Για τον λόγο αυτό χρησιμοποιείται συστηματικά η λογική `upsert`.

Το `upsert` συνδυάζει τη λειτουργία εισαγωγής και ενημέρωσης. Αν η εγγραφή δεν υπάρχει, εισάγεται στη βάση. Αν υπάρχει ήδη, ενημερώνεται. Το ποια εγγραφή θεωρείται ίδια καθορίζεται από το πεδίο ή τον συνδυασμό πεδίων που δίνεται στο `onConflict`. Για παράδειγμα, σε πίνακες όπως `players`, `teams`, `seasons` και `tournaments`, το βασικό πεδίο σύγκρουσης είναι το `api_id`. Αντίθετα, στους πίνακες στατιστικών χρησιμοποιούνται σύνθετα κλειδιά, επειδή μια εγγραφή καθορίζεται από περισσότερες από μία τιμές.

Εκτός από τη διαχείριση διπλότυπων, η εφαρμογή παρακολουθεί και την κατάσταση ενημέρωσης των δεδομένων. Αυτό γίνεται μέσω του πίνακα `entity_freshness`, στον οποίο αποθηκεύεται πότε ένα συγκεκριμένο entity ενημερώθηκε με επιτυχία ή πότε απέτυχε η ενημέρωσή του.

```
export async function markFresh({ entityType, entityKey }) {
const now = new Date().toISOString();
const { error } = await supabase.from("entity_freshness").upsert(
{
entity_type: entityType,
entity_key: entityKey,
last_fetched_at: now,
status: "success",
error: null,
updated_at: now,
},
{ onConflict: "entity_type,entity_key" },
);
if (error) throw error;
}
```

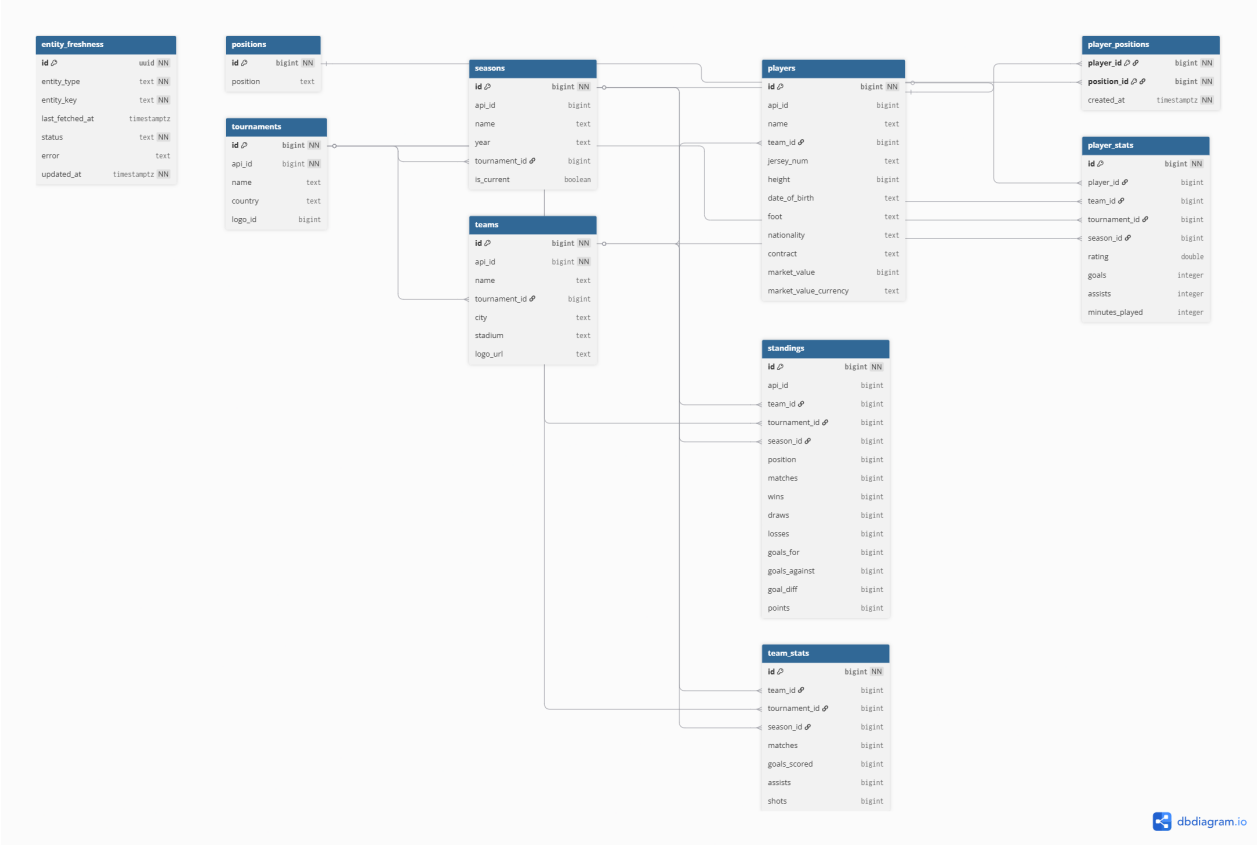
Listing 4.12: Upsert κατάστασης συγχρονισμού δεδομένων

Το παραπάνω απόσπασμα δείχνει ότι για κάθε συνδυασμό `entity_type` και `entity_key` υπάρχει μία εγγραφή κατάστασης. Αν το ίδιο entity ενημερωθεί ξανά, η υπάρχουσα εγγραφή ανανεώνεται αντί να δημιουργηθεί νέα. Αυτή η λογική είναι χρήσιμη για τα scheduled jobs, επειδή επιτρέπει στο σύστημα να γνωρίζει ποια δεδομένα έχουν ήδη ενημερωθεί πρόσφατα και ποια χρειάζονται νέα συλλογή.

4.8 Τελική δομή του συνόλου δεδομένων / βάσης δεδομένων

Η τελική δομή της βάσης οργανώνεται γύρω από τις βασικές έννοιες του ποδοσφαίρου: διοργανώσεις, σεζόν, ομάδες, παίκτες, βαθμολογίες και στατιστικά. Η πληροφορία δεν αποθηκεύεται

σε έναν ενιαίο μεγάλο πίνακα, αλλά κατανέμεται σε επιμέρους πίνακες, ώστε να μειώνεται η επανάληψη και να είναι πιο εύκολη η επεξεργασία των δεδομένων.



Σχήμα 4.1: Διάγραμμα οντοτήτων-συσχετίσεων της βάσης δεδομένων της εφαρμογής TransferMind

Όπως φαίνεται στο Σχήμα 4.1, η βάση δεδομένων αποτελείται από πίνακες που αναπαριστούν τις βασικές οντότητες της εφαρμογής, όπως διοργανώσεις, σεζόν, ομάδες, παίκτες, βαθμολογίες και στατιστικά. Οι πίνακες `tournaments`, `seasons` και `teams` σχηματίζουν τον βασικό κορμό της βάσης, ενώ οι πίνακες `standings`, `team_stats` και `player_stats` αποθηκεύουν τα αγωνιστικά και στατιστικά δεδομένα. Παράλληλα, οι πίνακες `positions` και `player_positions` χρησιμοποιούνται για την αναπαράσταση των θέσεων των παικτών. Για λόγους ευκρίνειας, στο διάγραμμα δεν εμφανίζονται όλα τα πεδία των πινάκων `team_stats` και `player_stats`. Οι δύο αυτοί πίνακες περιλαμβάνουν περισσότερες στατιστικές στήλες, όπως μετρήσεις για γκολ, ασίστ, πάσες, σουτ, τάκλιν, κατοχή και ποσοστά ακρίβειας. Στο διάγραμμα παρουσιάζονται κυρίως τα βασικά πεδία και τα κλειδιά που δείχνουν πώς οι πίνακες συνδέονται με τις ομάδες, τους παίκτες, τις διοργανώσεις και τις σεζόν.

Ο πίνακας `tournaments` αποθηκεύει τις διοργανώσεις, ενώ ο πίνακας `seasons` αποθηκεύει τις σεζόν που αντιστοιχούν σε κάθε διοργάνωση. Ο πίνακας `standings` συνδέει ομάδες με συγκεκριμένη διοργάνωση και σεζόν και αποθηκεύει τη βαθμολογική τους εικόνα. Ο πίνακας `teams` κρατά τα βασικά στοιχεία κάθε ομάδας, ενώ ο πίνακας `team_stats` αποθηκεύει

τα αγωνιστικά στατιστικά της ομάδας σε συγκεκριμένο πλαίσιο.

Αντίστοιχα, ο πίνακας `players` αποθηκεύει τα στοιχεία των παικτών και ο πίνακας `player_stats` τα στατιστικά τους. Οι πίνακες `positions` και `player_positions` χρησιμοποιούνται για την πιο σωστή αναπαράσταση των θέσεων, επιτρέποντας τη σύνδεση ενός παίκτη με μία ή περισσότερες θέσεις.

Οι κανόνες μοναδικότητας στη βάση είναι σημαντικοί, επειδή στηρίζουν πρακτικά τη λογική του `upsert`. Χωρίς αυτούς τους κανόνες, η βάση δεν θα μπορούσε να γνωρίζει με ασφάλεια πότε μια εγγραφή θεωρείται ήδη υπάρχουσα.

```
ALTER TABLE ONLY "public"."entity_freshness"
ADD CONSTRAINT "entity_freshness_entity_type_entity_key_key"
UNIQUE ("entity_type", "entity_key");
ALTER TABLE ONLY "public"."player_stats"
ADD CONSTRAINT "player_stats_player_team_season_tournament_uniq"
UNIQUE ("player_id", "team_id", "season_id", "tournament_id");
ALTER TABLE ONLY "public"."players"
ADD CONSTRAINT "players_api_id_key" UNIQUE ("api_id");
ALTER TABLE ONLY "public"."seasons"
ADD CONSTRAINT "seasons_api_id_key" UNIQUE ("api_id");
ALTER TABLE ONLY "public"."standings"
ADD CONSTRAINT "standings_api_id_key" UNIQUE ("api_id");
ALTER TABLE ONLY "public"."team_stats"
ADD CONSTRAINT "team_stats_unique_key"
UNIQUE ("team_id", "season_id", "tournament_id");
ALTER TABLE ONLY "public"."teams"
ADD CONSTRAINT "teams_api_id_key" UNIQUE ("api_id");
ALTER TABLE ONLY "public"."tournaments"
ADD CONSTRAINT "tournaments_api_id_key" UNIQUE ("api_id");
```

Listing 4.13: Κανόνες μοναδικότητας της βάσης δεδομένων

Το παραπάνω SQL απόσπασμα συνοψίζει τους βασικούς κανόνες ακεραιότητας που χρησιμοποιούνται στο dataset. Οι πίνακες που βασίζονται σε αντικείμενα του Sofascore, όπως παίκτες, ομάδες, διοργανώσεις, σεζόν και βαθμολογίες, χρησιμοποιούν το `api_id` ως μοναδικό πεδίο. Οι πίνακες στατιστικών χρησιμοποιούν σύνθετα μοναδικά κλειδιά, επειδή η μοναδικότητα δεν καθορίζεται μόνο από την ομάδα ή τον παίκτη, αλλά και από τη σεζόν και τη διοργάνωση.

Συνολικά, η διαδικασία που παρουσιάστηκε στο κεφάλαιο αυτό μετατρέπει τα αρχικά δεδομένα του Sofascore API σε ένα σταθερό και οργανωμένο σύνολο δεδομένων. Το API χρησιμοποιείται ως εξωτερική πηγή, τα ingestion scripts αναλαμβάνουν τη συλλογή και τη μετατροπή των απαντήσεων, ενώ η βάση Supabase/PostgreSQL αποθηκεύει τα δεδομένα με τρόπο που επιτρέπει επαναλαμβανόμενες ενημερώσεις χωρίς διπλότυπα. Αυτή η δομή αποτελεί τη βάση πάνω στην οποία στηρίζονται οι επόμενες λειτουργίες της εφαρμογής, όπως η παρουσίαση προφίλ ομάδων και παικτών, η σύγκριση στατιστικών και η εφαρμογή αλγορίθμων ανάλυσης δεδομένων.

Κεφάλαιο 5

Σχεδίαση και Υλοποίηση

5.1 Ανάλυση Απαιτήσεων

Η ανάλυση απαιτήσεων του TransferMind καθορίζει τις βασικές λειτουργίες που πρέπει να παρέχει η εφαρμογή, καθώς και τα ποιοτικά χαρακτηριστικά που πρέπει να ικανοποιεί. Η εφαρμογή δεν περιορίζεται στην απλή προβολή ποδοσφαιρικών δεδομένων, αλλά συνδυάζει αναζήτηση, παρουσίαση στατιστικών, σύγκριση ομάδων, clustering, association rules mining και οπτικοποίηση αποτελεσμάτων. Για τον λόγο αυτό, οι απαιτήσεις χωρίζονται σε λειτουργικές απαιτήσεις, μη λειτουργικές απαιτήσεις και user stories, ώστε να αποτυπωθούν τόσο οι τεχνικές ανάγκες του συστήματος όσο και οι ανάγκες του τελικού χρήστη.

Η ανάλυση απαιτήσεων αποτελεί βασικό στάδιο στη σχεδίαση μιας εφαρμογής, καθώς καθορίζει τις λειτουργίες που πρέπει να υποστηρίζει το σύστημα και τις ανάγκες που καλείται να καλύψει. Στην περίπτωση του TransferMind, οι απαιτήσεις προκύπτουν από τον σκοπό της εφαρμογής, δηλαδή την παρουσίαση και ανάλυση ποδοσφαιρικών δεδομένων για ομάδες και παίκτες, καθώς και από τις λειτουργίες που έχουν υλοποιηθεί στο frontend, στο backend και στη βάση δεδομένων. Η ενότητα αυτή δεν εστιάζει στον κώδικα της υλοποίησης, αλλά περιγράφει τι πρέπει να προσφέρει το σύστημα στον χρήστη και με ποιον τρόπο οργανώνονται οι βασικές λειτουργικές του ανάγκες.

Το TransferMind σχεδιάστηκε ως διαδικτυακή εφαρμογή ανάλυσης ποδοσφαιρικών δεδομένων. Ο χρήστης μπορεί να αναζητήσει ομάδες και παίκτες, να προβάλει βαθμολογίες διοργανώσεων, να εξετάσει προφίλ ομάδων και παικτών, να συγκρίνει ομάδες ανά σεζόν και να αξιοποιήσει αλγορίθμους εξόρυξης δεδομένων με εργαλεία ανάλυσης, όπως K-Means clustering, ιεραρχική συσταδοποίηση και Apriori association rules. Οι βασικοί χρήστες της εφαρμογής μπορούν να είναι φίλαθλοι που ενδιαφέρονται για στατιστικά ομάδων και παικτών, χρήστες που θέλουν να συγκρίνουν ομάδες μεταξύ τους, καθώς και φοιτητές ή αναλυτές που επιθυμούν να εξετάσουν ποδοσφαιρικά δεδομένα μέσα από τεχνικές εξόρυξης δεδομένων. Για τον λόγο αυτό, η εφαρμογή πρέπει να παραμένει κατανοητή και εύχρηστη, χωρίς όμως να αφαιρεί τις πιο σύνθετες δυνατότητες ανάλυσης.

Σε επίπεδο λειτουργικού πεδίου, η εφαρμογή περιλαμβάνει αρχική σελίδα με αναζήτηση ομάδων και παικτών, σελίδα βαθμολογιών, κατάλογο ομάδων, προβολή προφίλ ομάδας, προβολή παικτών, προφίλ παίκτη και ξεχωριστή ενότητα σύγκρισης ομάδων. Η ενότητα σύγκρισης ομάδων αποτελεί ένα από τα πιο σημαντικά σημεία της εφαρμογής, καθώς επιτρέπει στον χρήστη να επιλέξει ομάδες, σεζόν και στατιστικά, ώστε να δημιουργήσει συγκρίσεις και να εφαρμόσει αλγορίθμους ανάλυσης. Τα δεδομένα που απαιτούνται για τη λειτουργία της εφαρμογής οργανώνονται γύρω από τις βασικές οντότητες του ποδοσφαιρικού πεδίου, δηλαδή τις ομάδες, τους παίκτες, τις διοργανώσεις, τις σεζόν, τις βαθμολογίες, τα στατιστικά ομάδων, και τα στατιστικά παικτών. Επιπλέον, για τις λειτουργίες σύγκρισης και εξόρυξης δεδομένων χρησιμοποιούνται εγγραφές τύπου team-season, δηλαδή συνδυασμοί ομάδας, διοργάνωσης και σεζόν. Με αυτόν τον τρόπο, η σύγκριση και η ανάλυση δεν πραγματοποιούνται γενικά για μια ομάδα, αλλά σε συγκεκριμένο αγωνιστικό πλαίσιο.

Ιδιαίτερη απαίτηση της εφαρμογής είναι η υποστήριξη αλγορίθμων ανάλυσης δεδομένων. Για το K-Means clustering ο χρήστης πρέπει να μπορεί να επιλέγει ομάδες, σεζόν, στατιστικά και μέγιστο αριθμό συστάδων για τη μέθοδο elbow. Στη συνέχεια, πρέπει να μπορεί να εκτελεί την τελική συσταδοποίηση και να βλέπει τα αποτελέσματα σε μορφή αναθέσεων ομάδων σε clusters και σχετικών οπτικοποιήσεων. Για την ιεραρχική συσταδοποίηση, η εφαρμογή πρέπει να υποστηρίζει επιλογή αριθμού συστάδων και μεθόδου linkage, ενώ το αποτέλεσμα πρέπει να παρουσιάζεται με dendrogram και πίνακες αποτελεσμάτων. Για τον Apriori, ο χρήστης πρέπει να μπορεί να ορίζει παραμέτρους όπως support, confidence και lift, ώστε να εντοπίζει κανόνες συσχέτισης στα επιλεγμένα στατιστικά.

Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τις βασικές λειτουργίες τις οποίες προσφέρει η εφαρμογή στον χρήστη. Στο TransferMind, οι βασικές λειτουργίες αφορούν την αναζήτηση ομάδων και παικτών, την προβολή προφίλ ομάδων και παικτών, την παρουσίαση βαθμολογιών, τη σύγκριση ομάδων και την εκτέλεση αναλυτικών αλγορίθμων, οι οποίες αποτυπώνονται αναλυτικά στον Πίνακα 5.1.

Κωδικός	**Λειτουργική απαίτηση**	**Περιγραφή**	**Χρήστης/ Ρόλος**	**Προτεραιότητα**
ΛΑ-01	Προβολή αρχικής σελίδας	Το σύστημα πρέπει να παρέχει αρχική σελίδα με πρόσβαση στις βασικές ενότητες της εφαρμογής.	Επισκέπτης / Χρήστης	Υψηλή
ΛΑ-02	Προβολή βαθμολογιών	Το σύστημα πρέπει να εμφανίζει βαθμολογίες ποδοσφαιρικών διοργανώσεων ανά τουρνουά, σεζόν, στάδιο και όμιλο.	Χρήστης	Υψηλή
ΛΑ-03	Φιλτράρισμα βαθμολογιών	Ο χρήστης πρέπει να μπορεί να φιλτράρει τις βαθμολογίες με βάση διαθέσιμα κριτήρια, όπως διοργάνωση, σεζόν, στάδιο ή όμιλο.	Χρήστης	Υψηλή

Κωδικός	**Λειτουργική απαίτηση**	**Περιγραφή**	**Χρήστης/ Ρόλος**	**Προτεραιότητα**
ΛΑ-04	Προβολή λίστας ομάδων	Το σύστημα πρέπει να εμφανίζει κατάλογο ομάδων που υπάρχουν στη βάση δεδομένων.	Χρήστης	Υψηλή
ΛΑ-05	Αναζήτηση ομάδων	Ο χρήστης πρέπει να μπορεί να αναζητά ομάδες με βάση το όνομα ή άλλα διαθέσιμα στοιχεία.	Χρήστης	Μεσαία
ΛΑ-06	Προβολή προφίλ ομάδας	Το σύστημα πρέπει να εμφανίζει αναλυτική σελίδα ομάδας με βασικές πληροφορίες, ρόστερ, βαθμολογική εικόνα και στατιστικά ανά σεζόν.	Χρήστης	Υψηλή
ΛΑ-07	Προβολή στατιστικών ομάδας	Το σύστημα πρέπει να εμφανίζει στατιστικά στοιχεία ομάδας για συγκεκριμένη σεζόν και διοργάνωση.	Χρήστης	Υψηλή
ΛΑ-08	Προβολή λίστας παικτών	Το σύστημα πρέπει να εμφανίζει κατάλογο παικτών που έχουν συλλεχθεί από την πηγή δεδομένων.	Χρήστης	Μεσαία
ΛΑ-09	Προβολή προφίλ παίκτη	Το σύστημα πρέπει να εμφανίζει αναλυτική σελίδα παίκτη με βασικές πληροφορίες και διαθέσιμα στατιστικά.	Χρήστης	Μεσαία
ΛΑ-10	Επιλογή ομάδων ανά σεζόν	Το σύστημα πρέπει να επιτρέπει την επιλογή ομάδων σε επίπεδο ομάδας-σεζόν και όχι μόνο σε επίπεδο ομάδας.	Χρήστης	Υψηλή
ΛΑ-11	Επιλογή στατιστικών σύγκρισης	Ο χρήστης πρέπει να μπορεί να επιλέγει τα στατιστικά που θα χρησιμοποιηθούν στη σύγκριση ομάδων.	Χρήστης	Υψηλή
ΛΑ-12	Προβολή ακατέργαστων τιμών στατιστικών	Το σύστημα πρέπει να εμφανίζει τις πραγματικές τιμές των στατιστικών, ώστε ο χρήστης να βλέπει τα αρχικά δεδομένα.	Χρήστης	Υψηλή
ΛΑ-13	Προβολή σχετικών/κανονικοποιημένων βαθμολογιών	Το σύστημα πρέπει να εμφανίζει σχετικές ή κανονικοποιημένες βαθμολογίες, ώστε να είναι δυνατή η σύγκριση ομάδων μεταξύ τους.	Χρήστης	Υψηλή
ΛΑ-14	Οπτικοποίηση συγκρίσεων	Το σύστημα πρέπει να παρουσιάζει τις συγκρίσεις ομάδων μέσω διαγραμμάτων και γραφικών απεικονίσεων.	Χρήστης	Υψηλή
ΛΑ-15	Προσαρμοσμένη σύγκριση ομάδων	Ο χρήστης πρέπει να μπορεί να δημιουργεί προσαρμοσμένη σύγκριση επιλέγοντας συγκεκριμένες ομάδες, σεζόν και στατιστικά.	Χρήστης	Υψηλή

Κωδικός	**Λειτουργική απαίτηση**	**Περιγραφή**	**Χρήστης/ Ρόλος**	**Προτεραιότητα**
ΛΑ-16	Ανάλυση συστάδων με K-Means	Το σύστημα πρέπει να υποστηρίζει ομαδοποίηση ομάδων με χρήση του αλγορίθμου K-Means.	Χρήστης	Υψηλή
ΛΑ-17	Υποστήριξη μεθόδου Elbow	Το σύστημα πρέπει να παρέχει ανάλυση Elbow για την εκτίμηση κατάλληλου αριθμού συστάδων στο K-Means.	Χρήστης	Υψηλή
ΛΑ-18	Ιεραρχική ομαδοποίηση	Το σύστημα πρέπει να υποστηρίζει ανάλυση συστάδων με ιεραρχική ομαδοποίηση και προβολή σχετικών αποτελεσμάτων.	Χρήστης	Υψηλή
ΛΑ-19	Εξόρυξη κανόνων συσχέτισης	Το σύστημα πρέπει να υποστηρίζει εξόρυξη κανόνων συσχέτισης με βάση μοτίβα στατιστικών ομάδων-σεζόν.	Χρήστης	Υψηλή
ΛΑ-20	Προβολή αποτελεσμάτων μηχανικής μάθησης	Το σύστημα πρέπει να εμφανίζει τα αποτελέσματα των αλγορίθμων ανάλυσης με τρόπο κατανοητό και αξιοποιήσιμο από τον χρήστη.	Χρήστης	Υψηλή
ΛΑ-21	Ανάκτηση δεδομένων μέσω backend API	Το frontend πρέπει να ανακτά όλα τα δεδομένα μέσω του backend API και όχι απευθείας από τη βάση δεδομένων.	Σύστημα	Υψηλή
ΛΑ-22	Συλλογή δεδομένων από εξωτερική πηγή	Το σύστημα πρέπει να διαθέτει μηχανισμούς εισαγωγής δεδομένων για διοργανώσεις, σεζόν, βαθμολογίες, ομάδες, λογότυπα, παίκτες και στατιστικά.	Διαχειριστής / Σύστημα	Υψηλή
ΛΑ-23	Περιοδική ανανέωση δεδομένων	Το σύστημα πρέπει να υποστηρίζει προγραμματισμένες εργασίες ανανέωσης για τρέχουσες βαθμολογίες, στατιστικά ομάδων και στατιστικά παικτών.	Διαχειριστής / Σύστημα	Υψηλή
ΛΑ-24	Διαχείριση ιστορικών δεδομένων	Το σύστημα πρέπει να διατηρεί ιστορικά δεδομένα σεζόν, ομάδων, παικτών και στατιστικών χωρίς να περιορίζεται αποκλειστικά στην τρέχουσα σεζόν.	Σύστημα	Υψηλή

Πίνακας 5.1: Λειτουργικές απαιτήσεις της εφαρμογής TransferMind

Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις δεν περιγράφουν συγκεκριμένες λειτουργίες, αλλά τα ποιοτικά χαρακτηριστικά που πρέπει να έχει το σύστημα. Για το TransferMind, σημαντικά στοιχεία είναι η χρηστικότητα, η απόκριση, η αξιοπιστία, η συντηρησιμότητα, η επεκτασιμότητα και ο καθαρός διαχωρισμός των επιπέδων της εφαρμογής. Στον Πίνακα 5.2 παρουσιάζονται οι βασικότερες μη λειτουργικές απαιτήσεις.

Κωδικός	**Μη λειτουργική απαίτηση**	**Περιγραφή**	**Κατηγορία**	**Προτεραιότητα**
ΜΛΑ-01	Διαχωρισμός frontend και backend	Το σύστημα πρέπει να διατηρεί σαφή διαχωρισμό μεταξύ frontend εφαρμογής και backend API. Το frontend δεν πρέπει να έχει άμεση πρόσβαση στη βάση δεδομένων.	Αρχιτεκτονική	Υψηλή
ΜΛΑ-02	Χρήση backend API για πρόσβαση σε δεδομένα	Όλα τα δεδομένα που εμφανίζονται στο frontend πρέπει να ανακτώνται μέσω του backend API.	Ασφάλεια / Αρχιτεκτονική	Υψηλή
ΜΛΑ-03	Προστασία ευαίσθητων στοιχείων	Κλειδιά API, Supabase service keys, RapidAPI credentials και αρχεία `.env` δεν πρέπει να εκτίθενται ή να αποθηκεύονται δημόσια στο αποθετήριο.	Ασφάλεια	Υψηλή
ΜΛΑ-04	Απόκρυψη εξωτερικών API ids	Τα εξωτερικά αναγνωριστικά της πηγής δεδομένων πρέπει να παραμένουν εσωτερική λεπτομέρεια του backend και να μην εκτίθενται στο frontend ή στα δημόσια API responses.	Ασφάλεια / Συντηρησιμότητα	Υψηλή
ΜΛΑ-05	Συντηρήσιμη αρχιτεκτονική backend	Το backend πρέπει να ακολουθεί τη ροή route → controller → service → repository, ώστε ο κώδικας να παραμένει οργανωμένος και επεκτάσιμος.	Συντηρησιμότητα	Υψηλή
ΜΛΑ-06	Λεπτοί controllers	Οι controllers πρέπει να περιορίζονται στην ανάγνωση παραμέτρων, στην κλήση των services και στην επιστροφή HTTP απαντήσεων.	Συντηρησιμότητα	Μεσαία
ΜΛΑ-07	Διαχωρισμός επιχειρησιακής λογικής	Η επιχειρησιακή λογική, η κανονικοποίηση δεδομένων και η μορφοποίηση απαντήσεων πρέπει να υλοποιούνται στα services.	Συντηρησιμότητα	Υψηλή

Κωδικός	**Μη λειτουργική απαίτηση**	**Περιγραφή**	**Κατηγορία**	**Προτεραιότητα**
ΜΛΑ-08	Απομόνωση πρόσβασης στη βάση	Τα repositories πρέπει να είναι το βασικό σημείο πρόσβασης στη Supabase/PostgreSQL βάση δεδομένων.	Συντηρησιμότητα	Υψηλή
ΜΛΑ-09	Αξιοπιστία δεδομένων	Το σύστημα πρέπει να χρησιμοποιεί μηχανισμούς ανανέωσης και freshness checks, όπου υπάρχουν, ώστε τα δεδομένα να παραμένουν επίκαιρα.	Αξιοπιστία	Υψηλή
ΜΛΑ-10	Διατήρηση ιστορικότητας	Οι εργασίες ανανέωσης δεν πρέπει να διαγράφουν αυθαίρετα ιστορικά δεδομένα σεζόν, ομάδων, παικτών ή στατιστικών.	Αξιοπιστία / Ακεραιότητα δεδομένων	Υψηλή
ΜΛΑ-11	Επεκτασιμότητα ανάλυσης	Το σύστημα πρέπει να μπορεί να επεκταθεί με νέες μεθόδους ανάλυσης ή μηχανικής μάθησης χωρίς σημαντική αναδόμηση της εφαρμογής.	Επεκτασιμότητα	Μεσαία
ΜΛΑ-12	Απόδοση διεπαφής χρήστη	Η frontend εφαρμογή πρέπει να φορτώνει και να ανταποκρίνεται σε εύλογο χρόνο κατά την προβολή λιστών, προφίλ, στατιστικών και διαγραμμάτων.	Απόδοση	Υψηλή
ΜΛΑ-13	Απόδοση API	Το backend API πρέπει να επιστρέφει δεδομένα σε αποδεκτούς χρόνους απόκρισης για τις βασικές λειτουργίες, όπως ομάδες, παίκτες, βαθμολογίες και συγκρίσεις.	Απόδοση	Υψηλή
ΜΛΑ-14	Κλιμακωτή ανάκτηση δεδομένων	Για μεγάλα σύνολα δεδομένων, όπως λίστες παικτών, το σύστημα πρέπει να υποστηρίζει αποδοτική ανάκτηση και κατάλληλα ευρετήρια στη βάση.	Απόδοση / Κλιμάκωση	Μεσαία
ΜΛΑ-15	Φορητότητα περιβάλλοντος	Το σύστημα πρέπει να μπορεί να εκτελεστεί τοπικά και σε περιβάλλον παραγωγής με χρήση μεταβλητών περιβάλλοντος.	Φορητότητα	Υψηλή
ΜΛΑ-16	Υποστήριξη deployment	Το frontend πρέπει να μπορεί να αναπτυχθεί σε Vercel και το backend σε Render, σύμφωνα με τη δομή του έργου.	Λειτουργικότητα ανάπτυξης	Μεσαία

Κωδικός	**Μη λειτουργική απαίτηση**	**Περιγραφή**	**Κατηγορία**	**Προτεραιότητα**
ΜΛΑ-17	Διαθεσιμότητα ελέγχου υγείας	Το backend πρέπει να παρέχει endpoint ελέγχου υγείας, ώστε να είναι δυνατή η παρακολούθηση της βασικής λειτουργίας του API.	Διαθεσιμότητα	Μεσαία
ΜΛΑ-18	Χρήση migrations για αλλαγές βάσης	Οι αλλαγές στο σχήμα της βάσης δεδομένων πρέπει να γίνονται μέσω νέων migrations και όχι με τροποποίηση παλιών εφαρμοσμένων migrations.	Ακεραιότητα / Συντηρησιμότητα	Υψηλή
ΜΛΑ-19	Συμβατότητα με σύγχρονο περιβάλλον Node.js	Το σύστημα πρέπει να υποστηρίζει εκτέλεση με Node.js 20, όπως χρησιμοποιείται στις εργασίες αυτοματοποίησης.	Συμβατότητα	Μεσαία
ΜΛΑ-20	Συμβατότητα Python αναλύσεων	Οι λειτουργίες clustering και association rules πρέπει να εκτελούνται σε περιβάλλον Python με εγκατεστημένες τις απαιτούμενες εξαρτήσεις.	Συμβατότητα	Μεσαία
ΜΛΑ-21	Αυτοματοποιημένες εργασίες ανανέωσης	Το σύστημα πρέπει να υποστηρίζει προγραμματισμένη ή χειροκίνητη εκτέλεση εργασιών ανανέωσης μέσω GitHub Actions.	Αυτοματοποίηση	Μεσαία
ΜΛΑ-22	Περιορισμός φόρτου εξωτερικού API	Οι εργασίες εισαγωγής δεδομένων πρέπει να σέβονται περιορισμούς όγκου αιτημάτων προς την εξωτερική ποδοσφαιρική API πηγή.	Αξιοπιστία / Απόδοση	Υψηλή
ΜΛΑ-23	Ευχρηστία διεπαφής	Η εφαρμογή πρέπει να παρέχει καθαρή και κατανοητή διεπαφή για εξερεύνηση ομάδων, παικτών, βαθμολογιών και στατιστικών αναλύσεων.	Ευχρηστία	Υψηλή
ΜΛΑ-24	Οπτική κατανόηση δεδομένων	Τα διαγράμματα και οι οπτικοποιήσεις πρέπει να βοηθούν τον χρήστη να ερμηνεύει στατιστικές συγκρίσεις και αποτελέσματα αναλύσεων.	Ευχρηστία	Υψηλή
ΜΛΑ-25	Τυποποιημένη μορφή API responses	Τα επιτυχημένα API responses πρέπει να ακολουθούν συνεπή μορφή, όπως χρήση envelope τύπου `{ data: ... }` όπου εφαρμόζεται.	Συνέπεια / Συντηρησιμότητα	Μεσαία

Πίνακας 5.2: Μη λειτουργικές απαιτήσεις της εφαρμογής TransferMind

User Stories

Τα user stories περιγράφουν τις απαιτήσεις από την οπτική του χρήστη. Στην περίπτωση του TransferMind, ο βασικός χρήστης μπορεί να θεωρηθεί ένας αναλυτής ποδοσφαιρικών δεδομένων, ο οποίος θέλει να εξετάζει ομάδες, παίκτες, στατιστικά και μοτίβα απόδοσης.

User Stories του TransferMind

As an analyst, I want to search for teams and players, so that I can quickly access their profiles and statistics.

As an analyst, I want to compare multiple teams using selected statistics, so that I can identify performance differences.

As an analyst, I want to run K-Means clustering, so that I can discover groups of teams with similar statistical profiles.

As an analyst, I want to view an elbow curve, so that I can choose a suitable number of clusters.

As an analyst, I want to run Agglomerative clustering and view a dendrogram, so that I can understand hierarchical similarities between teams.

As an analyst, I want to apply Association Rules Mining, so that I can discover frequent relationships between performance indicators.

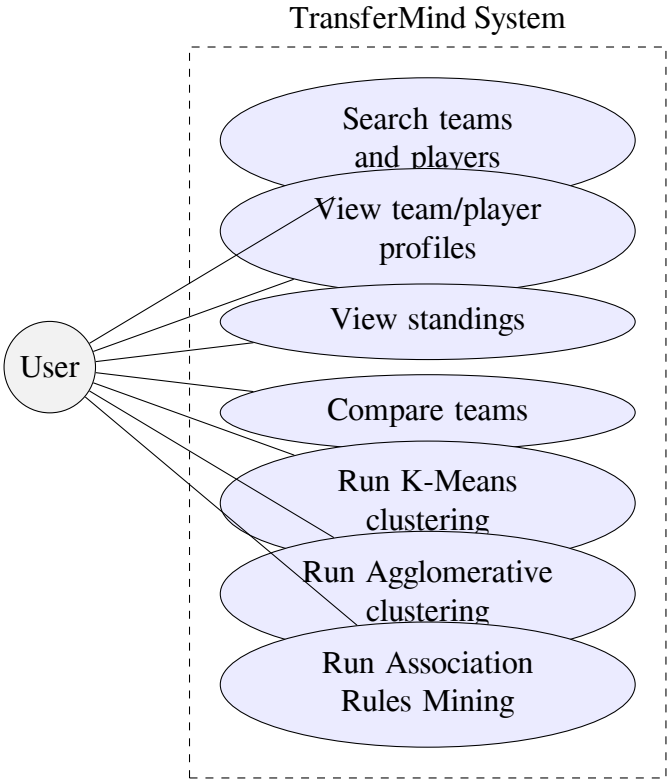
As a user, I want to receive clear warnings when my selections are not sufficient, so that I can correct the input parameters.

Σχήμα 5.1: User stories της εφαρμογής

Τα user stories του Σχήματος 5.1 δείχνουν ότι η εφαρμογή σχεδιάζεται γύρω από αναλυτικές ανάγκες. Ο χρήστης δεν ενδιαφέρεται μόνο για την απλή εμφάνιση δεδομένων, αλλά και για την εξαγωγή συμπερασμάτων. Για παράδειγμα, μέσω του clustering μπορεί να εντοπίσει ομάδες με παρόμοια αγωνιστική συμπεριφορά, ενώ μέσω του Association Rules Mining μπορεί να παρατηρήσει συχνές σχέσεις ανάμεσα σε στατιστικούς δείκτες.

Use Case Diagram

Το ακόλουθο use case diagram συνοψίζει τις βασικές αλληλεπιδράσεις ανάμεσα στον χρήστη και το σύστημα.



Σχήμα 5.2: Use case diagram του TransferMind

Το Σχήμα 5.2 συγκεντρώνει τις βασικές λειτουργίες της εφαρμογής από την πλευρά του χρήστη. Ο χρήστης αλληλεπιδρά με ένα ενιαίο σύστημα, όμως εσωτερικά η εφαρμογή χωρίζεται σε διαφορετικά επίπεδα: το frontend για την παρουσίαση, το backend για τη διαχείριση των αιτημάτων και τα Python modules για την εκτέλεση των αναλυτικών αλγορίθμων.

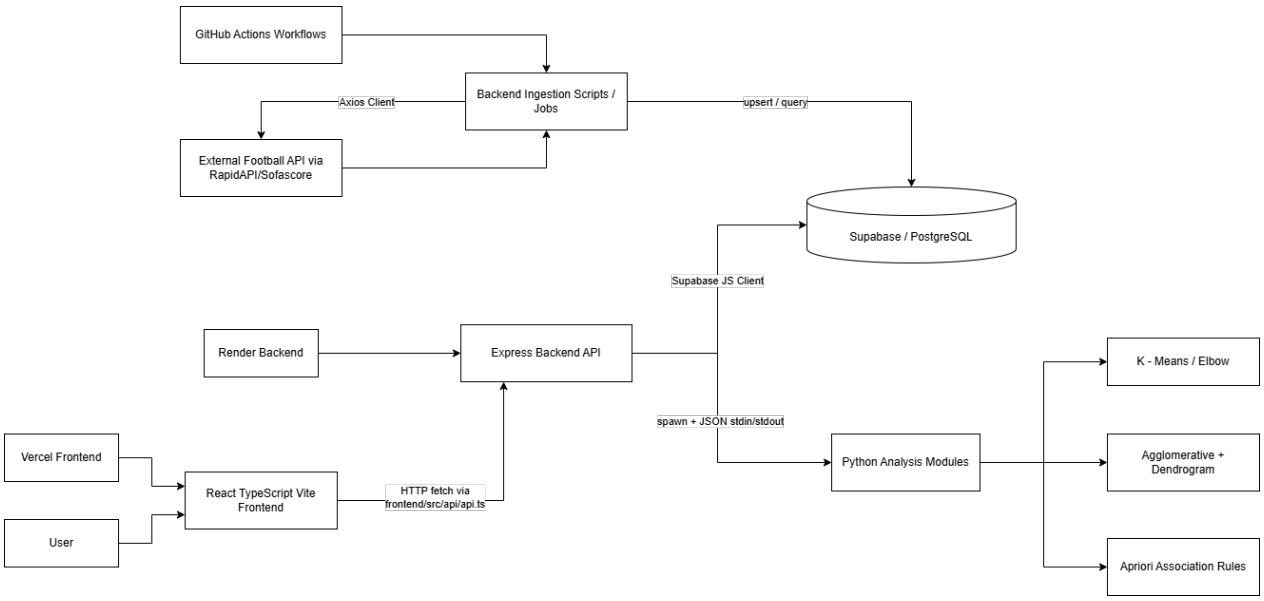
Συνολικά, η ανάλυση απαιτήσεων δείχνει ότι το TransferMind σχεδιάστηκε ως εφαρμογή ανάλυσης και όχι απλής προβολής δεδομένων. Οι λειτουργικές απαιτήσεις καλύπτουν τις βασικές ενέργειες του χρήστη, οι μη λειτουργικές απαιτήσεις εξασφαλίζουν την ποιότητα της υλοποίησης και τα user stories συνδέουν τις τεχνικές λειτουργίες με πραγματικές ανάγκες ανάλυσης ποδοσφαιρικών δεδομένων.

5.2 Αρχιτεκτονική

Η εφαρμογή TransferMind ακολουθεί πολυεπίπεδη αρχιτεκτονική, με διακριτό διαχωρισμό ανάμεσα στη διεπαφή χρήστη, το backend API, τη βάση δεδομένων, τους μηχανισμούς εισαγωγής δεδομένων και τις αναλυτικές λειτουργίες. Η συγκεκριμένη προσέγγιση επιλέχθηκε ώστε η εφαρμογή να παραμένει οργανωμένη, επεκτάσιμη και πιο εύκολη στη συντήρηση. Κάθε επίπεδο του συστήματος έχει συγκεκριμένο ρόλο και επικοινωνεί με τα υπόλοιπα μέσω καθορισμένων διεπαφών.

Σε γενικό επίπεδο, το TransferMind αποτελείται από ένα frontend υλοποιημένο με React και TypeScript, ένα backend API σε Node.js/Express, μία βάση δεδομένων Supabase/PostgreSQL και ξεχωριστά Python modules για την εκτέλεση αλγορίθμων ανάλυσης. Επιπλέον, το σύστημα περιλαμβάνει μηχανισμούς συλλογής και ανανέωσης δεδομένων από εξωτερική ποδοσφαιρική πηγή, καθώς και υποδομή deployment και αυτοματοποίησης.

Γενική διάρθρωση του συστήματος



Σχήμα 5.3: Block Diagram της αρχιτεκτονικής της εφαρμογής TransferMind

Η γενική αρχιτεκτονική του TransferMind βασίζεται στη λογική διαχωρισμού αρμοδιοτήτων. Το frontend είναι υπεύθυνο για την αλληλεπίδραση με τον χρήστη και την παρουσίαση των δεδομένων. Το backend αναλαμβάνει την επεξεργασία των αιτημάτων, την επικοινωνία με τη βάση δεδομένων, την προετοιμασία των απαντήσεων και τη διασύνδεση με τα Python scripts. Η βάση δεδομένων αποθηκεύει τα ποδοσφαιρικά δεδομένα, ενώ τα Python modules χρησιμοποιούνται για πιο εξειδικευμένες αναλυτικές διαδικασίες, όπως η συσταδοποίηση και η εξόρυξη κανόνων συσχέτισης.

Η βασική ροή επικοινωνίας ξεκινά από τον χρήστη, ο οποίος αλληλεπιδρά με τη διεπαφή της

εφαρμογής. Το frontend στέλνει HTTP αιτήματα στο backend API, το οποίο με τη σειρά του ανακτά ή επεξεργάζεται δεδομένα από τη βάση. Όταν απαιτείται αναλυτική επεξεργασία, το backend προετοιμάζει τα δεδομένα και καλεί τα αντίστοιχα Python scripts. Τα αποτελέσματα επιστρέφονται στο frontend, όπου παρουσιάζονται με πίνακες, κάρτες και διαγράμματα.

Επίπεδο διεπαφής χρήστη

Το frontend της εφαρμογής έχει υλοποιηθεί με React, TypeScript και Vite. Περιλαμβάνει σελίδες για την αρχική αναζήτηση, τις βαθμολογίες, τις ομάδες, τους παίκτες, τα προφίλ ομάδων και παικτών, καθώς και τη σελίδα σύγκρισης ομάδων. Η πλοήγηση οργανώνεται με routes, όπως `/standings`, `/teams`, `/players`, `/team/:id`, `/player/:id` και `/teams-comparison`.

Η επικοινωνία του frontend με το backend πραγματοποιείται μέσω κεντρικού API client. Με αυτόν τον τρόπο, τα επιμέρους components της εφαρμογής δεν επικοινωνούν απευθείας με τη βάση δεδομένων, αλλά αντλούν τα απαραίτητα δεδομένα μέσω των διαθέσιμων endpoints του backend. Αυτή η επιλογή ενισχύει τον διαχωρισμό των επιπέδων και περιορίζει την έκθεση εσωτερικών λεπτομερειών της βάσης στο frontend.

Στο frontend χρησιμοποιούνται επίσης επαναχρησιμοποιήσιμα components για πίνακες, κάρτες, προφίλ, βαθμολογίες, συγκρίσεις και οπτικοποιήσεις. Για την παρουσίαση στατιστικών δεδομένων αξιοποιούνται διαγράμματα, όπως bar charts, radar charts και line charts, ώστε τα αποτελέσματα να γίνονται πιο κατανοητά στον χρήστη.

Επίπεδο backend API

Το backend υλοποιείται με Node.js και Express και λειτουργεί ως το κεντρικό επίπεδο διασύνδεσης της εφαρμογής. Εκθέτει REST endpoints για αναζήτηση, ομάδες, παίκτες, βαθμολογίες, σύγκριση ομάδων, clustering και association rules. Επιπλέον, παρέχει endpoint ελέγχου λειτουργίας, ώστε να μπορεί να επιβεβαιωθεί ότι το API εκτελείται κανονικά.

Η εσωτερική οργάνωση του backend ακολουθεί τη ροή:

$$\text{Route} \rightarrow \text{Controller} \rightarrow \text{Service} \rightarrow \text{Repository} \rightarrow \text{Database}$$

Τα routes ορίζουν τα διαθέσιμα endpoints, οι controllers διαχειρίζονται τα εισερχόμενα αιτήματα και οι services περιέχουν την κύρια λογική της εφαρμογής. Τα repositories αποτελούν το σημείο πρόσβασης στη βάση δεδομένων. Με αυτή τη δομή, ο κώδικας παραμένει πιο οργανωμένος, καθώς κάθε επίπεδο έχει συγκεκριμένη ευθύνη.

Επίπεδο δεδομένων

Η βάση δεδομένων της εφαρμογής βασίζεται σε Supabase/PostgreSQL. Σε αυτήν αποθηκεύονται οι βασικές οντότητες του συστήματος, όπως διοργανώσεις, σεζόν, ομάδες, παίκτες, βαθμολογίες, στατιστικά ομάδων, στατιστικά παικτών και θέσεις παικτών. Η σχεσιακή δομή της βάσης επιτρέπει τη σύνδεση των δεδομένων μεταξύ τους, ώστε οι λειτουργίες της εφαρμογής να μπορούν να εμφανίζουν πληροφορίες σε επίπεδο ομάδας, παίκτη, διοργάνωσης και σεζόν.

Ιδιαίτερη σημασία έχει η χρήση εσωτερικών αναγνωριστικών της βάσης σε συνδυασμό με τα εξωτερικά αναγνωριστικά της πηγής δεδομένων. Τα εξωτερικά `api_id` χρησιμοποιούνται κυρίως κατά την εισαγωγή και ενημέρωση των δεδομένων, ενώ το frontend χειρίζεται κυρίως τα δεδομένα μέσα από τα endpoints του backend. Έτσι, η βάση και τα εξωτερικά ids παραμένουν εσωτερική λεπτομέρεια του συστήματος.

Εισαγωγή και ανανέωση δεδομένων

Η εφαρμογή διαθέτει μηχανισμούς εισαγωγής και ανανέωσης δεδομένων από εξωτερική ποδοσφαιρική πηγή. Οι διαδικασίες αυτές αφορούν διοργανώσεις, σεζόν, βαθμολογίες, ομάδες, λογότυπα, στατιστικά ομάδων, παίκτες και στατιστικά παικτών. Η εισαγωγή των δεδομένων εκτελείται από scripts και jobs στο backend, τα οποία επικοινωνούν με την εξωτερική API πηγή και αποθηκεύουν τα δεδομένα στη βάση.

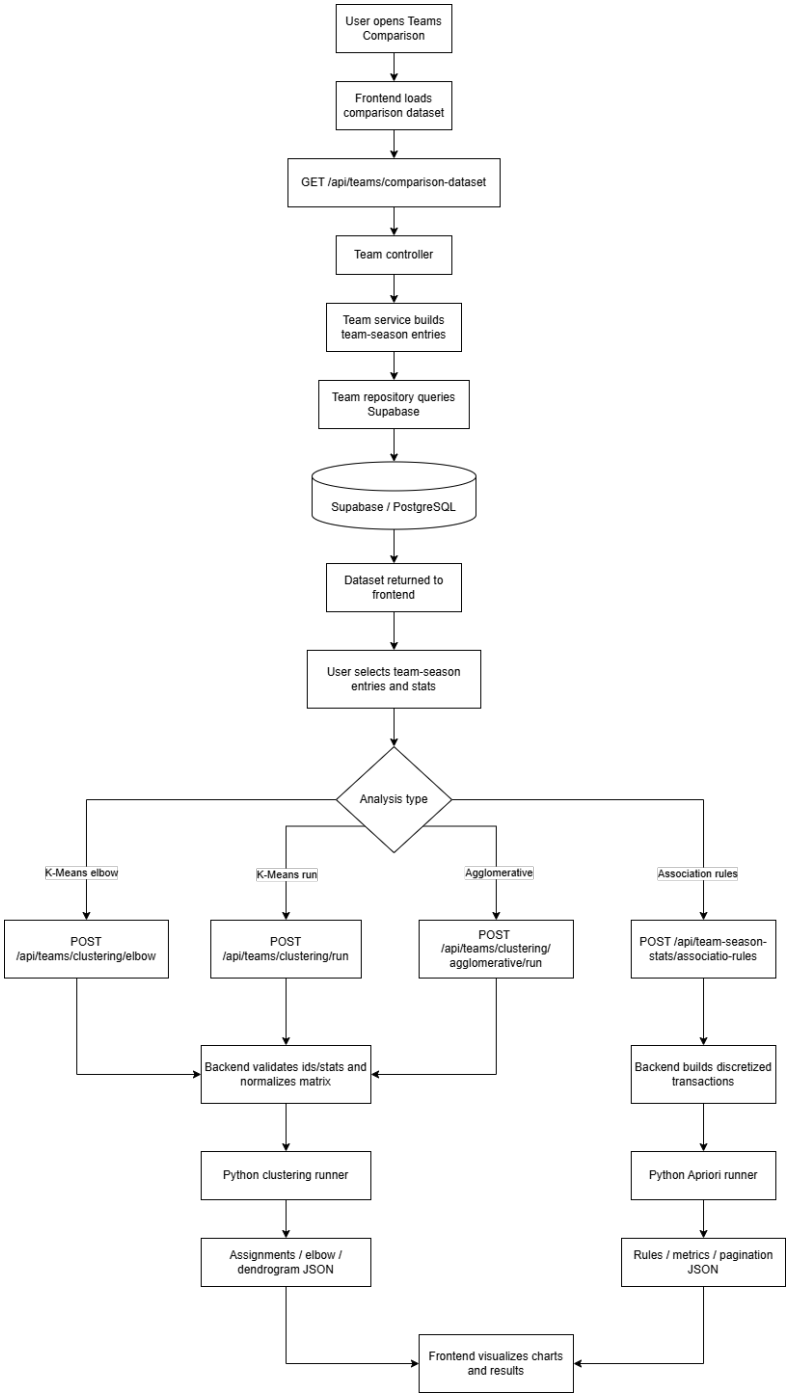
Για την αποφυγή διπλότυπων εγγραφών χρησιμοποιείται λογική ενημέρωσης υπαρχόντων δεδομένων όπου αυτό είναι απαραίτητο. Παράλληλα, η αρχιτεκτονική λαμβάνει υπόψη τη διατήρηση ιστορικών δεδομένων, ώστε η εφαρμογή να μην περιορίζεται αποκλειστικά στην τρέχουσα σεζόν. Οι διαδικασίες ανανέωσης μπορούν να εκτελούνται χειροκίνητα ή αυτοματοποιημένα, μέσω GitHub Actions.

Επίπεδο ανάλυσης δεδομένων

Οι αναλυτικές λειτουργίες του TransferMind εκτελούνται με συνεργασία backend και Python scripts. Το backend είναι υπεύθυνο για την παραλαβή των επιλογών του χρήστη, την ανάκτηση των κατάλληλων δεδομένων από τη βάση και την προετοιμασία τους σε μορφή κατάλληλη για επεξεργασία. Στη συνέχεια, καλεί τα Python modules, τα οποία εκτελούν τους αντίστοιχους αλγορίθμους.

Στην περίπτωση του K-Means, το σύστημα μπορεί να υπολογίσει τιμές για τη μέθοδο elbow και να εκτελέσει την τελική συσταδοποίηση. Για την ιεραρχική συσταδοποίηση, το backend καλεί Python script που επιστρέφει τις συστάδες και το dendrogram. Για την εξόρυξη κανόνων συσχέτισης, τα δεδομένα μετατρέπονται σε κατάλληλη μορφή συναλλαγών και στη συνέχεια εφαρμόζεται ο Apriori. Τα αποτελέσματα επιστρέφονται στο frontend και παρουσιάζονται με πίνακες και διαγράμματα.

Ροή αναλυτικής διαδικασίας



Σχήμα 5.4: Ροή εκτέλεσης αναλυτικής διαδικασίας στο TransferMind

Μία βασική ροή χρήσης αφορά την εκτέλεση ανάλυσης μέσα από τη σελίδα σύγκρισης ομάδων. Αρχικά, ο χρήστης επιλέγει ομάδες, σεζόν και στατιστικά. Το frontend στέλνει το αίτημα στο backend, το οποίο ελέγχει τις επιλογές, ανακτά τα αντίστοιχα δεδομένα και δημιουργεί τον πίνακα χαρακτηριστικών. Στη συνέχεια, ανάλογα με τον αλγόριθμο που έχει επιλεγεί, το

backend καλεί το αντίστοιχο Python script. Τα αποτελέσματα επιστρέφονται στο backend και στη συνέχεια αποστέλλονται στο frontend για οπτική παρουσίαση.

Η ροή αυτή δείχνει τον τρόπο με τον οποίο συνεργάζονται τα διαφορετικά επίπεδα της εφαρμογής. Το frontend δεν εκτελεί τους αλγορίθμους απευθείας, αλλά λειτουργεί ως επίπεδο εισόδου και παρουσίασης. Το backend οργανώνει και ελέγχει τα δεδομένα, ενώ τα Python scripts αναλαμβάνουν την υπολογιστική επεξεργασία.

Ανάπτυξη και αυτοματοποίηση

Η αρχιτεκτονική της εφαρμογής υποστηρίζει διαχωρισμένο deployment. Το frontend μπορεί να φιλοξενηθεί σε Vercel, ενώ το backend μπορεί να αναπτυχθεί σε Render. Η βάση δεδομένων φιλοξενείται στο Supabase. Τα ευαίσθητα στοιχεία, όπως κλειδιά API και στοιχεία σύνδεσης με τη βάση, διατηρούνται σε μεταβλητές περιβάλλοντος και δεν αποθηκεύονται στον πηγαίο κώδικα.

Επιπλέον, η χρήση GitHub Actions επιτρέπει την εκτέλεση εργασιών ανανέωσης δεδομένων είτε χειροκίνητα είτε με προγραμματισμό. Με αυτόν τον τρόπο, η εφαρμογή μπορεί να υποστηρίζει περιοδική ενημέρωση δεδομένων χωρίς να απαιτείται άμεση παρέμβαση στον κώδικα της κύριας εφαρμογής.

5.3 Backend

Το backend της εφαρμογής TransferMind υλοποιήθηκε με Node.js και Express και αποτελεί το κεντρικό επίπεδο διασύνδεσης ανάμεσα στο frontend, τη βάση δεδομένων, τα scripts εισαγωγής δεδομένων και τις αναλυτικές λειτουργίες που εκτελούνται μέσω Python. Η βασική του ευθύνη είναι να δέχεται αιτήματα από το frontend, να ανακτά ή να επεξεργάζεται δεδομένα, να εφαρμόζει την απαραίτητη λογική της εφαρμογής και να επιστρέφει οργανωμένες αποκρίσεις σε μορφή JSON.

Η εφαρμογή χρησιμοποιεί ES modules και το κύριο σημείο εκκίνησης βρίσκεται στο αρχείο `backend/index.js`. Μέσα από αυτό αρχικοποιείται η Express εφαρμογή, ενεργοποιείται το CORS, ορίζεται η υποστήριξη JSON αιτημάτων, προστίθεται endpoint ελέγχου λειτουργίας και προσαρτώνται τα API routes κάτω από το πρόθεμα `/api`. Η χρήση ξεχωριστού backend επιτρέπει στο frontend να μην επικοινωνεί απευθείας με τη βάση δεδομένων, αλλά να λαμβάνει τα δεδομένα μέσω ελεγχόμενων endpoints.

```
1 export const app = express();
2 const port = Number(process.env.PORT || 3001);
3
4 app.use(cors({ origin: true }));
5 app.use(express.json());
6
```

```
app.get("/health", (req, res) => {
  res.status(200).json({ data: { status: "ok" } });
});

app.use("/api", routes);
app.use(handleError);
```

Listing 5.1: Αρχικοποίηση Express εφαρμογής και health endpoint

Το παραπάνω απόσπασμα δείχνει τη βασική αρχικοποίηση του backend. Το endpoint `/health` χρησιμοποιείται για απλό έλεγχο διαθεσιμότητας της υπηρεσίας, ενώ όλα τα κύρια endpoints της εφαρμογής οργανώνονται κάτω από το `/api`. Ο κεντρικός μηχανισμός διαχείρισης σφαλμάτων τοποθετείται στο τέλος, ώστε να μπορεί να χειρίζεται σφάλματα από όλα τα προηγούμενα routes.

Δομή και επίπεδα υλοποίησης

Η υλοποίηση του backend ακολουθεί οργανωμένη δομή φακέλων. Ο φάκελος `api` περιέχει τον ορισμό των routes, οι `controllers` αναλαμβάνουν την επεξεργασία των HTTP αιτημάτων, οι `services` περιλαμβάνουν την κύρια λογική της εφαρμογής, ενώ οι `repositories` απομονώνουν την πρόσβαση στη βάση δεδομένων. Επιπλέον, ο φάκελος `lib` περιλαμβάνει κοινές βοηθητικές λειτουργίες, όπως HTTP utilities, Supabase client και Python clients, ενώ οι φάκελοι `jobs` και `ingestion` χρησιμοποιούνται για εργασίες συλλογής και ανανέωσης δεδομένων.

Η βασική ροή ενός αιτήματος στο backend μπορεί να περιγραφεί ως εξής:

$$\text{Request} \rightarrow \text{Route} \rightarrow \text{Controller} \rightarrow \text{Service} \rightarrow \text{Repository} \rightarrow \text{Database} \rightarrow \text{Response}$$

Η προσέγγιση αυτή βοηθά στον καθαρό διαχωρισμό αρμοδιοτήτων. Τα routes καθορίζουν τη δημόσια επιφάνεια του API, οι controllers παραμένουν απλοί και καλούν τα κατάλληλα services, τα services υλοποιούν τους κανόνες της εφαρμογής και τα repositories εκτελούν τα queries προς τη Supabase/PostgreSQL βάση.

Οργάνωση των API routes

Το αρχείο `backend/api/routes.js` συγκεντρώνει τα διαθέσιμα endpoints του backend. Σε αυτά περιλαμβάνονται endpoints για αναζήτηση ομάδων και παικτών, προβολή ομάδων, παικτών και βαθμολογιών, ανάκτηση δεδομένων σύγκρισης ομάδων, εκτέλεση clustering και παραγωγή κανόνων συσχέτισης. Κάθε route συνδέεται με τον αντίστοιχο controller και τυλίγεται με `asyncHandler`, ώστε τα ασύγχρονα σφάλματα να προωθούνται στον κεντρικό error handler.

```
router.get("/search", asyncHandler(searchController));

router.get("/teams", asyncHandler(listTeamsController));

router.get(
  "/teams/comparison-dataset",
  asyncHandler(getTeamsComparisonDatasetController),
);

router.post(
  "/teams/clustering/run",
  asyncHandler(runTeamClustersController),
);
```

Listing 5.2: Παράδειγμα ορισμού routes στο backend

Το συγκεκριμένο παράδειγμα δείχνει ότι το backend δεν περιέχει τη λογική της εφαρμογής απευθείας μέσα στα routes. Αντίθετα, κάθε route παραπέμπει σε controller, διατηρώντας έτσι το αρχείο δρομολόγησης καθαρό και εύκολο στη συντήρηση.

Controllers

Οι controllers αποτελούν το πρώτο επίπεδο επεξεργασίας ενός HTTP αιτήματος. Ο ρόλος τους είναι να διαβάζουν παραμέτρους από το URL, το query string ή το request body, να τις μετατρέπουν σε κατάλληλη μορφή και να καλούν το αντίστοιχο service. Δεν περιέχουν τη βασική επιχειρησιακή λογική της εφαρμογής, καθώς αυτή βρίσκεται στο service layer.

Χαρακτηριστικό παράδειγμα αποτελεί ο controller για το προφίλ ομάδας. Ο controller διαβάζει το `id` της ομάδας, ελέγχει προαιρετικά το `seasonId` και στη συνέχεια καλεί το service που συνθέτει το πλήρες προφίλ της ομάδας.

```
export async function getTeamProfileController(req, res) {
  const id = parseInteger(req.params.id, "id");
  const seasonId =
    req.query.seasonId !== undefined
      ? parseInteger(req.query.seasonId, "seasonId")
      : undefined;

  const profile = await getTeamProfile(id, seasonId);

  res.status(200).json({ data: profile });
}
```

Listing 5.3: Controller για το προφίλ ομάδας

Η μορφή της απάντησης ακολουθεί το σχήμα `{ data: ... }`, το οποίο χρησιμοποιείται σε πολλά endpoints της εφαρμογής. Με αυτόν τον τρόπο το frontend μπορεί να χειρίζεται τις απόκρισεις με πιο συνεπή τρόπο.

Service layer

Το service layer είναι υπεύθυνο για την κύρια λογική της εφαρμογής. Σε αυτό το επίπεδο γίνεται η σύνθεση δεδομένων από διαφορετικές πηγές, η επικύρωση των επιλογών του χρήστη και η μορφοποίηση των δεδομένων σε μορφή κατάλληλη για το frontend. Για παράδειγμα, το service που δημιουργεί το προφίλ μιας ομάδας δεν επιστρέφει απλώς μία εγγραφή από τη βάση, αλλά συνδυάζει πληροφορίες ομάδας, διαθέσιμες σεζόν, ρόστερ, στατιστικά και βαθμολογική εικόνα.

```
const [squad, stats, standingsRows] = await Promise.all([
  selectedSeason
    ? getTeamSquad(id, selectedSeason.season_id)
    : getTeamSquad(id),
  selectedSeason ? getOptionalTeamStats(id, selectedSeason.season_id)
   : null,
  selectedSeason
    ? getOptionalStandings(team?.tournament_id,
   selectedSeason.season_id)
    : [],
]);
```

Listing 5.4: Παράλληλη ανάκτηση δεδομένων για το προφίλ ομάδας

Το παραπάνω απόσπασμα δείχνει ότι το backend ανακτά παράλληλα τα δεδομένα που χρειάζονται για το προφίλ ομάδας. Η χρήση του `Promise.all` επιτρέπει τη μείωση του συνολικού χρόνου αναμονής, καθώς το ρόστερ, τα στατιστικά και οι βαθμολογίες μπορούν να ανακτηθούν ανεξάρτητα.

Στο ίδιο επίπεδο υλοποιούνται και λειτουργίες που σχετίζονται με τη σύγκριση ομάδων και την ανάλυση δεδομένων. Για παράδειγμα, το backend ελέγχει τις επιλεγμένες ομάδες, σεζόν και στατιστικά, δημιουργεί πίνακες χαρακτηριστικών, κανονικοποιεί τιμές όπου χρειάζεται και προετοιμάζει τα δεδομένα πριν σταλούν στα Python scripts.

Repository layer και πρόσβαση στη βάση

Η πρόσβαση στη βάση δεδομένων έχει απομονωθεί στο repository layer. Τα repositories περιέχουν τα Supabase queries και είναι υπεύθυνα για την ανάκτηση, το φιλτράρισμα και τη χαρτογράφηση των δεδομένων. Με αυτόν τον τρόπο, τα services δεν χρειάζεται να γνωρίζουν τις λεπτομέρειες κάθε query, ενώ το frontend δεν έχει καμία άμεση πρόσβαση στη βάση.

Η σύνδεση με τη Supabase δημιουργείται κεντρικά στο `backend/lib/supabaseClient.js`. Για τη σύνδεση χρησιμοποιούνται οι μεταβλητές περιβάλλοντος `SUPABASE_URL` και `SUPABASE_SERVICE_KEY`. Η χρήση service key σημαίνει ότι η πρόσβαση αυτή πρέπει να παραμένει αποκλειστικά στο backend και να μην εκτίθεται ποτέ στον browser.

```
import { createClient } from "@supabase/supabase-js";
import "./env.js";
import { getRequiredEnv } from "./env.js";

const supabaseUrl = getRequiredEnv("SUPABASE_URL");
const supabaseKey = getRequiredEnv("SUPABASE_SERVICE_KEY");

export const supabase = createClient(supabaseUrl, supabaseKey);
```

Listing 5.5: Αρχικοποίηση Supabase client στο backend

Τα repositories αναλαμβάνουν επίσης τη διαχείριση των εσωτερικών ids της βάσης και των εξωτερικών `api_id` που προέρχονται από την πηγή δεδομένων. Αυτός ο διαχωρισμός είναι σημαντικός, καθώς τα εξωτερικά ids χρησιμοποιούνται κυρίως για εισαγωγή και ενημέρωση δεδομένων, ενώ το frontend εργάζεται με ελεγχόμενες αποκρίσεις του backend.

```
const { data, error } = await supabase
  .from("seasons")
  .select("id, api_id")
  .eq("id", seasonId)
  .eq("tournament_id", tournament.api_id)
  .maybeSingle();
```

Listing 5.6: Παράδειγμα repository query με χρήση Supabase

Το παράδειγμα δείχνει ότι το repository μπορεί να λαμβάνει εσωτερικά ids από το backend, αλλά να χρησιμοποιεί εξωτερικά `api_id` όπου αυτό απαιτείται από το σχήμα της βάσης ή από τις σχέσεις των δεδομένων. Έτσι, η πολυπλοκότητα της αντιστοίχισης παραμένει στο backend.

Διαχείριση σφαλμάτων

Η διαχείριση σφαλμάτων γίνεται κεντρικά μέσω βοηθητικών συναρτήσεων στο `backend/lib/http.js`. Η χρήση του `asyncHandler` επιτρέπει τη μεταφορά σφαλμάτων από ασύγχρονες λειτουργίες στον κεντρικό error handler, χωρίς να απαιτείται επαναλαμβανόμενος κώδικας `try/catch` σε κάθε route. Παράλληλα, η χρήση ειδικών HTTP errors βοηθά στην επιστροφή κατάλληλων status codes και μηνυμάτων προς το frontend.

Η κεντρική διαχείριση σφαλμάτων κάνει το backend πιο συνεπές, επειδή όλα τα σφάλματα μπορούν να επιστρέφονται με παρόμοια μορφή. Αυτό διευκολύνει και το frontend, καθώς μπορεί να εμφανίζει μηνύματα αποτυχίας με πιο προβλέψιμο τρόπο.

Διασύνδεση με Python scripts

Οι αναλυτικές λειτουργίες του TransferMind, όπως το K-Means clustering, η ιεραρχική συσταδοποίηση και οι κανόνες Apriori, εκτελούνται σε Python. Το backend δεν υλοποιεί απευθείας τους αλγορίθμους, αλλά λειτουργεί ως ενδιάμεσο επίπεδο που προετοιμάζει τα δεδομένα, καλεί τα αντίστοιχα Python scripts και επιστρέφει τα αποτελέσματα στο frontend.

Η επικοινωνία Node.js και Python γίνεται μέσω ξεχωριστών διεργασιών. Το backend χρησιμοποιεί `child_process.spawn`, στέλνει τα δεδομένα εισόδου σε μορφή JSON μέσω `stdin` και λαμβάνει τα αποτελέσματα μέσω `stdout`. Με αυτόν τον τρόπο, τα Python modules παραμένουν απομονωμένα από το HTTP layer.

```
export function runPythonApriori(payload, options = {}) {
  const timeoutMs = options.timeoutMs ?? DEFAULT_TIMEOUT_MS;

  return new Promise((resolveResult, reject) => {
    const child = spawn(PYTHON_BIN, [RUNNER_PATH], {
      stdio: ["pipe", "pipe", "pipe"],
    });
```

Listing 5.7: Εκτέλεση Python runner από το backend

Η προσέγγιση αυτή επιτρέπει στο backend να διατηρεί τον έλεγχο της επικύρωσης, των timeouts και της μορφοποίησης της τελικής απόκρισης, ενώ η υπολογιστική επεξεργασία εκτελείται σε περιβάλλον Python. Επομένως, κάθε τεχνολογία χρησιμοποιείται στο σημείο όπου είναι πιο κατάλληλη: το Node.js για το API και η Python για τις αναλυτικές εργασίες.

Jobs και μηχανισμοί ενημέρωσης δεδομένων

Εκτός από τα endpoints που εξυπηρετούν άμεσα αιτήματα χρηστών, το backend περιλαμβάνει scripts και jobs για την εισαγωγή και ανανέωση ποδοσφαιρικών δεδομένων. Ο φάκελος `backend/ingestion` περιλαμβάνει διαδικασίες αρχικής εισαγωγής, όπως διοργανώσεις, σεζόν, βαθμολογίες, ομάδες, λογότυπα, παίκτες και στατιστικά. Ο φάκελος `backend/jobs` περιλαμβάνει εργασίες ανανέωσης, όπως εβδομαδιαίο refresh, μηνιαία ενημέρωση στατιστικών παικτών και ετήσια αλλαγή σεζόν.

Οι εργασίες αυτές είναι διαχωρισμένες από το κανονικό request handling του API. Αυτό σημαίνει ότι οι πιο βαριές διαδικασίες συλλογής και ανανέωσης δεδομένων δεν εκτελούνται τη στιγμή που ο χρήστης αλληλεπιδρά με την εφαρμογή. Αντίθετα, μπορούν να εκτελούνται χειροκίνητα ή προγραμματισμένα, μέσω scripts και GitHub Actions. Ο διαχωρισμός αυτός συμβάλλει στη σταθερότητα του backend και στη διατήρηση καλύτερης απόκρισης για τις βασικές λειτουργίες της εφαρμογής.

Συνολικά, η υλοποίηση του backend στο TransferMind ακολουθεί μία οργανωμένη και επεκτάσιμη δομή. Ο διαχωρισμός σε routes, controllers, services και repositories επιτρέπει καλύτερη

συντήρηση του κώδικα, ενώ η χρήση ξεχωριστών Python runners και ingestion jobs επιτρέπει την υποστήριξη πιο σύνθετων λειτουργιών χωρίς να επιβαρύνεται η βασική λειτουργία του API.

5.4 Python ML Modules

Στο TransferMind, τα `Python ML Modules` αποτελούν το αναλυτικό και υπολογιστικό επίπεδο του συστήματος. Η Python δεν χρησιμοποιείται για routing, για άμεση επικοινωνία με το frontend ή για πρόσβαση στη βάση δεδομένων. Αυτές οι λειτουργίες παραμένουν στο Express backend και στο React frontend. Το frontend παρέχει τα εργαλεία επιλογής ομάδων, σεζόν, στατιστικών, αλγορίθμων και παραμέτρων, ενώ το backend αναλαμβάνει την επικύρωση των αιτημάτων, την ανάκτηση των δεδομένων και τη διαμόρφωση των εισόδων. Η Python χρησιμοποιείται αποκλειστικά για την εκτέλεση των αλγορίθμων μηχανικής μάθησης και επιστρέφει τα αποτελέσματα σε μορφή συμβατή με JSON μέσω `stdout`.

Η γενική ροή ξεκινά από τη σελίδα `Teams Comparison`. Ο χρήστης επιλέγει τις ομάδες ή τις εγγραφές ομάδας-σεζόν που θέλει να αναλύσει, τα διαθέσιμα στατιστικά πεδία και τις παραμέτρους του αντίστοιχου αλγορίθμου. Για το clustering χρησιμοποιούνται endpoints όπως `POST /api/teams/clustering/elbow`, `POST /api/teams/clustering/run` και `POST /api/teams/clustering/agglomerative/run`. Για το Association Rules Mining χρησιμοποιείται το endpoint `POST /api/team-season-stats/association-rules`. Τα αιτήματα έχουν εσωτερικά αναγνωριστικά, όπως `teamId`, `tournamentId` και `seasonId`, καθώς και επιλεγμένα `statKeys` και παραμέτρους όπως `k`, `linkage`, `minSupport`, `minConfidence` και `minLift`.

Το backend λειτουργεί ως ενδιάμεσο επίπεδο μεταξύ του frontend και των Python modules. Αρχικά ελέγχει την εγκυρότητα των παραμέτρων, στη συνέχεια ανακτά τα απαραίτητα στατιστικά από το repository layer και δημιουργεί την κατάλληλη είσοδο για τον αλγόριθμο. Στην περίπτωση του clustering, η είσοδος είναι ένας αριθμητικός πίνακας, όπου κάθε γραμμή αντιστοιχεί σε ένα team-season entry και κάθε στήλη σε ένα επιλεγμένο στατιστικό. Στην περίπτωση του Apriori, η είσοδος είναι ένα σύνολο transactions, όπου κάθε transaction περιγράφει την απόδοση μίας ομάδας σε μία συγκεκριμένη σεζόν μέσα από κατηγορικά items. Στη συνέχεια, το backend καλεί τον αντίστοιχο Python runner, περνώντας τα δεδομένα ως JSON μέσω `stdin`, και διαβάζει το αποτέλεσμα από το `stdout`. Το αποτέλεσμα εμπλουτίζεται με metadata, raw values, normalized values και warnings πριν επιστραφεί στο frontend.

Βασικό στάδιο πριν από το clustering είναι η κανονικοποίηση των δεδομένων. Στο TransferMind χρησιμοποιείται Min-Max normalization και όχι `StandardScaler`. Κάθε επιλεγμένο στατιστικό μετατρέπεται στο διάστημα 0–1, επειδή οι ποδοσφαιρικές μετρικές έχουν διαφορετικές κλίμακες. Για παράδειγμα, τα γκολ, τα σουτ, η κατοχή μπάλας, η ακρίβεια πάσας, οι κάρτες και τα γκολ κατά δεν μπορούν να συγκριθούν άμεσα χωρίς προεπεξεργασία. Η κανονικοποίηση επιτρέπει στους αλγορίθμους να αντιμετωπίζουν όλα τα χαρακτηριστικά με συγκρίσιμο βάρος.

Επιπλέον, το σύστημα λαμβάνει υπόψη στατιστικά αρνητικής κατεύθυνσης. Σε τέτοια πεδία, όπως `goals_conceded`, `fouls`, `yellowcards`, `redcards`, `errors_to_goals` και γενικότερα στατιστικά τύπου "against", η μεγαλύτερη αρχική τιμή δεν σημαίνει καλύτερη απόδοση. Για τον λόγο αυτό, μετά την Min-Max κανονικοποίηση οι τιμές αντιστρέφονται, ώστε μία υψηλότερη normalized τιμή να σημαίνει σταθερά καλύτερη αγωνιστική εικόνα. Αυτό είναι σημαντικό, επειδή επιτρέπει στα clustering modules και στο Apriori module να ερμηνεύουν ομοιόμορφα όλα τα στατιστικά.

```
def require_number_matrix(value):
    if not isinstance(value, list) or len(value) == 0:
        raise ValueError("points must be a non-empty matrix.")
    if len(value) < 3:
        raise ValueError("points must include at least three rows.")

    matrix = pd.DataFrame(value)
    numeric_matrix = matrix.apply(pd.to_numeric, errors="coerce")
    values = numeric_matrix.to_numpy(dtype=float)

    if not np.isfinite(values).all():
        raise ValueError("points must contain only finite numeric
   values.")
    if (values < 0).any() or (values > 1).any():
        raise ValueError("points must contain normalized values
   between 0 and 1.")

    return numeric_matrix
```

Listing 5.8: Έλεγχος κανονικοποιημένου πίνακα στον Agglomerative runner

Το απόσπασμα 5.8 δείχνει τον έλεγχο ποιότητας που εκτελείται πριν από την ανάλυση. Η αρχική Min-Max κανονικοποίηση πραγματοποιείται στο backend, όμως ο Python runner επιβεβαιώνει ότι τα δεδομένα που λαμβάνει είναι αριθμητικά, πεπερασμένα και βρίσκονται στο αναμενόμενο εύρος 0–1. Με αυτόν τον τρόπο αποφεύγεται η εκτέλεση αλγορίθμων πάνω σε λανθασμένα ή ελλιπή δεδομένα.

Το module `kmeans_runner.py` χρησιμοποιεί τον αλγόριθμο `KMeans` της python βιβλιοθήκης `scikit-learn`. Κάθε εγγραφή ομάδας-σεζόν μετατρέπεται σε διάνυσμα χαρακτηριστικών, με βάση τα στατιστικά που έχει επιλέξει ο χρήστης. Ο στόχος του KMeans είναι να ομαδοποιήσει ομάδες ή team-season entries με παρόμοιο στατιστικό προφίλ. Έτσι, ο χρήστης μπορεί να εντοπίσει ομάδες που παρουσιάζουν κοινά χαρακτηριστικά, ακόμη και αν δεν ανήκουν στην ίδια διοργάνωση ή δεν βρίσκονται στην ίδια θέση βαθμολογίας.

Το KMeans module υποστηρίζει δύο βασικές λειτουργίες. Η πρώτη είναι η λειτουργία `elbow`, όπου ο αλγόριθμος εκτελείται για πολλαπλές τιμές του `k`. Για κάθε τιμή του `k` επιστρέφεται η τιμή `inertia`, δηλαδή το άθροισμα των εσωτερικών αποστάσεων των σημείων από το κέντρο του cluster στο οποίο ανήκουν. Η δεύτερη λειτουργία είναι η τελική ομαδοποίηση, όπου ο χρήστης δίνει συγκεκριμένο `k` και το σύστημα επιστρέφει assignments, centroids, inertia και

αριθμό επαναλήψεων.

```
def run(payload):
    mode = payload.get("mode")
    points = require_number_matrix(payload.get("points"))
    random_state = require_integer(payload.get("randomState", 42), "randomState")
    max_iter = require_integer(payload.get("maxIter", 100), "maxIter", minimum=1)

    if mode == "elbow":
        max_k = require_integer(payload.get("maxK"), "maxK",
                                minimum=1, maximum=len(points))
        elbow = []
        for k in range(1, max_k + 1):
            model = fit_kmeans(KMeans, points, k, random_state, max_iter)
            elbow.append({
                "k": k,
                "inertia": float(model.inertia_),
                "iterations": int(model.n_iter_)
            })
        return {"elbow": elbow}

    if mode == "cluster":
        k = require_integer(payload.get("k"), "k", minimum=2, maximum=len(points))
        model = fit_kmeans(KMeans, points, k, random_state, max_iter)
        return {
            "assignments": [int(x) for x in model.labels_.tolist()],
            "centroids": model.cluster_centers_.astype(float).tolist(),
            "inertia": float(model.inertia_),
            "iterations": int(model.n_iter_)
        }
```

Listing 5.9: Λειτουργίες elbow και τελικού KMeans clustering

Το απόσπασμα 5.9 παρουσιάζει τη βασική ροή του KMeans runner. Στη λειτουργία elbow, ο runner επιστρέφει μία λίστα τιμών `k` και `inertia`, ώστε το frontend να δημιουργήσει την καμπύλη elbow. Στη λειτουργία `cluster`, επιστρέφονται τα τελικά clusters και τα κέντρα τους. Το backend στη συνέχεια μετατρέπει τα labels σε πιο φιλικά cluster ids και εμπλουτίζει το αποτέλεσμα με raw και normalized στατιστικά, ώστε το frontend να μπορεί να δημιουργήσει πίνακες, cluster summaries και γραφήματα.

Η καμπύλη elbow βοηθά τον χρήστη να επιλέξει έναν λογικό αριθμό clusters. Όσο αυξάνεται το `k`, το inertia μειώνεται, επειδή τα σημεία μοιράζονται σε περισσότερες ομάδες. Ωστόσο, μετά από ένα σημείο, η βελτίωση γίνεται μικρότερη. Το σημείο αυτό είναι το λεγόμενο “elbow”. Στο

TransferMind, εκτός από την προβολή της καμπύλης, το backend υπολογίζει ένα `suggestedK`, εντοπίζοντας το σημείο με τη μεγαλύτερη απόσταση από την ευθεία που ενώνει το πρώτο και το τελευταίο σημείο της καμπύλης. Η τελική επιλογή, όμως, παραμένει στον χρήστη.

Το δεύτερο clustering module αφορά την ιεραρχική ομαδοποίηση. Το module `agglomerative_runner.py` δεν βασίζεται στο `sklearn AgglomerativeClustering`, αλλά στη βιβλιοθήκη `scipy.cluster.hierarchy`. Πιο συγκεκριμένα, χρησιμοποιεί τις συναρτήσεις `linkage`, `fcluster` και `dendrogram`. Η μέθοδος αυτή ξεκινά θεωρώντας κάθε team-season entry ως ξεχωριστό cluster και στη συνέχεια συγχωνεύει σταδιακά τα πιο κοντινά clusters, μέχρι να δημιουργηθεί μία πλήρης ιεραρχία. Το `k` χρησιμοποιείται για να κοπεί αυτή η ιεραρχία σε συγκεκριμένο αριθμό ομάδων.

Το Agglomerative module υποστηρίζει διαφορετικές linkage methods, όπως `ward`, `complete`, `average` και `single`. Η επιλογή linkage επηρεάζει τον τρόπο με τον οποίο υπολογίζεται η απόσταση ανάμεσα σε clusters. Για παράδειγμα, η `ward` προσπαθεί να ελαχιστοποιήσει την εσωτερική διακύμανση, ενώ άλλες μέθοδοι χρησιμοποιούν διαφορετικούς ορισμούς απόστασης ανάμεσα στα σημεία των clusters.

```
def run(payload):
    points = require_number_matrix(payload.get("points"))
    k = require_integer(payload.get("k"), "k", minimum=2,
   maximum=len(points))
    linkage_method = require_linkage(payload.get("linkage"))
    labels = require_labels(payload.get("labels"), len(points))

    point_values = points.to_numpy(dtype=float)
    linkage_matrix = linkage(point_values, method=linkage_method)
    cluster_labels = fcluster(linkage_matrix, t=k,
   criterion="maxclust")
    assignments = to_zero_based_labels(cluster_labels)

    return {
        "assignments": assignments,
        "dendrogramSvg": build_dendrogram_svg(
            linkage_matrix, labels, linkage_method, k
        ),
        "linkageMatrix": linkage_matrix.tolist(),
        "warnings": []
    }
```

Listing 5.10: Agglomerative clustering με linkage matrix και δενδρόγραμμα

Το απόσπασμα 5.10 δείχνει ότι το αποτέλεσμα της ιεραρχικής ομαδοποίησης δεν περιορίζεται μόνο στα τελικά cluster assignments. Ο runner επιστρέφει επίσης το `linkageMatrix` και ένα `dendrogramSvg`. Το δενδρόγραμμα παράγεται με τη βοήθεια της `matplotlib` και επιστρέφεται ως SVG string. Στο frontend εμφανίζεται ως εικόνα, προσφέροντας μία οπτική αναπαράσταση της διαδικασίας συγχώνευσης. Αυτό είναι ιδιαίτερα χρήσιμο, επειδή ο χρήστης

μπορεί να δει όχι μόνο ποιες ομάδες καταλήγουν μαζί, αλλά και σε ποιο επίπεδο ομοιότητας συγχωνεύονται.

Το τρίτο Python module αφορά το Apriori και την εξόρυξη κανόνων συσχέτισης. Σε αντίθεση με το clustering, το Apriori δεν χρησιμοποιεί συνεχείς αριθμητικούς πίνακες. Το backend μετατρέπει κάθε team-season entry σε transaction, δηλαδή σε ένα σύνολο από items. Για κάθε επιλεγμένο στατιστικό δημιουργείται κατηγορία τύπου `low`, `medium` ή `high`, με βάση τις direction-adjusted normalized τιμές. Η διακριτοποίηση πραγματοποιείται με όρια 1/3 και 2/3. Έτσι, οι συνεχείς τιμές μετατρέπονται σε κατηγορικές περιγραφές που μπορούν να χρησιμοποιηθούν από τον Apriori.

Η διαδικασία αυτή επιτρέπει στο σύστημα να εντοπίζει συχνά μοτίβα απόδοσης. Για παράδειγμα, μπορεί να εξετάσει αν ομάδες με υψηλή επιθετική παραγωγή εμφανίζουν συχνά και υψηλή ακρίβεια πάσας ή αν συγκεκριμένοι συνδυασμοί αμυντικών στατιστικών συνδέονται με άλλα χαρακτηριστικά. Ο στόχος δεν είναι η πρόβλεψη, αλλά η ανάδειξη σχέσεων που εμφανίζονται συχνά μέσα στο επιλεγμένο σύνολο ομάδων και σεζόν.

```
def generate_rules(frequent_itemsets, min_confidence, min_lift):
    if frequent_itemsets.empty:
        return []

    rules_frame = association_rules(
        frequent_itemsets,
        metric="confidence",
        min_threshold=min_confidence
    )

    if rules_frame.empty:
        return []

    rules = []
    for _, row in rules_frame.iterrows():
        lift = float(row["lift"])
        if min_lift is not None and lift < min_lift:
            continue

        rules.append({
            "antecedents": sorted(row["antecedents"]),
            "consequents": sorted(row["consequents"]),
            "support": float(row["support"]),
            "confidence": float(row["confidence"]),
            "lift": lift
        })

    return sorted(
        rules,
        key=lambda r: (-r["lift"], -r["confidence"], -r["support"])
```

```
31      )
```

Listing 5.11: Παραγωγή association rules με confidence και lift filtering

Το απόσπασμα 5.11 δείχνει τη δημιουργία κανόνων από τα συχνά itemsets. Το `support` δείχνει πόσο συχνά εμφανίζεται ένα μοτίβο στο επιλεγμένο dataset. Το `confidence` δείχνει πόσο συχνά εμφανίζεται το consequent όταν εμφανίζεται το antecedent. Το `lift` δείχνει αν η σχέση είναι ισχυρότερη από μία τυχαία συνύπαρξη. Οι παράμετροι `minSupport`, `minConfidence` και `minLift` επιτρέπουν στον χρήστη να ελέγχει την αυστηρότητα της εξόρυξης κανόνων. Όσο αυξάνονται οι τιμές αυτές, τόσο λιγότεροι αλλά ισχυρότεροι κανόνες επιστρέφονται.

Στο TransferMind, το `minLift` είναι προαιρετικό στο frontend. Όταν δεν δίνεται ή είναι κενό, το backend το μετατρέπει σε προεπιλεγμένη τιμή 1.01, ώστε να αποφεύγονται κανόνες που δεν είναι ισχυρότεροι από τυχαία συνύπαρξη. Αν δεν βρεθούν κανόνες, ο Python runner δεν επιστρέφει σφάλμα, αλλά κενή λίστα `rules: []`. Με αυτόν τον τρόπο, το frontend μπορεί να εμφανίσει κατάλληλο empty state και να ενημερώσει τον χρήστη ότι οι επιλεγμένες παράμετροι ήταν πιθανώς πολύ αυστηρές ή ότι τα δεδομένα δεν περιέχουν αρκετά επαναλαμβανόμενα μοτίβα.

Η υλοποίηση δίνει ιδιαίτερη έμφαση στην επικύρωση των δεδομένων. Το backend απορρίπτει ανεπαρκείς επιλογές entries ή στατιστικών, άκυρα ids, άκυρες τιμές `k`, `k` μεγαλύτερο από τον αριθμό των διαθέσιμων σημείων, άκυρο `linkage` και thresholds εκτός αποδεκτών ορίων. Αντίστοιχα, οι Python runners ελέγχουν για κενά matrices, μη αριθμητικές τιμές, `NaN`, τιμές εκτός του διαστήματος 0–1, λάθος πλήθος labels και κενά transactions. Τα warnings χρησιμοποιούνται για περιπτώσεις που δεν απαιτούν υποχρεωτική διακοπή, όπως σταθερές στήλες ή ελλιπείς τιμές που μπορούν να αντιμετωπιστούν με ασφαλή τρόπο.

Ο διαχωρισμός των Python ML modules από το υπόλοιπο σύστημα προσφέρει σημαντικά αρχιτεκτονικά πλεονεκτήματα. Το frontend παραμένει υπεύθυνο για την εμπειρία χρήστη και τις οπτικοποιήσεις, το backend για τα API contracts, την επικύρωση και την πρόσβαση στα δεδομένα, ενώ η Python επικεντρώνεται στους αριθμητικούς και αλγοριθμικούς υπολογισμούς. Έτσι, το σύστημα αξιοποιεί το οικοσύστημα της Python για μηχανική μάθηση και στατιστική ανάλυση, χωρίς να μεταφέρει λογική παρουσίασης ή routing στους runners.

Τα αποτελέσματα των Python modules παρουσιάζονται στο frontend με διαφορετικούς τρόπους. Το KMeans εμφανίζεται μέσα από elbow curve, cluster assignment tables, cluster membership summaries, average cluster profiles και parallel coordinates plots. Το Agglomerative clustering προσθέτει το δενδρόγραμμα, το οποίο βοηθά στην κατανόηση της ιεραρχικής σχέσης ανάμεσα στις ομάδες. Το Apriori εμφανίζει πίνακα κανόνων συσχέτισης και γραφήματα των ισχυρότερων κανόνων με βάση το `lift`. Συνολικά, τα Python ML modules μετατρέπουν τα στατιστικά δεδομένα των ομάδων σε αναλυτικά αποτελέσματα που μπορούν να ερμηνευτούν οπτικά και να αξιοποιηθούν από τον τελικό χρήστη.

5.5 Frontend

Το frontend του TransferMind υλοποιήθηκε ως εφαρμογή React με TypeScript και Vite, ακολουθώντας τη λογική μίας σύγχρονης single-page application. Ο βασικός στόχος της διεπαφής ήταν να παρουσιάζει τα ποδοσφαιρικά δεδομένα με τρόπο κατανοητό, οργανωμένο και διαδραστικό, ώστε ο χρήστης να μπορεί να αναζητά ομάδες και παίκτες, να εξετάζει στατιστικά, να συγκρίνει επιδόσεις και να αξιοποιεί αποτελέσματα αναλυτικών αλγορίθμων, όπως το clustering και η εξόρυξη κανόνων συσχέτισης. Επομένως, το frontend δεν λειτουργεί απλώς ως ένα επίπεδο προβολής δεδομένων, αλλά ως το βασικό περιβάλλον αλληλεπίδρασης ανάμεσα στον χρήστη, τα δεδομένα και τα αποτελέσματα που παράγονται από το backend.

Η οργάνωση της εφαρμογής βασίζεται σε καθαρό διαχωρισμό ανάμεσα σε σελίδες, επαναχρησιμοποιήσιμα components, βοηθητικές συναρτήσεις και τύπους δεδομένων. Οι βασικές σελίδες βρίσκονται στον φάκελο `frontend/src/pages` και περιλαμβάνουν την αρχική σελίδα `Home`, τις βαθμολογίες `Standings`, τη λίστα ομάδων `Teams`, τη λίστα παικτών `Players`, το προφίλ ομάδας `TeamProfile`, το προφίλ παίκτη `PlayerProfile` και τέλος τη σελίδα `TeamsComparison`. Η πλοήγηση υλοποιείται με το `react-router-dom`, επιτρέποντας την εναλλαγή σελίδων χωρίς πλήρη επαναφόρτωση της εφαρμογής. Το σταθερό `Navbar` παρέχει πρόσβαση στις κύριες ενότητες και διατηρεί κοινή εμπειρία πλοήγησης σε όλο το περιβάλλον.

Η component-based αρχιτεκτονική αποτελεί βασικό στοιχείο της υλοποίησης. Αντί κάθε σελίδα να περιέχει όλη τη λογική και τη δομή της διεπαφής, η εφαρμογή χωρίζεται σε μικρότερα και επαναχρησιμοποιήσιμα τμήματα. Components όπως τα `PageShell`, `PageHeader`, `SearchInput`, `SegmentedTabs`, `StandingsTable`, `PlayerRosterTable`, `TeamHeader` και `TeamStatsPanel` χρησιμοποιούνται για την ομοιομορφία της διεπαφής, τη μείωση επαναλαμβανόμενου κώδικα και την ευκολότερη συντήρηση. Για παράδειγμα, το `SegmentedTabs` μπορεί να χρησιμοποιηθεί σε διαφορετικά σημεία της εφαρμογής, όπως σε επιλογές σταδίων βαθμολογίας, κατηγορίες στατιστικών ή tabs αναλυτικών εργαλείων.

Η αρχική σελίδα `Home` λειτουργεί ως σημείο εισόδου στην εφαρμογή. Μέσω της αναζήτησης, ο χρήστης μπορεί να εντοπίσει ομάδες ή παίκτες και να μεταβεί στα αντίστοιχα προφίλ. Η σελίδα `Standings` παρουσιάζει βαθμολογίες με φίλτρα διοργάνωσης, σεζόν και stage, επιτρέποντας την εξερεύνηση της κατάταξης ομάδων σε διαφορετικά αγωνιστικά πλαίσια. Η σελίδα `Teams` οργανώνει τις ομάδες με φίλτρα και αναζήτηση, ενώ η σελίδα `Players` επιτρέπει την προβολή παικτών με βάση την ομάδα και υποστηρίζει σταδιακή φόρτωση αποτελεσμάτων.

Ιδιαίτερη σημασία έχει το `TeamProfile`, καθώς συγκεντρώνει πληροφορίες για μία ομάδα, διαθέσιμες σεζόν, στατιστικά, σύντομη βαθμολογία και ρόστερ. Το `TeamStatsPanel` παρουσιάζει τις επιδόσεις της ομάδας σε κατηγορίες όπως επίθεση, πάσες, άμυνα και πειθαρχία, δίνοντας τη δυνατότητα σύγκρισης με τις υπόλοιπες ομάδες της ίδιας διοργάνωσης και σεζόν. Αντίστοιχα, το `PlayerProfile` εμφανίζει βασικά στοιχεία παίκτη και χωρίζει τα στατιστικά ανάλογα με τον ρόλο του, για παράδειγμα σε common, outfield ή goalkeeper sections.

Η πιο σύνθετη αναλυτική ενότητα είναι η `TeamsComparison`. Σε αυτήν ενσωματώνονται οι βασικές λειτουργίες σύγκρισης και ανάλυσης της εφαρμογής. Το `Custom Comparison` επιτρέπει στον χρήστη να επιλέξει ομάδες, σεζόν και στατιστικά, ώστε να συγκρίνει raw values και

relative scores. Το `Cluster Analysis` υποστηρίζει τεχνικές όπως k-means και agglomerative clustering, παρουσιάζοντας elbow curve, cluster profiles, parallel coordinates plot και σύνοψη συμμετοχής ομάδων στα clusters. Το `Association Rules Mining` επιτρέπει την επιλογή team-season entries, στατιστικών και thresholds, όπως min support, min confidence και min lift, ώστε να παραχθούν κανόνες συσχέτισης με βάση τις επιδόσεις των ομάδων.

Οι οπτικοποιήσεις αποτελούν κρίσιμο μέρος του frontend, επειδή μετατρέπουν σύνθετα στατιστικά αποτελέσματα σε ευκολότερα κατανοητές αναπαραστάσεις. Η εφαρμογή χρησιμοποιεί τη βιβλιοθήκη Recharts για γραφήματα όπως `BarChart`, `LineChart` και `RadarChart`. Στο cluster analysis, για παράδειγμα, η elbow curve βοηθά στην επιλογή αριθμού clusters, ενώ τα average cluster profiles και το parallel coordinates plot δείχνουν τις διαφορές ανάμεσα στις ομάδες ή στις ομάδες-σεζόν. Παράλληλα, ορισμένες οπτικοποιήσεις, όπως το ranking chart στο `TeamStatsPanel`, υλοποιούνται με custom HTML και CSS, ώστε να προσαρμόζονται καλύτερα στο περιεχόμενο της εφαρμογής.

Η επικοινωνία με το backend συγκεντρώνεται στο αρχείο `frontend/src/api/api.ts`. Το frontend δεν συνδέεται απευθείας με τη βάση δεδομένων, αλλά χρησιμοποιεί typed API calls προς το backend. Με αυτόν τον τρόπο διαχωρίζεται καθαρά η ευθύνη της διεπαφής από την επιχειρησιακή λογική και την πρόσβαση στα δεδομένα. Η χρήση TypeScript types για ομάδες, παίκτες, προφίλ, βαθμολογίες, clustering payloads και association rules μειώνει την πιθανότητα λαθών και κάνει πιο σαφές ποια δεδομένα αναμένει κάθε component. Συνολικά, το frontend του TransferMind λειτουργεί ως οργανωμένο και διαδραστικό επίπεδο παρουσίασης, το οποίο συνδέει τα δεδομένα, τα αναλυτικά μοντέλα και τον τελικό χρήστη μέσα από μία ενιαία εφαρμογή.

Ενδεικτικά αποσπάσματα κώδικα

```
function App() {
  return (
    <Router>
      <Navbar />
      <main>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/standings" element={<Standings />} />
          <Route path="/teams" element={<Teams />} />
          <Route path="/players" element={<Players />} />
          <Route path="/team/:id" element={<TeamProfile />} />
          <Route path="/player/:id" element={<PlayerProfile />} />
          <Route path="/teams-comparison" element={<TeamsComparison
   />} />
        </Routes>
      </main>
    </Router>
  );
```

```
}
```

Listing 5.12: Οργάνωση των βασικών routes της εφαρμογής

Το απόσπασμα 5.12 παρουσιάζει την κεντρική οργάνωση της πλοήγησης στο frontend. Κάθε βασική λειτουργία του TransferMind αντιστοιχίζεται σε ξεχωριστό route, ενώ τα δυναμικά paths `/team/:id` και `/player/:id` επιτρέπουν την προβολή συγκεκριμένων προφίλ ομάδας και παίκτη.

```
export type SegmentedTabItem<T extends string | number> = {
  value: T;
  label: ReactNode;
  disabled?: boolean;
};

function SegmentedTabs<T extends string | number>({
  items,
  value,
  onChange,
}: SegmentedTabsProps<T>) {
  return (
    <div>
      {items.map((item) => (
        <TabButton
          key={item.value}
          active={value === item.value}
          onClick={() => onChange(item.value)}
          disabled={item.disabled}
        >
          {item.label}
        </TabButton>
      ))}
    </div>
  );
}
```

Listing 5.13: Επαναχρησιμοποιήσιμο component για segmented tabs

Το απόσπασμα 5.13 δείχνει τη λογική επαναχρησιμοποίησης components στο frontend. Στο παράδειγμά μας, το `SegmentedTabs` είναι generic component και μπορεί να χρησιμοποιηθεί σε διαφορετικά σημεία της εφαρμογής, όπως σε tabs, φίλτρα, κατηγορίες στατιστικών και επιλογές ανάλυσης.

```
<ResponsiveContainer width="100%" height="100%">
  <LineChart data={elbowResult.elbow}>
    <XAxis dataKey="k" allowDecimals={false} />
    <YAxis label={{ value: "Inertia / WCSS", angle: -90 }} />
```

```
    <Tooltip content={<ElbowTooltip />} />

    {elbowResult.suggestedK ? (
      <ReferenceLine
        x={elbowResult.suggestedK}
        label={{ value: `Suggested K=${elbowResult.suggestedK}` }}
      />
    ) : null}

    <Line type="monotone" dataKey="inertia" />
  </LineChart>
</ResponsiveContainer>
```

Listing 5.14: Οπτικοποίηση elbow method στο cluster analysis

Το απόσπασμα 5.14 αφορά την οπτικοποίηση της elbow method στο cluster analysis. Η καμπύλη παρουσιάζει τη μεταβολή του inertia/WCSS για διαφορετικές τιμές του k, βοηθώντας τον χρήστη να αξιολογήσει ποιος αριθμός clusters είναι καταλληλότερος.

```
async function request<T>(
  path: string,
  options?: RequestInit
): Promise<T> {
  const response = await fetch(`${apiBaseUrl}${path}`, options);
  const payload = await response.json().catch(() => null);

  if (!response.ok) {
    throw new Error(payload?.error || "Request failed.");
  }

  return payload?.data as T;
}
```

Listing 5.15: Κοινή βοηθητική συνάρτηση για API requests

Το απόσπασμα 5.15 παρουσιάζει την κοινή λογική επικοινωνίας με το backend. Η συνάρτηση `request` αναλαμβάνει την εκτέλεση του `fetch`, την ανάγνωση του JSON response και τον χειρισμό πιθανών σφαλμάτων.

```
export type TeamAssociationRulesPayload = {
  context: {
    selectedEntryCount: number;
    selectedStatCount: number;
    ruleCount: number;
  };
  settings: {
    minSupport: number;
```

```
    minConfidence: number;
    minLift: number | null;
  };
  statKeys: TeamStatKey[];
  rules: TeamAssociationRule[];
  warnings: string[];
};
```

Listing 5.16: Τύπος δεδομένων για Association Rules Mining

Το απόσπασμα 5.16 δείχνει πώς το frontend περιγράφει τα δεδομένα που επιστρέφονται από τη διαδικασία Association Rules Mining. Εκτός από τους ίδιους τους κανόνες, περιλαμβάνονται πληροφορίες για τις επιλεγμένες εγγραφές, τις ρυθμίσεις του αλγορίθμου και πιθανές προειδοποιήσεις, ώστε το UI να μπορεί να παρουσιάσει πιο ολοκληρωμένα το αποτέλεσμα.

5.6 GitHub Repository, Deployment και Αυτοματισμοί

Η εφαρμογή TransferMind οργανώθηκε σε ενιαίο GitHub repository, με σαφή διαχωρισμό ανάμεσα στο frontend, το backend, τη βάση δεδομένων, την τεκμηρίωση και τους αυτοματισμούς συντήρησης. Η συγκεκριμένη οργάνωση βοηθά στη διαχείριση του έργου, καθώς κάθε βασικό μέρος της εφαρμογής βρίσκεται σε ξεχωριστό φάκελο και έχει διακριτό ρόλο. Το frontend βρίσκεται στον φάκελο `frontend/`, το backend στον φάκελο `backend/`, οι μεταβολές της βάσης δεδομένων στον φάκελο `supabase/migrations/`, ενώ οι αυτοματισμοί του GitHub Actions βρίσκονται στον φάκελο `.github/workflows/`.

Οργάνωση του repository

Η δομή του repository ακολουθεί τη λογική διαχωρισμού ευθυνών. Το frontend περιλαμβάνει την εφαρμογή χρήστη, η οποία έχει υλοποιηθεί με React, TypeScript και Vite. Το backend περιλαμβάνει τον Express API server, τα services, τα repositories, τα jobs εισαγωγής και ανανέωσης δεδομένων, καθώς και τους Python runners που χρησιμοποιούνται για τις αναλυτικές λειτουργίες. Η βάση δεδομένων τεκμηριώνεται μέσω SQL migrations, ώστε οι αλλαγές στο σχήμα της βάσης να μπορούν να παρακολουθούνται και να αναπαράγονται.

Φάκελος / αρχείο	**Ρόλος**
`frontend/`	Περιλαμβάνει την React/Vite εφαρμογή και το frontend API layer.
`backend/`	Περιλαμβάνει το Express API, τη σύνδεση με Supabase, τα ingestion jobs και τους Python runners.
`supabase/migrations/`	Περιλαμβάνει τα SQL migrations για το σχήμα, τα views και τους πίνακες καταγραφής.
`docs/`	Περιλαμβάνει οδηγούς deployment και λειτουργικής τεκμηρίωσης.
`.github/workflows/`	Περιλαμβάνει workflows για χειροκίνητη και προγραμματισμένη ανανέωση δεδομένων.
`README.md`	Περιγράφει το έργο, την εγκατάσταση, τα scripts και το deployment.

Πίνακας 5.3: Βασική οργάνωση του repository TransferMind

Η ύπαρξη ξεχωριστών φακέλων για κάθε επίπεδο του συστήματος διευκολύνει τόσο τη συντήρηση όσο και τη μελλοντική επέκταση της εφαρμογής. Παράλληλα, η χρήση Git επιτρέπει την παρακολούθηση αλλαγών στον κώδικα, στην τεκμηρίωση και στα migration files της βάσης.

Το repository περιλαμβάνει κεντρικό `README.md`, το οποίο περιγράφει τον σκοπό της εφαρμογής, τη δομή του έργου, τις βασικές τεχνολογίες, τα απαραίτητα scripts και τις οδηγίες εκτέλεσης. Επιπλέον, υπάρχουν αρχεία `.env.example` για το frontend και το backend, τα οποία λειτουργούν ως πρότυπα για τις απαιτούμενες μεταβλητές περιβάλλοντος. Τα πραγματικά αρχεία `.env` δεν πρέπει να αποθηκεύονται στο repository, καθώς περιέχουν ευαίσθητες τιμές, όπως API keys και στοιχεία σύνδεσης με τη βάση.

Η τεκμηρίωση βοηθά έναν νέο developer να κατανοήσει τη γενική λειτουργία του project και να το εκτελέσει τοπικά, εφόσον έχει πρόσβαση στις απαραίτητες εξωτερικές υπηρεσίες. Ωστόσο, η πλήρης αναπαραγωγή του project απαιτεί πρόσβαση σε Supabase project και σε εξωτερική ποδοσφαιρική API πηγή.

Deployment frontend και backend

Το deployment του TransferMind βασίζεται σε διαχωρισμένη φιλοξενία των βασικών μερών της εφαρμογής. Το frontend μπορεί να αναπτυχθεί στο Vercel, το backend στο Render και η βάση δεδομένων στη Supabase. Η διαχωρισμένη αυτή προσέγγιση ταιριάζει με την αρχιτεκτονική της εφαρμογής, καθώς το frontend και το backend αποτελούν ανεξάρτητα deployable τμήματα.

Τμήμα	**Πλατφόρμα**	**Βασική ρύθμιση**
Frontend	Vercel	Root directory `frontend`, build command `npm run build`, output directory `dist`.
Backend	Render	Root directory `backend`, start command `npm start`, health check path `/health`.
Database	Supabase	PostgreSQL βάση δεδομένων με migrations για schema και views.
Automation	GitHub Actions	Manual και scheduled workflows για εργασίες ανανέωσης δεδομένων.

Πίνακας 5.4: Σύνοψη deployment της εφαρμογής TransferMind

Για τη σωστή λειτουργία των client-side routes του React Router στο Vercel χρησιμοποιείται αρχείο `vercel.json`. Η ρύθμιση αυτή ανακατευθύνει όλα τα routes στο `index.html`, ώστε η ανανέωση μιας εσωτερικής σελίδας, όπως `/teams-comparison`, να μην οδηγεί σε σφάλμα 404.

```
{
  "rewrites": [
    {
      "source": "/(.*)",
      "destination": "/index.html"
    }
  ]
}
```

Listing 5.17: Ρύθμιση Vercel για υποστήριξη SPA routes

Το frontend χρησιμοποιεί Vite και παράγεται για production με την εντολή `npm run build`. Η εντολή αυτή εκτελεί πρώτα έλεγχο TypeScript και στη συνέχεια δημιουργεί το τελικό build της εφαρμογής.

```
"scripts": {
  "dev": "vite",
  "build": "tsc && vite build",
  "preview": "vite preview"
}
```

Listing 5.18: Build script του frontend

Το backend εκκινεί με την εντολή `npm start`, η οποία εκτελεί το `index.js`. Επιπλέον, διαθέτει endpoint ελέγχου λειτουργίας στο `/health`, το οποίο μπορεί να χρησιμοποιηθεί για βασική παρακολούθηση της διαθεσιμότητας της υπηρεσίας.

```
app.get("/health", (req, res) => {
  res.status(200).json({ data: { status: "ok" } });
});
```

Listing 5.19: Backend health check endpoint

Μεταβλητές περιβάλλοντος και προστασία secrets

Η εφαρμογή χρησιμοποιεί μεταβλητές περιβάλλοντος για τη διαχείριση ρυθμίσεων και ευαίσθητων στοιχείων. Στο frontend, η βασική μεταβλητή είναι η `VITE_API_BASE_URL`, η οποία δηλώνει τη διεύθυνση του backend API. Στο backend χρησιμοποιούνται οι μεταβλητές περιβάλλοντος `SUPABASE_URL`, `SUPABASE_SERVICE_KEY`, `API_BASE`, `RAPIDAPI_KEY` και `RAPIDAPI_HOST`. Το Supabase service key χρησιμοποιείται μόνο στο backend και δεν πρέπει να εκτίθεται στο frontend.

Πεδίο	**Μεταβλητές**
Frontend	`VITE_API_BASE_URL`
Backend server	`PORT`, `CORS_ORIGIN`
Supabase	`SUPABASE_URL`, `SUPABASE_SERVICE_KEY`
External football API	`API_BASE`, `RAPIDAPI_KEY`, `RAPIDAPI_HOST`
Python runners	`TRANSFERMIND_PYTHON_BIN`

Πίνακας 5.5: Κατηγορίες μεταβλητών περιβάλλοντος

Τα πραγματικά αρχεία `.env` εξαιρούνται από το Git μέσω `.gitignore`, ενώ τα αρχεία `.env.example` παραμένουν στο repository ως ασφαλή πρότυπα. Με αυτόν τον τρόπο, το έργο μπορεί να τεκμηριώνει τις απαραίτητες μεταβλητές χωρίς να αποκαλύπτει πραγματικές τιμές.

```
.env
.env.*
backend/.env
backend/.env.*
!.env.example
!backend/.env.example
```

Listing 5.20: Εξαίρεση αρχείων περιβάλλοντος από το Git

GitHub Actions και ανανέωση δεδομένων

Το TransferMind χρησιμοποιεί GitHub Actions για την εκτέλεση εργασιών συντήρησης δεδομένων. Τα workflows μπορούν να εκτελεστούν χειροκίνητα μέσω `workflow_dispatch`, αλλά

και αυτόματα με βάση προγραμματισμένα cron schedules. Με τον τρόπο αυτό, η ενημέρωση δεδομένων μπορεί να εκτελείται χωρίς άμεση παρέμβαση στον βασικό κώδικα της εφαρμογής.

Workflow	**Trigger**	**Σκοπός**
manual-weekly-refresh.yml	Manual και εβδομαδιαίο cron	Ανανέωση τρεχουσών βαθμολογιών και στατιστικών ομάδων.
manual-monthly-player-stats-refresh.yml	Manual και μηνιαίο cron	Ανανέωση στατιστικών παικτών για την τρέχουσα σεζόν.
manual-player-refresh.yml	Manual και προγραμματισμένες ημερομηνίες	Ανανέωση παικτών της τρέχουσας σεζόν.
manual-annual-season-reset.yml	Manual και ετήσιο cron	Ετήσια συντήρηση σεζόν, διοργανώσεων, ομάδων και λογότυπων.

Πίνακας 5.6: GitHub Actions workflows για ανανέωση δεδομένων

Ένα τυπικό workflow ορίζει χειροκίνητη και προγραμματισμένη εκτέλεση, χρησιμοποιεί περιβάλλον `ubuntu-latest`, εγκαθιστά Node.js και εκτελεί τις εντολές από τον φάκελο `backend/`. Τα πραγματικά credentials περνούν στο workflow μέσω GitHub Secrets.

```
on:
  workflow_dispatch:
  schedule:
    - cron: "0 0 * * 2"

jobs:
  weekly-refresh:
    runs-on: ubuntu-latest
    defaults:
      run:
        working-directory: backend
```

Listing 5.21: Παράδειγμα trigger σε GitHub Actions workflow

Η χρήση GitHub Actions διαχωρίζει τις εργασίες συντήρησης από την καθημερινή χρήση της εφαρμογής. Οι χρήστες αλληλεπιδρούν με το frontend και το backend API, ενώ οι διαδικασίες ενημέρωσης δεδομένων εκτελούνται ανεξάρτητα, σε προγραμματισμένα χρονικά διαστήματα ή χειροκίνητα από τον διαχειριστή του repository.

Open-source χαρακτηριστικά και περιορισμοί

Το repository διαθέτει αρκετά στοιχεία που ενισχύουν τη συνεργατική ανάπτυξη και την αναπαραγωγιμότητα του έργου. Υπάρχει κεντρικό `README.md`, οδηγός συνεισφοράς, αρχεία `.env.example`, deployment guides και οργανωμένη δομή φακέλων. Παράλληλα, η εξαίρεση των πραγματικών αρχείων περιβάλλοντος από το Git μειώνει τον κίνδυνο διαρροής ευαίσθητων πληροφοριών. Το repository διαθέτει αρχείο `LICENSE` στο root directory, μέσω του οποίου το έργο διατίθεται ως open-source λογισμικό. Η ύπαρξη άδειας χρήσης είναι σημαντική, καθώς καθορίζει με σαφήνεια τα δικαιώματα χρήσης, τροποποίησης και διανομής του κώδικα.

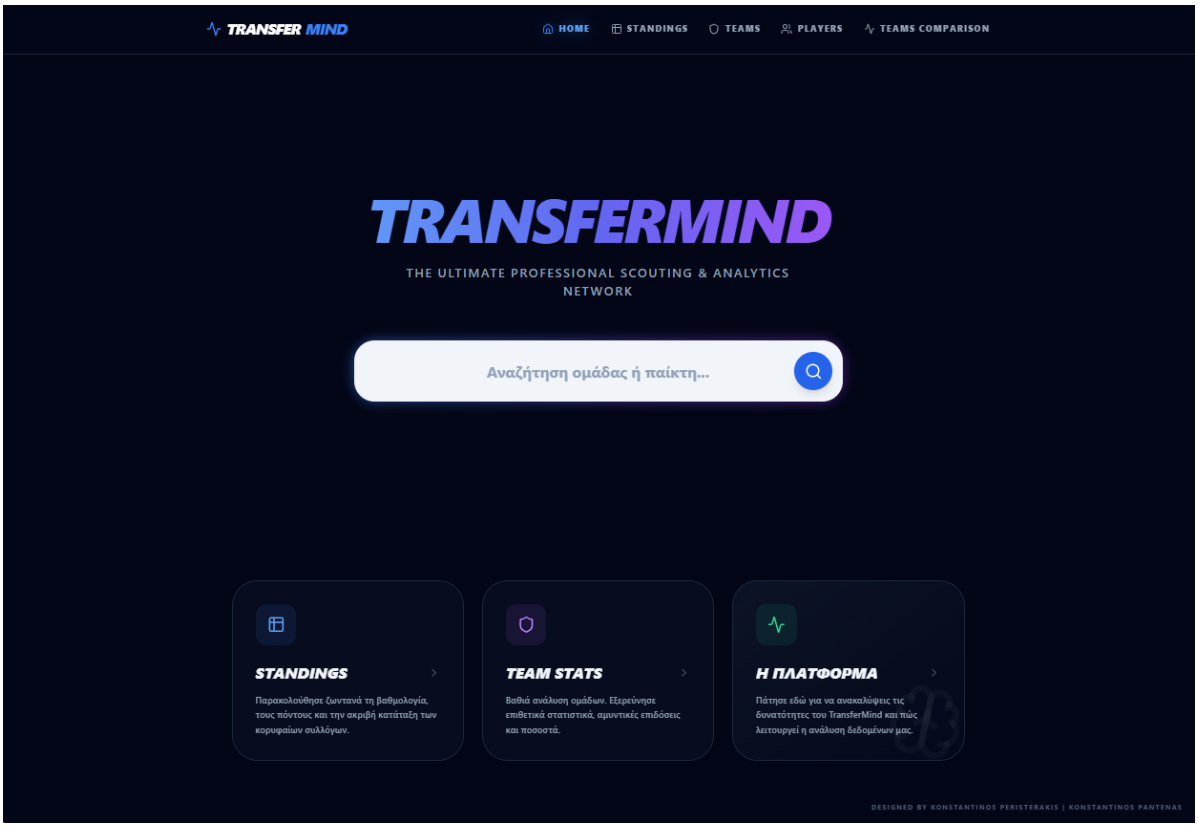
Συνολικά, η οργάνωση του repository, το διαχωρισμένο deployment και οι αυτοματισμοί μέσω GitHub Actions συμβάλλουν στη λειτουργική σταθερότητα του TransferMind. Το σύστημα μπορεί να αναπτυχθεί σε ανεξάρτητες πλατφόρμες, να διαχειρίζεται με ασφάλεια τα απαραίτητα secrets και να υποστηρίζει περιοδική ανανέωση δεδομένων χωρίς άμεση παρέμβαση στον κύριο κώδικα της εφαρμογής.

Κεφάλαιο 6

Παρουσίαση TransferMind

6.1 Homepage

Η αρχική σελίδα της εφαρμογής TransferMind αποτελεί το πρώτο σημείο επαφής του χρήστη με το σύστημα και έχει σχεδιαστεί με στόχο να παρουσιάζει άμεσα τον βασικό χαρακτήρα της πλατφόρμας. Όπως φαίνεται στην Εικόνα 6.1, η σελίδα ακολουθεί ένα μοντέρνο και καθαρό περιβάλλον χρήσης, με σκούρο φόντο, έντονες χρωματικές αντιθέσεις και κεντρική τοποθέτηση του ονόματος της εφαρμογής. Η επιλογή του συγκεκριμένου σχεδιασμού δεν είναι τυχαία, καθώς η εφαρμογή απευθύνεται σε ένα περιβάλλον ποδοσφαιρικής ανάλυσης, scouting και στατιστικής αξιολόγησης, όπου η άμεση πρόσβαση στην πληροφορία και η ευκολία πλοήγησης είναι ιδιαίτερα σημαντικές.



Σχήμα 6.1: Αρχική σελίδα της εφαρμογής TransferMind.

Στο επάνω μέρος της αρχικής σελίδας υπάρχει η βασική μπάρα πλοήγησης της εφαρμογής. Μέσω αυτής, ο χρήστης μπορεί να μεταβεί γρήγορα στις βασικές ενότητες του TransferMind, όπως η αρχική σελίδα, η βαθμολογία, οι ομάδες, οι παίκτες και η σύγκριση ομάδων. Με αυτόν τον τρόπο, η εφαρμογή δεν περιορίζεται σε μία απλή παρουσίαση δεδομένων, αλλά λειτουργεί ως ένα ενιαίο σύστημα εξερεύνησης ποδοσφαιρικών πληροφοριών. Η πλοήγηση είναι απλή και σταθερή, ώστε ο χρήστης να μπορεί να μετακινείται ανάμεσα στις διαφορετικές λειτουργίες χωρίς να χάνει τον προσανατολισμό του.

Κεντρικό στοιχείο της homepage είναι η μπάρα αναζήτησης, η οποία επιτρέπει την αναζήτηση ομάδας ή παίκτη. Η τοποθέτησή της στο κέντρο της σελίδας δείχνει ότι η αναζήτηση αποτελεί μία από τις βασικές λειτουργίες της εφαρμογής. Μέσα από αυτήν, ο χρήστης μπορεί να ξεκινήσει άμεσα την αλληλεπίδρασή του με τα δεδομένα, χωρίς να χρειάζεται πρώτα να περιηγηθεί σε πολλές διαφορετικές σελίδες. Η συγκεκριμένη επιλογή βελτιώνει τη χρηστικότητα της εφαρμογής, επειδή μειώνει τον χρόνο που απαιτείται για την πρόσβαση σε συγκεκριμένες πληροφορίες.

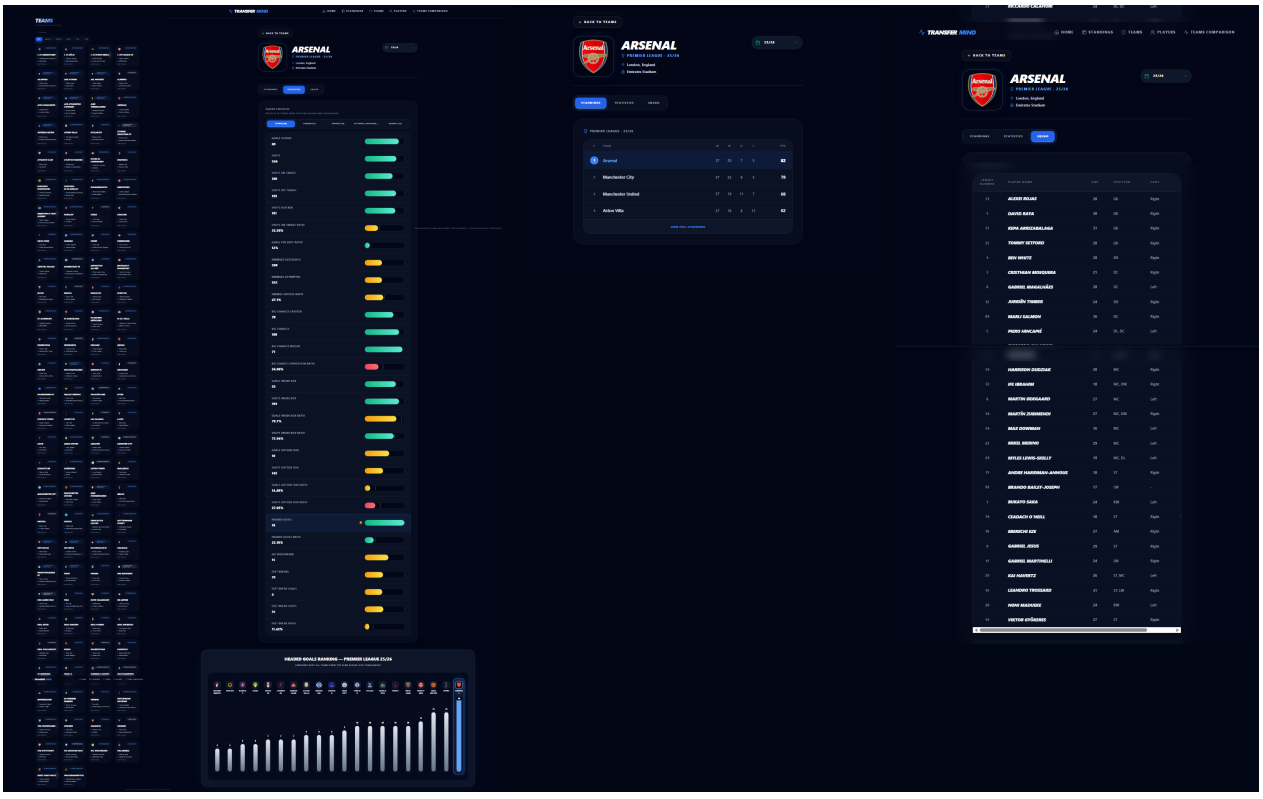
Στο κάτω μέρος της αρχικής σελίδας εμφανίζονται τρεις βασικές κάρτες λειτουργιών. Η πρώτη αφορά τη βαθμολογία, όπου ο χρήστης μπορεί να παρακολουθήσει τη θέση των ομάδων και τη συνολική εικόνα του πρωταθλήματος. Η δεύτερη κάρτα σχετίζεται με τα στατιστικά ομάδων και οδηγεί τον χρήστη σε πιο αναλυτική παρουσίαση επιδόσεων. Η τρίτη κάρτα παρουσιάζει συνοπτικά την πλατφόρμα και λειτουργεί ως εισαγωγικό σημείο για τις δυνατότητες του

TransferMind. Οι κάρτες αυτές βοηθούν τον χρήστη να κατανοήσει γρήγορα τις βασικές κατηγορίες πληροφορίας που παρέχει το σύστημα.

Συνολικά, η αρχική σελίδα λειτουργεί ως κεντρικός κόμβος της εφαρμογής. Από αυτήν ο χρήστης μπορεί να ξεκινήσει την αναζήτηση, να μεταβεί στις βασικές ενότητες και να αποκτήσει μία πρώτη εικόνα για τον σκοπό του TransferMind. Ο σχεδιασμός της homepage συνδυάζει αισθητική, απλότητα και λειτουργικότητα, στοιχεία που είναι απαραίτητα για μία εφαρμογή ανάλυσης δεδομένων, καθώς επιτρέπουν στον χρήστη να επικεντρωθεί στο περιεχόμενο και όχι στην πολυπλοκότητα του περιβάλλοντος.

6.2 Team Profile

Η ενότητα *Team Profile* παρουσιάζει συγκεντρωμένα τις βασικές πληροφορίες μιας ομάδας, τη βαθμολογική της θέση, το ρόστερ και τα διαθέσιμα στατιστικά της. Στην Εικόνα 6.2 παρουσιάζεται η ροή από τη σελίδα επιλογής ομάδας προς το αναλυτικό προφίλ της Arsenal για τη σεζόν Premier League 25/26. Η συγκεκριμένη οθόνη λειτουργεί ως κεντρικό σημείο πληροφόρησης για κάθε ομάδα, καθώς συγκεντρώνει δεδομένα που διαφορετικά θα έπρεπε να αναζητηθούν σε διαφορετικές ενότητες της εφαρμογής.



Σχήμα 6.2: Σελίδα Team Profile στην εφαρμογή TransferMind

Στο αριστερό τμήμα της εικόνας εμφανίζεται η σελίδα *Teams*, όπου οι ομάδες παρουσιάζονται

σε μορφή καρτών. Κάθε κάρτα περιλαμβάνει βασικά στοιχεία, όπως το έμβλημα, το όνομα, τη χώρα, την πόλη και το γήπεδο της ομάδας. Στο επάνω μέρος υπάρχουν φίλτρα και πεδίο αναζήτησης, ώστε ο χρήστης να μπορεί να εντοπίσει γρήγορα την ομάδα που τον ενδιαφέρει. Μετά την επιλογή μιας ομάδας, ο χρήστης μεταφέρεται στο αντίστοιχο προφίλ.

Στην κορυφή του *Team Profile* εμφανίζεται η βασική ταυτότητα της ομάδας. Περιλαμβάνονται το έμβλημα, το όνομα, η διοργάνωση, η σεζόν, η πόλη, η χώρα και το γήπεδο. Παράλληλα, υπάρχει κουμπί επιστροφής στη λίστα ομάδων και επιλογέας σεζόν, ώστε τα δεδομένα να προσαρμόζονται στην αγωνιστική περίοδο που επιλέγει ο χρήστης.

Το περιεχόμενο της σελίδας οργανώνεται σε τρεις βασικές καρτέλες: *Standings*, *Statistics* και *Squad*. Η καρτέλα *Standings* παρουσιάζει τη θέση της ομάδας στον βαθμολογικό πίνακα της διοργάνωσης. Με αυτόν τον τρόπο ο χρήστης αποκτά άμεσα εικόνα για την αγωνιστική κατάσταση της ομάδας, χωρίς να χρειάζεται να μεταβεί σε ξεχωριστή σελίδα βαθμολογίας.

Η καρτέλα *Statistics* παρουσιάζει αναλυτικά στατιστικά στοιχεία της ομάδας, οργανωμένα σε θεματικές κατηγορίες, όπως επίθεση, πάσες, άμυνα, στατικές φάσεις και προχωρημένα στατιστικά. Η χρήση κατηγοριών κάνει την πληροφορία πιο ευανάγνωστη και επιτρέπει στον χρήστη να εξετάζει συγκεκριμένες πλευρές της αγωνιστικής απόδοσης. Για κάθε στατιστικό εμφανίζεται η αριθμητική τιμή και μία οπτική μπάρα, η οποία διευκολύνει τη γρήγορη κατανόηση της έντασης ή της σημασίας της μέτρησης.

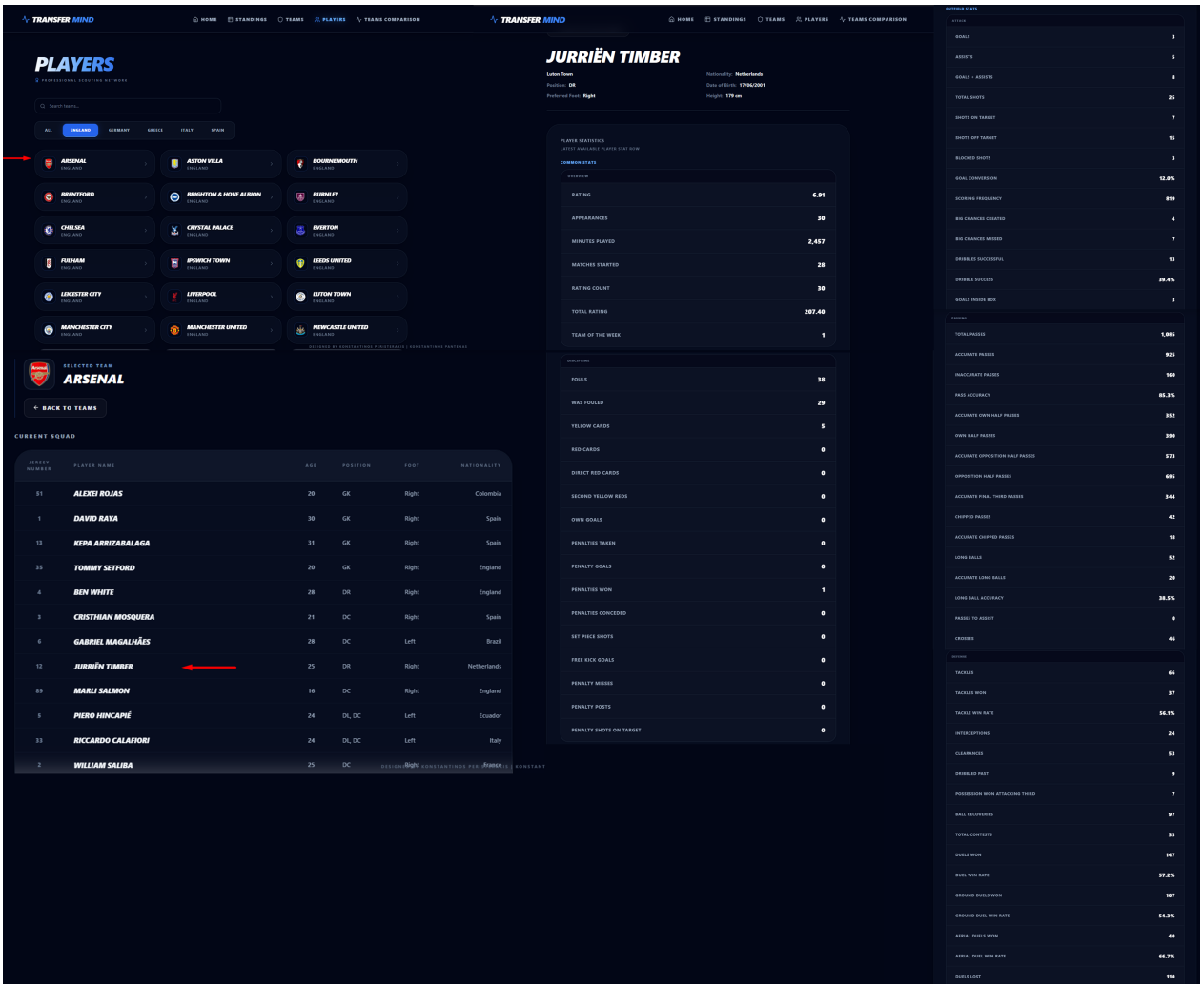
Ιδιαίτερο στοιχείο της καρτέλας *Statistics* είναι η ένδειξη με το κύπελλο. Η ένδειξη αυτή εμφανίζεται όταν η ομάδα έχει την καλύτερη επίδοση στη λίγκα σε κάποιο στατιστικό. Έτσι, ο χρήστης μπορεί να εντοπίσει άμεσα τα σημεία στα οποία η ομάδα ξεχωρίζει σε σχέση με τις υπόλοιπες. Κάτω από τα στατιστικά εμφανίζεται και σχετικό γράφημα κατάταξης, το οποίο συγκρίνει όλες τις ομάδες της διοργάνωσης ως προς το επιλεγμένο στατιστικό. Η οπτικοποίηση αυτή λειτουργεί συμπληρωματικά προς τις αριθμητικές τιμές και κάνει τη σύγκριση πιο άμεση.

Η καρτέλα *Squad* παρουσιάζει το ρόστερ της ομάδας σε μορφή πίνακα. Ο πίνακας περιλαμβάνει βασικές πληροφορίες για κάθε παίκτη, όπως αριθμό φανέλας, όνομα, ηλικία, θέση και δυνατό πόδι. Η συγκεκριμένη παρουσίαση επιτρέπει στον χρήστη να εξετάζει γρήγορα τη σύνθεση της ομάδας και να αναγνωρίζει τη βασική κατανομή των παικτών ανά θέση.

Συνολικά, η ενότητα *Team Profile* συνδυάζει τρεις βασικούς άξονες πληροφόρησης: βαθμολογική εικόνα, αγωνιστικά στατιστικά και ρόστερ. Η χρήση καρτελών, φίλτρων, χρωματικών μπαρών και γραφημάτων κάνει την παρουσίαση πιο καθαρή και περιορίζει την ανάγκη για μεγάλα κείμενα ή χειροκίνητη σύγκριση δεδομένων. Με αυτόν τον τρόπο, το προφίλ ομάδας λειτουργεί ως σύντομη αλλά πλήρης εικόνα της απόδοσης και της ταυτότητας κάθε ομάδας μέσα στην εφαρμογή TransferMind.

6.3 Player Profile

Η ενότητα *Players* της εφαρμογής TransferMind παρουσιάζει τους ποδοσφαιριστές με οργανωμένο τρόπο, ώστε ο χρήστης να μπορεί να επιλέγει πρώτα μια ομάδα και στη συνέχεια να βλέπει το αντίστοιχο ρόστερ. Στην Εικόνα 6.3 εμφανίζεται η γενική λίστα των ομάδων και η προβολή των παικτών της επιλεγμένης ομάδας. Η συγκεκριμένη δομή βοηθά στη σταδιακή πλοήγηση, καθώς αποφεύγεται η εμφάνιση ενός πολύ μεγάλου πίνακα με όλους τους παίκτες της εφαρμογής.



Σχήμα 6.3: Σελίδα Players και προβολή ρόστερ επιλεγμένης ομάδας στην εφαρμογή TransferMind

Στο αριστερό τμήμα της εικόνας εμφανίζεται η αρχική προβολή της σελίδας *Players*. Οι ομάδες παρουσιάζονται σε μορφή λίστας, όπου κάθε εγγραφή περιλαμβάνει βασικά στοιχεία αναγνώρισης, όπως το έμβλημα και το όνομα της ομάδας. Στο επάνω μέρος υπάρχει πεδίο αναζήτησης, το οποίο επιτρέπει τη γρήγορη εύρεση ομάδας. Παράλληλα, τα διαθέσιμα φίλτρα ανά διοργά-

νωση περιορίζουν τα αποτελέσματα και κάνουν την πλοήγηση πιο άμεση.

Μετά την επιλογή ομάδας, εμφανίζεται η αντίστοιχη προβολή ρόστερ. Στην κορυφή της σελίδας υπάρχει ο τίτλος της ενότητας, πεδίο αναζήτησης παικτών, η ένδειξη της επιλεγμένης ομάδας και κουμπί επιστροφής στη λίστα ομάδων. Με αυτόν τον τρόπο ο χρήστης γνωρίζει κάθε στιγμή ποια ομάδα εξετάζει και μπορεί εύκολα να επιστρέψει στο προηγούμενο βήμα.

Το κύριο περιεχόμενο της σελίδας είναι ένας πίνακας παικτών. Ο πίνακας περιλαμβάνει βασικές πληροφορίες, όπως αριθμό φανέλας, όνομα, ηλικία, θέση, δυνατό πόδι και εθνικότητα. Η μορφή αυτή επιτρέπει την άμεση ανάγνωση του ρόστερ και παρέχει μια σύντομη εικόνα για τη σύνθεση της ομάδας. Η θέση του παίκτη εμφανίζεται με συντομογραφίες, όπως GK, DC, DL, DR, MC, DM, AM, RW, LW και ST, ώστε ο πίνακας να παραμένει σύντομος και ευανάγνωστος.

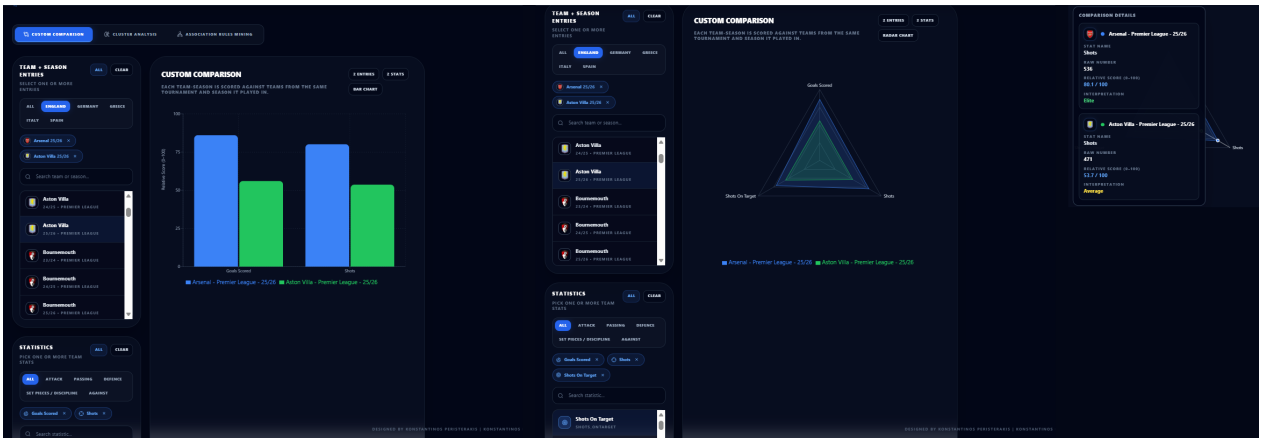
Το πεδίο αναζήτησης πάνω από τον πίνακα διευκολύνει τον εντοπισμό συγκεκριμένου ποδοσφαιριστή μέσα στο ρόστερ, χωρίς να απαιτείται χειροκίνητη αναζήτηση σε όλες τις εγγραφές. Η λειτουργία αυτή είναι χρήσιμη κυρίως σε ομάδες με μεγάλο αριθμό παικτών ή σε περιπτώσεις όπου ο χρήστης θέλει να μεταβεί γρήγορα σε ένα συγκεκριμένο προφίλ.

Σχεδιαστικά, η ενότητα ακολουθεί το ίδιο ύφος με τις υπόλοιπες σελίδες της εφαρμογής. Το σκούρο φόντο, οι μπλε τονισμοί, οι καθαρές γραμμές του πίνακα και η σταθερή μπάρα πλοήγησης δημιουργούν ενιαία οπτική ταυτότητα. Η ενεργή ενότητα *Players* ξεχωρίζει στη μπάρα πλοήγησης, ώστε ο χρήστης να γνωρίζει σε ποιο σημείο της εφαρμογής βρίσκεται.

Συνολικά, η σελίδα *Players* λειτουργεί ως οργανωμένο σημείο πρόσβασης στα ρόστερ των ομάδων. Η πληροφορία παρουσιάζεται σε δύο απλά βήματα: πρώτα επιλογή ομάδας και έπειτα προβολή παικτών. Έτσι, η εφαρμογή αποφεύγει την υπερφόρτωση δεδομένων και προσφέρει έναν καθαρό τρόπο αναζήτησης και παρουσίασης ποδοσφαιριστών.

6.4 Team Comparison

Η ενότητα *Team Comparison* της εφαρμογής TransferMind επιτρέπει τη σύγκριση ομάδων με βάση επιλεγμένα στατιστικά και συγκεκριμένες σεζόν. Στην Εικόνα 6.4 παρουσιάζεται η λειτουργία *Custom Comparison*, όπου ο χρήστης μπορεί να επιλέξει ομάδες, αγωνιστικές περιόδους και στατιστικά. Η σύγκριση βασίζεται σε σχετικές βαθμολογίες από 0 έως 100, ώστε τα δεδομένα να είναι πιο εύκολα συγκρίσιμα, ακόμη και όταν οι αρχικές τιμές έχουν διαφορετική κλίμακα.



Σχήμα 6.4: Σελίδα Team Comparison με ραβδόγραμμα, radar chart και αναλυτικά στοιχεία σύγκρισης

Στο επάνω μέρος της σελίδας εμφανίζονται οι βασικές επιλογές της ενότητας: *Custom Comparison*, *Cluster Analysis* και *Association Rules Mining*. Στην εικόνα είναι ενεργή η επιλογή *Custom Comparison*, η οποία αφορά τη χειροκίνητη σύγκριση ομάδων. Μέσα από αυτήν τη λειτουργία ο χρήστης μπορεί να ορίσει ο ίδιος ποιες ομάδες και ποια στατιστικά θα εξεταστούν.

Στην αριστερή πλευρά υπάρχει το πλαίσιο *Team + Season Entries*, όπου επιλέγονται οι ομάδες και οι αντίστοιχες σεζόν. Τα φίλτρα ανά χώρα ή διοργάνωση, καθώς και το πεδίο αναζήτησης, βοηθούν στον γρήγορο εντοπισμό των διαθέσιμων εγγραφών. Κάτω από αυτό βρίσκεται η περιοχή *Statistics*, στην οποία ο χρήστης επιλέγει τα στατιστικά της σύγκρισης. Τα στατιστικά οργανώνονται σε κατηγορίες, όπως επίθεση, πάσες, άμυνα, στατικές φάσεις και στατιστικά εναντίον, ώστε η επιλογή να γίνεται πιο εύκολα και πιο στοχευμένα.

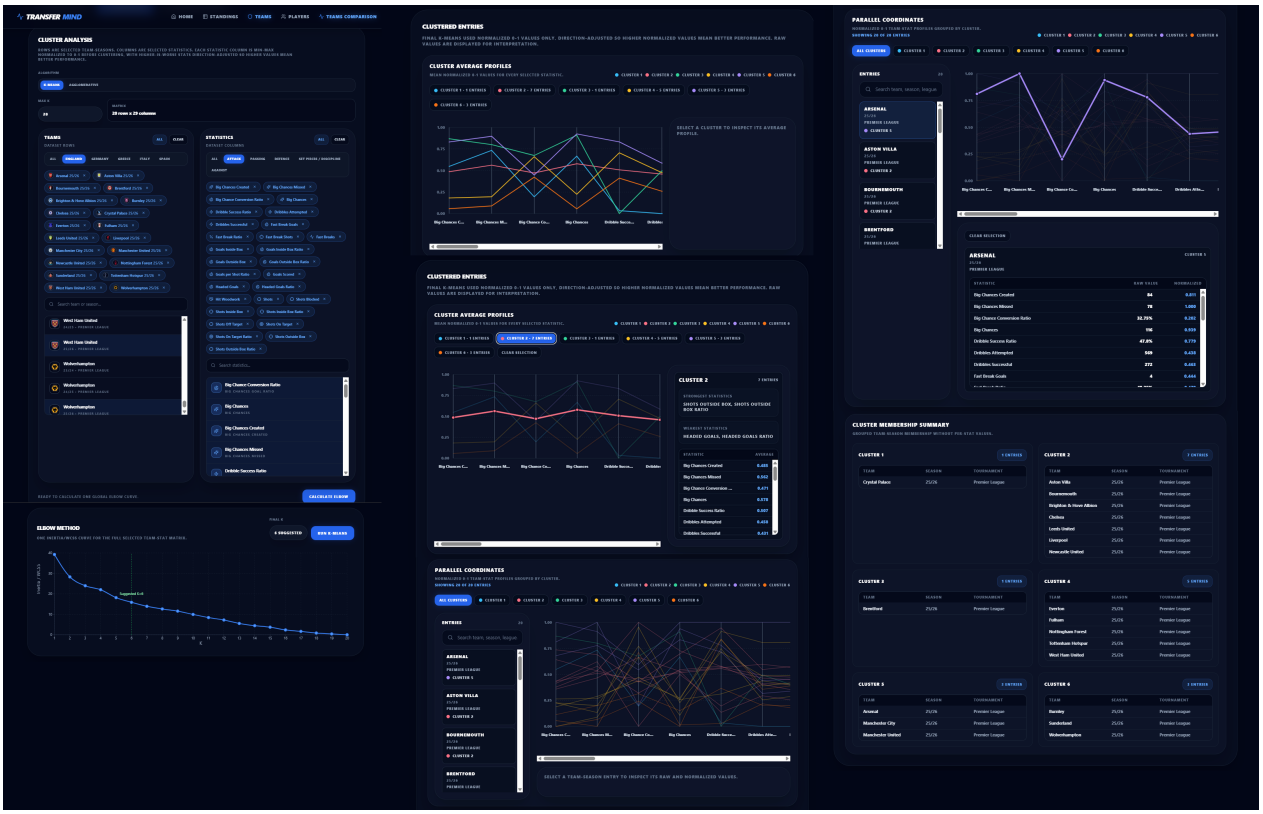
Στο κεντρικό μέρος εμφανίζονται τα αποτελέσματα της σύγκρισης. Η εφαρμογή υποστηρίζει διαφορετικούς τρόπους οπτικοποίησης, όπως ραβδόγραμμα και *radar chart*. Το ραβδόγραμμα είναι κατάλληλο για άμεση σύγκριση ανά στατιστικό, καθώς κάθε ομάδα εμφανίζεται με διαφορετική μπάρα. Αντίστοιχα, το *radar chart* προσφέρει μια πιο συνολική εικόνα, επειδή παρουσιάζει ταυτόχρονα την απόδοση των ομάδων σε πολλαπλά στατιστικά.

Στη δεξιά πλευρά εμφανίζεται το πλαίσιο *Comparison Details*. Εκεί παρουσιάζονται για κάθε επιλεγμένη ομάδα η αρχική τιμή του στατιστικού, η σχετική βαθμολογία και μία σύντομη λεκτική ερμηνεία της επίδοσης. Η προσθήκη ερμηνειών, όπως υψηλή ή μέση επίδοση, βοηθά τον χρήστη να κατανοήσει πιο γρήγορα τη σημασία των αριθμών χωρίς να χρειάζεται να αναλύσει μόνος του όλες τις τιμές.

Συνολικά, η ενότητα *Team Comparison* λειτουργεί ως εργαλείο συγκριτικής αξιολόγησης ομάδων. Ο χρήστης επιλέγει ομάδες, σεζόν και στατιστικά, ενώ η εφαρμογή παρουσιάζει τα αποτελέσματα με οπτικό και αναλυτικό τρόπο. Έτσι, η σύγκριση δεν περιορίζεται στην απλή εμφάνιση αριθμών, αλλά υποστηρίζει την καλύτερη κατανόηση των διαφορών ανάμεσα στις ομάδες.

6.5 Cluster Analysis

Η ενότητα *Cluster Analysis* της εφαρμογής TransferMind επιτρέπει την ομαδοποίηση ομάδων με βάση κοινά στατιστικά χαρακτηριστικά. Σε αντίθεση με την απλή σύγκριση ομάδων, η λειτουργία αυτή εξετάζει περισσότερες ομάδες ταυτόχρονα και εντοπίζει ποιες παρουσιάζουν παρόμοιο αγωνιστικό προφίλ. Με αυτόν τον τρόπο, ο χρήστης μπορεί να αναγνωρίσει γενικότερα μοτίβα μέσα στα δεδομένα και όχι μόνο μεμονωμένες αριθμητικές διαφορές.



Σχήμα 6.5: Σελίδα Cluster Analysis με επιλογή ομάδων, στατιστικών, Elbow Method και οπτικοποίηση clusters

Στην Εικόνα 6.5 παρουσιάζεται η συνολική ροή της ενότητας. Ο χρήστης επιλέγει τον αλγόριθμο ομαδοποίησης, τις ομάδες, τις σεζόν και τα στατιστικά που θα χρησιμοποιηθούν στην ανάλυση. Η εφαρμογή υποστηρίζει τόσο *K-Means* όσο και *Agglomerative Clustering*, δίνοντας τη δυνατότητα διαφορετικής προσέγγισης στο ίδιο σύνολο δεδομένων.

Στο αριστερό μέρος της σελίδας υπάρχουν οι βασικές ρυθμίσεις της ανάλυσης. Το πλαίσιο *Teams* χρησιμοποιείται για την επιλογή των ομάδων και των αγωνιστικών περιόδων, ενώ το πλαίσιο *Statistics* επιτρέπει την επιλογή των μεταβλητών που θα συμμετέχουν στην ομαδοποίηση. Τα στατιστικά οργανώνονται σε κατηγορίες, όπως επίθεση, πάσες, άμυνα και στατικές φάσεις, ώστε η επιλογή να γίνεται πιο εύκολα. Η επιλογή των στατιστικών είναι σημαντική, επειδή καθορίζει το είδος των ομοιοτήτων που θα αναζητήσει ο αλγόριθμος.

Για την περίπτωση του *K-Means*, η εφαρμογή παρέχει τη μέθοδο *Elbow Method*. Το γράφημα αυτό δείχνει τη μεταβολή του WCSS/Inertia για διαφορετικές τιμές του αριθμού clusters. Η εφαρμογή προτείνει μία ενδεικτική τιμή για το πλήθος των clusters, ενώ ο χρήστης μπορεί να επιλέξει την τελική τιμή και να εκτελέσει τον αλγόριθμο. Με αυτόν τον τρόπο συνδυάζεται η αυτοματοποιημένη υποστήριξη με τον χειροκίνητο έλεγχο του χρήστη.

Μετά την εκτέλεση της ανάλυσης, εμφανίζονται τα αποτελέσματα των clusters. Η προβολή *Cluster Average Profiles* παρουσιάζει τις μέσες κανονικοποιημένες τιμές κάθε cluster στα επιλεγμένα στατιστικά. Έτσι, ο χρήστης μπορεί να κατανοήσει το γενικό προφίλ κάθε ομάδας ομάδων, για παράδειγμα αν ένα cluster χαρακτηρίζεται από υψηλή επιθετική παραγωγή ή χαμηλότερες τιμές σε συγκεκριμένους δείκτες.

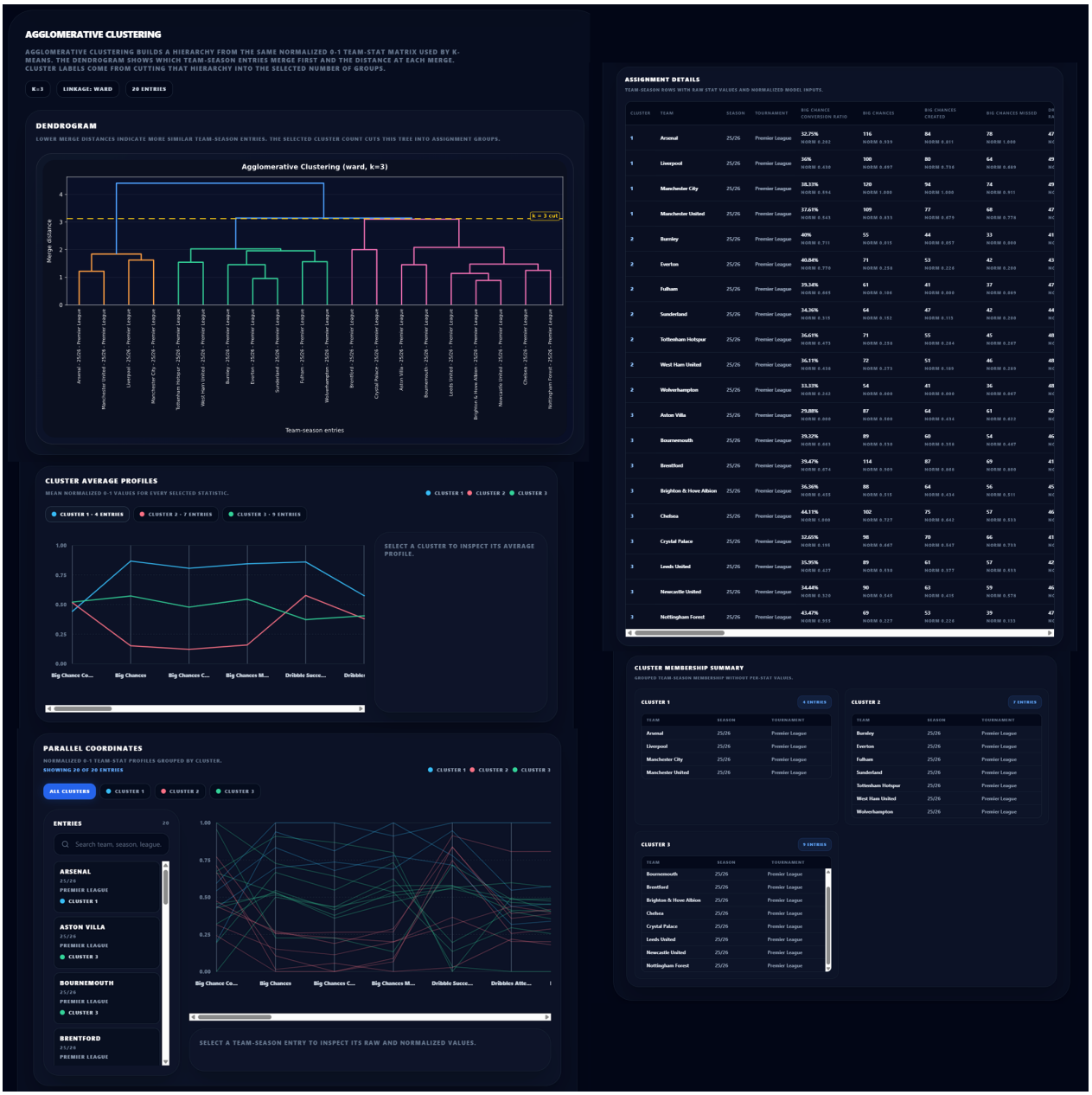
Η προβολή *Parallel Coordinates* χρησιμοποιείται για την οπτική σύγκριση των ομάδων σε πολλά στατιστικά ταυτόχρονα. Κάθε στατιστικό εμφανίζεται ως ξεχωριστός άξονας και κάθε ομάδα-σεζόν αποτυπώνεται ως γραμμή που περνά από όλους τους άξονες. Η μορφή αυτή βοηθά τον χρήστη να δει αν οι ομάδες ενός cluster έχουν παρόμοια συμπεριφορά ή αν υπάρχουν σημαντικές αποκλίσεις.

Στο κάτω μέρος της ενότητας εμφανίζεται το *Cluster Membership Summary*, το οποίο δείχνει ποιες ομάδες ανήκουν σε κάθε cluster. Η πληροφορία αυτή είναι απαραίτητη, επειδή συνδέει το αποτέλεσμα του αλγορίθμου με τις πραγματικές ομάδες που συμμετέχουν στην ανάλυση. Παράλληλα, όταν επιλεγεί μία συγκεκριμένη εγγραφή, η εφαρμογή μπορεί να εμφανίσει τις αρχικές και τις κανονικοποιημένες τιμές της, ώστε ο χρήστης να κατανοήσει καλύτερα τη θέση της μέσα στο cluster.

Συνολικά, η ενότητα *Cluster Analysis* προσφέρει μια πιο σύνθετη μορφή ανάλυσης στο TransferMind. Ο χρήστης ορίζει το δείγμα και τα στατιστικά, ενώ η εφαρμογή υπολογίζει ομάδες με παρόμοια αγωνιστικά χαρακτηριστικά και παρουσιάζει τα αποτελέσματα μέσα από γραφήματα, πίνακες και συνοπτικές περιγραφές. Έτσι, η λειτουργία δεν περιορίζεται στην απλή προβολή στατιστικών, αλλά υποστηρίζει την εξαγωγή συμπερασμάτων για ομοιότητες και διαφορές μεταξύ ομάδων.

6.6 Agglomerative Clustering

Η ενότητα *Agglomerative Clustering* αποτελεί τη δεύτερη μέθοδο ομαδοποίησης που υποστηρίζεται στο περιβάλλον *Cluster Analysis*. Σε αντίθεση με το *K-Means*, η συγκεκριμένη μέθοδος ακολουθεί ιεραρχική προσέγγιση. Αρχικά, κάθε ομάδα-σεζόν θεωρείται ξεχωριστή μονάδα και στη συνέχεια οι πιο κοντινές μονάδες ενώνονται σταδιακά, μέχρι να σχηματιστεί η τελική δομή των clusters. Έτσι, ο χρήστης μπορεί να εξετάσει όχι μόνο το τελικό αποτέλεσμα, αλλά και τον τρόπο με τον οποίο δημιουργήθηκαν οι ομάδες ομοιότητας. :contentReference[oaicite:0]index=0



Σχήμα 6.6: Σελίδα Agglomerative Clustering με dendrogram, μέσους όρους clusters, Parallel Coordinates και σύνοψη συμμετοχής ομάδων

Στην Εικόνα 6.6 παρουσιάζεται η λειτουργία *Agglomerative Clustering*. Στο επάνω μέρος εμφανίζονται οι βασικές ρυθμίσεις της ανάλυσης, όπως ο αριθμός των τελικών clusters, η μέθοδος σύνδεσης και το πλήθος των εγγραφών που συμμετέχουν. Οι ρυθμίσεις αυτές καθορίζουν τον τρόπο με τον οποίο θα γίνει η ιεραρχική ομαδοποίηση.

Το βασικό γράφημα της ενότητας είναι το *dendrogram*. Στον οριζόντιο άξονα εμφανίζονται οι ομάδες-σεζόν που συμμετέχουν στην ανάλυση, ενώ στον κάθετο άξονα εμφανίζεται η απόσταση συγχώνευσης. Όσο χαμηλότερα ενώνονται δύο εγγραφές, τόσο μεγαλύτερη θεωρείται

η ομοιότητά τους. Η διακεκομμένη γραμμή δείχνει το σημείο κοπής της ιεραρχίας και ορίζει πόσα clusters θα δημιουργηθούν τελικά.

Στη δεξιά πλευρά παρουσιάζονται τα *Cluster Average Profiles*, δηλαδή οι μέσες κανονικοποιημένες τιμές κάθε cluster στα επιλεγμένα στατιστικά. Η προβολή αυτή βοηθά τον χρήστη να κατανοήσει το γενικό αγωνιστικό προφίλ κάθε ομάδας ομάδων και να εντοπίσει σε ποια χαρακτηριστικά ένα cluster εμφανίζει υψηλότερες ή χαμηλότερες τιμές.

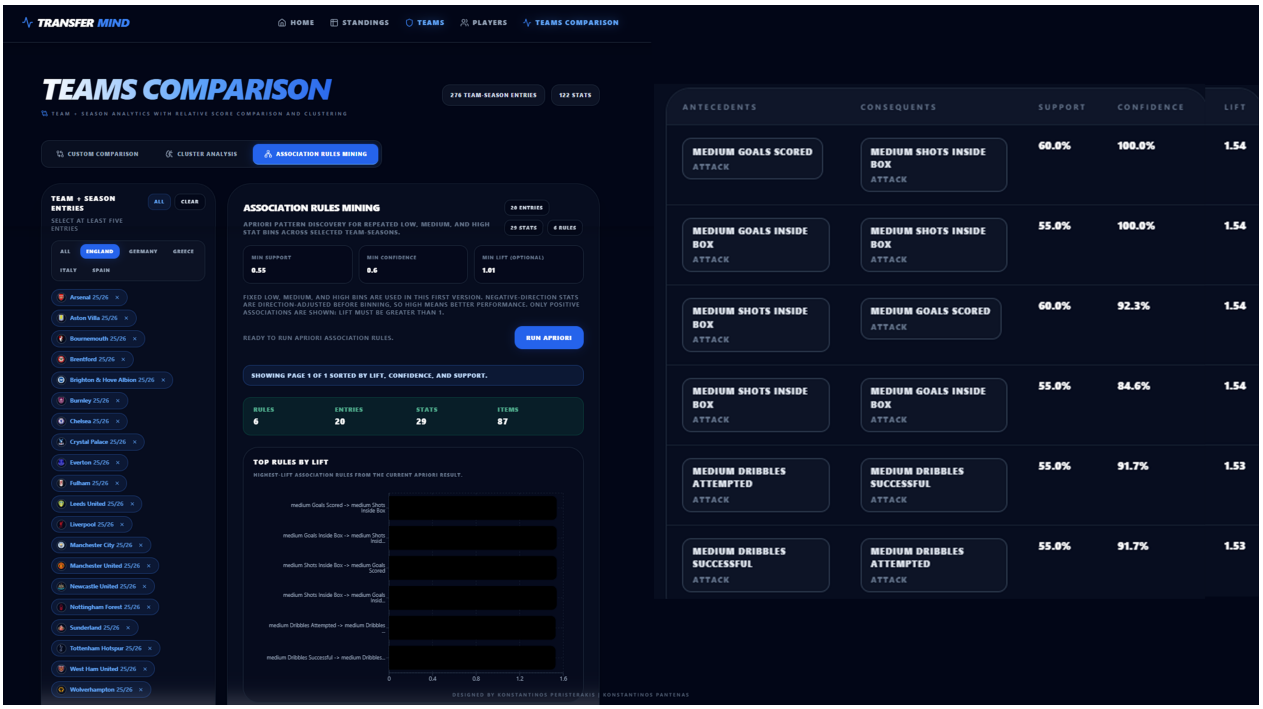
Η οπτικοποίηση *Parallel Coordinates* χρησιμοποιείται με τον ίδιο τρόπο όπως και στην προηγούμενη ενότητα *Cluster Analysis*. Κάθε στατιστικό εμφανίζεται ως ξεχωριστός άξονας και κάθε ομάδα-σεζόν ως γραμμή που περνά από όλους τους άξονες. Οι γραμμές χρωματίζονται ανάλογα με το cluster, ώστε να είναι πιο εύκολη η σύγκριση της συμπεριφοράς των ομάδων μέσα και ανάμεσα στα clusters.

Στο κάτω μέρος εμφανίζονται πίνακες με τις αναλυτικές τιμές και το *Cluster Membership Summary*. Οι πίνακες παρουσιάζουν σε ποιο cluster ανήκει κάθε ομάδα, καθώς και τις βασικές τιμές των στατιστικών που χρησιμοποιήθηκαν στην ανάλυση. Με αυτόν τον τρόπο, το αποτέλεσμα του αλγορίθμου συνδέεται άμεσα με τις πραγματικές εγγραφές και γίνεται πιο εύκολο να ερμηνευθεί.

Συνολικά, η ενότητα *Agglomerative Clustering* επεκτείνει την ανάλυση ομοιοτήτων με μια ιεραρχική προσέγγιση. Το *dendrogram* δείχνει τη διαδικασία συγχώνευσης των ομάδων, τα *Cluster Average Profiles* συνοψίζουν το προφίλ κάθε cluster, τα *Parallel Coordinates* παρουσιάζουν τη συμπεριφορά των ομάδων στα επιλεγμένα στατιστικά και το *Cluster Membership Summary* εμφανίζει καθαρά τη συμμετοχή κάθε ομάδας στο αντίστοιχο cluster.

6.7 Association Rules Mining

Η ενότητα *Association Rules Mining* της εφαρμογής TransferMind παρουσιάζει κανόνες συσχέτισης που προκύπτουν από τα διαθέσιμα δεδομένα ομάδων και στατιστικών. Σε αντίθεση με τη σύγκριση ομάδων ή την ομαδοποίηση, η συγκεκριμένη λειτουργία δεν εξετάζει μόνο αν δύο ομάδες μοιάζουν μεταξύ τους, αλλά αναζητά επαναλαμβανόμενα μοτίβα μέσα στα δεδομένα. Έτσι, η εφαρμογή μπορεί να εντοπίσει σχέσεις του τύπου «όταν εμφανίζεται ένα σύνολο χαρακτηριστικών, τότε είναι πιθανό να εμφανιστεί και ένα άλλο χαρακτηριστικό».



Σχήμα 6.7: Σελίδα Association Rules Mining με λίστα κανόνων συσχέτισης στην εφαρμογή TransferMind

Στην Εικόνα 6.7 εμφανίζεται η λίστα των κανόνων συσχέτισης που έχουν παραχθεί από την εφαρμογή. Κάθε γραμμή αντιστοιχεί σε έναν κανόνα και παρουσιάζει το αρχικό σύνολο χαρακτηριστικών και το αποτέλεσμα που συνδέεται με αυτό. Στην ορολογία των association rules, το πρώτο μέρος ονομάζεται *antecedent*, ενώ το δεύτερο *consequent*. Με αυτόν τον τρόπο, ο χρήστης μπορεί να δει ποιες καταστάσεις εμφανίζονται συχνά μαζί.

Για κάθε κανόνα παρουσιάζονται βασικές μετρικές αξιολόγησης, όπως *support*, *confidence* και *lift*. Το *support* δείχνει πόσο συχνά εμφανίζεται ο κανόνας στο συνολικό σύνολο δεδομένων. Το *confidence* δείχνει πόσο πιθανό είναι να εμφανιστεί το αποτέλεσμα όταν ισχύει το αρχικό σύνολο χαρακτηριστικών. Το *lift* δείχνει αν η σχέση είναι ισχυρότερη από μια τυχαία συνύπαρξη, επομένως βοηθά στην αναγνώριση πιο ουσιαστικών συσχετίσεων.

Η παρουσίαση των κανόνων σε μορφή λίστας κάνει την ανάλυση πιο εύκολη στην ανάγνωση. Ο χρήστης μπορεί να εξετάσει κάθε κανόνα ξεχωριστά, να συγκρίνει τις μετρικές του και να εντοπίσει ποιοι κανόνες έχουν μεγαλύτερη σημασία. Η διάταξη ακολουθεί το ίδιο οπτικό ύφος με τις υπόλοιπες ενότητες της εφαρμογής, διατηρώντας την πληροφορία οργανωμένη και ευανάγνωστη.

Συνολικά, η λειτουργία *Association Rules Mining* λειτουργεί συμπληρωματικά προς το *Team Comparison* και το *Cluster Analysis*. Η πρώτη λειτουργία υποστηρίζει την άμεση σύγκριση ομάδων, η δεύτερη εντοπίζει ομάδες με παρόμοιο προφίλ, ενώ το *Association Rules Mining* εστιάζει στις σχέσεις μεταξύ χαρακτηριστικών. Με αυτόν τον τρόπο, το TransferMind δεν περιορίζεται στην παρουσίαση στατιστικών, αλλά προσφέρει και έναν μηχανισμό εξαγωγής συμπερασμά-

των από επαναλαμβανόμενα μοτίβα στα δεδομένα.

6.8 Visualizations

Η ενότητα *Visualizations* συνοψίζει τα βασικά οπτικά στοιχεία που χρησιμοποιούνται στην εφαρμογή TransferMind. Η εφαρμογή δεν παρουσιάζει τα δεδομένα μόνο ως αριθμούς, αλλά αξιοποιεί γραφήματα, μπάρες και πίνακες, ώστε η πληροφορία να γίνεται πιο κατανοητή και πιο εύκολη στη σύγκριση.

Στο *Team Profile* χρησιμοποιούνται οριζόντιες μπάρες και ranking charts, ώστε ο χρήστης να βλέπει γρήγορα τη σχετική επίδοση μιας ομάδας σε συγκεκριμένα στατιστικά. Στο *Team Comparison* χρησιμοποιούνται bar charts και radar charts, τα οποία βοηθούν στη σύγκριση ομάδων τόσο ανά στατιστικό όσο και ως συνολικό αγωνιστικό προφίλ.

Στις λειτουργίες ομαδοποίησης, η εφαρμογή χρησιμοποιεί πιο σύνθετες οπτικοποιήσεις. Στο *K-Means*, το γράφημα *Elbow Method* βοηθά στην επιλογή του αριθμού των clusters, ενώ τα *Cluster Average Profiles* παρουσιάζουν το μέσο προφίλ κάθε cluster. Αντίστοιχα, στο *Agglomerative Clustering*, το *dendrogram* δείχνει την ιεραρχική διαδικασία με την οποία ενώνονται οι ομάδες.

Η προβολή *Parallel Coordinates* χρησιμοποιείται στις ενότητες ομαδοποίησης για την παρουσίαση πολλών στατιστικών ταυτόχρονα. Κάθε στατιστικό εμφανίζεται ως ξεχωριστός άξονας και κάθε ομάδα-σεζόν ως γραμμή. Με αυτόν τον τρόπο, ο χρήστης μπορεί να συγκρίνει πολυδιάστατα δεδομένα χωρίς να χρειάζεται να εξετάζει κάθε στατιστικό ξεχωριστά.

Τέλος, οι πίνακες της εφαρμογής συμπληρώνουν τα γραφήματα με ακριβείς τιμές και λεπτομέρειες. Χρησιμοποιούνται για το ρόστερ, τα στοιχεία σύγκρισης, τη συμμετοχή στα clusters και τους κανόνες συσχέτισης. Συνολικά, οι οπτικοποιήσεις βοηθούν τον χρήστη να κατανοεί πιο γρήγορα τα δεδομένα, να εντοπίζει σημαντικές διαφορές και να εξάγει συμπεράσματα από τα στατιστικά της εφαρμογής.

Κεφάλαιο 7

Discussion

7.1 SUS

Το SUS (System Usability Scale) είναι ένα σύντομο και ευρέως χρησιμοποιούμενο ερωτηματολόγιο αξιολόγησης της ευχρηστίας ενός συστήματος. Χρησιμοποιείται για να αποτυπώσει την εμπειρία των χρηστών μετά τη χρήση μιας εφαρμογής, ενός ιστότοπου ή ενός πληροφοριακού συστήματος. Αποτελείται από δέκα ερωτήσεις, στις οποίες οι χρήστες απαντούν συνήθως με βάση μια κλίμακα συμφωνίας από το 1 έως το 5. Μέσα από τις απαντήσεις αυτές προκύπτει μια συνολική βαθμολογία, η οποία βοηθά στην εκτίμηση του πόσο εύχρηστο, κατανοητό και φιλικό θεωρείται το σύστημα από τους χρήστες. Στην παρούσα εργασία, το SUS χρησιμοποιείται ως εργαλείο αξιολόγησης της εφαρμογής TransferMind, ώστε να καταγραφεί με απλό και δομημένο τρόπο η άποψη των χρηστών για την εμπειρία χρήσης.

Ερωτηματολόγιο

Το ερωτηματολόγιο που χρησιμοποιήθηκε περιλάμβανε τις δέκα τυπικές ερωτήσεις του SUS, προσαρμοσμένες στην εφαρμογή TransferMind. Οι ερωτήσεις ήταν οι εξής:

1. Πιστεύω ότι θα ήθελα να χρησιμοποιώ συχνά την εφαρμογή TransferMind.
2. Βρήκα την εφαρμογή TransferMind αδικαιολόγητα πολύπλοκη.
3. Θεωρώ ότι η εφαρμογή TransferMind ήταν εύκολη στη χρήση.
4. Πιστεύω ότι θα χρειαζόμουν τη βοήθεια κάποιου τεχνικού ατόμου για να μπορέσω να χρησιμοποιήσω την εφαρμογή TransferMind.
5. Βρήκα ότι οι διάφορες λειτουργίες της εφαρμογής TransferMind ήταν καλά ενσωματωμένες μεταξύ τους.

6. Θεωρώ ότι υπήρχε μεγάλη ασυνέπεια στον τρόπο λειτουργίας της εφαρμογής TransferMind.
7. Πιστεύω ότι οι περισσότεροι χρήστες θα μάθαιναν να χρησιμοποιούν την εφαρμογή TransferMind πολύ γρήγορα.
8. Βρήκα την εφαρμογή TransferMind δύσχρηστη και κουραστική στη χρήση.
9. Ένιωσα σιγουριά κατά τη χρήση της εφαρμογής TransferMind.
10. Χρειάστηκε να μάθω αρκετά πράγματα πριν μπορέσω να χρησιμοποιήσω αποτελεσματικά την εφαρμογή TransferMind.

Οι ερωτήσεις 1, 3, 5, 7 και 9 είναι θετικά διατυπωμένες, ενώ οι ερωτήσεις 2, 4, 6, 8 και 10 είναι αρνητικά διατυπωμένες. Αυτή η εναλλαγή θετικών και αρνητικών ερωτήσεων βοηθά στην πιο ισορροπημένη αξιολόγηση της εμπειρίας χρήσης.

Συμμετέχοντες

Το ερωτηματολόγιο απαντήθηκε από 50 συμμετέχοντες. Οι συμμετέχοντες χρησιμοποίησαν ή είδαν την εφαρμογή TransferMind και στη συνέχεια απάντησαν στις ερωτήσεις με βάση την προσωπική τους εμπειρία. Ο αριθμός των συμμετεχόντων θεωρείται επαρκής για την εξαγωγή μιας πιο ολοκληρωμένης εικόνας σχετικά με τη χρηστικότητα της εφαρμογής, καθώς επιτρέπει τον υπολογισμό μέσων όρων και την παρατήρηση γενικών τάσεων στις απαντήσεις.

Υπολογισμός SUS Score

Ο υπολογισμός του SUS score δεν γίνεται με απλό μέσο όρο όλων των απαντήσεων. Κάθε ερώτηση μετατρέπεται πρώτα σε επιμέρους συνεισφορά από 0 έως 4. Για τις θετικά διατυπωμένες ερωτήσεις αφαιρείται το 1 από την απάντηση του χρήστη. Για τις αρνητικά διατυπωμένες ερωτήσεις, η απάντηση αφαιρείται από το 5.

- Για τις θετικές ερωτήσεις: score = απάντηση - 1
- Για τις αρνητικές ερωτήσεις: score = 5 - απάντηση

Στη συνέχεια, οι δέκα επιμέρους συνεισφορές αθροίζονται και το αποτέλεσμα πολλαπλασιάζεται με 2.5, ώστε το τελικό SUS score να εκφραστεί σε κλίμακα από 0 έως 100.

$$SUS = \left(\sum_{i=1}^{10} score_i \right) \times 2.5 \tag{7.1}$$

7.2 Αποτελέσματα

Ο Πίνακας 7.1 παρουσιάζει συγκεντρωτικά τις απαντήσεις των 50 συμμετεχόντων για κάθε ερώτηση του SUS. Για κάθε ερώτηση εμφανίζεται ο αριθμός απαντήσεων ανά τιμή της κλίμακας, ο μέσος όρος και η μέση συνεισφορά της ερώτησης στον τελικό υπολογισμό του SUS score.

Πίνακας 7.1: Συγκεντρωτικά αποτελέσματα SUS

Ερ.	**Περιγραφή**	**1**	**2**	**3**	**4**	**5**	**Μ.Ο.**	**SUS συνεισφορά**
1	Συχνή χρήση της εφαρμογής	0	1	6	10	33	4.50	3.50
2	Αδικαιολόγητη πολυπλοκότητα	31	12	5	1	1	1.58	3.42
3	Ευκολία χρήσης	0	0	2	16	32	4.60	3.60
4	Ανάγκη τεχνικής υποστήριξης	33	9	3	3	2	1.64	3.36
5	Καλή ενσωμάτωση λειτουργιών	0	0	2	17	31	4.58	3.58
6	Ασυνέπεια στη λειτουργία	35	6	2	5	2	1.66	3.34
7	Γρήγορη εκμάθηση	0	0	4	12	34	4.60	3.60
8	Δύσχρηστη και κουραστική χρήση	39	6	1	4	0	1.40	3.60
9	Σιγουριά κατά τη χρήση	0	0	3	18	29	4.52	3.52
10	Ανάγκη εκμάθησης πριν τη χρήση	30	12	4	3	1	1.66	3.34

Με βάση τις μέσες συνεισφορές των δέκα ερωτήσεων, το άθροισμα είναι:

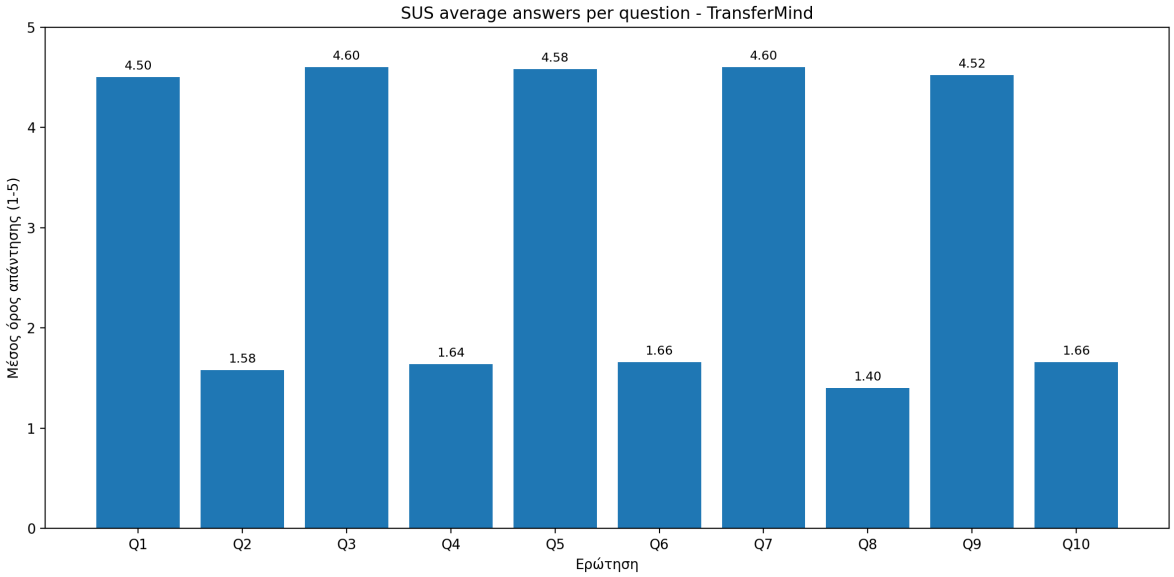
$$3.50 + 3.42 + 3.60 + 3.36 + 3.58 + 3.34 + 3.60 + 3.60 + 3.52 + 3.34 = 34.86 \tag{7.2}$$

Επομένως, το συνολικό SUS score της εφαρμογής TransferMind είναι:

$$SUS = 34.86 \times 2.5 = 87.15 \tag{7.3}$$

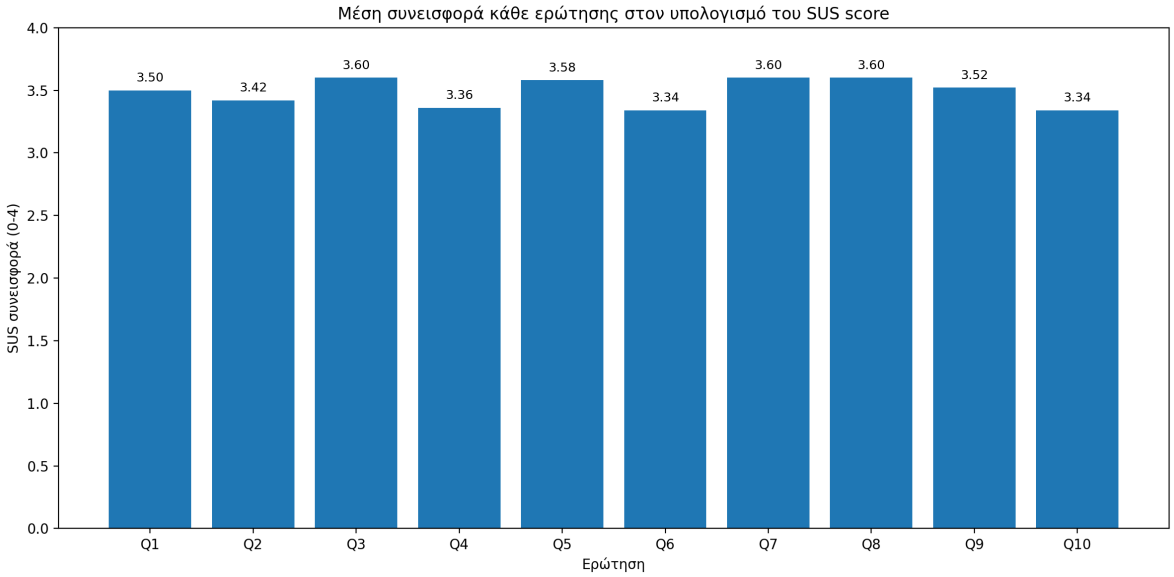
SUS Score = 87.15 / 100

Με βάση τις 50 απαντήσεις που συλλέχθηκαν, το συνολικό SUS score της εφαρμογής TransferMind υπολογίστηκε σε 87.15/100. Η τιμή αυτή δείχνει πολύ υψηλό επίπεδο αντιλαμβανόμενης χρηστικότητας. Οι χρήστες αξιολόγησαν θετικά την ευκολία χρήσης, τη γρήγορη εκμάθηση, τη συνοχή των λειτουργιών και την αίσθηση σιγουριάς κατά τη χρήση της εφαρμογής. Παράλληλα, οι χαμηλές τιμές στις αρνητικά διατυπωμένες ερωτήσεις δείχνουν ότι η εφαρμογή δεν θεωρήθηκε ιδιαίτερα πολύπλοκη, δύσχρηστη ή κουραστική.



Σχήμα 7.1: Μέσοι όροι απαντήσεων ανά ερώτηση SUS

Στο Σχήμα 7.1 παρουσιάζονται οι μέσοι όροι των απαντήσεων για κάθε ερώτηση του ερωτηματολογίου. Στις θετικά διατυπωμένες ερωτήσεις παρατηρούνται υψηλοί μέσοι όροι, γεγονός που δείχνει θετική αντίληψη των χρηστών για την εφαρμογή. Αντίθετα, στις αρνητικά διατυπωμένες ερωτήσεις οι μέσοι όροι είναι χαμηλοί, κάτι που είναι επίσης θετικό, καθώς δείχνει ότι οι χρήστες δεν θεώρησαν την εφαρμογή περίπλοκη, ασυνεπή ή δύσχρηστη.

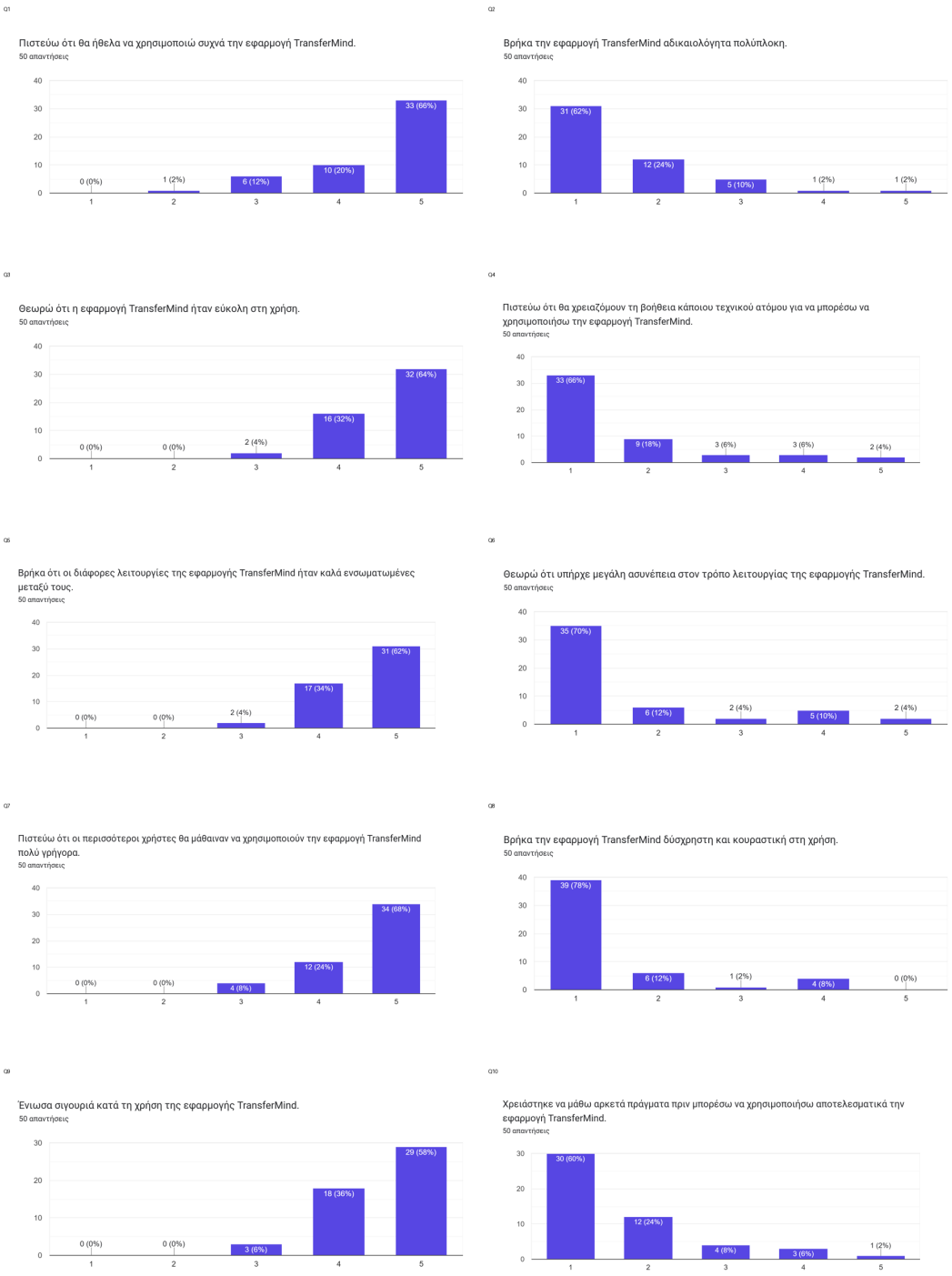


Σχήμα 7.2: Μέση συνεισφορά κάθε ερώτησης στον υπολογισμό του SUS score

Στο Σχήμα 7.2 παρουσιάζεται η μέση συνεισφορά κάθε ερώτησης στον τελικό υπολογισμό του

SUS score. Οι περισσότερες ερωτήσεις έχουν συνεισφορά μεγαλύτερη από 3.3 σε μέγιστη τιμή 4, γεγονός που επιβεβαιώνει τη συνολικά θετική αξιολόγηση της εφαρμογής. Η υψηλότερη συνεισφορά εμφανίζεται στις ερωτήσεις που σχετίζονται με την ευκολία χρήσης, τη γρήγορη εκμάθηση και την εύχρηστη αλληλεπίδραση με την εφαρμογή.

Στο Σχήμα 7.3 παρουσιάζονται συγκεντρωτικά τα γραφήματα των απαντήσεων για όλες τις ερωτήσεις του ερωτηματολογίου. Η εικόνα αυτή χρησιμοποιείται για να αποτυπωθεί συνολικά η κατανομή των απαντήσεων που συλλέχθηκαν μέσω του Google Forms.



Σχήμα 7.3: Συγκεντρωτικά γραφήματα απαντήσεων του ερωτηματολογίου SUS

Η συνολική εικόνα επιβεβαιώνει ότι οι θετικές ερωτήσεις συγκεντρώνουν κυρίως απαντήσεις στις τιμές 4 και 5, ενώ οι αρνητικές ερωτήσεις συγκεντρώνουν κυρίως απαντήσεις στις τιμές 1 και 2.

Συνολικά, τα αποτελέσματα δείχνουν ότι η εφαρμογή TransferMind ανταποκρίνεται σε μεγάλο βαθμό στις βασικές απαιτήσεις χρηστικότητας. Το SUS score 87.15/100 αποτελεί ένδειξη ότι οι χρήστες αντιλήφθηκαν την εφαρμογή ως εύχρηστη, κατανοητή και καλά οργανωμένη. Ιδιαίτερα θετικά αξιολογήθηκαν η ευκολία χρήσης, η συνοχή των λειτουργιών, η γρήγορη εκμάθηση και η σιγουριά κατά την πλοήγηση στην εφαρμογή.

Παράλληλα, οι χαμηλές τιμές στις αρνητικά διατυπωμένες ερωτήσεις δείχνουν ότι η εφαρμογή δεν θεωρήθηκε ιδιαίτερα πολύπλοκη, ασυνεπής ή κουραστική. Επομένως, η εμπειρία χρήσης μπορεί να χαρακτηριστεί θετική, ενώ τα αποτελέσματα της αξιολόγησης δείχνουν ότι η εφαρμογή έχει υψηλό επίπεδο αποδοχής από τους συμμετέχοντες.

Κεφάλαιο 8

Συμπεράσματα και Μελλοντικές Επεκτάσεις

8.1 Συμπεράσματα

Η ανάπτυξη του TransferMind έδειξε ότι τα ποδοσφαιρικά στατιστικά μπορούν να αποκτήσουν μεγαλύτερη αξία όταν δεν παρουσιάζονται απλώς ως αριθμοί, αλλά οργανώνονται με τρόπο που βοηθά τον χρήστη να τα κατανοήσει και να τα ερμηνεύσει. Μέσα από τις λειτουργίες της εφαρμογής, ο χρήστης μπορεί να εξετάσει ομάδες και παίκτες, να πραγματοποιήσει συγκρίσεις και να παρατηρήσει στοιχεία που δεν είναι πάντα εμφανή με την πρώτη ματιά. Έτσι, η εφαρμογή επιχειρεί να μετατρέψει την απλή πληροφορία σε πιο χρήσιμη γνώση. Η εργασία που ξεκίνησε από την ανάγκη καλύτερης αξιοποίησης των ποδοσφαιρικών δεδομένων, κατέληξε στη δημιουργία ενός ολοκληρωμένου διαδικτυακού συστήματος, το οποίο συνδυάζει στατιστική πληροφόρηση, διαδραστική παρουσίαση και τεχνικές ανάλυσης δεδομένων.

Το πιο σημαντικό συμπέρασμα της εργασίας είναι ότι τεχνικές όπως το K-Means Clustering, η ιεραρχική συσταδοποίηση και ο αλγόριθμος Apriori μπορούν να εφαρμοστούν με πρακτικό τρόπο σε δεδομένα ποδοσφαίρου. Οι αλγόριθμοι αυτοί δεν χρησιμοποιήθηκαν μόνο θεωρητικά, αλλά ενσωματώθηκαν σε ένα πραγματικό σύστημα, με στόχο να αναδείξουν ομοιότητες, σχέσεις και μοτίβα μέσα στα διαθέσιμα δεδομένα. Με αυτόν τον τρόπο, η εργασία συνέδεσε τη θεωρητική γνώση με την πρακτική εφαρμογή.

Επιπλέον, η υλοποίηση της εφαρμογής ανέδειξε τη σημασία της συνεργασίας διαφορετικών τεχνολογιών σε ένα ενιαίο έργο. Η συλλογή δεδομένων, η αποθήκευσή τους σε βάση, η επεξεργασία τους και η παρουσίασή τους στον τελικό χρήστη αποτέλεσαν στάδια που έπρεπε να λειτουργήσουν συνδυαστικά. Μέσα από αυτή τη διαδικασία, το TransferMind αποτέλεσε ένα παράδειγμα ολοκληρωμένης διαδικτυακής εφαρμογής που αξιοποιεί τεχνολογίες ανάπτυξης λογισμικού, βάσεις δεδομένων και μεθόδους ανάλυσης.

Ωστόσο, η συνεισφορά της εργασίας δεν περιορίζεται μόνο στην τεχνική υλοποίηση της εφαρμογής. Ιδιαίτερη έμφαση δόθηκε στη δημιουργία ενός περιβάλλοντος που μπορεί να χρησι-

μοποιηθεί από διαφορετικές κατηγορίες χρηστών, χωρίς να απαιτείται εξειδικευμένη γνώση στατιστικής ή μηχανικής μάθησης. Με αυτόν τον τρόπο, η εφαρμογή μπορεί να αξιοποιηθεί τόσο από φιλάθλους που επιθυμούν να εξερευνήσουν ποδοσφαιρικά δεδομένα, όσο και από φοιτητές, ερευνητές ή αναλυτές που ενδιαφέρονται για την εφαρμογή τεχνικών ανάλυσης δεδομένων σε πραγματικά σύνολα πληροφοριών.Τα αποτελέσματα της αξιολόγησης της εφαρμογής μέσω του ερωτηματολογίου System Usability Scale (SUS) έδειξαν ότι οι βασικές λειτουργίες της εφαρμογής μπορούν να χρησιμοποιηθούν αποτελεσματικά, επιβεβαιώνοντας ότι η ενσωμάτωση σύνθετων αναλυτικών διαδικασιών μπορεί να πραγματοποιηθεί με τρόπο τέτοιο, ώστε να μην επιβαρύνεται η εμπειρία χρήσης.

Συνολικά, η εργασία απέδειξε ότι οι τεχνικές εξόρυξης δεδομένων και μηχανικής μάθησης μπορούν να ενσωματωθούν επιτυχώς σε μια διαδικτυακή εφαρμογή ποδοσφαιρικής ανάλυσης, προσφέροντας στον χρήστη περισσότερες δυνατότητες κατανόησης και αξιοποίησης των διαθέσιμων δεδομένων. Το TransferMind αποτελεί μια προσπάθεια γεφύρωσης του χάσματος ανάμεσα στην απλή παρουσίαση στατιστικών στοιχείων και στην ουσιαστική ανάλυση δεδομένων, συμβάλλοντας στην ευρύτερη διάδοση εργαλείων ποδοσφαιρικής ανάλυσης σε ένα ευρύτερο και πιο ανομοιογενές κοινό, με κεντρικό γνώμονα την αγάπη για το άθλημα.

8.2 Περιορισμοί

Παρότι η εφαρμογή TransferMind πέτυχε τους βασικούς στόχους που τέθηκαν κατά τη διάρκεια της εργασίας, υπάρχουν ορισμένοι περιορισμοί οι οποίοι πρέπει να ληφθούν υπόψη κατά την αξιολόγηση των αποτελεσμάτων της. Οι περιορισμοί αυτοί δεν αναιρούν τη χρησιμότητα της εφαρμογής, αλλά αναδεικνύουν ορισμένα σημεία στα οποία τα αποτελέσματα επηρεάζονται από παράγοντες που δεν μπορούν να ελεγχθούν πλήρως.

Ένας από τους σημαντικότερους περιορισμούς αφορά τα δεδομένα που χρησιμοποιούνται από την εφαρμογή. Η ποιότητα των αναλύσεων εξαρτάται άμεσα από την ποιότητα, την πληρότητα και την επικαιρότητα των διαθέσιμων στατιστικών στοιχείων. Εφόσον τα δεδομένα προέρχονται από εξωτερικές πηγές, τυχόν ελλείψεις, αλλαγές ή σφάλματα σε αυτά μπορούν να επηρεάσουν τα αποτελέσματα που παρουσιάζονται στον χρήστη.

Επιπλέον, οι αλγόριθμοι ανάλυσης που ενσωματώθηκαν στην εφαρμογή βασίζονται αποκλειστικά στα διαθέσιμα αριθμητικά χαρακτηριστικά ομάδων και παικτών. Παράγοντες όπως οι τακτικές επιλογές ενός προπονητή, οι τραυματισμοί, η ψυχολογική κατάσταση των αθλητών ή οι ιδιαίτερες συνθήκες ενός αγώνα δεν μπορούν να αποτυπωθούν πλήρως μέσα από τα στατιστικά δεδομένα. Για τον λόγο αυτό, τα αποτελέσματα των αλγορίθμων θα πρέπει να αντιμετωπίζονται ως υποστηρικτικά εργαλεία ανάλυσης και όχι ως απόλυτες αξιολογήσεις της πραγματικής αγωνιστικής εικόνας.

Τέλος, το TransferMind αναπτύχθηκε στο πλαίσιο μιας ακαδημαϊκής εργασίας και όχι ως εμπορικό προϊόν. Ως εκ τούτου, το εύρος των λειτουργιών, η ποσότητα των διαθέσιμων δεδομένων και οι δυνατότητες παραμετροποίησης είναι περιορισμένες σε σύγκριση με επαγγελματικές πλατφόρμες ποδοσφαιρικής ανάλυσης που χρησιμοποιούνται από μεγάλους αθλητικούς οργα-

νισμούς. Παρ' όλα αυτά, οι περιορισμοί αυτοί δεν επηρεάζουν τον βασικό στόχο της εργασίας, ο οποίος ήταν η δημιουργία ενός λειτουργικού και εύχρηστου συστήματος που αναδεικνύει τις δυνατότητες της ανάλυσης ποδοσφαιρικών δεδομένων μέσω τεχνικών εξόρυξης δεδομένων και μηχανικής μάθησης.

8.3 Μελλοντικές Επεκτάσεις

Παρότι το TransferMind καλύπτει τους βασικούς στόχους που τέθηκαν κατά τη διάρκεια της εργασίας, υπάρχουν αρκετές δυνατότητες περαιτέρω ανάπτυξης που θα μπορούσαν να βελτιώσουν τόσο το εύρος των λειτουργιών του όσο και την αξία των αποτελεσμάτων που προσφέρει στους χρήστες.

Μία πρώτη κατεύθυνση αφορά την επέκταση του διαθέσιμου συνόλου δεδομένων. Η ενσωμάτωση περισσότερων πρωταθλημάτων, διοργανώσεων και ιστορικών στοιχείων από προηγούμενες αγωνιστικές περιόδους θα επέτρεπε την πραγματοποίηση πιο ολοκληρωμένων αναλύσεων και τη διερεύνηση μακροχρόνιων τάσεων στην απόδοση ομάδων και παικτών. Παράλληλα, η αξιοποίηση επιπλέον κατηγοριών στατιστικών θα μπορούσε να εμπλουτίσει τα δεδομένα που χρησιμοποιούνται από τους αλγορίθμους της εφαρμογής.

Μια δεύτερη πιθανή επέκταση σχετίζεται με την ενσωμάτωση πρόσθετων τεχνικών μηχανικής μάθησης και εξόρυξης δεδομένων. Η ανάπτυξη μοντέλων πρόβλεψης αποτελεσμάτων αγώνων, η αξιολόγηση παικτών μέσω συστημάτων βαθμολόγησης ή η εφαρμογή πιο σύνθετων αλγορίθμων συσταδοποίησης θα μπορούσαν να προσφέρουν νέες δυνατότητες ανάλυσης. Παράλληλα, η αξιοποίηση σύγχρονων τεχνικών τεχνητής νοημοσύνης θα μπορούσε να συμβάλει στην παραγωγή πιο εξατομικευμένων και επεξηγηματικών αποτελεσμάτων.

Επίσης, ενδιαφέρον παρουσιάζει η αξιοποίηση δεδομένων γεγονότων αγώνα (event data), όπως πάσες, σουτ, ανακτήσεις μπάλας και θέσεις ενεργειών στον αγωνιστικό χώρο. Η ενσωμάτωση τέτοιων δεδομένων θα επέτρεπε πιο λεπτομερή ανάλυση της αγωνιστικής συμπεριφοράς ομάδων και παικτών, καθώς και την ανάπτυξη νέων μορφών οπτικοποίησης που θα παρείχαν βαθύτερη κατανόηση των δεδομένων.

Τέλος, μελλοντικές βελτιώσεις μπορούν να πραγματοποιηθούν και στο επίπεδο της εμπειρίας χρήσης. Δυνατότητες όπως η δημιουργία προσωπικού λογαριασμού, η αποθήκευση αγαπημένων ομάδων και παικτών, η εξαγωγή αναφορών ή η παρακολούθηση συγκεκριμένων δεικτών απόδοσης θα μπορούσαν να καταστήσουν την εφαρμογή περισσότερο εξατομικευμένη και χρήσιμη για τον τελικό χρήστη.

Συνολικά, το TransferMind μπορεί να αποτελέσει τη βάση για την ανάπτυξη ενός ακόμη πιο ολοκληρωμένου συστήματος ποδοσφαιρικής ανάλυσης. Οι επεκτάσεις αυτές θα μπορούσαν να ενισχύσουν τόσο το ερευνητικό όσο και το πρακτικό ενδιαφέρον της εφαρμογής, προσφέροντας νέες δυνατότητες αξιοποίησης των ποδοσφαιρικών δεδομένων και διευρύνοντας το κοινό στο οποίο απευθύνεται.

Βιβλιογραφία

[1] R. Mackenzie και C. Cushion, "Performance analysis in football: A critical review and implications for future research", *Journal of Sports Sciences*, τόμ. 31, αρθμ. 6, σσ. 639–676, 2013. doi: 10.1080/02640414.2012.746720. διεύθν.: https://doi.org/10.1080/02640414.2012.746720.

[2] H. Sarmento, R. Marcelino, M. T. Anguera, J. Campaniço, N. Matos και J. C. Leitão, "Match analysis in football: A systematic review", *Journal of Sports Sciences*, τόμ. 32, αρθμ. 20, σσ. 1831–1843, 2014. doi: 10.1080/02640414.2014.898852. διεύθν.: https://doi.org/10.1080/02640414.2014.898852.

[3] R. Rein και D. Memmert, "Big data and tactical analysis in elite soccer: Future challenges and opportunities for sports science", *SpringerPlus*, τόμ. 5, αρθμ. 1, σ. 1410, 2016. doi: 10.1186/s40064-016-3108-2. διεύθν.: https://doi.org/10.1186/s40064-016-3108-2.

[4] T. Decroos, L. Bransen, J. Van Haaren και J. Davis, "Actions Speak Louder than Goals: Valuing Player Actions in Soccer", στο *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, σσ. 1851–1861. doi: 10.1145/3292500.3330758. διεύθν.: https://doi.org/10.1145/3292500.3330758.

[5] H. Liu, W. Hopkins, A. M. Gómez και S. J. Molinuevo, "Inter-operator reliability of live football match statistics from OPTA Sportsdata", *International Journal of Performance Analysis in Sport*, τόμ. 13, αρθμ. 3, σσ. 803–821, 2013. doi: 10.1080/24748668.2013.11868690. διεύθν.: https://doi.org/10.1080/24748668.2013.11868690.

[6] Stats Perform, *ProVision for Analysis*, https://www.statsperform.com/team-performance/performance-solutions-for-football/match-analysis/provision-analysis/, Accessed: 2026-05-30, 2026.

[7] L. Pappalardo κ.ά., "A public data set of spatio-temporal match events in soccer competitions", *Scientific Data*, τόμ. 6, σ. 236, 2019. doi: 10.1038/s41597-019-0247-7. διεύθν.: https://doi.org/10.1038/s41597-019-0247-7.

[8] J. Ball, M. Huynh και M. C. Varley, "Comparing player rating systems as a metric for assessing individual performance in soccer", *Journal of Sports Sciences*, τόμ. 43, αρθμ. 7, σσ. 676–686, 2025. doi: 10.1080/02640414.2025.2471208. διεύθν.: https://doi.org/10.1080/02640414.2025.2471208.

[9] O. Müller, A. Simons και M. Weinmann, "Beyond crowd judgments: Data-driven estimation of market value in association football", *European Journal of Operational Research*, τόμ. 263, αρθμ. 2, σσ. 611–624, 2017. doi: 10.1016/j.ejor.2017.05.005. διεύθν.: https://doi.org/10.1016/j.ejor.2017.05.005.

[10] T. Peeters, "Testing the wisdom of crowds in the field: Transfermarkt valuations and international soccer results", *International Journal of Forecasting*, τόμ. 34, αρθμ. 1, σσ. 17–29, 2018. doi: 10.1016/j.ijforecast.2017.08.002. διεύθν.: https://doi.org/10.1016/j.ijforecast.2017.08.002.

[11] Hudl, *Hudl StatsBomb FAQ*, https://www.hudl.com/products/statsbomb/faq, Accessed: 2026-05-30, 2026.

[12] J. Gudmundsson και M. Horton, "Spatio-Temporal Analysis of Team Sports", *ACM Comput. Surv.*, τόμ. 50, αρθμ. 2, Απρ. 2017, issn: 0360-0300. doi: 10.1145/3054132. διεύθν.: https://doi.org/10.1145/3054132.

[13] M. Rico-González, J. Pino Ortega, A. Méndez, F. Clemente και A. Baca, "Machine learning application in soccer: a systematic review", *Biology of Sport*, τόμ. 40, σσ. 249–263, Ιαν. 2023. doi: 10.5114/biolsport.2023.112970.

[14] J. Brooke, "SUS: A Quick and Dirty Usability Scale", στο *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, I. L. McClelland και B. Weerdmeester, επιμελητές, Taylor & Francis, 1996, σσ. 189–194. διεύθν.: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale.

[15] D. Ancona, L. Franceschini, G. Delzanno, M. Leotta, M. Ribaudo και F. Ricca, "Towards Runtime Monitoring of Node.js and Its Application to the Internet of Things", *Electronic Proceedings in Theoretical Computer Science*, τόμ. 264, σσ. 27–42, Φεβ. 2018, issn: 2075-2180. doi: 10.4204/eptcs.264.4. διεύθν.: http://dx.doi.org/10.4204/EPTCS.264.4.

[16] A. J. Poulter, S. Johnston και S. Cox, "Using the MEAN Stack to implement a RESTful service for an Internet of Things Application", στο *IEEE World Forum on Internet of Things (14/12/15 - 16/12/15)*, Φεβ. 2016. διεύθν.: https://eprints.soton.ac.uk/383487/.

[17] M. Wicha και B. Pa'nczyk, "Performance Analysis of REST API Technologies Using Spring and Express.js Examples", *Journal of Computer Sciences Institute*, τόμ. 29, σσ. 352–359, 2023. doi: 10.35784/jcsi.3796. διεύθν.: https://ph.pollub.pl/index.php/jcsi/article/view/3796.

[18] R. T. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures", Διδακτορική διατρ., University of California, Irvine, 2000. διεύθν.: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[19] H. Ed-Douibi, J. L. C'anovas Izquierdo, A. G'omez, M. Tisi και J. Cabot, "EMF-REST: Generation of RESTful APIs from Models", στο *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, σειρά SAC '16, ACM, 2016, σσ. 1446–1453. doi: 10.1145/2851613.2851782. διεύθν.: https://arxiv.org/abs/1504.03498.

[20] M. Zhang, B. Marculescu και A. Arcuri, "Resource and Dependency Based Test Case Generation for RESTful Web Services", *Empirical Software Engineering*, τόμ. 26, αρθμ. 76, 2021. doi: 10.1007/s10664-020-09937-1. διεύθν.: https://link.springer.com/article/10.1007/s10664-020-09937-1.

[21] S. Raschka, J. Patterson και C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence", *Information*, τόμ. 11, αρθμ. 4, σ. 193, 2020. doi: 10.3390/info11040193. διεύθν.: https://www.mdpi.com/2078-2489/11/4/193.

[22] W. McKinney, "Data Structures for Statistical Computing in Python", στο *Proceedings of the 9th Python in Science Conference*, 2010, σσ. 56–61. doi: 10.25080/Majora-92bf1922-00a. διεύθν.: https://proceedings.scipy.org/articles/Majora-92bf1922-00a.

[23] C. R. Harris κ.ά., "Array Programming with NumPy", *Nature*, τόμ. 585, σσ. 357–362, 2020. doi: 10.1038/s41586-020-2649-2. διεύθν.: https://www.nature.com/articles/s41586-020-2649-2.

[24] F. Pedregosa κ.ά., "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, τόμ. 12, σσ. 2825–2830, 2011. διεύθν.: https://www.jmlr.org/papers/v12/pedregosa11a.html.

[25] P. Virtanen κ.ά., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, τόμ. 17, σσ. 261–272, 2020. doi: 10.1038/s41592-019-0686-2. διεύθν.: https://www.nature.com/articles/s41592-019-0686-2.

[26] F. Murtagh και P. Contreras, "Methods of Hierarchical Clustering", *CoRR*, τόμ. abs/1105.0121, 2011. arXiv: 1105.0121. διεύθν.: http://arxiv.org/abs/1105.0121.

[27] F. Murtagh και P. Legendre, "Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?", *Journal of Classification*, τόμ. 31, αρθμ. 3, σσ. 274–295, Οκτ. 2014, issn: 1432-1343. doi: 10.1007/s00357-014-9161-z. διεύθν.: http://dx.doi.org/10.1007/s00357-014-9161-z.

[28] S. Raschka, "MLxtend: Providing Machine Learning and Data Science Utilities and Extensions to Python's Scientific Computing Stack", *Journal of Open Source Software*, τόμ. 3, αρθμ. 24, σ. 638, 2018. doi: 10.21105/joss.00638. διεύθν.: https://joss.theoj.org/papers/10.21105/joss.00638.

[29] R. Agrawal και R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases", στο *Proceedings of the 20th International Conference on Very Large Data Bases*, Morgan Kaufmann, 1994, σσ. 487–499. διεύθν.: https://www.vldb.org/dblp/db/conf/vldb/vldb94-487.html.

[30] R. Agrawal, T. Imieli'nski και A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", στο *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ACM, 1993, σσ. 207–216. doi: 10.1145/170035.170072. διεύθν.: https://dl.acm.org/doi/10.1145/170035.170072.

[31] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", *Computing in Science & Engineering*, τόμ. 9, αρθμ. 3, σσ. 90–95, 2007. doi: 10.1109/MCSE.2007.55. διεύθν.: https://doi.org/10.1109/MCSE.2007.55.

[32] M. Stonebraker και G. Kemnitz, "The POSTGRES Next Generation DBMS", αδημοσίευτη ερευνητική εργασία UCB/ERL M91/62, Ιούλ. 1991. διεύθν.: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1991/1784.html.

[33] D. Yatsenko, E. Y. Walker και A. S. Tolias, *DataJoint: A Simpler Relational Data Model*, 2018. arXiv: 1807.11104 [cs.DB]. διεύθν.: https://arxiv.org/abs/1807.11104.

[34] H. Zhao, Z. Benomar, T. Pfandzelter και N. Georgantas, "Supporting Multi-Cloud in Serverless Computing", στο *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, IEEE, Δεκ. 2022, σσ. 285–290. doi: 10.1109/ucc56403.2022.00051. διεύθν.: http://dx.doi.org/10.1109/UCC56403.2022.00051.

[35] O. Kolesnikov, G. Golovko, V. Yastreba και Y. Piatyntsev, "Leveraging Cloud Technologies and Serverless Architecture for Efficient Web Development: A Case Study from Real-World Application", *Control, Navigation and Communication Systems. Academic Journal*, τόμ. 1, αρθμ. 75, σσ. 98–102, 2024. doi: 10.26906/SUNZ.2024.1.098. διεύθν.: https://journals.nupp.edu.ua/sunz/en/article/view/3277.

[36] A. K. Y. S. Mohamed, D. Auer, D. Hofer και J. Küng, "A systematic literature review of authorization and access control requirements and current state of the art for different database models", *International Journal of Web Information Systems*, τόμ. 20, αρθμ. 1, σσ. 1–23, Οκτ. 2023, issn: 1744-0084. doi: 10.1108/IJWIS-04-2023-0072. eprint: https://www.emerald.com/ijwis/article-pdf/20/1/1/9580267/ijwis-04-2023-0072.pdf. διεύθν.: https://doi.org/10.1108/IJWIS-04-2023-0072.

[37] C. Dar, M. Hershcovitch και A. Morrison, "RLS Side Channels: Investigating Leakage of Row-Level Security Protected Data Through Query Execution Time", *Proc. ACM Manag. Data*, τόμ. 1, αρθμ. 1, Μάι. 2023. doi: 10.1145/3588943. διεύθν.: https://doi.org/10.1145/3588943.

[38] M. Madsen, O. Lhoták και F. Tip, "A Semantics for the Essence of React", στο *34th European Conference on Object-Oriented Programming (ECOOP 2020)*, R. Hirschfeld και T. Pape, επιμελητές, σειρά Leibniz International Proceedings in Informatics (LIPIcs), τόμ. 166, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 12:1–12:26, isbn: 978-3-95977-154-2. doi: 10.4230/LIPIcs.ECOOP.2020.12. διεύθν.: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECOOP.2020.12.

[39] A. Mesbah και A. van Deursen, "An Architectural Style for Ajax", στο *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*, IEEE, Ιαν. 2007, σσ. 9–9. doi: 10.1109/wicsa.2007.7. διεύθν.: http://dx.doi.org/10.1109/WICSA.2007.7.

[40] G. Bierman, M. Abadi και M. Torgersen, "Understanding TypeScript", στο *ECOOP 2014 – Object-Oriented Programming*, σειρά Lecture Notes in Computer Science, τόμ. 8586, Springer, 2014, σσ. 257–281. doi: 10.1007/978-3-662-44202-9_11. διεύθν.: https://gavinbierman.github.io/assets/pdf/ecoop2014.pdf.

[41] Z. Gao, C. Bird και E. T. Barr, "To Type or Not to Type: Quantifying Detectable Bugs in JavaScript", στο *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017, σσ. 758–769. doi: 10.1109/ICSE.2017.75.

[42] M. Bostock, V. Ogievetsky και J. Heer, "D3: Data-Driven Documents", *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011. doi: 10.1109/TVCG.2011.185. διεύθν.: https://idl.uw.edu/papers/d3.

[43] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means", *Pattern Recognition Letters*, τόμ. 31, αρθμ. 8, σσ. 651–666, 2010. doi: 10.1016/j.patrec.2009.09.011.

[44] L. Kaufman και P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990. doi: 10.1002/9780470316801.

[45] scikit-learn developers. "Clustering", επίσκεψη 24 Μάι. 2026. διεύθν.: https://scikit-learn.org/stable/modules/clustering.html.

[46] A. C. Benabdellah, A. Benghabrit και I. Bouhaddou, "A Survey of Clustering Algorithms for an Industrial Context", *Procedia Computer Science*, τόμ. 148, σσ. 291–302, 2019. doi: 10.1016/j.procs.2019.01.022. επίσκεψη 29 Μάι. 2026. διεύθν.: https://doi.org/10.1016/j.procs.2019.01.022.

[47] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations", στο *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, τόμ. 1, University of California Press, 1967, σσ. 281–297. επίσκεψη 24 Μάι. 2026. διεύθν.: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fifth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992.

[48] X. Jin και J. Han, "K-Means Clustering", στο *Encyclopedia of Machine Learning*, C. Sammut και G. I. Webb, επιμελητές, Boston, MA: Springer, 2011, σσ. 563–564. doi: 10.1007/978-0-387-30164-8_425. επίσκεψη 29 Μάι. 2026. διεύθν.: https://doi.org/10.1007/978-0-387-30164-8_425.

[49] M. Ahmed, R. Seraj και S. M. S. Islam, "The k-means Algorithm: A Comprehensive Survey and Performance Evaluation", *Electronics*, τόμ. 9, αρθμ. 8, σ. 1295, 2020. doi: 10.3390/electronics9081295. επίσκεψη 29 Μάι. 2026. διεύθν.: https://doi.org/10.3390/electronics9081295.

[50] D. Arthur και S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding", στο *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, σσ. 1027–1035. επίσκεψη 24 Μάι. 2026. διεύθν.: https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf.

[51] P. J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis", *Journal of Computational and Applied Mathematics*, τόμ. 20, σσ. 53–65, 1987. doi: 10.1016/0377-0427(87)90125-7. επίσκεψη 29 Μάι. 2026. διεύθν.: https://doi.org/10.1016/0377-0427(87)90125-7.

[52] F. Murtagh και P. Contreras, "Algorithms for Hierarchical Clustering: An Overview, II", *WIREs Data Mining and Knowledge Discovery*, 2017. επίσκεψη 24 Μάι. 2026. διεύθν.: https://eprints.hud.ac.uk/id/eprint/32552/1/DWD_Fmurtagh_31.pdf.

[53] SciPy Developers. “scipy.cluster.hierarchy.linkage”, επίσκεψη 29 Μάι. 2026. διεύθν.: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html.

[54] J. H. Ward, “Hierarchical Grouping to Optimize an Objective Function”, *Journal of the American Statistical Association*, τόμ. 58, αρθμ. 301, σσ. 236–244, 1963. doi: 10.1080/01621459.1963.10500845. επίσκεψη 24 Μάι. 2026. διεύθν.: https://iv.cns.iu.edu/sw/data/ward.pdf.

[55] SciPy Developers. “scipy.cluster.hierarchy.dendrogram”, επίσκεψη 29 Μάι. 2026. διεύθν.: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html.

[56] B. Badhon, M. M. J. Kabir, S. Xu και M. Kabir, “A Survey on Association Rule Mining Based on Evolutionary Algorithms”, *International Journal of Computers and Applications*, τόμ. 43, αρθμ. 8, σσ. 775–785, 2021. doi: 10.1080/1206212X.2019.1612993. επίσκεψη 29 Μάι. 2026. διεύθν.: https://doi.org/10.1080/1206212X.2019.1612993.

[57] G. Li, X. Li, X. Wang και Y. Zhang, “A Survey on Particle Swarm Optimization for Association Rule Mining”, *Electronics*, τόμ. 11, αρθμ. 19, σ. 3044, 2022. doi: 10.3390/electronics11193044. επίσκεψη 29 Μάι. 2026. διεύθν.: https://doi.org/10.3390/electronics11193044.

[58] S. Raschka. “apriori: Frequent Itemsets via the Apriori Algorithm”, επίσκεψη 29 Μάι. 2026. διεύθν.: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/.

[59] S. Raschka. “association_rules: Generation of Association Rules from Frequent Itemsets”, επίσκεψη 29 Μάι. 2026. διεύθν.: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/.