Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Έξυπνη Επιλογή Υπηρεσιών Διαδικτύου με εξασφάλιση δυναμικών χαρακτηριστικών ποιότητας QoS (Quality of Service)** **Smart Web Services Selection utilizing dynamic Quality of Service Characteristics**
Ονοματεπώνυμο Φοιτητή	**Δήμητρα Κάλλη**
Πατρώνυμο	**Ρήγας**
Αριθμός Μητρώου	**ΜΠΠΛ18019**
Επιβλέπων	**Ευάγγελος Σακκόπουλος**

Ημερομηνία Παράδοσης **Ιούνιος 2024**

Τριμελής Εξεταστική Επιτροπή

Ευάγγελος Σακκόπουλος	Διονύσιος Σωτηρόπουλος	Κωνσταντίνα Χρυσαφιάδου
Αναπληρωτής Καθηγητής	Επίκουρος Καθηγητής	Επίκουρη Καθηγήτρια

Περιεχόμενα

Κατάλογος Εικόνων/ Σχημάτων

Περίληψη – Abstract

Περίληψη

Παρουσιάζεται μια Εφαρμογή η οποία βοηθά τους χρήστες να επιλέξουν την κατάλληλη Διαδικτυακή Υπηρεσία από ένα σύνολο 365 υπηρεσιών λαμβάνοντας υπόψιν τις ανάγκες και προτιμήσεις τους χρησιμοποιώντας προηγούμενες βαθμολογίες ή/και βαθμολογίες όμοιων χρηστών.

Ξεκινώντας παρουσιάζεται μια ιστορική αναδρομή των Υπηρεσιών Διαδικτύου και των Συστημάτων Συστάσεων. Αναλύεται ο Αλγόριθμος KNN Πλησιέστερων Γειτόνων και στην συνέχεια γίνεται χρήση του με σκοπό την ομαδοποίηση χρηστών των Διαδικτυακών Υπηρεσιών, έτσι ώστε να προσφέρουμε εγκυρότερες μελλοντικές προτάσεις. Παρουσιάζονται τα εργαλεία και οι μέθοδοι ανάπτυξης της εφαρμογής, όπως και κάποια σενάρια εκτέλεσης.

Η εφαρμογή αναπτύχθηκε στην γλώσσα προγραμματισμού C# με την βοήθεια κάποιων πακέτων NuGet που εξυπηρετούν την δημιουργία γραφημάτων, συνδυαστικά με μία Βάση Δεδομένων Postgresql.

Abstract

This paper introduces an application designed to aid users in identifying the most suitable web service from a collection of 365 options. This can be achieved by taking into consideration the End User's preferences, which are inferred from historical rating data.

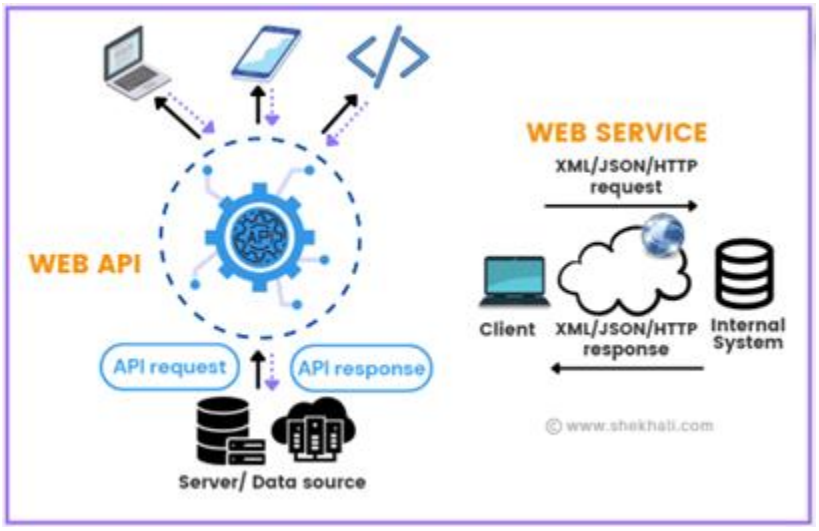
The dissertation begins with a retrospective examination of the evolution of Web Services and the development of recommendation systems. By employing the K-Nearest Neighbors (KNN) algorithm, the application effectively groups web service consumers, thereby enhancing the accuracy of subsequent service recommendations.

The software is developed in C# and leverages a selection of NuGet packages for graph generation, operating in conjunction with a Postgresql database backend.

1. Εισαγωγή

1.1 Σύντομη Παρουσίαση Υπηρεσιών Διαδικτύου (Web Services)

Οι Υπηρεσίες Διαδικτύου ή Υπηρεσίες Ιστού (Web Services) αποτελούν ένα σημαντικό τμήμα της σύγχρονης τεχνολογίας πληροφορικής και δικτύων επιτρέποντας την αποτελεσματική επικοινωνία μεταξύ εφαρμογών και συσκευών στο Internet [1]. Παράγουν ακατέργαστα δεδομένα (raw data) που θα χρησιμοποιηθούν από κάποια εφαρμογή σε αντίθεση με κάποια ιστοσελίδα που προορίζεται για κατανάλωση περιεχομένου. Αποτελούν έναν τύπο API (Application Programming Interface), ενώ αξίζει να σημειωθεί ότι δεν είναι όλες οι διεπαφές (APIs) υπηρεσίες διαδικτύου, παρά ένας τρόπος με τον οποίο μπορούν να εφαρμοστούν οι διασυνδέσεις (APIs).



Εικόνα 1. Υπηρεσίες Ιστού vs API [2]

Συνοπτικά η Emma Jagger σε σχετικό άρθρο περιγράφει τις υπηρεσίες διαδικτύου ως ένα είδος λογισμικού που επιτρέπει σε διάφορες εφαρμογές και συσκευές να επικοινωνούν μεταξύ τους. Χρησιμοποιούν πρωτόκολλα επικοινωνίας, όπως το HTTP (Hypertext Transfer Protocol) για τη μετάδοση δεδομένων και συνήθως λειτουργούν στο πλαίσιο της αρχιτεκτονικής του πελάτη-διακομιστή (Client/ Server), όπου ο πελάτης (Client) αιτείται δεδομένα από τον εξυπηρετητή/ διακομιστή (Server) [3].

Οι υπηρεσίες Ιστού χρησιμοποιούν SOAP και XML για να επιτρέπουν την επικοινωνία μεταξύ διαφορετικών γλωσσών προγραμματισμού και διαφορετικών εφαρμογών σε HTTP. Ένα μήνυμα SOAP είναι απλώς XML που αποστέλλεται μεταξύ πελάτη και διακομιστή.

Απλό πρωτόκολλο πρόσβασης αντικειμένων (SOAP)

Το SOAP χρησιμοποιεί την XML ως μέρος ενός τυποποιημένου πρωτοκόλλου επικοινωνίας που επιτρέπει την ανταλλαγή δομημένων πληροφοριών σε κατανεμημένα περιβάλλοντα. Το SOAP επιτρέπει σε εφαρμογές που εκτελούνται σε διαφορετικά λειτουργικά συστήματα και σε διαφορετικές γλώσσες προγραμματισμού να επικοινωνούν μεταξύ τους.

Για τα μηνύματα που αποστέλλονται και λαμβάνονται από έναν API πελάτη το SOAP παρέχει τέσσερις διαφορετικές διαστάσεις [4]:

- Φάκελος (Envelope): Ορίζει τη δομή του μηνύματος. Κάθε μήνυμα που αποστέλλεται ή λαμβάνεται έχει μια συγκεκριμένη δομή, η οποία είναι κρίσιμη για την επιτυχή επικοινωνία μέσω SOAP.
- Κωδικοποίηση (Encoding): Καθορίζει τους κανόνες για την έκφραση του τύπου των δεδομένων κι εξασφαλίζει ότι τα δεδομένα ερμηνεύονται σωστά από τον παραλήπτη.
- Αιτήματα (Requests): Ορίζει τον τρόπο δόμησης κάθε αίτησης SOAP API. Αυτό περιλαμβάνει τη δομή και την οργάνωση των δεδομένων στο αίτημα.
- Απαντήσεις (Responses): Καθορίζει τον τρόπο δόμησης κάθε απάντησης SOAP API. Αντίστοιχα με τα αιτήματα, οι απαντήσεις πρέπει να δομηθούν με τρόπο που να είναι κατανοητές και εύκολα ερμηνεύσιμες.

Πώς όμως γνωρίζει ο πελάτης πού ζει η υπηρεσία Ιστού;

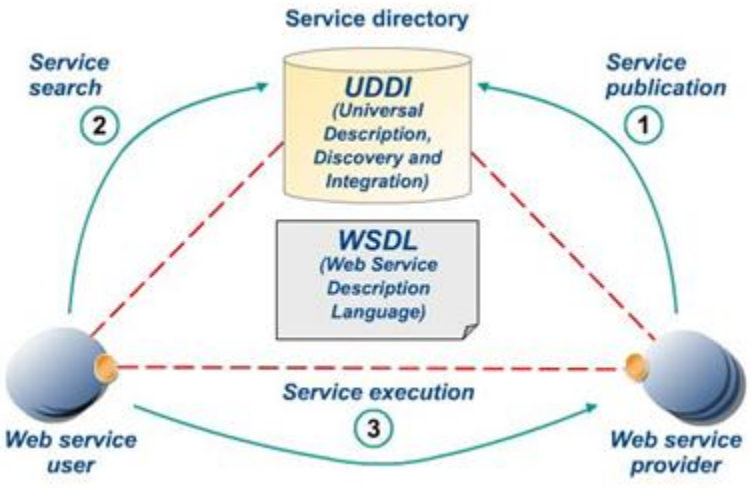
Γλώσσα περιγραφής υπηρεσιών Ιστού (WSDL)

Η Γλώσσα Περιγραφής Υπηρεσιών Ιστού ή WSDL είναι ένα έγγραφο XML που λέει στον πελάτη (Client) πού ζει η υπηρεσία Ιστού και τι κάνει. Το WSDL είναι ένα αρκετά περιεκτικό έγγραφο XML, που περιγράφει κάθε αλληλεπίδραση SOAP που δέχεται, επομένως δημιουργείται αυτόματα όταν δημιουργείται μια νέα υπηρεσία Ιστού για εξοικονόμηση χρόνου. Ένας πελάτης που θέλει να δει το WSDL μιας υπηρεσίας Ιστού στέλνει ένα αίτημα ως εξής: `http://webservice.example:1234/foo?WSDL` για να λάβει το αρχείο WSDL αυτής της υπηρεσίας Ιστού. Αυτό προϋποθέτει ότι η υπηρεσία Ιστού έχει ρυθμιστεί για να παρέχει το WSDL της. Για να μπορούμε να βρούμε ένα μήνυμα SOAP θα πρέπει να καλέσουμε το UDDI.

UDDI

Το Universal Description, Discovery and Integration (UDDI) είναι ένα μητρώο που βασίζεται σε XML μέσω του οποίου οι επιχειρήσεις σε όλο τον κόσμο μπορούν να καταχωρηθούν στο Διαδίκτυο. Αποτελεί επίσης ένα μηχανισμό εγγραφής και εντοπισμού εφαρμογών υπηρεσιών web. Το UDDI δίνει τη δυνατότητα στις επιχειρήσεις να δημοσιεύουν λίστες υπηρεσιών και να ανακαλύπτουν η μία την άλλη και να ορίζουν πώς

αλληλεπιδρούν οι υπηρεσίες ή οι εφαρμογές λογισμικού μέσω του Διαδικτύου. Έχει σχεδιαστεί για να καλείται από μηνύματα SOAP και να παρέχει πρόσβαση σε έγγραφα WSDL για υπηρεσίες web. Το UDDI είναι ένας κατάλογος APIs που παραθέτει διευθύνσεις και δυνατότητες για υπηρεσίες ιστού.



Εικόνα 2. Υποστήριξη Διαδικτυακών Υπηρεσιών - SAP NetWeaver [5]

Σε αυτό το σημείο θα πρέπει ν'αναλύσουμε την Υπηρεσιοστρεφή Αρχιτεκτονική (Service-oriented Architecture), η οποία σε συνδυασμό με τις υπηρεσίες διαδικτύου στο Cloud έχει μετασχηματίσει τον τρόπο με τον οποίο αναπτύσσονται, διαχειρίζονται και χρησιμοποιούνται οι υπηρεσίες στον ψηφιακό κόσμο. Θα εξετάσουμε επιπλέον την σημασία της δημοσίευσης (Publish), της αναζήτησης (Find) και της σύνδεσης (Bind) υπηρεσιών διαδικτύου στο Cloud, καθώς και τα πλεονεκτήματα που προσφέρει στον ψηφιακό κόσμο.

Υπηρεσιοστρεφής Αρχιτεκτονική (SOA):

Η αρχιτεκτονική SOA αναπαριστά έναν προσεκτικά δομημένο τρόπο για την ανάπτυξη εφαρμογών, όπου οι λειτουργίες και οι διαδικασίες παρουσιάζονται ως ανεξάρτητες υπηρεσίες. Κάθε υπηρεσία εκτελεί συγκεκριμένες λειτουργίες και μπορεί να επικοινωνεί με άλλες υπηρεσίες. Η αρχιτεκτονική SOA διευκολύνει την επαναχρησιμοποίηση και την ολοκλήρωση των υπηρεσιών σε πολλές εφαρμογές, ενισχύοντας την ευελιξία και την αποδοτικότητα [6]. Ας δούμε μια αναλυτικότερη επεξήγηση των βημάτων που ακολουθούνται παρακάτω:

Publish (Δημοσίευση):

Το στάδιο της Δημοσίευσης σε μια αρχιτεκτονική SOA αναφέρεται στην πράξη του προσφοράς υπηρεσιών ώστε να είναι διαθέσιμες για χρήση από άλλες εφαρμογές ή χρήστες. Οι Πάροχοι Υπηρεσιών

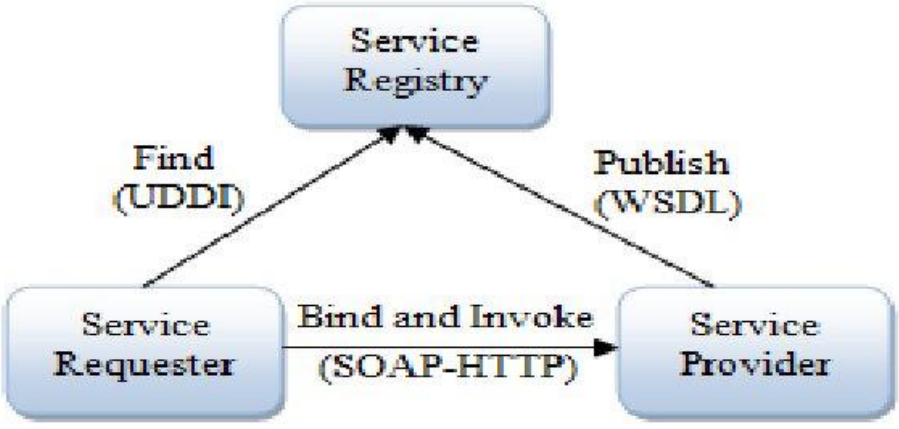
(Service Producers) δημοσιεύουν τις υπηρεσίες τους στον κατάλογο, τον οποίο μπορούν να αναζητούν άλλες εφαρμογές και χρήστες. Η δημοσίευση συνήθως συμπεριλαμβάνει την διατήρηση των μεταδεδομένων (Metadata) των υπηρεσιών, όπως την περιγραφή των λειτουργιών τους και τα πρότυπα επικοινωνίας.

Find (Αναζήτηση):

Η Αναζήτηση αφορά τη δυνατότητα εντοπισμού υπηρεσιών που έχουν δημοσιευθεί και είναι διαθέσιμες για χρήση. Αυτή η διαδικασία συνήθως εκτελείται μέσω καταλόγων υπηρεσιών ή κεντρικών καταλόγων μεταδεδομένων. Οι εφαρμογές (Service Consumers) μπορούν να ψάξουν τον κατάλογο για να βρουν τις υπηρεσίες που ταιριάζουν στις ανάγκες τους βάσει των μεταδεδομένων που περιέχονται σε αυτές τις υπηρεσίες. Μπορούν επίσης να επιλέξουν τον τρόπο με τον οποίο θα χρησιμοποιήσουν την υπηρεσία.

Bind (Σύνδεση):

Η Σύνδεση είναι η διαδικασία όπου μια εφαρμογή ή μια υπηρεσία χρησιμοποιεί μια άλλη υπηρεσία που έχει ανακαλυφθεί μέσω της αναζήτησης. Σε αυτό το στάδιο επιτυγχάνεται η επικοινωνία μεταξύ εφαρμογών και υπηρεσιών για την ανταλλαγή δεδομένων και την εκτέλεση λειτουργιών. Η σύνδεση συνήθως γίνεται με τη χρήση προκαθορισμένων προτύπων επικοινωνίας, όπως το HTTP και το SOAP.



Εικόνα 3. Οι θεμελιώδεις λειτουργίες και τα στοιχεία στην Υπηρεσιοστρεφή Αρχιτεκτονική [7]

1.2 Ανακάλυψη Υπηρεσιών Ιστού (Web Services Discovery)

Οι Μηχανισμοί Ανακάλυψης Υπηρεσιών Ιστού αντιπροσωπεύουν ένα σύνολο τεχνικών και προτύπων που επιτρέπουν στις εφαρμογές και τις υπηρεσίες να ανακαλύπτουν, εντοπίζουν και αλληλεπιδρούν με υπηρεσίες στον Παγκόσμιο Ιστό [8]. Αυτοί οι μηχανισμοί είναι απαραίτητοι στον σύγχρονο κυβερνοχώρο για την αποτελεσματική επικοινωνία, την ανταλλαγή δεδομένων και την ολοκλήρωση διαφορετικών υπηρεσιών [9].

Ένας από τους πιο γνωστούς μηχανισμούς ανακάλυψης υπηρεσιών είναι το UDDI (Universal Description, Discovery, and Integration). Το UDDI είναι όχι μόνο ένα πρωτόκολλο, αλλά και μια βάση δεδομένων που επιτρέπει σε παρόχους υπηρεσιών να δημοσιεύουν περιγραφές των υπηρεσιών τους, ενώ οι καταναλωτές μπορούν να αναζητούν και να βρίσκουν αυτές τις υπηρεσίες μέσω ενός κεντρικού καταλόγου. Αυτό επιτρέπει την εύρεση και την ενσωμάτωση των υπηρεσιών σε διάφορες εφαρμογές, επιτρέποντας την δημιουργία ευέλικτων και ανοικτών συστημάτων.

Δυστυχώς το UDDI δεν υιοθετήθηκε ευρέως, οπότε η υπηρεσία καταργήθηκε το 2010 [10].

Αξίζει σε αυτό το σημείο όμως ν' αναφερθούμε στην Ποιότητα της Παροχής Υπηρεσιών στον Παγκόσμιο Ιστό (Quality of Web Service Provisioning) που θα μας βοηθήσει να προχωρήσουμε την διερεύνηση και αξιολόγηση του επιπέδου ποιότητας των Web Services (QoWS), εξασφαλίζοντας ότι πληρούν τα απαιτούμενα πρότυπα και τις προσδοκίες[11]. Αυτή η έννοια είναι ζωτική στον σύγχρονο ψηφιακό χώρο, όπου οι υπηρεσίες Ιστού διαδραματίζουν σημαντικό ρόλο στη σύνδεση εφαρμογών, συστημάτων και συσκευών σε όλο το διαδίκτυο[12].

Υπάρχουν αρκετοί βασικοί παράγοντες που πρέπει να ληφθούν υπόψη κατά την αξιολόγηση της ποιότητας της παροχής υπηρεσιών ιστού:

- Απόδοση (Performance): Αποτελεί έναν από τους σημαντικότερους παράγοντες στην ποιότητα της υπηρεσίας ιστού. Περιλαμβάνει τον Χρόνο Απόκρισης της υπηρεσίας (Response Time), τη Διαθεσιμότητά της (Availability) και τη Δυνατότητά της να χειρίζεται ταυτόχρονα αιτήματα (Throughput). Μια υπηρεσία υψηλής απόδοσης εξασφαλίζει μια άνετη εμπειρία χρήστη και ελαχιστοποιεί την Καθυστέρηση (Latency).
- Αξιοπιστία (Reliability): Αφορά την συνέπεια μιας υπηρεσίας ιστού όταν καλείται αν παράγει τα αναμενόμενα αποτελέσματα. Οι τελικοί χρήστες πρέπει να μπορούν να βασίζονται στην υπηρεσία για να λειτουργεί όπως υποσχέθηκε, χωρίς απροσδόκητη διακοπή ή σφάλματα.
- Επεκτασιμότητα (Scalability): Μια υψηλής ποιότητας υπηρεσία ιστού πρέπει να είναι Επεκτάσιμη καλύπτοντας τις απαιτήσεις των χρηστών όταν υπάρχει αυξημένη επισκεψιμότητα. Θα πρέπει με άλλα λόγια να εξασφαλίζουμε ότι η προσφερόμενη υπηρεσία μπορεί να διαχειρίζεται αυξημένη κυκλοφορία χωρίς να επηρεάζεται η απόδοση ή η αξιοπιστία της.
- Ασφάλεια (Security): Η διασφάλιση της Εχεμύθειας των Δεδομένων (Access Control), της Ακεραιότητας (Data integrity) και της Αυθεντικοποίησης (Authentication) είναι απαραίτητη για την

προστασία ευαίσθητων πληροφοριών και τη διατήρηση της εμπιστοσύνης μεταξύ των χρηστών [13].

- Διαλειτουργικότητα (Interoperability): Οι υπηρεσίες ιστού πρέπει να σχεδιάζονται έτσι ώστε να λειτουργούν άψογα με άλλες υπηρεσίες και εφαρμογές. Αυτό περιλαμβάνει την τήρηση τυποποιημένων πρωτοκόλλων και μορφών δεδομένων για την ευκολότερη ενσωμάτωση σε διάφορα συστήματα.
- Χρηστικότητα (Usability): Πόσο εύκολα μπορούν οι χρήστες να αλληλεπιδράσουν με την Υπηρεσία Ιστού και να την ενσωματώσουν στις εφαρμογές τους; Ένα καλά τεκμηριωμένο και εύχρηστο API θα προτιμηθεί έναντι κάποιου άλλου που δεν προσφέρει επαρκή καθοδήγηση.
- Οικονομική αποδοτικότητα (Cost-effectiveness): Η παροχή υπηρεσιών ιστού πρέπει να είναι οικονομικά αποδοτική, προσφέροντας αξία για τους διαθέσιμους πόρους. Αυτό περιλαμβάνει τη βελτιστοποίηση της χρήσης των πόρων και την ελαχιστοποίηση των λειτουργικών δαπανών.
- Παρακολούθηση και Διαχείριση (Monitoring and Management): Σε μια προσπάθεια εξασφάλισης της συνεχούς προσφοράς ποιοτικών υπηρεσιών, θα πρέπει να εξασφαλίσουμε ότι η λύση περιλαμβάνει ισχυρά εργαλεία παρακολούθησης και διαχείρισης για την παρακολούθηση της απόδοσης, τον εντοπισμό προβλημάτων και τη διασφάλιση προληπτικής συντήρησης.

Λαμβάνοντας υπόψιν το γεγονός ότι ο απώτερος στόχος μας είναι η εύρεση της καταλληλότερης Υπηρεσίας Διαδικτύου ανάλογα με την ανάγκη, οι αριθμητικές τιμές που ορίζουν την Απόκριση, Διαθεσιμότητα, Απόδοση, Αναμονή και Αξιοπιστία θα πρέπει να εξεταστούν διεξοδικά έτσι ώστε να πραγματοποιηθεί αντιστοίχιση με τη χρήση συνδυασμένων ιστορικών δεδομένων QoS. Αυτή η προσέγγιση θα μας επιτρέψει την κατηγοριοποίηση, ομαδοποίηση και αντιστοίχιση Υπηρεσιών Ιστού.

1.3 Στόχος της διπλωματικής εργασίας

Η παρούσα διπλωματική εργασία αποσκοπεί στην διευκόλυνση εύρεσης της διαδικτυακής υπηρεσίας που κάποιοι χρήστες θ' αναζητήσουν σε ένα ευρύτερο σύνολο προσφερόμενων Διαδικτυακών υπηρεσιών. Χρησιμοποιώντας τον Αλγόριθμο ταξινόμησης KNN κατανοούμε το σύνολο των δεδομένων που έχουμε στην διάθεσή μας & προσφέρουμε μια λύση στο πρόβλημα ταξινόμησης νέων δειγμάτων δεδομένων, όπου θα πρέπει να βασιστούμε στις πιο κοντινές παρατηρήσεις (ή «γείτονες»). Ο αλγόριθμος εξετάζει τις κατηγορίες των k πιο κοντινών γειτόνων και αναθέτει στο νέο δείγμα την κατηγορία που δείχνει ν' ανταποκρίνεται καλύτερα στα χαρακτηριστικά της υπηρεσίας.

Στο πλαίσιο αυτό, η έρευνα επικεντρώνεται στην κατανόηση και βελτίωση των διαδικασιών επιλογής διαδικτυακών υπηρεσιών, με ειδική έμφαση στα ποιοτικά χαρακτηριστικά που αυτές προσφέρουν. Η προσέγγιση αυτή δίνει μεγάλη σημασία στην αξιοποίηση των Συστημάτων Συστάσεων βασισμένων στο Συνεργατικό Φιλτράρισμα, με στόχο την ακριβέστερη αναγνώριση και επιλογή των κατάλληλων Διαδικτυακών Υπηρεσιών.

Η αναγνώριση των ιδανικών υπηρεσιών μέσα από τη μεθοδολογία του Συνεργατικού Φιλτραρίσματος είναι ένας τρόπος για τη βελτίωση της εμπειρίας του χρήστη, παρέχοντας προσαρμοσμένες και στοχευμένες λύσεις.

1.4 Διάρθρωση της διπλωματικής εργασίας

Η παρούσα Διπλωματική Εργασία αποτελείται από 5 Κεφάλαια:

Στο Κεφάλαιο 1 παρουσιάζονται συνοπτικά οι Υπηρεσίες Διαδικτύου, ο Στόχος και η Δομή της Μεταπτυχιακής Διατριβής.

Στο Κεφάλαιο 2 παρουσιάζονται τα είδη των Συστημάτων Συστάσεων αναλυτικότερα με επεξήγηση των Αλγορίθμων που χρησιμοποιούνται ευρέως από την επιστημονική κοινότητα.

Στο Κεφάλαιο 3 μπορούμε να βρούμε την ανάλυση του Αλγορίθμου k-NN, ο οποίος ταξινομεί αντικείμενα με βάση την ομοιότητα τους με τα k πλησιέστερα δείγματα ενός συνόλου δεδομένων, προβλέποντας κατηγορίες ή τιμές.

Στο Κεφάλαιο 4 παρουσιάζεται η Υλοποίηση της Βάσης Δεδομένων και της εφαρμογής χρησιμοποιώντας τον Αλγόριθμο Κ Πλησιέστερων Γειτόνων (KNN) σε γλώσσα προγραμματισμού C#. Γίνεται αναφορά στα εργαλεία που χρησιμοποιήθηκαν και τα Αποτελέσματα των εκτελέσεων. Τέλος, εδώ μπορούμε να βρούμε τα Αρχεία πηγαίου κώδικα με την επεξήγηση.

Στο Κεφάλαιο 5 υπάρχουν τα Συμπεράσματα και Μελλοντικές Επεκτάσεις και οι Βιβλιογραφικές Αναφορές.

2. Συστήματα Συστάσεων (Recommender Systems)

Τα Συστήματα Συστάσεων (Recommender Systems), αναπτύχθηκαν για να παρέχουν προτάσεις σχετικά με προϊόντα, υπηρεσίες, ή περιεχόμενο που θα ενδιέφερε κάποιο συγκεκριμένο κοινό με βάση τις προτιμήσεις ή/και τη συμπεριφορά τους. Αξιοποιώντας ποικίλες μεθόδους και τεχνικές, παρέχουν εξατομικευμένες συμβουλές βελτιώνοντας την εμπειρία των χρηστών, όπως και την αλληλεπίδρασή τους με την εκάστοτε πλατφόρμα. Ένα από τα κύρια πλεονεκτήματά τους είναι η ικανότητά τους να ανακαλύπτουν προτιμήσεις που ο χρήστης ίσως δεν γνωρίζει.

Οι προτάσεις μπορούν να αφορούν πολλούς τομείς, όπως ταινίες, μουσική, βιβλία, προϊόντα, ειδήσεις, και πολλά άλλα. Τα Συστήματα Συστάσεων είναι τόσο βαθιά ριζωμένα στη ζωή μας που έχουμε πια συνηθίσει να περιμένουμε αυτές τις εξατομικευμένες προτάσεις στην καθημερινότητά μας [14].

Οι χρήστες καταναλώνουν περιεχόμενο καθημερινά από μια πληθώρα επιλογών, συνεπώς επιλέγουν την καταλληλότερη πλατφόρμα που θα "μαθαίνει" μαζί τους. Ένα τρανταχτό παράδειγμα αποτελούν οι πλατφόρμες κοινωνικής δικτύωσης, οι οποίες χρησιμοποιούν συστήματα συστάσεων για να καθορίσουν τι θα εμφανιστεί στο News Feed του χρήστη, να προτείνουν νέες επαφές κλπ.

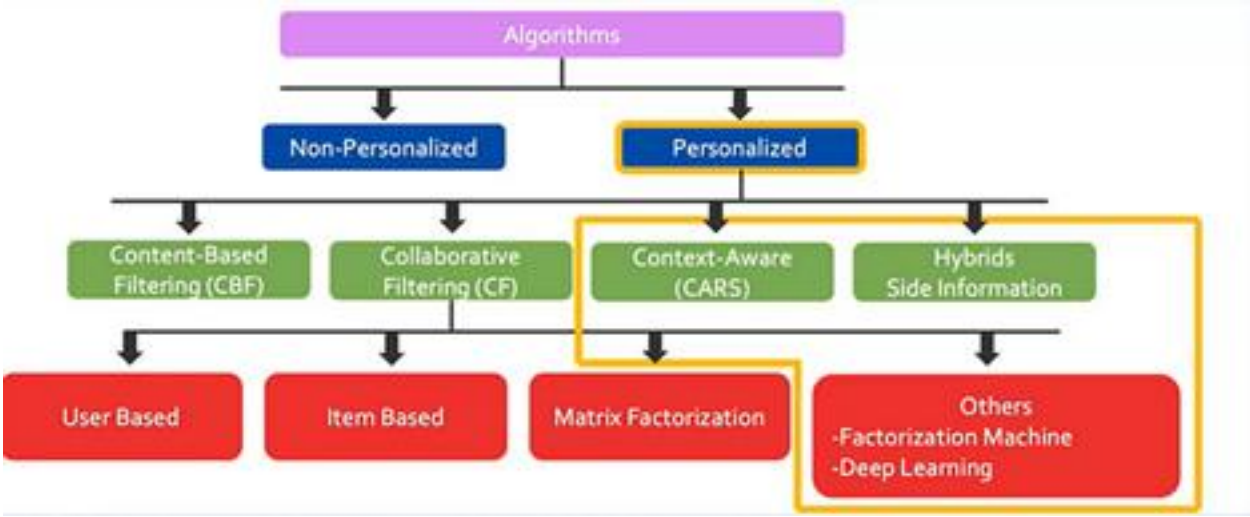
Στον επιχειρηματικό κόσμο, βλέπουμε πως βασική επιδίωξη των σύγχρονων τεχνολογικών κολοσσών (πχ Netflix, Meta, Google, Amazon, Microsoft κ.α.) είναι να κρατήσουν τους χρήστες δεσμευμένους με τα προϊόντα και τις υπηρεσίες τους για ν' αυξήσουν την κερδοφορία τους. Στο ηλεκτρονικό εμπόριο, για παράδειγμα, τα συστήματα συστάσεων μπορούν να κατευθύνουν τους πελάτες σε προϊόντα που είναι πιο πρόθυμοι να αγοράσουν με βάση τις προηγούμενες συμπεριφορές και αγορές τους. Οι επιχειρήσεις μπορούν επίσης να μάθουν πολλά για τους πελάτες τους με βάση αυτά τα δεδομένα και να τα χρησιμοποιήσουν για την ενημέρωση άλλων αποφάσεων.

Κατ' ουσίαν, τα Συστήματα Συστάσεων χειρίζονται τους χρήστες και τα αντικείμενα σαν δύο ξεχωριστές οντότητες, όπου κάθε χρήστης δίνει μια βαθμολογία (ή τιμή προτίμησης) σε ένα αντικείμενο (ή προϊόν) και οι αξιολογήσεις συλλέγονται έμμεσα (implicitly) ή άμεσα (explicitly). Κάθε χρήστης αλληλεπιδρά με τα αντικείμενα με τον ένα ή τον άλλο τρόπο (πχ ακούγοντας μουσική, βλέποντας ταινίες, αναζητώντας νέο περιεχόμενο κοκ), πράγμα που οδηγεί στην συλλογή έμμεσων (implicit) αξιολογήσεων. Αν πρόκειται για κάποια πλατφόρμα, ενδέχεται να συλλεχθούν δεδομένα σε σχέση με την επισκεψιμότητα ή τον χρόνο που περνάει κανείς σε κάποια σελίδα, την τοποθεσία του κοκ. Ο χρήστης μπορεί ακόμη να δώσει ρητά (explicitly) την αξιολόγησή του όταν παρουσιαστεί μια τιμή σε κάποια πεπερασμένη κλίμακα σημείων ή χαρακτηρισμένων τιμών διαστήματος. Οι αξιολογήσεις που παρέχει ο χρήστης διατάσσονται σε έναν πίνακα χρήστη-στοιχείου που ονομάζεται Πίνακας Ωφελιμότητας (Utility Matrix).

Ξεκινώντας την ανάλυσή τους, είναι σημαντικό ν' αναφέρουμε ότι τα Συστήματα Συστάσεων χωρίζονται σε δύο κατηγορίες, τις οποίες θα εξετάσουμε χωριστά παρακάτω:

- Τα Μη-Εξατομικευμένα Συστήματα Σύστασης (Non-Personalized RS)

- Τα Εξατομικευμένα Συστήματα Σύστασης (Personalized RS)



Εικόνα 4. Ανάλυση Κατηγοριών Αλγορίθμων που χρησιμοποιούνται από τα Συστήματα Σύστασης [15]

2.1 Μη-Εξατομικευμένα Συστήματα Συστάσεων (Non-Personalized Recommender Systems)

Τα Μη-Εξατομικευμένα Συστήματα Σύστασης δεν σχετίζονται με τις προτιμήσεις των χρηστών και θεωρούνται καταλληλότερα σε περιπτώσεις όπου η εξατομίκευση δεν είναι τόσο σημαντική ή όταν δεν υπάρχουν αρκετά δεδομένα για τον κάθε χρήστη. Συνοπτικά ξεχωρίζουν τα παρακάτω:

- Συστήματα συστάσεων βασισμένα σε γεωγραφική περιοχή: Μπορούν να προσφέρουν πληροφορίες βασισμένα στην τοποθεσία αναζήτησης, όπως τουριστικά αξιοθέατα ή εκδηλώσεις.
- Συστήματα συστάσεων βασισμένα στον χρόνο: Ένα καλό παράδειγμα είναι οι κατηγοριοποιημένες λίστες αναπαραγωγής σε υπηρεσίες ζωντανής μετάδοσης (streaming) μουσικής. Για παράδειγμα, μια λίστα αναπαραγωγής με τίτλο "Καλοκαιρινά Τραγούδια" περιέχει τραγούδια που είναι δημοφιλή την περίοδο του καλοκαιριού, αλλά δεν λαμβάνει υπόψη τις μουσικές προτιμήσεις του κάθε χρήστη. Ένα άλλο παράδειγμα θα ήταν μια εφαρμογή μετεωρολογίας που προτείνει ρούχα και αξεσουάρ που είναι κατάλληλα για τις πρόσφατες καιρικές συνθήκες.
- Συστήματα συστάσεων βασισμένα σε δημογραφικά στοιχεία: Σ' αυτή την περίπτωση οι συστάσεις προσαρμόζονται βάσει των δημογραφικών στοιχείων του χρήστη, όπως η ηλικία, η φυλή, και η γεωγραφική τοποθεσία. Για παράδειγμα, μια διαφημιστική πλατφόρμα μπορεί να

προβάλλει διαφημίσεις που ανταποκρίνονται στον στόχο αγοράς, με βάση το φύλο και την ηλικία του χρήστη.

Ενώ μπορεί κανείς να πιστεύει ότι η μη προσωποποιημένη σύσταση - όπως και τα περισσότερα δημοφιλή προϊόντα - είναι εύκολη, μπορεί να υπάρχουν πολλές παράμετροι που επηρεάζουν την απόδοσή της, όπως η περίοδος για την οποία υπολογίζεται η δημοτικότητα, ο τύπος (ή οι τύποι) αλληλεπίδρασης που λαμβάνονται υπόψιν. Επίσης, ας έχουμε κατά νου ότι περιπτώσεις όπως της αντιστοίχισης προϊόντος με προϊόν (item-to-item) που θεωρείται τυπική στο ηλεκτρονικό εμπόριο και είναι κατ' ουσίαν βασισμένη στα αντικείμενα και όχι τους χρήστες (item-driven versus user-driven).

2.2 Εξατομικευμένα Συστήματα Σύστασης (Personalized Recommender Systems)

Σε ένα συμβατικό Συστήμα Συστάσεων (Personalized Recommender System), οι προτάσεις που θα παρουσιαστούν στον χρήστη βασίζονται στις αξιολογήσεις που έχει λάβει αθροιστικά ένα αντικείμενο/ προϊόν κλπ. Εξετάζοντας την πρόσφατη Βιβλιογραφία παρόλ'αυτά, μπορούμε να εντοπίσουμε οφέλη εξατομικευμένων Συστημάτων Σύστασης σε νέους κλάδους, όπως σε εφαρμογές Ψυχικής Υγείας [16], στον έξυπνο Τουρισμό (e-Tourism) [17], σε εφαρμογές e-Learning [18] κ.ά.

Απολογιστικά μπορούμε να δούμε σε βιβλιογραφικές ανασκοπήσεις ότι σε αρχικά στάδια υλοποίησης θα πρέπει ν' αντιμετωπιστεί το πρόβλημα της ψυχρής εκκίνησης (Cold-start) και δεν υπολογίζονται περαιτέρω πληροφορίες σχετικά με το προφίλ του χρήστη κατά την διαμόρφωση των συστάσεων [19]. Γι'αυτό το λόγο θα άξιζε να εξετάσουμε τις κατηγορίες Εξατομικευμένων Συστημάτων Σύστασης ξεχωριστά.

2.2.1 Συστήματα Συστάσεων με βάση το Περιεχόμενο (Content-based/ Content-based Filtering Recommender Systems)

Χρησιμοποιώντας αυτή την μέθοδο (Content-Based και Content-Based Filtering Recommender System, CB RS ή CBF RS χάριν συντομίας) προσπαθούμε ν' αντιστοιχίσουμε τους Χρήστες μιας εφαρμογής με Αντικείμενα παρόμοια με αυτά που έχουν δηλώσει άμεσα ή έμμεσα προτίμηση στο παρελθόν. Σ' αυτή την περίπτωση η ομοιότητα δεν βασίζεται απαραίτητα σε συσχετίσεις που θα προέκυπταν από βαθμολόγηση, αλλά με βάση τα χαρακτηριστικά των αντικειμένων/προϊόντων. Βασιζόμενοι αποκλειστικά στον Χρήστη- Στόχο, ένα Content-based Recommender System εστιάζει σε μεγάλο βαθμό στις αξιολογήσεις του και τα χαρακτηριστικά των αντικειμένων που του αρέσουν. Ως εκ τούτου, οι άλλοι χρήστες έχουν ελάχιστο, αν όχι καθόλου, ρόλο στις προτάσεις που εκείνος,-η θα λάβει.

Αυτό το είδος συστήματος εξαρτάται από δύο πηγές δεδομένων [20]:

1. Μια περιγραφή των διαφόρων χαρακτηριστικών του αντικειμένου προερχόμενη απευθείας από τον κατασκευαστή του.

2. Ένα προφίλ χρήστη, το οποίο δημιουργείται από τα σχόλιά και την αλληλεπίδρασή του σε σχέση με διάφορα αντικείμενα/στοιχεία. Η μέθοδος έχει εφαρμοστεί σε πλήθος συστημάτων, συνεπώς η έννοια του Αντικειμένου μπορεί ν' αναφέρεται σε ταινίες, βιβλία, μαθήματα σε πλατφόρμες e-Learning κλπ, αλλά ακόμη και σε άρθρα, ειδήσεις, κ.ά. Η ανατροφοδότηση του χρήστη μπορεί να είναι ρητή (μέσω αξιολόγησης) ή σιωπηρή (κλικ, χρόνος παραμονής κα). Το προφίλ χρήστη συσχετίζει τα χαρακτηριστικά των διαφόρων στοιχείων με τα ενδιαφέροντα του.

Η κατηγοριοποίηση των προτιμήσεων του χρήστη βασίζεται στην δημιουργία ενός σετ σημαντικών λέξεων που μαρκάρουν κάθε αντικείμενο. Ένας καλό τρόπος επιλογής σημαντικών λέξεων μπορεί να επιτευχθεί με την χρήση του Ευρετικού Αλγορίθμου TF–IDF (term frequency–inverse document frequency), ο οποίος αποδίδει ένα μέτρο της σημαντικότητας μιας λέξης σε ένα έγγραφο προσαρμοσμένο έτσι ώστε να λαμβάνει υπόψιν ότι ορισμένες λέξεις εμφανίζονται συχνότερα. Ο όρος της Συχνότητας Όρου (Term frequency, tf(t,d)), αναφέρεται στην σχετική συχνότητα του όρου t εντός ενός εγγράφου d διαιρεμένη με το σύνολο των λέξεων που υπάρχουν στο έγγραφο όπως παρακάτω:

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

Ενώ ο όρος της Αντίστροφης Συχνότητας Εγγράφου (inverse document frequency, idf (t.d)) μετρά πόσο σημαντικός είναι ένας όρος σε ένα έγγραφο σε σχέση με μια συλλογή εγγράφων (δηλ. σε σχέση με ένα σώμα κειμένων, όπου N είναι ο συνολικός αριθμός των εγγράφων του σώματος):

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Συνεπώς μπορούμε να υπολογίσουμε το tf–idf ως εξής:

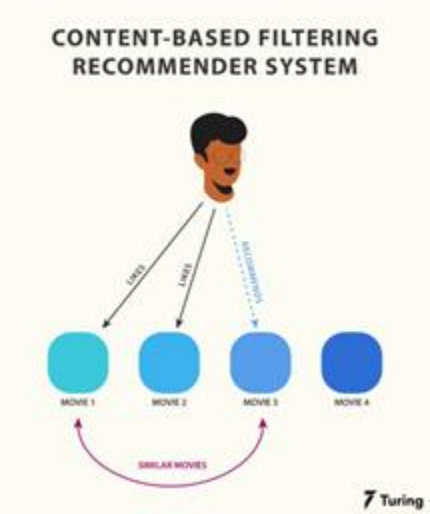
$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Ένα υψηλό βάρος στο tf-idf επιτυγχάνεται από μια υψηλή συχνότητα του όρου (στο συγκεκριμένο έγγραφο) και μια χαμηλή συχνότητα του όρου στο σύνολο της συλλογής εγγράφων- τα βάρη τείνουν επομένως να φιλτράρουν τους κοινούς όρους.

Για να επιτύχουμε στις προβλέψεις μας σε σχέση με το προφίλ του χρήστη, θα πρέπει να υπολογίσουμε την ομοιότητα συνημίτονου (cosine similarity), οποία είναι μια τιμή μεταξύ 0 και 1 και αναπαρίσταται από 2 διανύσματα χ και ψ σ' έναν δισδιάστατο άξονα. Το επιθυμητό είναι αυτή η τιμή να είναι πιο κοντά στο 1, υποδηλώνοντας μικρότερη γωνία μεταξύ των δύο διανυσμάτων (Χρήστη και Αντικειμένου). Όσο

μικρότερη η γωνία, τόσο περισσότερο μοιάζουν τα αντικείμενα μεταξύ τους, συνεπώς αποτελούν καλή πρόταση.

Θα ήταν δόκιμο να χρησιμοποιήσουμε ένα CB RS ή CBF RS σε σενάρια στα οποία έχουμε στην διάθεσή μας μια σημαντική ποσότητα πληροφοριών για τα χαρακτηριστικά και τις περιγραφές των αντικειμένων, οι οποίες σε πολλές περιπτώσεις μαρκάρονται από ετικέτες με λέξεις-κλειδιά,. Για παράδειγμα, σε περιπτώσεις όπου υπάρχει πλούτος κειμένου και μή-δομημένα πεδία, όπως η πρόταση περιεχομένου στο διαδίκτυο.



Εικόνα 5. Παράδειγμα Συνεργατικού Φιλτραρίσματος βασισμένου στο περιεχόμενο [21]

2.2.2 Συστήματα Σύστασης Συνεργατικού Φιλτραρίσματος (Collaborative Filtering Recommender Systems)

Το Συνεργατικό Φιλτράρισμα (Collaborative Filtering Recommender System, CF RS χάριν συντομίας) αναφέρεται στη διαδικασία καταλογογράφησης και συστάσεων που βασίζονται στη συνεργατική συμπεριφορά των χρηστών. Χρησιμοποιώντας CF RS προϋποθέτουμε ότι παρόμοιοι χρήστες θα εμφανίσουν όμοια μοτίβα αξιολόγησης και παρεμφερή αντικείμενα θα λάβουν κοντινές βαθμολογίες.

Αρχικά θα πρέπει να εντοπίσουμε μια την συλλογή του χρήστη X (ή ενός συνόλου χρηστών X) του οποίου οι προτιμήσεις, οι συμπάθειες και οι αντιπάθειες είναι παρόμοιες με αυτές του χρήστη Y. Συνεπώς το X ορίζεται ως γειτονιά του Y. Όταν προστεθούν νέα στοιχεία που αρέσουν στον X (ή στους περισσότερους χρήστες του X αν πρόκειται για σύνολο χρηστών), θα προταθούν στη συνέχεια και στον χρήστη Y. Η αποτελεσματικότητα ενός συνεργατικού αλγορίθμου εξαρτάται από το ποσοστό επιτυχίας

εύρεσης της γειτονιάς του χρήστη-στόχου. Η Αξιολόγηση Ομοιότητας (Similarity Evaluation) είναι ο καίριος παράγοντας στο CF και προσδιορίζεται από τον υπολογισμό της ομοιότητας μεταξύ χρηστών ή αντικειμένων (πχ χρησιμοποιώντας τους Αλγόριθμους Pearson, Jaccard, κ.λπ.) Σύμφωνα με τον Aggarwal [22] οι συνεργατικές προσεγγίσεις χωρίζονται και πάλι σε δύο τύπους: προσεγγίσεις που βασίζονται σε μοντέλα και προσεγγίσεις που βασίζονται στην μνήμη.

Συστήματα Σύστασης Συνεργατικού Φιλτραρίσματος βασισμένου σε Μοντέλα (Model-based)

Ξεκινάμε εξετάζοντας τις τεχνικές που χρησιμοποιούν Συστήματα Σύστασης Συνεργατικού Φιλτραρίσματος βασισμένου σε Μοντέλα (Model-based CF RS), τα οποία αποτελούν μια εξελιγμένη προσέγγιση των Συστήματα Συστάσεων. Εστιάζουν στην ανάλυση δεδομένων δημιουργώντας μοντέλα που μπορούν να προβλέπουν τις προτιμήσεις των χρηστών χωρίς να έχουν άμεση πρόσβαση σε όλα τα δεδομένα μια συγκεκριμένη χρονική στιγμή. Η ανάγκη για την δημιουργία τους προέκυψε λόγω της δυσκολίας μοντέλων Συνεργατικού Φιλτραρίσματος βασισμένου στην Μνήμη (Memory-based CF) να φορτώνουν στοιχεία από μια Βάση Δεδομένων αποτελεσματικά, ιδιαίτερα όταν υπήρχε χρήση από έναν ιδιαίτερα μεγάλο αριθμό χρηστών την ίδια στιγμή [23].

- Συσταδοποίηση (Clustering CF): Στόχος μας είναι η ομαδοποίηση στοιχείων και η ανίχνευση ομάδων παρόμοιων αντικειμένων ή δεδομένων σε ένα σύνολο. Προϋποθέτουμε ότι οι χρήστες που ανήκουν στην ίδια ομάδα θ' αποδώσουν κοντινές βαθμολογίες λόγω κοινών ενδιαφερόντων, ενώ μπορούμε να υπολογίσουμε το μέτρο ανομοιογένειας μεταξύ τους χρησιμοποιώντας την Ευκλείδια απόσταση ή την απόσταση Manhattan. Εάν υποθέσουμε ότι αναφερόμαστε ότι κάθε χρήστης αναπαρίσταται ως διάνυσμα αξιολόγησης που συμβολίζεται με ui = (ri1, ri2,..., rin), τότε το μέτρο ανομοιότητας μεταξύ δύο χρηστών είναι η μεταξύ τους απόσταση. Με άλλα λόγια, όσο μικρότερη η απόσταση μεταξυ των u1 και u2, τόσο μεγαλύτερη η ομοιότητά τους.

$$distance_{Euclidian}(u_1, u_2) = \sqrt{\sum_j (r_{1j} - r_{2j})^2}$$

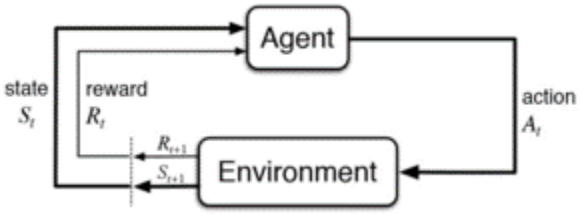
- Ταξινόμηση (Classification CF): Τα Μοντέλα Ταξινόμησης αναθέτουν τα αντικείμενα σε μια από προκαθορισμένες κατηγορίες ή κλάσεις με βάση τα χαρακτηριστικά και τις ετικέτες τους. Η διαδικασία εκπαίδευσης των μοντέλων ταξινόμησης περιλαμβάνει τη χρήση εκπαιδευτικών συνόλων δεδομένων και την αξιολόγηση της απόδοσής τους με χρήση μετρικών, όπως η

ακρίβεια και η ανάκληση. Εφαρμόζονται σε πολλούς επιστημονικούς τομείς, όπως η αναγνώριση προτύπων, η επεξεργασία εικόνας, η ιατρική διάγνωση, και η αναγνώριση φωνής.

Αξίζει ν' αναφερθούμε στο δίκτυο Bayes (Bayesian network), το οποίο είναι ένα κατευθυνόμενο ακυκλικό γράφημα που αποτελείται από ένα σύνολο κόμβων και ένα σύνολο κατευθυνόμενων τόξων, όπου κάθε τόξο αντιπροσωπεύει την εξάρτηση μεταξύ δύο κόμβων. Αναπαριστά τις σχέσεις μεταξύ μεταβλητών και χρησιμοποιεί τη θεωρία της πιθανότητας για να προβλέψει τις τιμές μεταβλητών βάσει των γνωστών τιμών άλλων μεταβλητών. Τα Bayesian Networks είναι χρήσιμα για την ανάλυση δεδομένων, την κατηγοριοποίηση, και την πρόβλεψη. Το Bayesian Collaborative Filtering είναι μια προσέγγιση στα συστήματα συστάσεων που βασίζεται στην αξιοποίηση της πιθανοτικής πληροφορίας. Αξιολογεί την πιθανότητα που ένας χρήστης θα επιλέξει ή θα αξιολογήσει ένα αντικείμενο, λαμβάνοντας υπόψη τις προτιμήσεις του και τις προηγούμενες αξιολογήσεις του. Το Bayesian CF επιτρέπει την εξατομίκευση των συστάσεων βάσει της πιθανοτικής προσέγγισης.

- Μοντέλο Ανάλυσης Λανθανουσών Κλάσεων (Latent Class Model CF): Συνδυάζει την έννοια των Μοντέλων Λανθανουσών Κλάσεων (Latent Class Model CF) με την ανάλυση δεδομένων για την παροχή εξατομικευμένων συστάσεων. Η υπόθεση εδώ είναι πως υπάρχουν κρυφές κλάσεις χρηστών και αντικειμένων. Αυτές οι κρυφές κλάσεις αντιστοιχούν σε προτιμήσεις και χαρακτηριστικά που δεν είναι ορατά στα δεδομένα και καλούμαστε να τις ανακαλύψουμε κατά την εκπαίδευση μαζί με τις σχέσεις μεταξύ εκείνων και των δεδομένων αξιολόγησης. Μπορούμε να δούμε παραδείγματα εφαρμογών στην εκπαιδευτική αξιολόγηση, σε ιατρικές γνωματεύσεις και στην αξιολόγηση απόδοσης σε απαιτητικά επαγγέλματα [24].

- Διαδικασία Απόφασεων Μαρκόβ (Markov decision process-based CF): Προϋποθέτουμε ότι το να δώσουμε συμβουλές στον χρήστη είναι μια πεπερασμένη διαδικασία που αποτελείται από πολλά στάδια. Ξεκινάμε με τον καθορισμό της τρέχουσας κατάστασης, η οποία αντιπροσωπεύει το περιβάλλον και την κατάσταση του χρήστη. Αυτή η κατάσταση μπορεί να περιλαμβάνει προηγούμενες αλληλεπιδράσεις, προτιμήσεις και πληροφορίες σχετικά με τα αντικείμενα εάν υπάρχουν ήδη αποθηκευμένα δεδομένα. Εάν πρόκειται για νέο χρήστη ή νέα αντικείμενα, τότε θα πρέπει να ξεκινήσει από κάποια τυχαία ή ορισμένη αφετηρία. Επιλέγοντας μια δράση από εκείνες που έχει διαθέσιμες, ο χρήστης μπορεί να εξερευνήσει τις επόμενες επιλογές που ανοίγονται σε κάθε νέα απόφαση/ βήμα. Μετά την επιλογή κάθε της δράσης, το σύστημα απολαμβάνει μια ανταμοιβή βάσει της ανταπόκρισης του χρήστη στις συστάσεις. Αυτή η ανταμοιβή μπορεί να βαθμολογηθεί ανάλογα με τον βαθμό ικανοποίησης του χρήστη. Εφαρμόζεται συνεπώς μια μαθησιακή διαδικασία για τη βελτίωση των μελλοντικών αποφάσεων. Κατά τη διάρκεια αυτής της διαδικασίας, το μοντέλο αποκτά γνώση για τη βέλτιστη ακολουθία δράσεων που θα οδηγήσει σε υψηλότερες ανταμοιβές.



Εικόνα 6. Σχηματική απεικόνιση της διαδικασίας αποφάσεων Markov [25]

- Παραγοντοποίηση Πίνακα (Matrix Factorization CF): Αρχικά ορίζουμε πως οι χρήστες και τα αντικείμενα αναπαρίστανται σε έναν πίνακα αξιολογήσεων, όπου οι γραμμές αντιστοιχούν στους χρήστες και οι στήλες στα αντικείμενα. Θα πρέπει στη συνέχεια να διασπάσουμε αυτόν τον πίνακα σε δύο χαμηλότερης διάστασης πίνακες: έναν για τους χρήστες και έναν για τα αντικείμενα. Αυτή η διάσπαση γίνεται με τη χρήση προσαρμοστικών αλγορίθμων. Κατά την εκπαίδευση, οι παράμετροι των ήδη διασπασμένων πινάκων προσαρμόζονται έτσι ώστε ο πολλαπλασιασμός τους να προσεγγίσει τις πραγματικές αξιολογήσεις των χρηστών για τα αντικείμενα. Οι αλγόριθμοι όπως οι Stochastic Gradient Descent (SGD) και ο Alternating Least Squares (ALS) είναι δημοφιλή εργαλεία για αυτό τον σκοπό. Μόλις ολοκληρωθεί η εκπαίδευση μπορούν να δοθούν νέες συστάσεις. Για παράδειγμα, αν ένας χρήστης αξιολογήσει μερικές ταινίες, ενώ έχει αφήσει κενές αξιολογήσεις για άλλες. Το μοντέλο χρησιμοποιεί τις γνωστές αξιολογήσεις για να εκπαιδεύσει τους πίνακες διασπάσεων. Έπειτα, προβλέπει τις κενές αξιολογήσεις, δημιουργώντας μια προσωποποιημένη λίστα ταινιών που ο χρήστης ίσως απολαύσει. Γνωστά παραδείγματα Παραγοντοποίησης Πινάκων είναι τα Funk MF, SVD++, Asymmetric SVD, Group-specific SVD κα.

Συστήματα Σύστασης Συνεργατικού Φιλτραρίσματος βασισμένου στην Μνήμη (Memory-based)

Οι Αλγόριθμοι Συνεργατικού Φιλτραρίσματος Βασισμένου στην Μνήμη (Memory-Based CF) ή Αλγόριθμοι Γειτνίασης (Neighborhood-based CF RS) βασίζονται στο γεγονός ότι παρόμοιοι χρήστες εμφανίζουν παρόμοια μοτίβα αξιολόγησης και παρόμοια αντικείμενα λαμβάνουν παρόμοιες βαθμολογίες. Υποθέτουμε ότι ο πίνακας βαθμολογιών συμβολίζεται με R (για Ratings) και είναι ένας μονοδιάστατος m × n πίνακας που περιέχει *m* αριθμό Χρηστών και *n* αριθμό Αντικειμένων. Επομένως, η βαθμολογία του χρήστη u (User) για το στοιχείο j συμβολίζεται με *ruj*. Στην περίπτωση που κληθούμε να δουλέψουμε σ'ένα παρόμοιο σύνολο στοιχείων, θα βρούμε καταχωρημένο μόνο ένα μικρό υποσύνολο τιμών το οποίο χρησιμοποιούμε για την εκπαίδευση του μοντέλου, ενώ οι μη καθορισμένες καταχωρήσεις του πίνακα αναφέρονται ως δεδομένα δοκιμής.

Συνεπώς στοχεύουμε στην εύρεση των πλησιέστερων γειτόνων ενός ενεργού χρήστη. Στον πίνακα αξιολόγησης R που αναφέρθηκε παραπάνω, οι γραμμές υποδηλώνουν χρήστες και οι στήλες υποδηλώνουν αντικείμενα και κάθε κελί είναι η βαθμολογία που έδωσε ένας χρήστης σε ένα αντικείμενο. Κάθε γραμμή αντιπροσωπεύει ένα διάνυσμα χρήστη ή ένα διάνυσμα αξιολόγησης. Το διάνυσμα αξιολόγησης του ενεργού χρήστη ονομάζεται διάνυσμα ενεργού χρήστη. Σε περίπτωση που ορισμένα κελιά που ανήκουν στο διάνυσμα του ενεργού χρήστη είναι κενά, αυτό σημαίνει ότι ο ενεργός χρήστης δεν έχει βαθμολογήσει τα αντίστοιχα στοιχεία και θα πρέπει να προβλεφθούν οι τιμές που λείπουν.

Ας εξερευνήσουμε τις δύο τεχνικές με τις οποίες μπορούμε να εφαρμόσουμε τους Αλγορίθμους Συνεργατικού Φιλτραρίσματος Βασισμένου στην Μνήμη:

- Μοντέλο Γειτνίασης με βάση τα στοιχεία (Item-Based Neighborhood Model/ Item-based Collaborative Filtering)
- Μοντέλο Γειτνίασης με βάση τον χρήστη (User-Based Neighborhood Model/ User-based Collaborative Filtering)

Μοντέλο Γειτνίασης με βάση τα στοιχεία (Item-Based Neighborhood Model)

Πρόκειται για μια προηγμένη τεχνική στα συστήματα συστάσεων που βασίζεται στην ανάλυση της ομοιότητας μεταξύ αντικειμένων (items) που υπάρχουν σε κάποια Βάση Δεδομένων σε σχέση μ'εκείνα για τα οποία κάποιος χρήστης έχει ήδη κάποια καταχωρημένη αξιολόγηση [26]. Παρακάτω αναλύουμε τα βήματα που ακολουθούνται για την εύρεση αντικειμένων και την πρόβλεψη για μελλοντικές προτάσεις:

- Ομοιότητα Αντικειμένου προς Αντικείμενο (Item-to-Item Similarity): Αυτή η τεχνική αφορά τον υπολογισμό της ομοιότητας μεταξύ διαφορετικών αντικειμένων στη βάση δεδομένων. Συνήθως, χρησιμοποιούνται μετρικές όπως οι συντελεστές συσχέτισης (Correlation Coefficients). Μια υψηλή ομοιότητα υπονοεί ότι τα αντικείμενα είναι παρόμοια και θ' αποτελούσαν καλή σύσταση για τον χρήστη με βάση τις καταχωρημένες προτιμήσεις του. Αξίζει ν' αναφερθεί ότι συχνά χρησιμοποιείται η Ομοιότητα Συνημίτονου (Cosine Similarity), η οποία υπολογίζει τη γωνία μεταξύ δύο διανυσμάτων αντικειμένων. Σε άλλη περίπτωση μπορούμε να εφαρμόσουμε τον συντελεστή Pearson (Pearson Correlation Coefficient), ο οποίος μετρά το βαθμό κατεύθυνσης και την ισχύ της σχέσης μεταξύ δύο μεταβλητών. Κυμαίνεται από -1 (τέλεια αντίθεση) έως 1 (τέλεια θετική συσχέτιση), με το 0 να υποδηλώνει απουσία συσχέτισης.
- Υπολογισμός Προβλέψεων (Prediction Computation): Αφού έχουν υπολογιστεί οι ομοιότητες μεταξύ των αντικειμένων, μπορούμε να υπολογίσουμε τις προβλέψεις που αφορούν χρήστες. Συνήθως, αυτό γίνεται με τον υπολογισμό του σταθμισμένου μέσου όρου των αξιολογήσεων παρόμοιων αντικειμένων, στα οποία λαμβάνονται υπόψιν και οι αξιολογήσεις χρηστών για παρόμοια αντικείμενα.

Μοντέλο Γειτνίασης με βάση τον χρήστη (User-Based Neighborhood Model)

Χρησιμοποιούμε Αλγορίθμους ομοιότητας μεταξύ χρηστών, για να εντοπίσουμε τους χρήστες εκείνους μέσα στο σύνολο που ερευνούμε που εμφανίζουν την μεγαλύτερη ομοιότητα με τον χρήστη-στόχο για τον οποίο υπολογίζονται οι προβλέψεις αξιολόγησης. Επομένως αρχικά θα πρέπει να υπολογίσουμε την ομοιότητά του με τους άλλους χρήστες και γι' αυτό τον λόγο ορίζουμε μια συνάρτηση ομοιότητας, έχοντας κατά νου την ανομοιομορφία των χρηστών και τις διαφορετικές κλίμακες αξιολογήσεων που μπορεί ν' ακολουθούν (πχ θα ήταν λογικό κάποιοι χρήστες να είναι πιο αυστηροί ενώ άλλοι πιο χαλαροί).

Εκτός της ομοιότητας, θα ήταν σημαντικό να εντοπίσουμε και τις διαφορές τους. Θα ήταν δύσκολο δύο χρήστες να έχουν αλληλεπιδράσει με τα ίδια αντικείμενα έχοντας διαμορφώσει τις ίδιες απόψεις οπότε θα ήταν ωφέλιμο να επικεντρωθούμε στις αμοιβαία παρατηρούμενες αξιολογήσεις για να εξάγουμε ασφαλή συμπεράσματα.

- Υπολογισμός Ομοιότητας & Εύρεση Κοινών Αξιολογήσεων (Similarity Calculation & Mutually Observed Ratings): Έχοντας δημιουργήσει τον Πίνακα Ωφέλειας (Utility matrix) μεταξύ αντικειμένων & χρηστών, θα υπολογίσουμε την ομοιότητα μεταξύ κάποιων χρηστών, χρησιμοποιώντας τον συντελεστή Pearson Correlation. Υψηλός συντελεστής Pearson υπονοεί ότι οι δύο χρήστες έχουν παρόμοιες προτιμήσεις, ενώ χαμηλός υπονοεί αντίθετες, άρα θα πρέπει να υπολογίζουμε τις Αξιολογήσεις τους σε όμοια αντικείμενα.
- Υπολογισμός Προβλέψεων & Παραγωγή Συστάσεων (Prediction Computation & Recommendation): Με τον συντελεστή Pearson και τις κοινές αξιολογήσεις, υπολογίζουμε τις προβλέψεις για τα αντικείμενα που ο συγκεκριμένος χρήστης δεν έχει αξιολογήσει. Αυτές οι προβλέψεις υποδηλώνουν το πόσο πιθανό είναι ο χρήστης να αξιολογήσει θετικά ή αρνητικά ένα αντικείμενο. Οι προβλέψεις χρησιμοποιούνται για να δημιουργήσουμε συστάσεις για τον χρήστη. Τα αντικείμενα που έχουν τις υψηλότερες προβλέψεις προτείνονται στον χρήστη ως πιθανές επιλογές.

2.2.3 Συστήματα Σύστασης με Υβριδικό Φιλτράρισμα (Hybrid Recommender Systems)

Έχοντας εξετάσει τα είδη Συστημάτων Σύστασης μεμονωμένα, παρακάτω θα βρούμε τις τεχνικές Υβριδικού Φιλτραρίσματος που χρησιμοποιούνται ευρέως για ν' αντιμετωπίσουν κάποιες από τις δυσκολίες που προκύπτουν από την εφαρμογή Συνεργατικού Φιλτραρίσματος ή Φιλτραρίσματος Βασισμένου στο Περιεχόμενο.

- Βαρισμένο Υβριδικό Φιλτράρισμα (Weighted Hybrid): Σε αυτήν την προσέγγιση, κάθε μέθοδος φιλτραρίσματος αξιολογείται με ένα βάρος και τα αποτελέσματα των διαφόρων μεθόδων συνδυάζονται με βάση αυτά τα βάρη. Αυτό επιτρέπει στους χρήστες να προσαρμόσουν τη σημασία των διαφόρων παραγόντων στις συστάσεις.

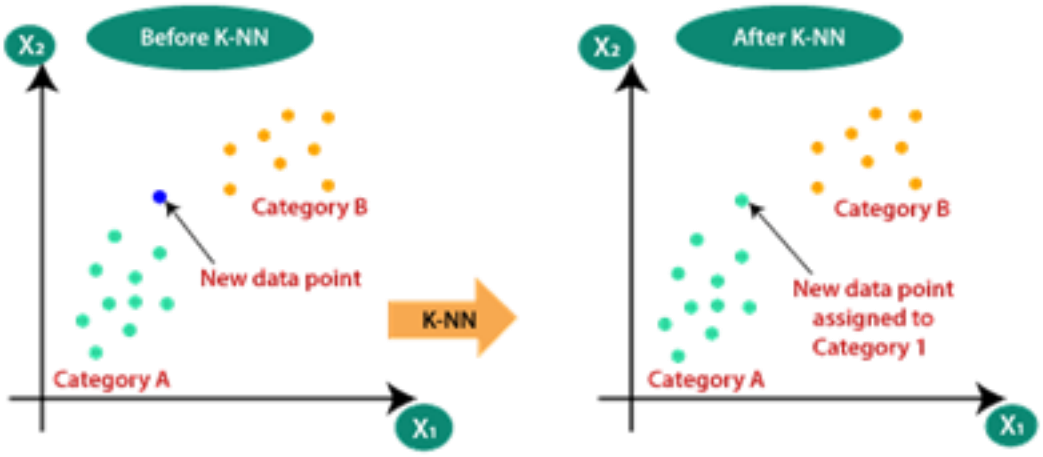
- Υβριδικά Συστήματα Με Εναλλαγή (Switching Hybrid Systems): Χρησιμοποιώντας αυτό το μοντέλο, διάφορες μέθοδοι φιλτραρίσματος λειτουργούν ως εναλλακτικές. Ανάλογα με τον χρήστη ή το περιβάλλον, επιλέγεται η καλύτερη μέθοδος για τη δημιουργία συστάσεων. Αυτό που παρέχει η συνεργατική τεχνική (Collaborative Filtering) εδώ είναι η ικανότητα να προκύπτουν συστάσεις που δεν είναι κοντά από σημασιολογική άποψη αλλά εξακολουθούν να είναι συναφείς [27].
- Μικτά Υβριδικά (Mixed Hybrid): Σε αυτό το μοντέλο, οι διάφορες μέθοδοι φιλτραρίσματος δημιουργούν ανεξάρτητα συστάσεις, και στη συνέχεια συνδυάζονται για να δημιουργηθεί η τελική σύσταση [28].
- Συνδυασμός Χαρακτηριστικών (Feature Combination): Σε αυτήν την προσέγγιση, τα χαρακτηριστικά από διάφορες πηγές (όπως το συνεργατικό φιλτράρισμα και το φιλτράρισμα βάσει περιεχομένου) συνδυάζονται σε ένα κοινό χώρο χαρακτηριστικών για τη δημιουργία συστάσεων.
- Ενίσχυση Χαρακτηριστικών (Feature Augmentation): Σε αυτήν την προσέγγιση, τα χαρακτηριστικά από μια μέθοδο φιλτραρίσματος ενισχύονται με πληροφορίες από μια άλλη μέθοδο για να δημιουργηθούν βελτιωμένες συστάσεις.
- Καταρροή (Cascade): Σε αυτό το μοντέλο, οι μέθοδοι φιλτραρίσματος εφαρμόζονται σε σειρά, με τα αποτελέσματα της μιας μεθόδου να χρησιμοποιούνται ως είσοδος για την επόμενη.
- Υβριδική Στρατηγική Επιπέδου Μετα-Συστήματος (Meta-level Hybrid Strategy): Σε αυτό το μοντέλο, ένα υψηλότερο επίπεδο συστήματος ελέγχει ποια μέθοδος φιλτραρίσματος θα χρησιμοποιηθεί για κάθε χρήστη ή κατάσταση.

3. Ο Αλγόριθμος των k-πλησιέστερων γειτόνων (k-NN)

3.1 Επεξήγηση του Αλγορίθμου των k-πλησιέστερων γειτόνων (k-NN)

Ο αλγόριθμος των k-πλησιέστερων γειτόνων (k-NN) χρησιμοποιείται στην Στατιστική για μη εποπτευόμενη Μηχανική Μάθηση. Αναπτύχθηκε αρχικά από την Evelyn Fix και τον Joseph Hodges το 1951 και στη συνέχεια διευρύνθηκε από τον Thomas Cover. Μπορούμε να βρούμε εφαρμογές του αλγορίθμου τόσο για ταξινόμηση (Classification) όσο και για παλινδρόμηση (Regression) με διαφορετικά αποτελέσματα ανάλογα με την χρήση.

Χρησιμοποιώντας τον Αλγόριθμο για Ταξινόμηση κάποιου στοιχείου σε σχέση με άλλα παρόμοια τα οποία έχουν ήδη ταξινομηθεί σ' ένα σύνολο δεδομένων θα λάβουμε σαν έξοδο μια κατηγορία (label) ανάλογα με το ποσοστό ομοιότητας ως προς τα στοιχεία των κοντινότερων γειτόνων του.



Εικόνα 7. Παράδειγμα ταξινόμησης χρησιμοποιώντας τον Αλγόριθμο KNN [29]

Στην περίπτωση της Παλινδρόμησης, θα περιμένουμε ότι το στοιχείο για το οποίο ενδιαφερόμαστε θα ταξινομηθεί ανάμεσα σε άλλα που εντάσσονται στην ίδια κατηγορία και το αποτέλεσμα που θα φτάσει σ' εμάς θα είναι μια τιμή κάποιας από τις ιδιότητές του.

Εάν θέλαμε να εφαρμόσουμε τον Αλγόριθμο, θα ξεκινούσαμε με την υπόθεση πως έχουμε ένα σύνολο ζευγών (X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n), όπου τα X_i είναι διανύσματα τιμών στον d-διάστατο χώρο R^d

και τα Y_i είναι ετικέτες κλάσης που παίρνουν τιμές από το σύνολο {1, 2}. Οι ετικέτες αυτές αντιστοιχούν σε δύο διαφορετικές πιθανοτικές κατανομές P_1 και P_2 για την κάθε κλάση.

Η διαδικασία ταξινόμησης ή παλινδρόμησης αρχίζει με τον καθορισμό μιας κανονικής (συνήθως Ευκλείδειας) απόστασης || . || στον R^d και ενός σημείου x στον ίδιο χώρο. Για να ταξινομηθεί το σημείο x, επιλέγουμε τα k πλησιέστερα δείγματα (X_i) από το σύνολο των δεδομένων εκπαίδευσης με βάση την απόστασή τους από το x. Αυτό γίνεται μέσω της αναδιάταξης των δεδομένων έτσι ώστε το (X_(1), Y_(1)) να είναι το πλησιέστερο στο x, το (X_(2), Y_(2)) το δεύτερο πλησιέστερο, και ούτω καθεξής, μέχρι το (X_(n), Y_(n)) που είναι το πιο μακρινό.

Στην φάση εξαγωγής χαρακτηριστικών (Feature Extraction Phase) αντλούνται περιγραφές των αντικειμένων, όπου συχνά επιλέγεται η εξαγωγή λέξεων-κλειδιών από τα υπάρχοντα δεδομένα, κυρίως λόγω της φυσικότητας και ευρείας διαθεσιμότητας αδόμητων κειμένων. Τα αντικείμενα μπορεί να περιγράφονται από πολλαπλά πεδία, όπως στην περίπτωση ενός πωλητή βιβλίων με περιγραφές και λέξεις-κλειδιά για το περιεχόμενο, τον τίτλο και τον συγγραφέα. Σε κάποιες περιπτώσεις, αυτές οι περιγραφές μετατρέπονται σε σύνολα λέξεων-κλειδιών, ενώ σε άλλες, εργάζονται με δομημένες πολυδιάστατες αναπαραστάσεις. Η αποτελεσματική ταξινόμηση απαιτεί κατάλληλο βάρος των πεδίων και σχετίζεται άμεσα με την επιλογή χαρακτηριστικών, όπου τα στοιχεία αξιολογούνται και βαρύνονται διαφορετικά ανάλογα με τη σημασία τους.

Θα μπορούσαμε επίσης να δούμε βελτίωση τόσο σε ποιότητα όσο και σε απόδοση χρησιμοποιώντας μεθόδους μείωσης των διαστάσεων μετατρέποντας τους αραιούς πίνακες αξιολόγησης σε πυκνές, χαμηλού-διάστασης αναπαραστάσεις με λανθάνοντες παράγοντες (latent factor). Με αυτό τον τρόπο μπορούμε να υπολογίσουμε την απόσταση χρηστών ακόμη και με λίγα κοινά αντικείμενα στις αξιολογήσεις τους αντιμετωπίζοντας το πρόβλημα της αραιότητας.

Μέσω της χρήσης του Αλγορίθμου KNN μπορούμε να επιτύχουμε σημαντική ευελιξία καθώς δεν απαιτεί προσδιορισμό μοντέλου a priori, επιτρέποντας την αναγνώριση μη γραμμικών σχέσεων ανάμεσα στα δεδομένα. Αποδεικνύεται ότι είναι ανθεκτικός σε θορυβώδη δεδομένα και φαίνεται πως η χρήση του σε εφαρμογές πραγματικού κόσμου δείχνει υψηλή πρακτική αξία.

Προχωρώντας με τα μειονεκτήματα βλέπουμε πως η αύξηση των διαστάσεων προκαλεί απώλεια της διακριτικής δύναμης της Ευκλείδειας απόστασης. Απαιτεί επίσης σημαντική υπολογιστική ισχύ και μνήμη κατά την κλασική υλοποίηση, καθώς απαιτεί την αποθήκευση και τη σύγκριση όλων των δεδομένων εκπαίδευσης κατά την κάθε πρόβλεψη, επιβραδύνοντας την πρόβλεψη σε μεγάλα σετ δεδομένων. Επιπλέον, η απόδοση του KNN μπορεί να επηρεαστεί από την ύπαρξη άνισων κατανομών κλάσεων ή αντιπροσωπευτικών δεδομένων.

4. Υλοποίηση του Αλγορίθμου KNN σε γλώσσα προγραμματισμού C#

4.1 Περιβάλλον και Εργαλεία Ανάπτυξης Εφαρμογής

Έπειτα από σχετική έρευνα, ένα μεγάλο μέρος υλοποιήσεων του Αλγορίθμου KNN στο διαδίκτυο είναι στην γλώσσα προγραμματισμού Python, κυρίως λόγω της δημοφιλίας μεταξύ των επιστημόνων δεδομένων, οι οποίοι και δημιουργούν το αντίστοιχο διαδικτυακό περιεχόμενο. Αναγνωρίζοντας τις προκλήσεις, η επιλογή της γλώσσας προγραμματισμού C# πραγματοποιήθηκε για να προσφέρει μια εναλλακτική πρόταση [30], καθώς η δυνατότητα φιλοξενίας εφαρμογών .NET σε υπηρεσίες νέφους παρουσιάζει αυξανόμενο ενδιαφέρον σε μεγάλους Οργανισμούς.

- .NET 8.0 Framework: Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε η νεότερη έκδοση του .NET 8.0 Framework, η οποία φέρνει μια σειρά από καινοτόμες βελτιώσεις και επεκτάσεις, καθιστώντας την ιδιαίτερα προσαρμοστική και αποδοτική για σύγχρονες εφαρμογές. Στην σχετική διαδικτυακή ενημέρωση της Microsoft επιβεβαιώνουμε πως οι βελτιώσεις αφορούν την αποδοτικότερη επεξεργασία δεδομένων διασφαλίζοντας την ασφάλεια και ακεραιότητά τους (Serialization). Θα βρούμε επίσης μεγαλύτερη ευελιξία και ακρίβεια στη διαχείριση χρόνου στις εφαρμογές, αφού υπήρξε επέκταση των δυνατοτήτων που σχετίζονται με την χρονική αφαίρεση (Time abstraction). Σημαντικές είναι και οι βελτιώσεις στην υποστήριξη UTF8 (UTF8 improvements), που προσφέρουν καλύτερη απόδοση και συμβατότητα με διάφορες κωδικοποιήσεις κειμένου. Έχουν εισαχθεί νέες μέθοδοι που καθιστούν την ανάπτυξη λειτουργιών με τυχαιότητα πιο αξιόπιστες και ασφαλείς (Methods for working with randomness). Επιπλέον, οι εστιασμένες στην απόδοση τύποι (Performance-focused types), συνεισφέρουν στη βελτιστοποίηση των εφαρμογών, ενισχύοντας την ταχύτητα και την απόδοση του λογισμικού. Η ενσωμάτωση βελτιωμένων μηχανισμών από τις βιβλιοθήκες System.Numerics και System.Runtime.Intrinsics παρέχει πιο ισχυρές λειτουργίες για αριθμητικούς υπολογισμούς και ενσωματωμένες λειτουργίες, ενώ οι βελτιώσεις στην επικύρωση δεδομένων (Data validation) και στην κρυπτογραφία ενισχύουν την ασφάλεια και την ακεραιότητα των εφαρμογών. Στον τομέα του Networking, προσφέρονται αναβαθμισμένες δυνατότητες για αποδοτικότερη διαχείριση δικτυακών συνδέσεων, ενώ οι βελτιωμένες stream-based ZipFile μέθοδοι παρέχουν πιο ευέλικτη διαχείριση συμπιεσμένων αρχείων.

  Το .NET Framework σε συνδυασμό με την γλώσσα προγραμματισμού C#, η οποία είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού που υποστηρίζει πολλαπλά προγραμματιστικά μοντέλα, μου έδωσε την δυνατότητα επανασχεδιασμού της λύσης από σχετικές πηγές που προσέφεραν παραδείγματα σε Python. Η C# σχεδιάστηκε από τον Anders Hejlsberg της Microsoft

το 2000 και αργότερα εγκρίθηκε ως διεθνές πρότυπο από την Ecma και την ISO/IEC. Παρόλο που αρχικά ήταν κλειστού κώδικα, η Microsoft αργότερα ανέπτυξε το Visual Studio Code, τον Roslyn και την ενοποιημένη πλατφόρμα .NET, υποστηρίζοντας το C# με ελεύθερο, ανοιχτού κώδικα λογισμικό.

- ScottPlot Library: Η βιβλιοθήκη ScottPlot αποτελεί μια σύγχρονη και ευέλικτη βιβλιοθήκη για τη δημιουργία γραφημάτων που χρησιμοποιείται από εφαρμογές που βασίζονται στη γλώσσα προγραμματισμού C#. Μια από τις βασικές λειτουργικότητές της είναι η δυνατότητα δημιουργίας μιας ευρείας γκάμας γραφημάτων, όπως γραμμικά, ιστογράμματα, διαγράμματα διασποράς κα, επιτρέποντάς μας να παρουσιάσουμε τα αποτελέσματα που λάβαμε με τον πιο αποτελεσματικό τρόπο. Μπορεί να χρησιμοποιηθεί σε διάφορα είδη εφαρμογών, από desktop εφαρμογές μέχρι πιο περίπλοκα συστήματα ανάλυσης δεδομένων.

- OxyPlot Library: Χρησιμοποιήθηκε επίσης εναλλακτικά η βιβλιοθήκη Oxyplot για την απεικόνιση και ανάλυση δεδομένων λόγω της προσαρμοστικότητάς της. Η βιβλιοθήκη Oxyplot είναι συμβατή με διάφορα περιβάλλοντα, όπως Windows Forms, WPF, Xamarin και ASP.NET. Είναι ανοικτού κώδικα, προσφέροντας μια επεκτάσιμη λύση για την οπτικοποίηση δεδομένων, που ενθαρρύνει την συμμετοχή της κοινότητας συμβάλλοντας στην συνεχή βελτίωσή της.

- PostgreSQL RDBMS: Η PostgreSQL είναι μια δωρεάν, ανοιχτού κώδικα σχεσιακή βάση δεδομένων, συμβατή με SQL. Οι χρήστες μπορούν να την εγκαταστήσουν σε διάφορα λειτουργικά συστήματα και υποστηρίζει πολλές γλώσσες προγραμματισμού. Προσφέρει ακεραιότητα δεδομένων και υποστηρίζει πολλαπλούς τύπους δεδομένων. Είναι επεκτάσιμη και ασφαλής, υποστηρίζοντας τόσο συγχρονισμένη όσο και ασύγχρονη αναπαραγωγή δεδομένων. Η ευελιξία και η ασφάλεια της καθιστούν κατάλληλη για εφαρμογές διαδικτύου και ευαίσθητα δεδομένα.

  Για τη διαχείριση της PostgreSQL, υπάρχουν εργαλεία όπως το phpPgAdmin και το pgAdmin για διαχείριση μέσω GUI, το PgFouine για ανάλυση logs και το pgDevOps για δημιουργία SQL queries και παρακολούθηση βάσεων δεδομένων.

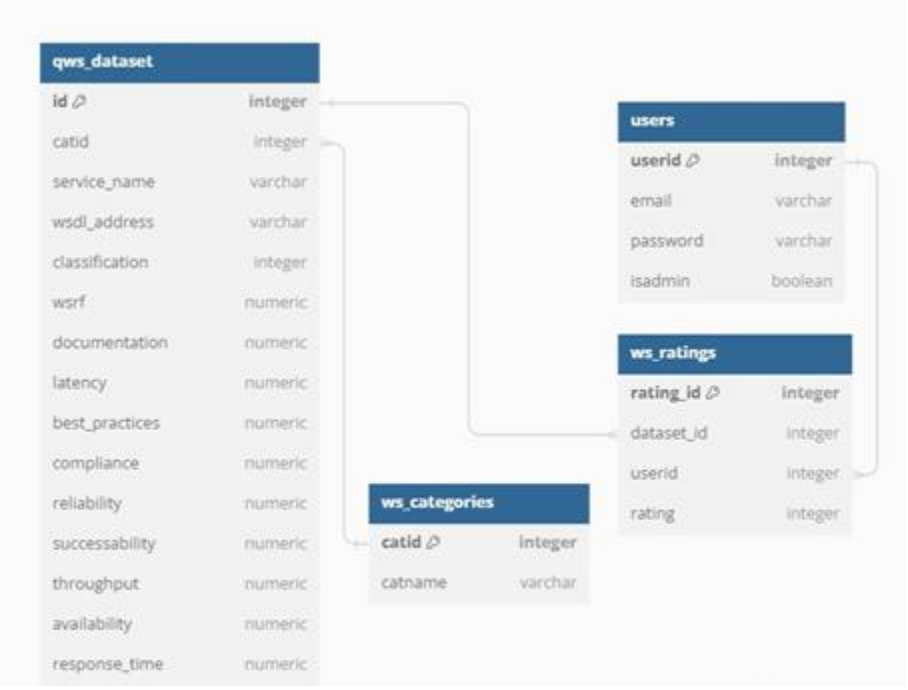
  Εξετάζοντας τα πλεονεκτήματα χρήσης της, η PostgreSQL θεωρείται ευκολότερη στην χρήση από άλλες σχεσιακές Βάσεις Δεδομένων, έχει χαμηλές απαιτήσεις συντήρησης, υπάρχει ελεύθερη διαθεσιμότητα του κώδικα για προσαρμογή στις ανάγκες της επιχείρησης σε περίπτωση εμπορικής χρήσης, υπάρχει δυνατότητα αλληλεπίδρασης με δυναμικές εφαρμογές διαδικτύου κι εμφανίζει υψηλή αντοχή σε κινδύνους.

4.2 Υλοποίηση Βάσης Δεδομένων

Ας ξεκινήσουμε με τον σχεδιασμό της Βάσης Δεδομένων, καθώς θα μας βοηθήσει να κατανοήσουμε καλύτερα τα δεδομένα που θα διαχειριστούμε και την υλοποίηση στο σύνολο.

Για την υλοποίηση και τον σχεδιασμό της Βάσης Δεδομένων (και κατά συνέπεια της εφαρμογής) χρησιμοποιήθηκαν τα δεδομένα που μας παρέχει το QWS Dataset [30], το οποίο αποτελεί μια συλλογή από δεδομένα αξιολόγησης για 365 διαδικτυακές υπηρεσίες, χρησιμοποιώντας μετρήσεις για την αξιοπιστία, την απόδοση και άλλους παράγοντες που εξετάζουμε κατά την αξιολόγηση της ποιότητας της παροχής υπηρεσιών ιστού, όπως αναφέρθηκαν στην σελίδα 8 κι 9. Το QWS Dataset έχει χρησιμοποιηθεί σε ακαδημαϊκά περιβάλλοντα και βιομηχανικές εφαρμογές για τη βελτίωση των συστημάτων SOA (Service-Oriented Architecture).

Σε ό,τι αφορά την κατασκευή της Βάσης Δεδομένων, υπάρχουν 4 πίνακες, των οποίων οι σχέσεις και αλληλεξαρτήσεις φαίνονται στο παρακάτω διάγραμμα:



Εικόνα 8. Διάγραμμα Σχέσης Οντοτήτων

Τα δεδομένα του QWS Dataset έχουν εισαχθεί στον πίνακα qws_dataset, στον οποίο έχει προστεθεί το πεδίο catid, το οποίο κάνει την αντιστοίχιση μεταξύ Κατηγοριών και Υπηρεσιών Ιστού.

Για να μπορούμε να συλλέξουμε επαρκή αποτελέσματα χρησιμοποιήθηκαν 10 κατηγορίες όπως παρακάτω:

1. WebService
2. Entertainment
3. Email
4. Weather
5. IT & Development
6. Finance
7. Religion
8. Location
9. News & Telcos
10. Academic

Από την αρχή φάνηκε να υπάρχει η ανάγκη βαθύτερης κατανόησης των δεδομένων, έτσι ώστε να βεβαιωθούμε ότι τα αποτελέσματα που λαμβάνουμε από την εφαρμογή είναι ορθά και ανταποκρίνονται στην πραγματικότητα. Ως εκ τούτου, δημιουργήθηκαν οι παρακάτω Όψεις (Views) από τους αρχικούς πίνακες για να υπάρξει ξεκάθαρος επιμερισμός δεδομένων και ταξινόμηση των Διαδικτυακών Υπηρεσιών σύμφωνα με την κατηγορία στην οποία ανήκουν.

Εικόνα 9. Όψεις βαθμολογιών χρηστών για υπηρεσίες διαδικτύου ταξινομημένες ως προς την κατηγορία

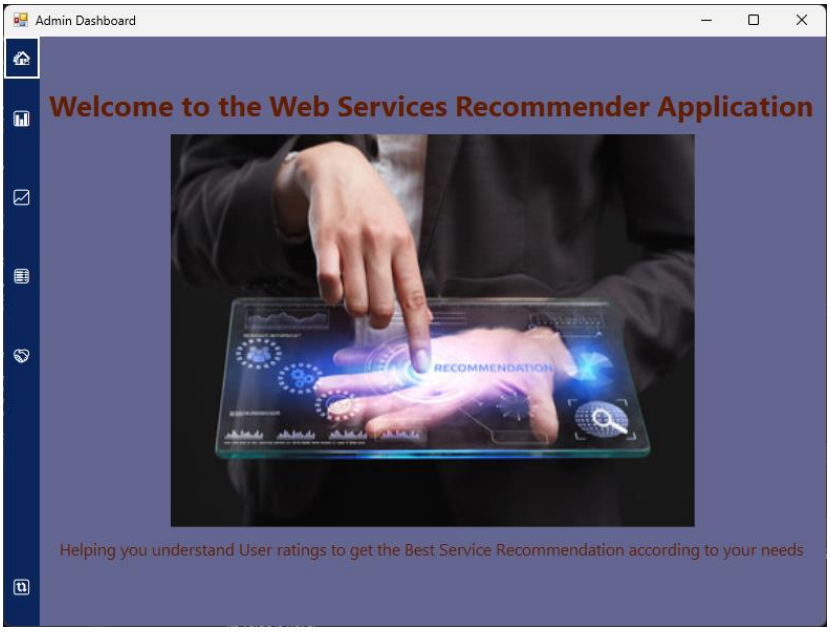
Εικόνα 10. Όψεις υπηρεσιών διαδικτύου ταξινομημένες ως προς την κατηγορία

Εικόνα 11. Ταξινόμηση Καλύτερων/ Χειρότερων Υπηρεσιών ως προς την διαθεσιμότητα

4.3 Υλοποίηση Εφαρμογής και Παραδείγματα Χρήσης

Παρακάτω μπορούμε να δούμε το Κύριο Μενού της εφαρμογής, όπου δίνονται οι παρακάτω επιλογές στον χρήστη από το μενού αριστερά:

- Πλοήγηση στην Αρχική Οθόνη (Home Button).
- Ανάκτηση και οπτικοποίηση αξιολογήσεων των χρηστών (User Ratings Graph Generator).
- Ανάκτηση και εμφάνιση των καλύτερων ή χειρότερων υπηρεσιών σύμφωνα με τη διαθεσιμότητα τους, ταξινομημένες ανά κατηγορία (Best/Worst Services According to their Availability).
- Εκτέλεση ταξινόμησης Πλησιέστερων γειτόνων (KNN Classification).
- Οπτικοποίηση των Ομοιοτήτων Χρηστών (User Similarities Visualization).
- Τέλος, οι χρήστες μπορούν να συνδεθούν, έτσι ώστε να λάβουν μια Σύσταση Υπηρεσίας (User Login Service Recommendation).

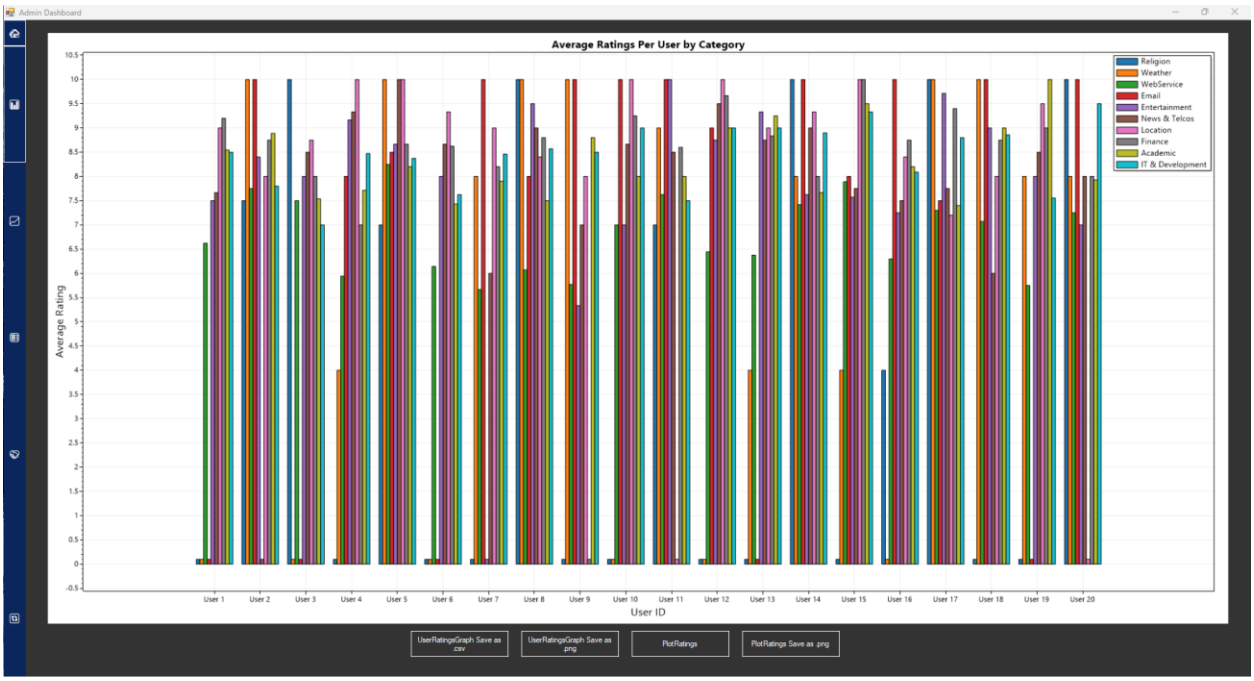


Εικόνα 12. Κεντρικό μενού εφαρμογής [32]

Ο χρήστης μπορεί να δει τις περιγραφές που αναφέρονται περνώντας το ποντίκι του πάνω από κάθε εικονίδιο του αριστερού πάνελ.

Επιλογή 1: Ανάκτηση και οπτικοποίηση των αξιολογήσεων χρηστών

Επιλέγοντας την πρώτη επιλογή, λαμβάνουμε την εικόνα παρακάτω:

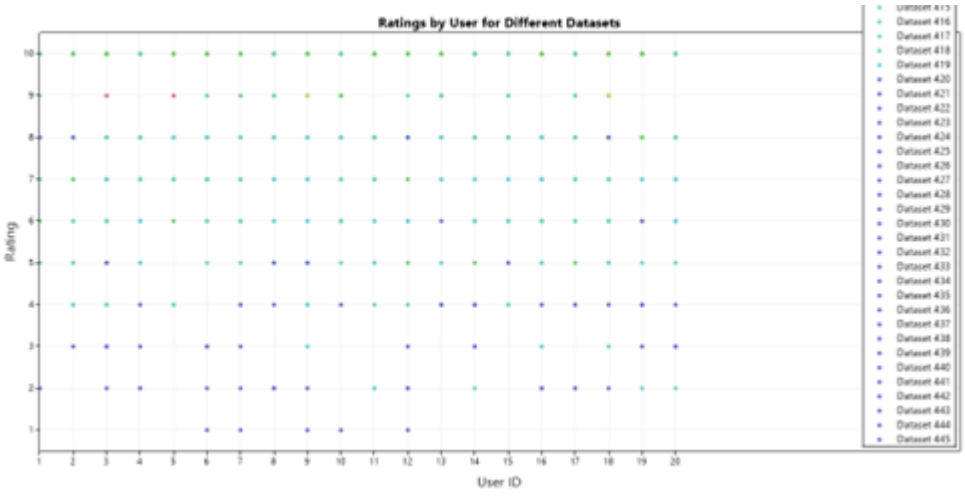


Εικόνα 13. Μενού ανάκτησης και οπτικοποίησης αξιολογήσεων

Η εφαρμογή παρουσιάζει δεδομένα της μέσης τιμής αξιολόγησης των χρηστών ταξινομημένα για κάθε μία από τις 10 κατηγορίες, πράγμα που μας δίνει μια γρήγορη εικόνα των προτιμήσεών τους.

Μέσα από την κλάση DataVisualizerUserratings, χρησιμοποιείται η μέθοδος PlotAverageRatingsPerCategory, η οποία δημιουργεί ένα διάγραμμα με τις μέσες αξιολογήσεις ανά κατηγορία για κάθε χρήστη παίρνοντας σαν είσοδο τα δεδομένα των Views που δημιουργήσαμε παραπάνω από το QWS Dataset. Ο πίνακας παραπάνω δημιουργείται σαν Bar Plot που προσφέρεται από την βιβλιοθήκη ScottPlot.

Στην συνέχεια, το κουμπί PlotRatings μας δίνει αυτή την εικόνα:



Εικόνα 14. Γράφημα αξιολογήσεων χρηστών για τις διαδικτυακές υπηρεσίες του qws_dataset

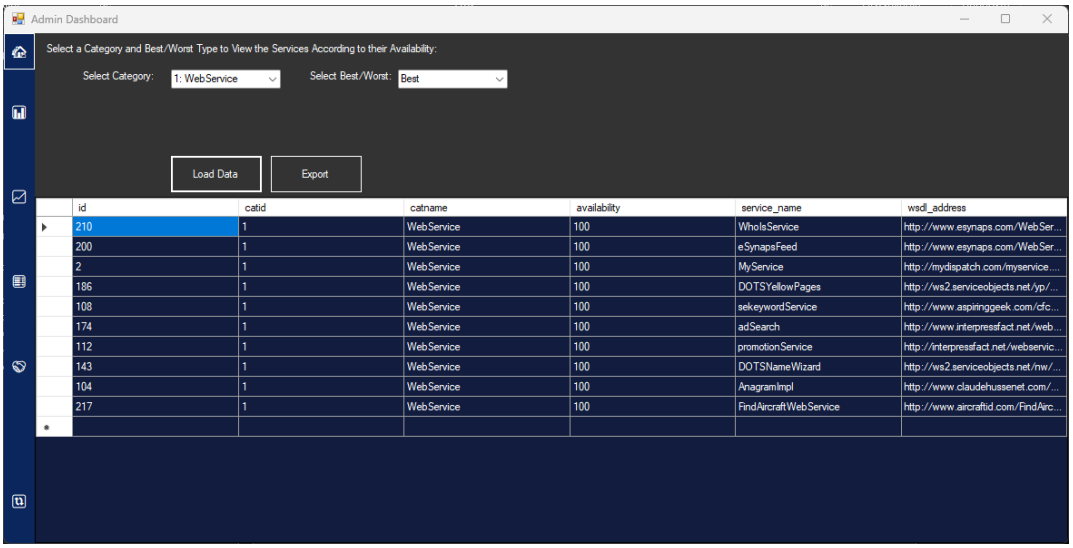
Το παραπάνω γράφημα δημιουργεί μια γρήγορη εικόνα του πόσες υπηρεσίες έχει αξιολογήσει ο κάθε χρήστης και ποια βαθμολογία έδωσε χωρίς να λαμβάνουμε υπόψιν τα ποιοτικά χαρακτηριστικά. Η συγκεκριμένη προσέγγιση μας φαίνεται χρήσιμη σε περιπτώσεις ψυχρής εκκίνησης όπου δεν υπάρχουν επαρκή δεδομένα για να δοθούν στην συνέχεια οι κατάλληλες συστάσεις.

Μέσα στην κλάση DataVisualizer βρίσκουμε την μέθοδο ColorFromHSL για τη δημιουργία χρωμάτων βασισμένων σε έναν δείκτη και για τη μετατροπή των τιμών HSL (Hue, Saturation, Lightness) σε χρώματα RGB, χρησιμοποιώντας μια αλγοριθμική διαδικασία η οποία αντιστοιχεί ένα διαφορετικό χρωματικό τόνο ανάλογα με το Id της Υπηρεσίας Ιστού που έχει βαθμολογήσει ο χρήστης. Η μέθοδος PlotRatings χρησιμοποιεί την βιβλιοθήκη ScottPlot για να δημιουργήσει ένα γράφημα που δείχνει τις αξιολογήσεις ανά χρήστη για διαφορετικά σετ δεδομένων. Αυτό γίνεται ανακτώντας τα μοναδικά ID σετ δεδομένων και χρήστη από τον πίνακα δεδομένων και σχεδιάζοντας τις αξιολογήσεις με διαφορετικά χρώματα για κάθε ID σετ δεδομένων. Ενδεικτικά βλέπουμε την διακύμανση στο παράθυρο δεξιά του γραφήματος.

Η μέθοδος PlotRatings προσαρμόζει τον τίτλο, τις ετικέτες των αξόνων και τις τιμές των δεικτών αποθηκεύοντας το τελικό γράφημα ως εικόνα. Επιπλέον, διαχειρίζεται τυχόν εξαιρέσεις που μπορεί να προκύψουν λόγω πιθανών λαθών κατά την αποθήκευση του γραφήματος.

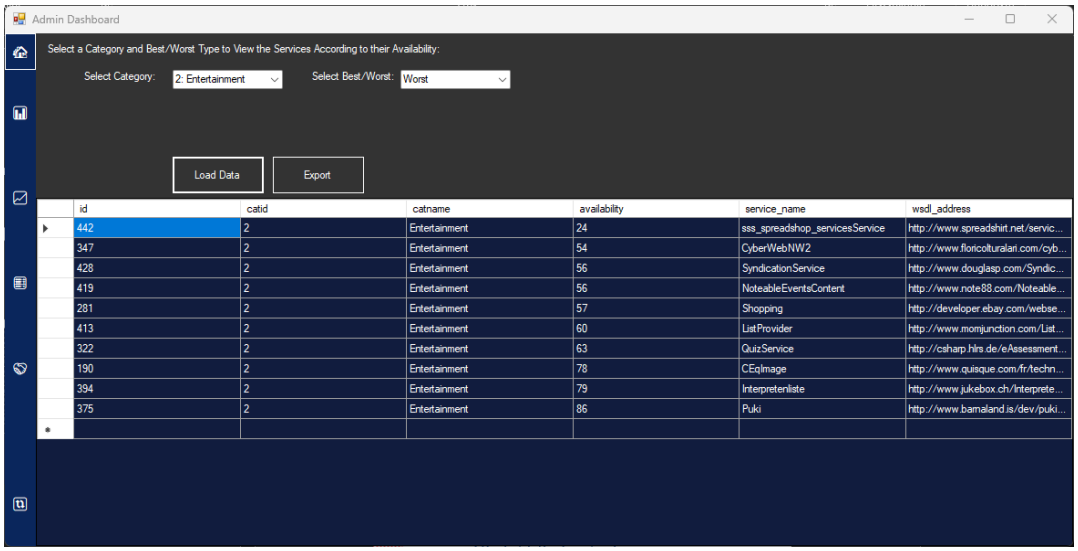
Τα κουμπιά στο κάτω μέρος προσφέρουν επιλογές αποθήκευσης των γραφημάτων και των αποτελεσμάτων.

Επιλογή 2: Ανάκτηση και εμφάνιση των καλύτερων ή χειρότερων υπηρεσιών σύμφωνα με τη διαθεσιμότητα τους, ταξινομημένες ανά κατηγορία.



id	catid	catname	availability	service_name	wsdl_address
210	1	WebService	100	WhoIsService	http://www.esynaps.com/WebSer...
200	1	WebService	100	eSynapsFeed	http://www.esynaps.com/WebSer...
2	1	WebService	100	MyService	http://mydispatch.com/myservice....
186	1	WebService	100	DOTSYellowPages	http://ws2.serviceobjects.net/yp/...
108	1	WebService	100	sekeywordService	http://www.aspiringgeek.com/cfc...
174	1	WebService	100	adSearch	http://www.interpressfact.net/web...
112	1	WebService	100	promotionService	http://interpressfact.net/webservic...
143	1	WebService	100	DOTSNameWizard	http://ws2.serviceobjects.net/nw/...
104	1	WebService	100	AnagramImpl	http://www.claudehussenet.com/...
217	1	WebService	100	FindAircraftWebService	http://www.aircraftid.com/FindAirc...

Εικόνα 15. Μενού εμφάνισης περισσότερο/λιγότερο προτιμητέων υπηρεσιών διαδικτύου σύμφωνα με την διαθεσιμότητά τους



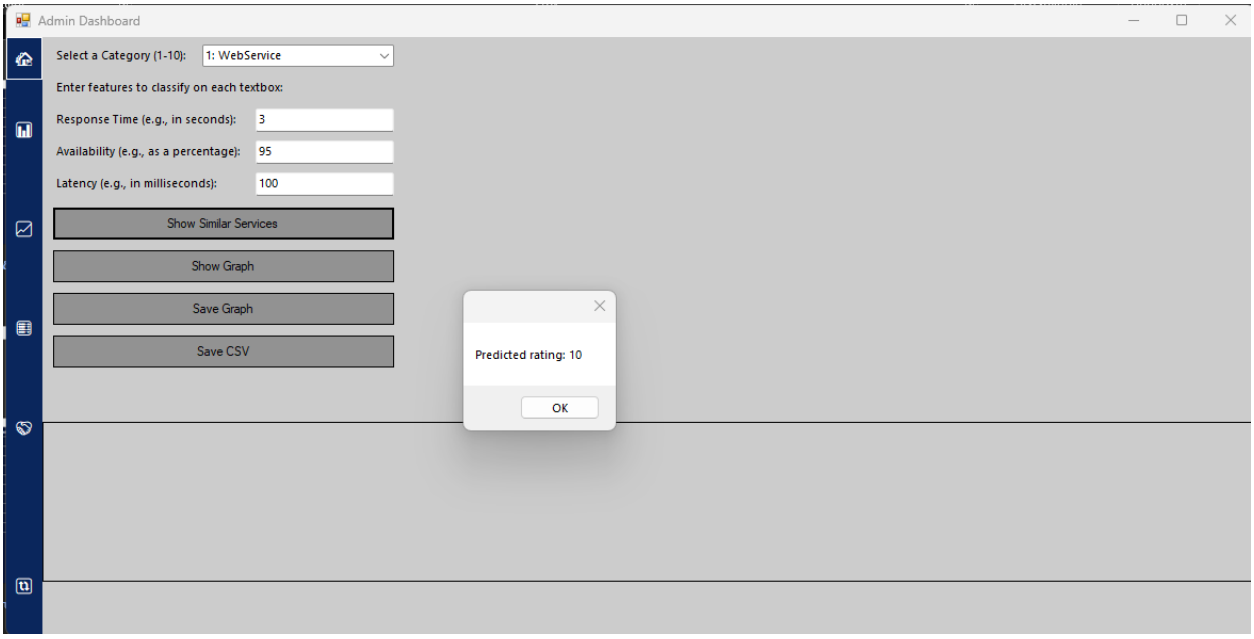
id	catid	catname	availability	service_name	wsdl_address
442	2	Entertainment	24	sss_spreadshop_servicesService	http://www.spreadshirt.net/servic...
347	2	Entertainment	54	CyberWebNW2	http://www.floricolturalari.com/cyb...
428	2	Entertainment	56	SyndicationService	http://www.douglasp.com/Syndic...
419	2	Entertainment	56	NoteableEventsContent	http://www.note88.com/Noteable...
281	2	Entertainment	57	Shopping	http://developer.ebay.com/webse...
413	2	Entertainment	60	ListProvider	http://www.momjunction.com/List...
322	2	Entertainment	63	QuizService	http://csharp.hlrs.de/eAssessment...
190	2	Entertainment	78	CEqImage	http://www.quisque.com/fr/techn...
394	2	Entertainment	79	Interpretenliste	http://www.jukebox.ch/Interprete...
375	2	Entertainment	86	Puki	http://www.bamaland.is/dev/puki...

Εικόνα 16. Αποτελέσματα περισσότερο/λιγότερο προτιμητέων υπηρεσιών διαδικτύου σύμφωνα με την διαθεσιμότητά τους

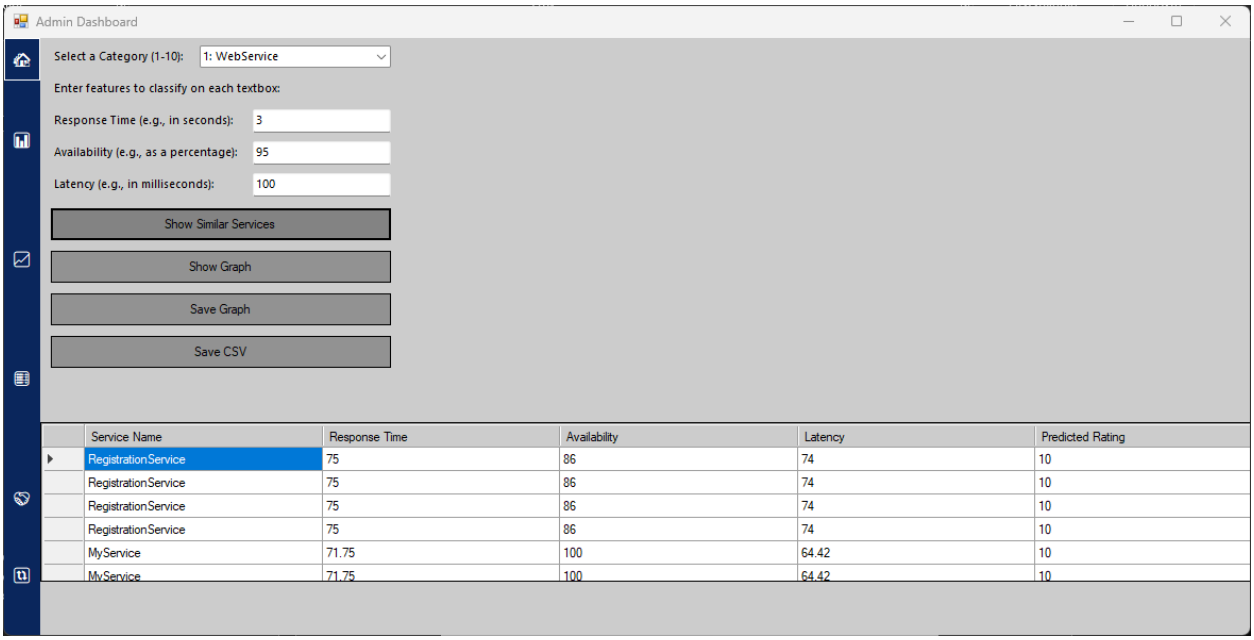
Για να υπάρξει καλύτερη αποτύπωση των δεδομένων, δημιουργήθηκαν κάποιες επιλογές ταξινόμησης. Επιλέγοντας μια κατηγορία διαδικτυακών υπηρεσιών, παρουσιάζεται η δυνατότητα να προβληθούν οι καλύτερες ή χειρότερες υπηρεσίες βάσει διαθεσιμότητας (Availability). Μετά την επιλογή, η μέθοδος GetServicesData καλείται με τις κατάλληλες πραμέτρους για να ανακτήσει τα δεδομένα από ένα σύνολο 20 προκαθορισμένων προβολών (Views) στη βάση δεδομένων. Τα δεδομένα επιστρέφονται ως ένας πίνακας δεδομένων (DataTable) και στη συνέχεια εμφανίζονται στον χρήστη. Η μέθοδος GetServicesData διαμορφώνει το όνομα της προβολής στη βάση δεδομένων ανάλογα με το αν ο χρήστης ζητήσει τις καλύτερες ή τις χειρότερες υπηρεσίες και εκτελεί ένα ερώτημα (query) για να λάβει τα απαραίτητα δεδομένα.

Υπάρχει επίσης η επιλογή για τον χρήστη να εξάγει τα αποτελέσματα σε ένα αρχείο CSV μέσω της επιλογής Export.

Επιλογή 3: Εκτέλεση Ταξινόμησης Πλησιέστερων γειτόνων (KNN)



Εικόνα 17. Μενού επιλογής ταξινόμησης πλησιέστερων γειτόνων – Πρόβλεψη βαθμολογίας

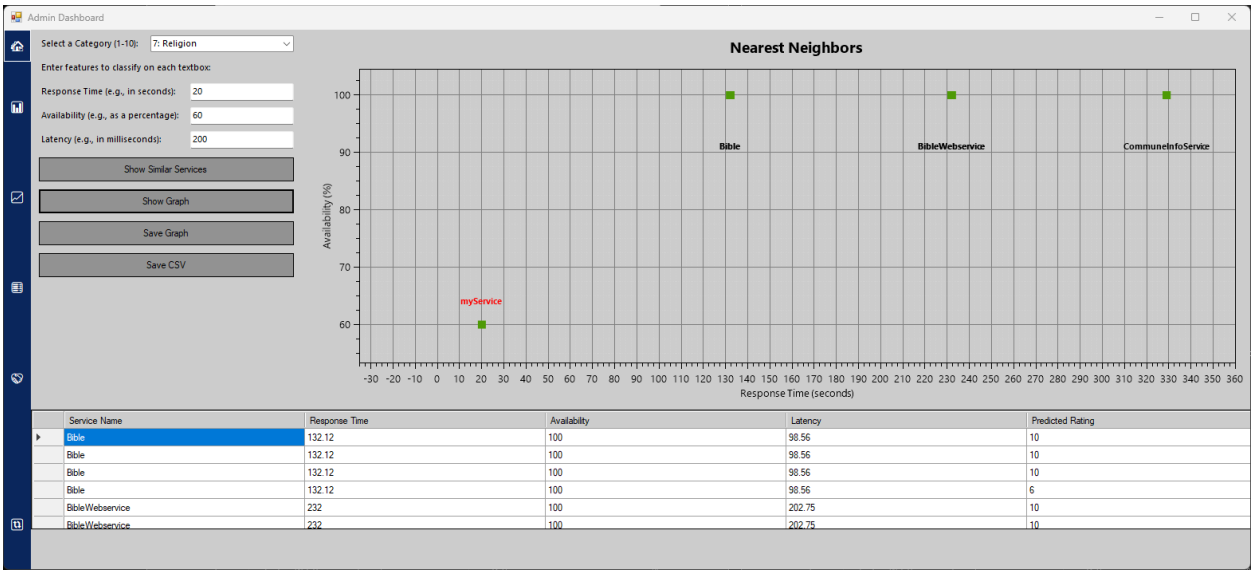


Εικόνα 18. Μενού επιλογής ταξινόμησης πλησιέστερων γειτόνων – Εμφάνιση Πλησιέσετερων γειτόνων σε πίνακα (σε σχέση με νέο στοιχείο)

Σε αυτό το μενού βλέπουμε την εφαρμογή του αλγορίθμου K-Nearest Neighbors (KNN) στις Διαδικτυακές Υπηρεσίες που βρίσκονται αποθηκευμένες στην Βάση Δεδομένων για την ταξινόμηση και την πρόβλεψη κατηγοριών διαδικτυακών υπηρεσιών βάσει χαρακτηριστικών όπως ο χρόνος απόκρισης, η διαθεσιμότητα και η καθυστέρηση.

Επιλέγοντας μια κατηγορία διαδικτυακής υπηρεσίας από τις διαθέσιμες, ο αλγόριθμος KNN χρησιμοποιεί τη λίστα knnData, η οποία περιέχει προηγουμένως ταξινομημένα δεδομένα, για να προβλέψει την πιθανότερη βαθμολογία που θα λάβει μια νέα υπηρεσία βάσει των δοθέντων χαρακτηριστικών και προηγούμενων βαθμολογήσεων που έχουν δοθεί για όμοιες Υπηρεσίες Διαδικτύου. Ο αλγόριθμος KNN λειτουργεί υπολογίζοντας τις ευκλείδειες αποστάσεις μεταξύ του νέου δείγματος και όλων των δεδομένων που έχει στη διάθεσή του επιλέγει τα πιο κοντινά δείγματα. Η ταξινόμηση του νέου δείγματος βασίζεται στην πιο συχνή ετικέτα (label) μεταξύ αυτών των κοντινών δειγμάτων.

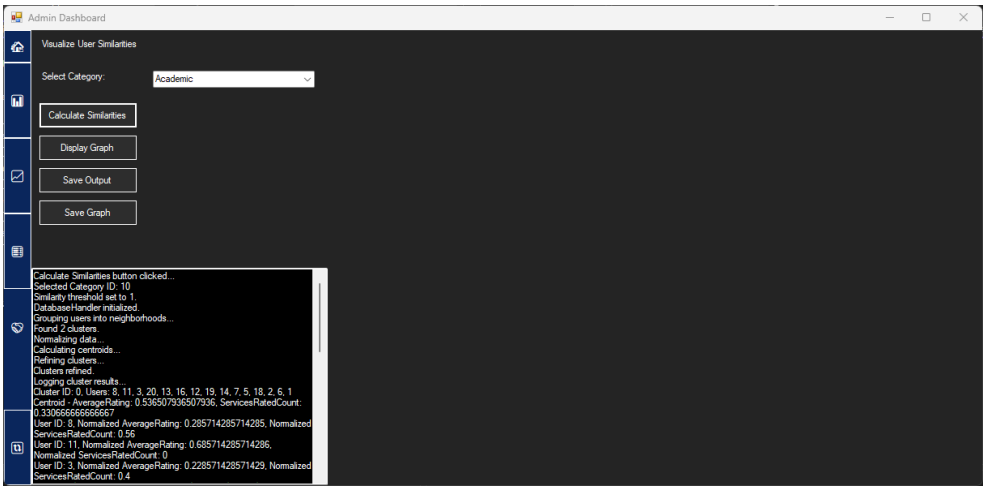
Εάν ο χρήστης επιθυμεί, υπάρχει η δυνατότητα δημιουργίας γραφήματος με τους 10 πλησιέστερες υπηρεσίες σ'εκείνη που έχει χρησιμοποιήσει σαν είσοδο. Το γράφημα χρησιμοποιεί την βιβλιοθήκη OxyPlot με την οποία δημιουργεί ένα διάγραμμα διασποράς στο οποίο εμφανίζονται οι κοντινότερες υπηρεσίες που βρίσκονται στην Βάση Δεδομένων. Επιπλέον, παρουσιάζεται η επιλογή εξαγωγής των δεδομένων σε αρχείο CSV και η αποθήκευση του γραφήματος.



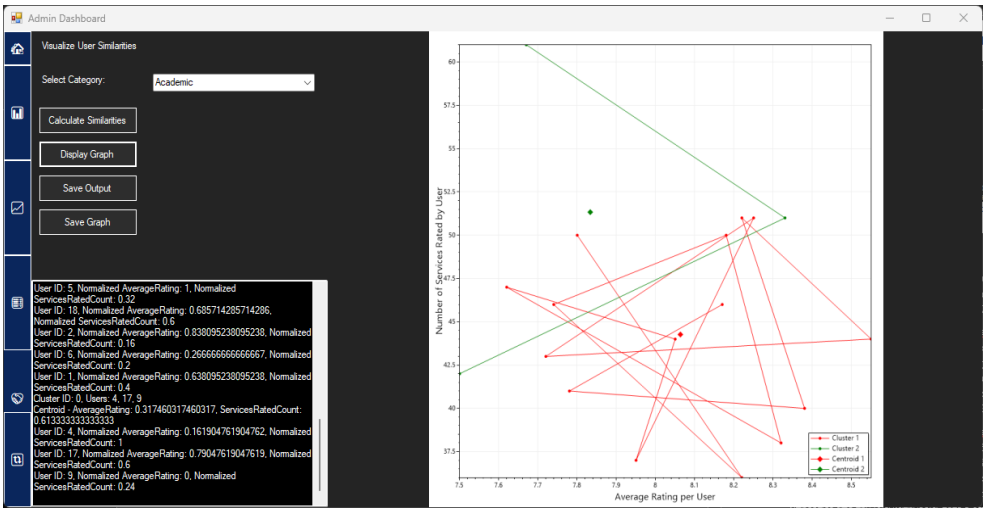
Εικόνα 19. Γράφημα τοποθέτησης νέου στοιχείου και εύρεσης των κοντινότερων γειτόνων του

Στο παράδειγμα βλέπουμε ότι πώς θα κατατασσόταν μια τυχαία υπηρεσία myService ανάμεσα σε άλλες της κατηγορίας 7, η οποία αφορά Web Services που ανήκουν στην κατηγορία "Religion". Οι κοντινότερες υπηρεσίες που βρέθηκαν στην Βάση Δεδομένων προορίζονται για μια Υπηρεσία Ιστού που θα είχε Χρόνο Απόκρισης (Response Time) 20 δευτερόλεπτα, Διαθεσιμότητα (Availability) 60% και Καθυστέρηση (Latency) 200 milliseconds.

Επιλογή 4: Οπτικοποίηση των Ομοιοτήτων Χρηστών



Εικόνα 20. Υπολογισμός Ομάδων Χρηστών – Μενού δημιουργίας Συστάδων Όμοιων Χρηστών με βάση την βαθμολόγηση Κοινών Υπηρεσιών για την Κατηγορία 10



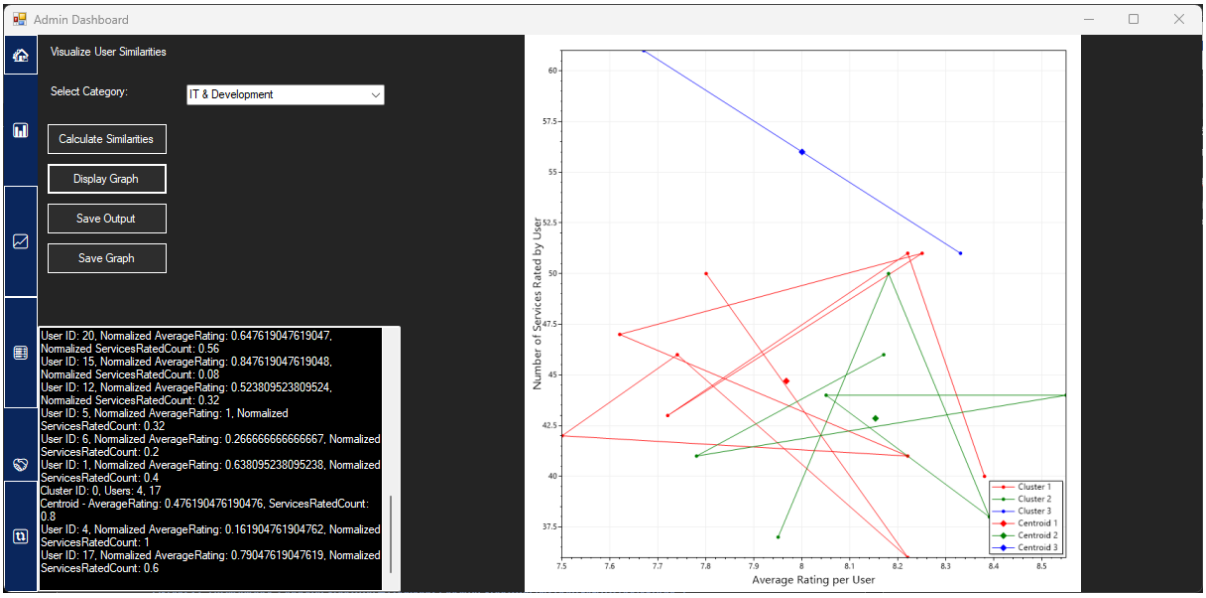
Εικόνα 21. Υπολογισμός Ομάδων Χρηστών – Γράφημα Όμοιων Χρηστών για συντελεστή συσχέτισης 1 για την Κατηγορία 10

Σε αυτό το σημείο η εφαρμογή επεξεργάζεται δεδομένα για να δημιουργήσει και να οπτικοποιήσει συστάδες χρηστών με βάση την ομοιότητα των αξιολογήσεών τους, απ' όπου μπορούμε ν' αποκτήσουμε καλύτερη αντίληψη της κατανομής των αξιολογήσεων.

Αρχικά, ορίζεται ο συντελεστής συσχέτισης (similarity threshold), το οποίο είναι η τιμή πάνω από την οποία δύο χρήστες θεωρούνται αρκετά όμοιοι ώστε να θεωρηθούν μέρος της ίδιας συστάδας. Στην δική μας περίπτωση έχει επιλεγεί το 1, μιας και οι χρήστες θα πρέπει να έχουν βαθμολογήσει τουλάχιστον μια κοινή υπηρεσία ιστού για να ορίζονται ως όμοιοι. Η διαδικασία ξεκινά με την κατηγορία των υπηρεσιών που έχει επιλεγεί (εδώ ως παράδειγμα έχει χρησιμοποιηθεί η κατηγορία 10 αρχικά) και συνεχίζεται με την ομαδοποίηση των χρηστών σε συστάδες με βάση την ομοιότητα των αξιολογήσεών τους.

Ο κώδικας χρησιμοποιεί την μέθοδο GroupUsersIntoNeighborhoods για να δημιουργήσει συστάδες (clusters) των χρηστών. Εντός αυτής της μεθόδου, γίνεται πρώτα ο υπολογισμός όλων των ομοιοτήτων μεταξύ των χρηστών και στη συνέχεια η επεξεργασία κάθε χρήστη ξεχωριστά. Εάν ένας χρήστης δεν έχει ήδη συμπεριληφθεί σε μια συστάδα, ελέγχεται για πιθανές ομοιότητες με άλλους χρήστες. Εάν η ομοιότητα με έναν άλλο χρήστη υπερβαίνει τον καθορισμένο συντελεστή και οι δύο έχουν αξιολογήσει υπηρεσίες από την ίδια κατηγορία, τότε ο δεύτερος χρήστης προστίθεται στη συστάδα του πρώτου.

Στην συνέχεια, η μέθοδος GetAllUserDataPoints ανακτά τα δεδομένα όλων των χρηστών, υπολογίζοντας τη μέση αξιολόγηση και το συνολικό αριθμό των αξιολογήσεων που έχει κάνει ο κάθε χρήστης. Τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν για να εντοπιστούν πρότυπα και τάσεις στην αξιολόγηση υπηρεσιών και για την ανάλυση σχέσεων μεταξύ χρηστών.



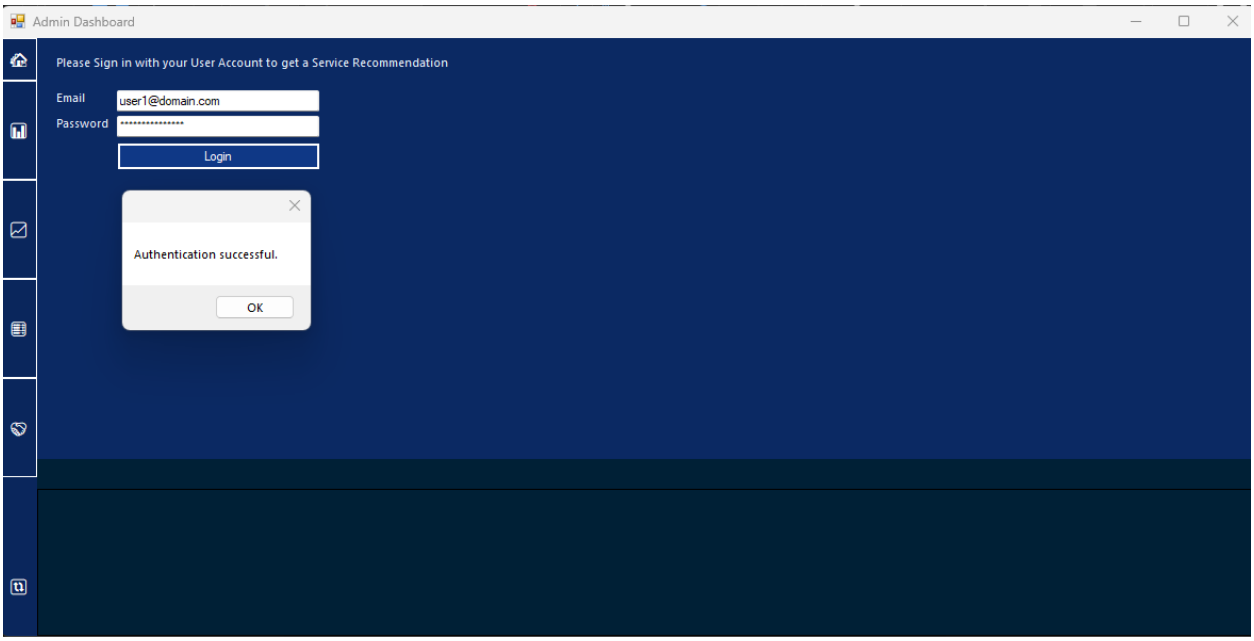
Εικόνα 22. Υπολογισμός Ομάδων Χρηστών – Γράφημα Όμοιων Χρηστών για συντελεστή συσχέτισης 1 για την Κατηγορία 5

Υπενθυμίζεται ότι οι υπολογισμοί μας αφορούν ένα δείγμα 20 χρηστών, οι οποίοι έχουν ομαδοποιηθεί σε 3 συστάδες χρηστών με παρόμοιες προτιμήσεις και συμπεριφορές βαθμολόγησης. Ο άξονας x

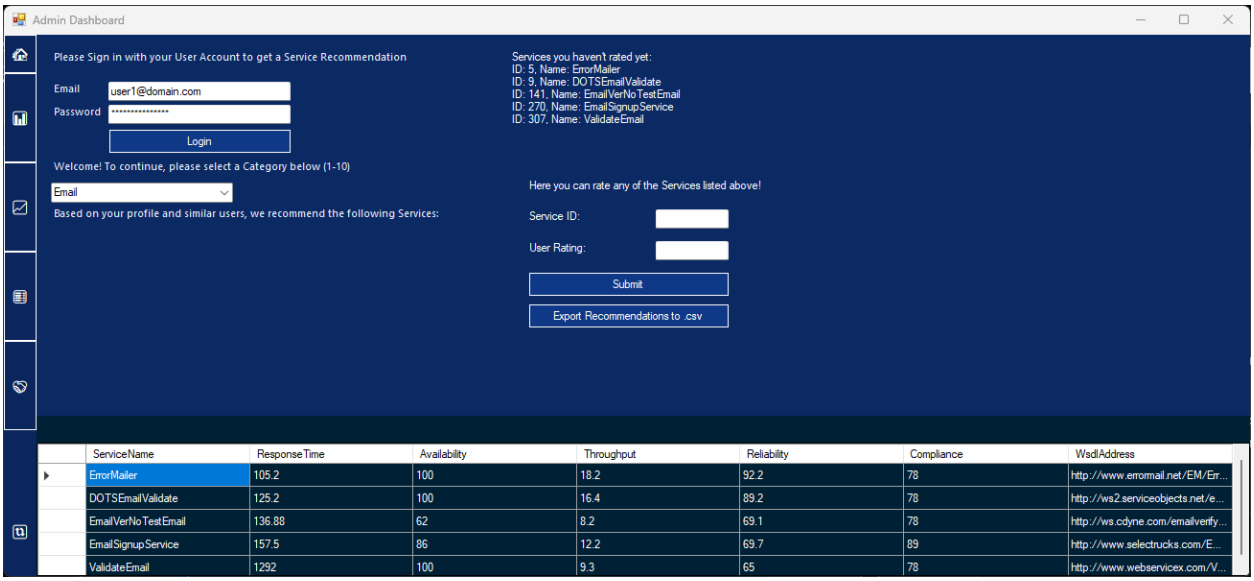
αναπαριστά την μέση βαθμολογία που έχει δώσει ο χρήστης σε υπηρεσίες web, που υπολογίζεται με την μέθοδο CalculateAverageRatingForUser. Αυτή η τιμή είναι μια ποσοτική μέτρηση της γενικής εκτίμησης που έχει ένας χρήστης για τις υπηρεσίες που έχει αξιολογήσει. Ο άξονας y αναπαριστά τον αριθμό των υπηρεσιών που έχει βαθμολογήσει ο χρήστης, που υπολογίζεται με την μέθοδο CountServicesRatedByUser.

Σαν μέτρο σύγκρισης με την κατηγορία 10 βλέπουμε την Κατηγορία 5. Από τις παραπάνω εικόνες καταλαβαίνουμε ότι σε κάποιες περιπτώσεις τ' αποτελέσματα εμφανίζουν μεγαλύτερη συνοχή απ' ότι σε άλλες, δείχνοντάς μας πιθανώς ότι οι βαθμολογήσεις δεν εγγυώνται την προτίμηση των χρηστών, πράγμα που θα γινόταν καλύτερα αντιληπτό σ'ένα μεγαλύτερο δείγμα χρηστών/ υπηρεσιών/ βαθμολογιών.

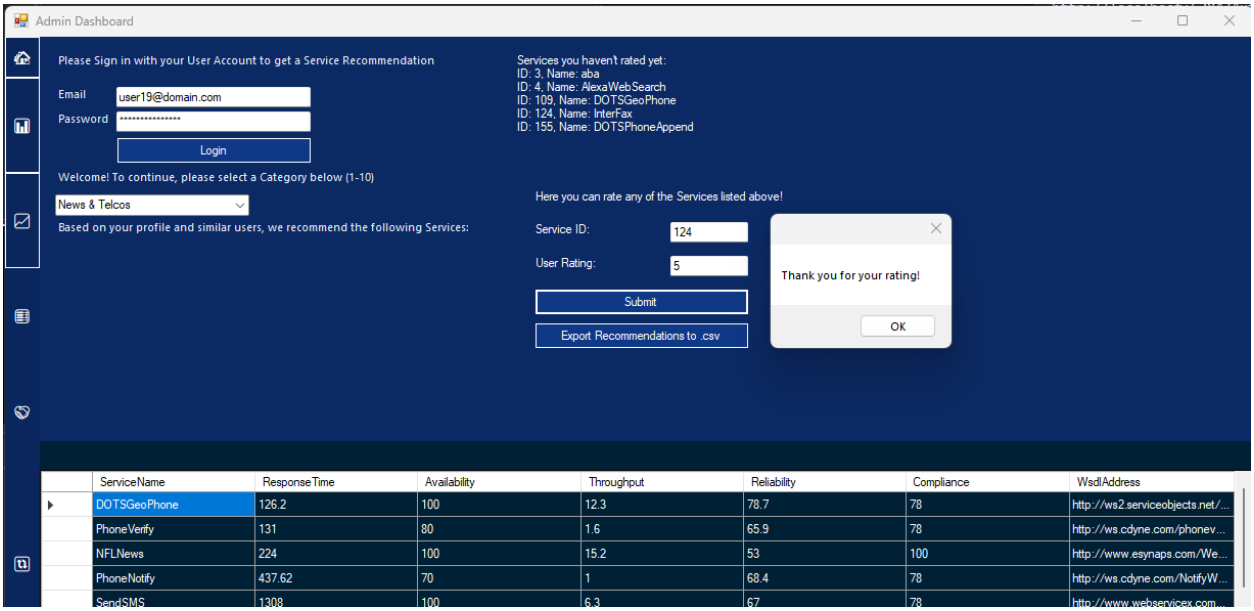
Επιλογή 5: Σύνδεση και λήψη Σύστασης Υπηρεσίας



Εικόνα 23. Μενού Αυθεντικοποίησης και Σύνδεσης Χρήστη



Εικόνα 24. Μενού Επιλογής Κατηγορίας κι εμφάνισης αποτελεσμάτων συστάσεων



Εικόνα 25. Μενού Εισαγωγής Βαθμολογίας

Ο χρήστης αρχικά συνδέεται χρησιμοποιώντας το email και τον κωδικό πρόσβασής του. Γίνεται έλεγχος αυθεντικοποίησης και μόλις ο χρήστης πιστοποιηθεί επιτυχώς του ζητείται να επιλέξει μια κατηγορία υπηρεσιών από μια λίστα. Στη συνέχεια, η εφαρμογή ανακτά μια λίστα πέντε υπηρεσιών που είναι κοντά

στις προτιμήσεις του κι ακόμη μία μ' εκείνες που δεν έχει αξιολογήσει ακόμη για την επιλεγμένη κατηγορία. Λαμβάνοντας μια νέα προτεινόμενη υπηρεσία δίνεται στον χρήστη η δυνατότητα να την αξιολογήσει σε μια κλίματα 1-10. Μόλις ο χρήστης δώσει την αξιολόγησή του, μπορούμε άμεσα να την βρούμε στην Βάση Δεδομένων στον πίνακα ws_ratings.

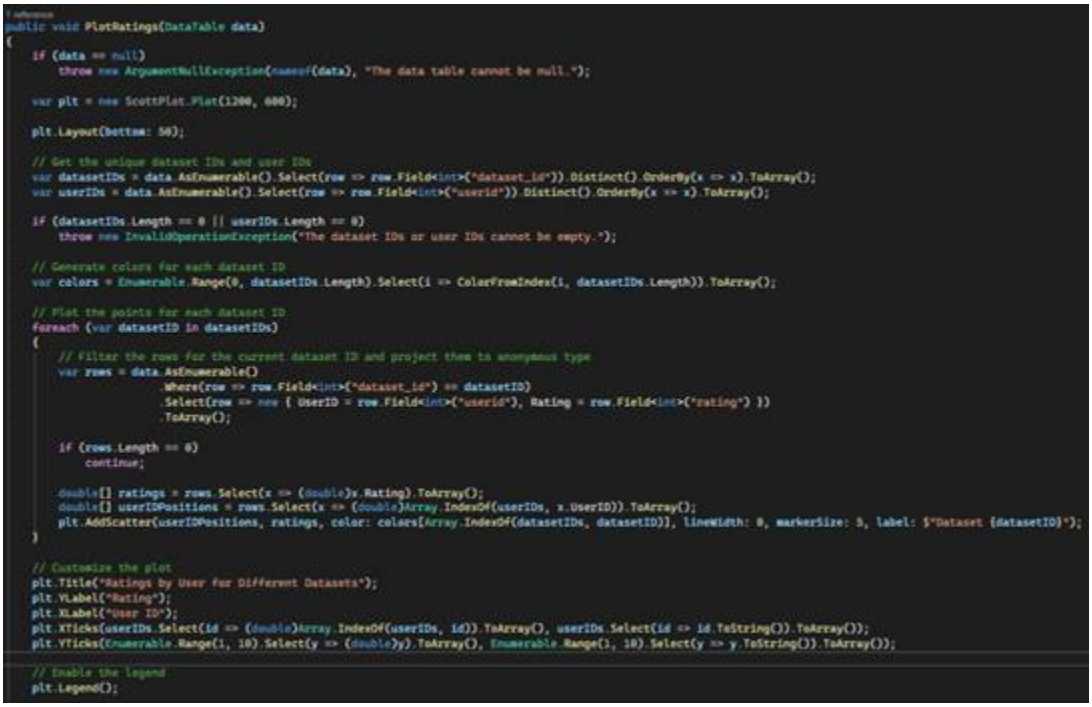
Για να κατανοήσουμε καλύτερα την διαδικασία της πρότασης υπηρεσιών ανατρέχουμε στην κλάση DatabaseHandler.cs και στην μέθοδο GetRecommendedServicesBasedOnSimilarUsers που επιστρέφει μια λίστα από προτεινόμενες υπηρεσίες για έναν χρήστη με βάση τις προτιμήσεις παρόμοιων χρηστών. Χρησιμοποιώντας τον αλγόριθμο KNN (K-Nearest Neighbors) βρίσκει όμοιους χρήστες & τους αποθηκεύει σε μια λίστα similarUserIds. Έπειτα βρίσκει τις υπηρεσίες που έχουν λάβει την καλύτερη βαθμολόγηση. Με ένα SQL Query επιλέγει μοναδικές υπηρεσίες που ανήκουν στην κατηγορία και έχουν βαθμολογία τουλάχιστον 4 από τους παρόμοιους χρήστες, αλλά δεν έχουν αξιολογηθεί από τον τρέχοντα χρήστη. Τα αποτελέσματα ταξινομούνται κατά id υπηρεσίας και φθίνουσα σειρά βαθμολογίας, και περιορίζονται σε 5 κορυφαίες υπηρεσίες.

4.4 Αποσπάσματα Εκτελέσιμου Κώδικα με Σχολιασμό

Επιλογή 1: Ανάκτηση και οπτικοποίηση των αξιολογήσεων χρηστών

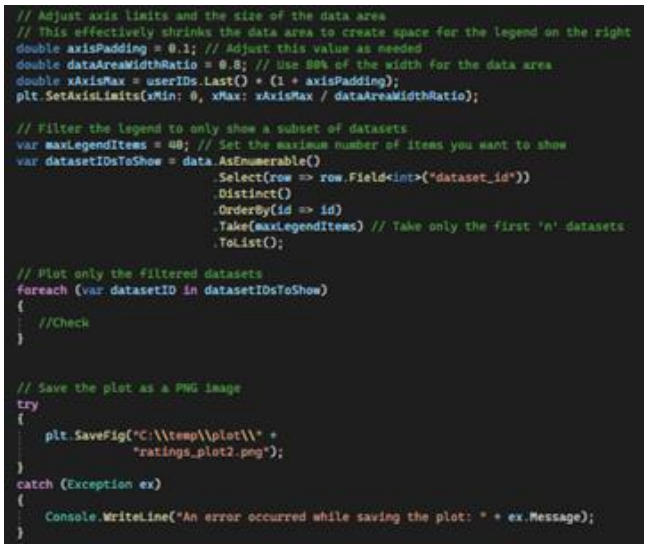
```
2 references
public class DataVisualizer
{
    // Helper method to generate a color based on the index
    1 reference
    private Color ColorFromIndex(int index, int total)
    {
        // Generate a color with varying hue
        int maxHue = 260; // Limit the hue to avoid colors that are too close to red
        int hue = (index * maxHue / total) % maxHue;
        return ColorFromHSL(hue, 0.5, 0.6);
    }

    // Convert HSL to RGB color (helper method)
    1 reference
    private Color ColorFromHSL(int hue, double saturation, double lightness)
    {
        // Convert HSL to RGB following the algorithm from https://en.wikipedia.org/wiki/HSL_and_HSV#HSL_to_RGB_alternative
        double chroma = (1 - Math.Abs(2 * lightness - 1)) * saturation;
        double x = chroma * (1 - Math.Abs((hue / 60 % 2) - 1));
        double m = lightness - chroma / 2;
        double r = 0, g = 0, b = 0;

        if (hue < 60) { r = chroma; g = x; }
        else if (hue < 120) { r = x; g = chroma; }
        else if (hue < 180) { g = chroma; b = x; }
        else if (hue < 240) { g = x; b = chroma; }
        else if (hue < 300) { r = x; b = chroma; }
        else { r = chroma; b = x; }

        r += m; g += m; b += m;
        return Color.FromArgb((int)(r * 255), (int)(g * 255), (int)(b * 255));
    }
```

Εικόνα 26. Βοηθητική μέθοδος για την δημιουργία γραφήματος που μετατρέπει τιμές HSL σε χρώματα RGB



```
public void PlotRatings(DataTable data)
{
    if (data == null)
        throw new ArgumentNullException(nameof(data), "The data table cannot be null.");

    var plt = new ScottPlot.Plot(1200, 600);

    plt.Layout(bottom: 50);

    // Get the unique dataset IDs and user IDs
    var datasetIDs = data.AsEnumerable().Select(row => row.Field<int>("dataset_id")).Distinct().OrderBy(x => x).ToArray();
    var userIDs = data.AsEnumerable().Select(row => row.Field<int>("userid")).Distinct().OrderBy(x => x).ToArray();

    if (datasetIDs.Length == 0 || userIDs.Length == 0)
        throw new InvalidOperationException("The dataset IDs or user IDs cannot be empty.");

    // Generate colors for each dataset ID
    var colors = Enumerable.Range(0, datasetIDs.Length).Select(i => ColorFromIndex(i, datasetIDs.Length)).ToArray();

    // Plot the points for each dataset ID
    foreach (var datasetID in datasetIDs)
    {
        // Filter the rows for the current dataset ID and project them to anonymous type
        var rows = data.AsEnumerable()
                       .Where(row => row.Field<int>("dataset_id") == datasetID)
                       .Select(row => new { UserID = row.Field<int>("userid"), Rating = row.Field<int>("rating") })
                       .ToArray();

        if (rows.Length == 0)
            continue;

        double[] ratings = rows.Select(x => (double)x.Rating).ToArray();
        double[] userIDPositions = rows.Select(x => (double)Array.IndexOf(userIDs, x.UserID)).ToArray();
        plt.AddScatter(userIDPositions, ratings, color: colors[Array.IndexOf(datasetIDs, datasetID)], lineWidth: 0, markerSize: 5, label: $"Dataset {datasetID}");
    }

    // Customize the plot
    plt.Title("Ratings by User for Different Datasets");
    plt.YLabel("Rating");
    plt.XLabel("User ID");
    plt.XTicks(userIDs.Select(id => (double)Array.IndexOf(userIDs, id)).ToArray(), userIDs.Select(id => id.ToString()).ToArray());
    plt.YTicks(Enumerable.Range(1, 10).Select(y => (double)y).ToArray(), Enumerable.Range(1, 10).Select(y => y.ToString()).ToArray());

    // Enable the legend
    plt.Legend();
```

Εικόνα 27. Δημιουργία γραφήματος χρησιμοποιώντας την βιβλιοθήκη ScottPlot για την απεικόνιση αξιολογήσεων χρηστών ανά σύνολο δεδομένων -1



```
// Adjust axis limits and the size of the data area
// This effectively shrinks the data area to create space for the legend on the right
double axisPadding = 0.1; // Adjust this value as needed
double dataAreaWidthRatio = 0.8; // Use 80% of the width for the data area
double xAxisMax = userIDs.Last() * (1 + axisPadding);
plt.SetAxisLimits(xMin: 0, xMax: xAxisMax / dataAreaWidthRatio);

// Filter the legend to only show a subset of datasets
var maxLegendItems = 40; // Set the maximum number of items you want to show
var datasetIDsToShow = data.AsEnumerable()
                           .Select(row => row.Field<int>("dataset_id"))
                           .Distinct()
                           .OrderBy(id => id)
                           .Take(maxLegendItems) // Take only the first 'n' datasets
                           .ToList();

// Plot only the filtered datasets
foreach (var datasetID in datasetIDsToShow)
{
    //Check
}


// Save the plot as a PNG image
try
{
    plt.SaveFig("C:\\temp\\plot\\" +
                "ratings_plot2.png");
}
catch (Exception ex)
{
    Console.WriteLine("An error occurred while saving the plot: " + ex.Message);
}
```

Εικόνα 28. Δημιουργία γραφήματος χρησιμοποιώντας την βιβλιοθήκη ScottPlot για την απεικόνιση αξιολογήσεων χρηστών ανά σύνολο δεδομένων -2

Επιλογή 2: Ανάκτηση και εμφάνιση των καλύτερων ή χειρότερων υπηρεσιών σύμφωνα με τη διαθεσιμότητα τους, ταξινομημένες ανά κατηγορία.

```
public DataTable GetServicesData(bool bestServices, int categoryId)
{
    DataTable dataTable = new DataTable();
    // Adjust the view name based on the user's selection
    string viewName = bestServices
        ? $"best_services_catid_{categoryId}_by_availability"
        : $"worst_services_catid{categoryId}_by_availability";

    //string query = $"SELECT * FROM {viewName}";
    string query = $"SELECT id, catid, catname, availability, service_name, wsdl_address FROM {viewName}";

    using (var connection = new NpgsqlConnection(connectionString))
    {
        using (var cmd = new NpgsqlCommand(query, connection))
        {
            connection.Open();
            using (NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd))
            {
                adapter.Fill(dataTable);
            }
        }
    }

    return dataTable;
}
```

Εικόνα 29. Ανάκτηση δεδομένων από την βάση δεδομένων για τις καλύτερες/χειρότερες διαδικτυακές υπηρεσίες βάσει διαθεσιμότητας

```
1 reference
static void ExportToCsv(DataTable dataTable, string filePath)
{
    StringBuilder csvData = new StringBuilder();

    // Column headers
    string[] columnNames = dataTable.Columns.Cast<DataColumn>().Select(column => column.ColumnName).ToArray();
    csvData.AppendLine(string.Join(",", columnNames));

    // Rows data
    foreach (DataRow row in dataTable.Rows)
    {
        IEnumerable<string> fields = row.ItemArray.Select(field => $"\"{field.ToString().Replace("\"", "\"\"")}\"");
        csvData.AppendLine(string.Join(",", fields));
    }

    // Write to file
    File.WriteAllText(filePath, csvData.ToString());
}
```

Εικόνα 30. Αποθήκευση σε αρχείο .csv

Επιλογή 3: Εκτέλεση Ταξινόμησης Πλησιέστερων γειτόνων (KNN)

```
namespace QWSAppGUI
{
    3 references
    public class KNN
    {
        private readonly List<KnnClassificationUserControl.DataPoint> _dataPoints;
        private readonly int _k;

        1 reference
        public KNN(int k, List<KnnClassificationUserControl.DataPoint> dataPoints)
        {
            _k = k;
            _dataPoints = dataPoints;
        }

        // Euclidean distance
        2 references
        private double ComputeDistance(double[] a, double[] b)
        {
            return Math.Sqrt(a.Zip(b, (x, y) => Math.Pow(x - y, 2)).Sum());
        }

        // Classification method returns the most common rating as a double
        1 reference
        public double Classify(double[] features)
        {
            var distances = _dataPoints.Select(p => new { Point = p, Distance = ComputeDistance(features, p.Features) });
            var nearestNeighbors = distances.OrderBy(d => d.Distance).Take(_k).ToList();

            // Group by label and count the occurrences of each label
            var mostCommonLabel = nearestNeighbors
                .GroupBy(x => x.Point.Label)
                .OrderByDescending(g => g.Count())
                .ThenBy(g => g.Key) // Tie-breaker if necessary
                .First().Key;

            // Parse the label to a double and return it
            return double.TryParse(mostCommonLabel, out double rating) ? rating : 0; // Returns 0 or a default value if parse fails
        }

        // Method to retrieve the nearest neighbors
        1 reference
        public List<KnnClassificationUserControl.DataPoint> GetNearestNeighbors(double[] features)
        {
            var distances = _dataPoints.Select(p => new { Point = p, Distance = ComputeDistance(features, p.Features) });
            return distances.OrderBy(d => d.Distance).Take(_k).Select(d => d.Point).ToList();
        }
    }
}
```

Εικόνα 31. Εφαρμογή του αλγόριθμου K-Nearest Neighbors

```
1 reference
private List<DataPoint> PerformKnnClassification(double[] newFeatures)
{
    int categoryId = cmbCategory.SelectedIndex + 1;
    List<DataPoint> knnData = dbHandler.GetKnnData(categoryId);
    int k = 10;
    KNN knn = new KNN(k, knnData);

    double predictedRating = knn.Classify(newFeatures);
    MessageBox.Show($"Predicted rating: {predictedRating}");

    var nearestNeighbors = knn.GetNearestNeighbors(newFeatures);

    // Ensure the returned DataPoints are correctly instantiated
    var dataPoints = nearestNeighbors.Select(nn => new DataPoint(nn.Features, nn.Label, nn.ServiceName)).ToList();
    return dataPoints;
}
```

Εικόνα 32. Εφαρμογή του αλγόριθμου K-Nearest Neighbors για την πρόβλεψη βαθμολογίας

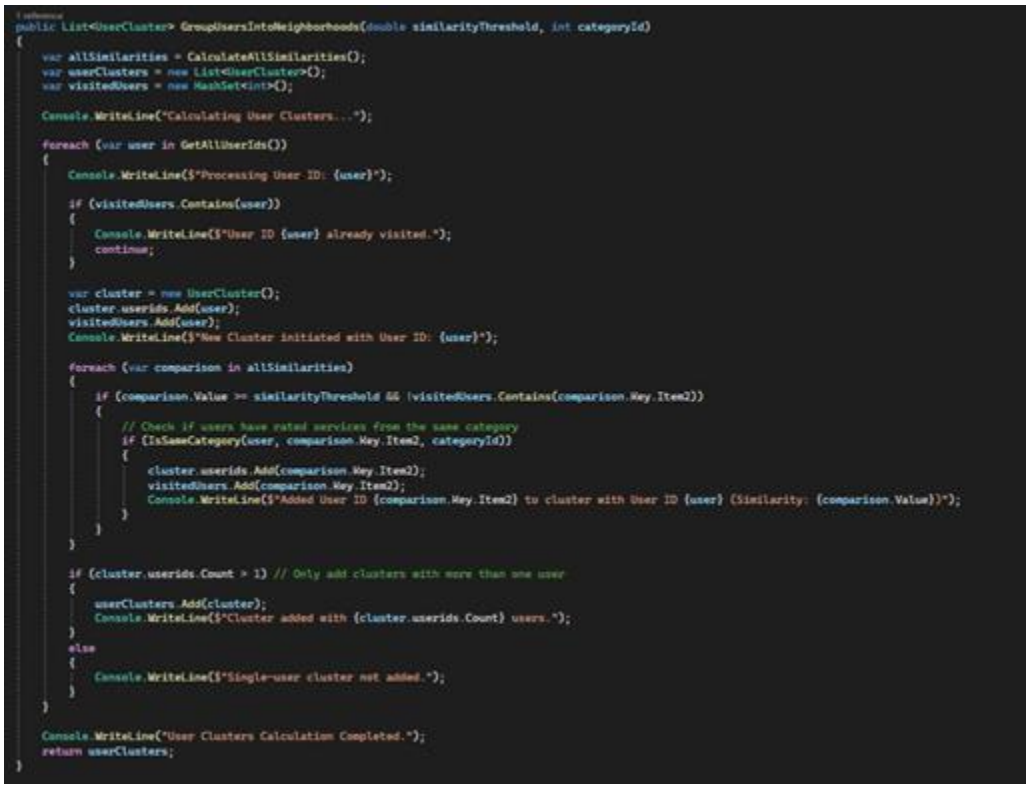
Επιλογή 4: Οπτικοποίηση των Ομοιοτήτων Χρηστών

```
13 references
public class UserCluster
{
    21 references
    public List<int> userids { get; set; } = new List<int>();
    12 references
    public int Id { get; set; }

    13 references
    public UserDataPoint Centroid { get; set; }

    3 references
    public static Dictionary<int, NormalizedUserDataPoint> NormalizeData(IEnumerable<int> allUserIds, DatabaseHandler dbHandler)
    {
        double maxRating = double.MinValue, minRating = double.MaxValue;
        double maxServices = double.MinValue, minServices = double.MaxValue;

        // Collect all user IDs from the clusters and remove duplicates
        HashSet<int> uniqueUserIds = new HashSet<int>(allUserIds);

        // First pass to find the min and max for each dimension
        foreach (var userId in uniqueUserIds)
        {
            double rating = dbHandler.CalculateAverageRatingForUser(userId);
            int services = dbHandler.CountServicesRatedByUser(userId);

            maxRating = Math.Max(maxRating, rating);
            minRating = Math.Min(minRating, rating);
            maxServices = Math.Max(maxServices, services);
            minServices = Math.Min(minServices, services);
        }

        var normalizedData = new Dictionary<int, NormalizedUserDataPoint>();

        // Second pass to normalize each user data point
        foreach (var userId in uniqueUserIds)
        {
            double rating = dbHandler.CalculateAverageRatingForUser(userId);
            int services = dbHandler.CountServicesRatedByUser(userId);

            double normalizedRating = (rating - minRating) / (maxRating - minRating);
            double normalizedServices = (services - minServices) / (maxServices - minServices);

            normalizedData[userId] = new NormalizedUserDataPoint
            {
                AverageRating = normalizedRating,
                ServicesRatedCount = normalizedServices
            };
        }

        return normalizedData;
    }
```

Εικόνα 33. Υπολογισμός ομάδων χρηστών - 1

```
4 references
public class NormalizedUserDataPoint
{
    4 references
    public double AverageRating { get; set; }
    4 references
    public double ServicesRatedCount { get; set; }
}

3 references
public void CalculateCentroid(Dictionary<int, NormalizedUserDataPoint> normalizedData)
{
    double totalAverageRating = 0;
    double totalServicesRatedCount = 0;
    int userCount = 0; // Initialize userCount to 0

    foreach (int userId in userids)
    {
        if (normalizedData.ContainsKey(userId))
        {
            totalAverageRating += normalizedData[userId].AverageRating;
            totalServicesRatedCount += normalizedData[userId].ServicesRatedCount;
            userCount++; // Only count this user if they were included in normalizedData
        }
        else
        {
            Console.WriteLine($"Warning: User ID {userId} was not found in normalized data and will be skipped.");
        }
    }

    if (userCount > 0)
    {
        Centroid = new UserDataPoint
        {
            AverageRating = totalAverageRating / userCount,
            ServicesRatedCount = totalServicesRatedCount / userCount
        };
    }
    else
    {
        // No users found
        Centroid = new UserDataPoint { AverageRating = 0, ServicesRatedCount = 0 };
    }
}
```

Εικόνα 34. Υπολογισμός ομάδων χρηστών - 2



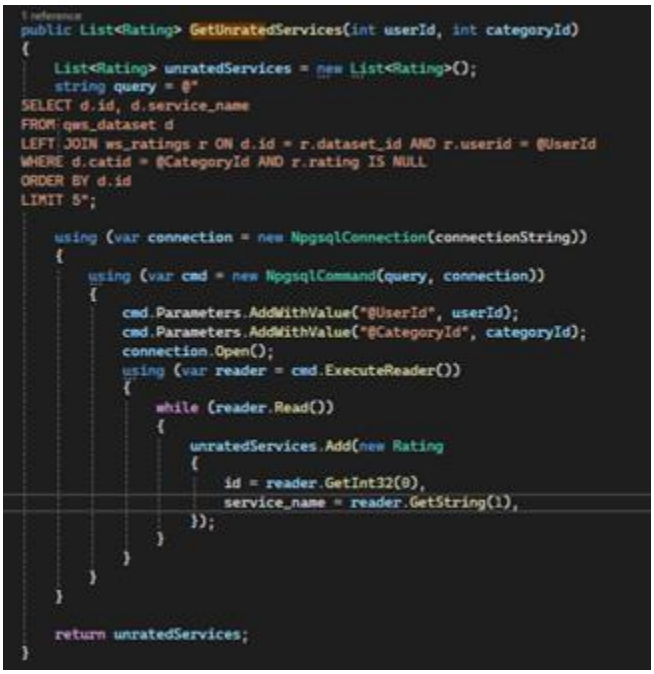
```
public List<UserCluster> GroupUsersIntoNeighborhoods(double similarityThreshold, int categoryId)
{
    var allSimilarities = CalculateAllSimilarities();
    var userClusters = new List<UserCluster>();
    var visitedUsers = new HashSet<int>();

    Console.WriteLine("Calculating User Clusters...");

    foreach (var user in GetAllUserIds())
    {
        Console.WriteLine($"Processing User ID: {user}");

        if (visitedUsers.Contains(user))
        {
            Console.WriteLine($"User ID {user} already visited.");
            continue;
        }

        var cluster = new UserCluster();
        cluster.userids.Add(user);
        visitedUsers.Add(user);
        Console.WriteLine($"New Cluster initiated with User ID: {user}");

        foreach (var comparison in allSimilarities)
        {
            if (comparison.Value >= similarityThreshold && !visitedUsers.Contains(comparison.Key.Item2))
            {
                // Check if users have rated services from the same category
                if (IsSameCategory(user, comparison.Key.Item2, categoryId))
                {
                    cluster.userids.Add(comparison.Key.Item2);
                    visitedUsers.Add(comparison.Key.Item2);
                    Console.WriteLine($"Added User ID {comparison.Key.Item2} to cluster with User ID {user} (Similarity: {comparison.Value})");
                }
            }
        }

        if (cluster.userids.Count > 1) // Only add clusters with more than one user
        {
            userClusters.Add(cluster);
            Console.WriteLine($"Cluster added with {cluster.userids.Count} users.");
        }
        else
        {
            Console.WriteLine($"Single-user cluster not added.");
        }
    }

    Console.WriteLine("User Clusters Calculation Completed.");
    return userClusters;
}
```

Εικόνα 35. Δημιουργία συστάδων χρηστών σύμφωνα με την ομοιότητα των αξιολογήσεών τους σε συγκεκριμένες κατηγορίες υπηρεσιών

Επιλογή 5: Σύνδεση και λήψη Σύστασης Υπηρεσίας

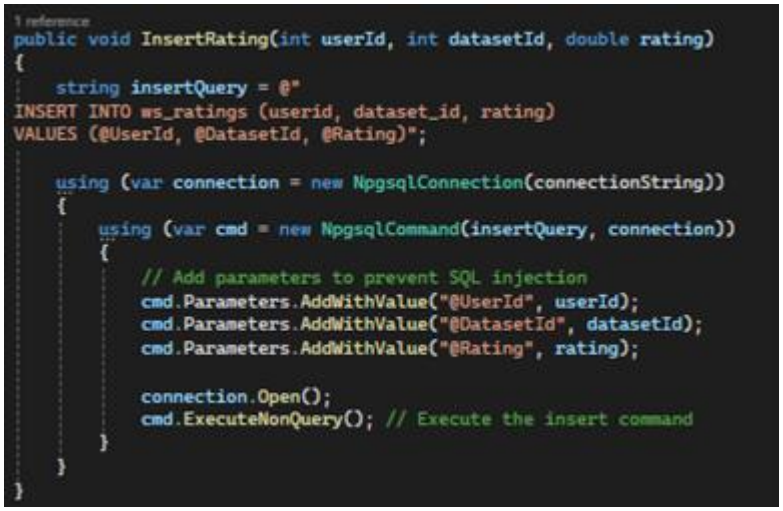
```
1 reference
public AuthenticateUser(string connectionString)
{
    this.connectionString = connectionString;
}

1 reference
public bool Authenticate(string email, string password)
{
    try
    {
        string query = "SELECT password FROM users WHERE email = @Email";

        using (var connection = new NpgsqlConnection(connectionString))
        {
            connection.Open();
            using (var cmd = new NpgsqlCommand(query, connection))
            {
                cmd.Parameters.AddWithValue("@Email", email);

                using (var reader = cmd.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        string storedPassword = reader.GetString(0);
                        return storedPassword == password;
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error during authentication: " + ex.Message);
    }

    return false;
}

1 reference
public int GetUserId(string email)
{
    try
    {
        string query = "SELECT userid FROM users WHERE email = @Email";

        using (var connection = new NpgsqlConnection(connectionString))
        {
            connection.Open();
            using (var cmd = new NpgsqlCommand(query, connection))
            {
                cmd.Parameters.AddWithValue("@Email", email);

                using (var reader = cmd.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return reader.GetInt32(0); // Assuming userid is the first column
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error during fetching user ID: " + ex.Message);
    }

    return -1; // Return -1 or another invalid value to indicate failure
}
```

Εικόνα 36. Αυθεντικοποίηση Χρήστη



```
1 reference
public List<Rating> GetUnratedServices(int userId, int categoryId)
{
    List<Rating> unratedServices = new List<Rating>();
    string query = @"
SELECT d.id, d.service_name
FROM qws_dataset d
LEFT JOIN ws_ratings r ON d.id = r.dataset_id AND r.userid = @UserId
WHERE d.catid = @CategoryId AND r.rating IS NULL
ORDER BY d.id
LIMIT 5";

    using (var connection = new NpgsqlConnection(connectionString))
    {
        using (var cmd = new NpgsqlCommand(query, connection))
        {
            cmd.Parameters.AddWithValue("@UserId", userId);
            cmd.Parameters.AddWithValue("@CategoryId", categoryId);
            connection.Open();
            using (var reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    unratedServices.Add(new Rating
                    {
                        id = reader.GetInt32(0),
                        service_name = reader.GetString(1),
                    });
                }
            }
        }
    }

    return unratedServices;
}
```

Εικόνα 37. Ανάκτηση λίστας υπηρεσιών που δεν έχουν αξιολογηθεί σε μια συγκεκριμένη κατηγορία από την Βάση Δεδομένων



```
1 reference
public void InsertRating(int userId, int datasetId, double rating)
{
    string insertQuery = @"
INSERT INTO ws_ratings (userid, dataset_id, rating)
VALUES (@UserId, @DatasetId, @Rating)";

    using (var connection = new NpgsqlConnection(connectionString))
    {
        using (var cmd = new NpgsqlCommand(insertQuery, connection))
        {
            // Add parameters to prevent SQL injection
            cmd.Parameters.AddWithValue("@UserId", userId);
            cmd.Parameters.AddWithValue("@DatasetId", datasetId);
            cmd.Parameters.AddWithValue("@Rating", rating);

            connection.Open();
            cmd.ExecuteNonQuery(); // Execute the insert command
        }
    }
}
```

Εικόνα 38. Εισαγωγή νέας αξιολόγησης στην Βάση Δεδομένων

5. Συμπεράσματα και Μελλοντικές Επεκτάσεις

5.1 Συμπεράσματα

Τα συστήματα συστάσεων αναπτύχθηκαν για να διαχειρίζονται ένα μεγάλο όγκο πληροφοριών, δίνοντας επιλογές για φιλτράρισμα υπηρεσιών ανάλογα με τις ανάγκες των χρηστών. Έχοντας μια σφαιρική αντίληψη των δεδομένων που παρουσιάζονται, μπορούμε να καταλήξουμε σε ασφαλέστερα συμπεράσματα και να δημιουργήσουμε συστήματα που εξυπηρετούν τις ανάγκες των χρηστών προσφέροντας εύχρηστες λύσεις. Χρησιμοποιώντας τον αλγόριθμο πλησιέστερων γειτόνων (k-NN) είχαμε την δυνατότητα να ταξινομήσουμε και να προβλέψουμε κάποια αποτελέσματα βάσει των χαρακτηριστικών και των προτιμήσεων των χρηστών.

5.2 Μελλοντικές Επεκτάσεις

Η ανάπτυξη της εφαρμογής έχει σχεδιαστεί έχοντας σαν βασικό άξονα την χρήση της από διαχειριστές σε εταιρικό περιβάλλον σε συνδυασμό με άλλες σουίτες εργαλείων και υπηρεσιών.

Σε αντίστοιχες περιπτώσεις βλέπουμε πως υπάρχει αυξημένο ενδιαφέρον μεταφοράς παρόμοιων υπηρεσιών σε υπηρεσίες νέφους λόγω των πλεονεκτημάτων που προσφέρουν, όπως η ευελιξία, διαθεσιμότητα και το προσαρμοσμένο κόστος. Η μεταφορά εφαρμογών σε υπηρεσίες νέφους προσφέρει αυξημένη κλιμακωσιμότητα (Scalability), επιτρέποντας στις επιχειρήσεις να προσαρμόζουν τους πόρους ανάλογα με τις ανάγκες τους. Επιπλέον, ενισχύει την ασφάλεια δεδομένων μέσω προηγμένων μεθόδων κρυπτογράφησης και διαχείρισης πρόσβασης. Η διαχείριση και συντήρηση της υποδομής μειώνεται, καθώς οι πάροχοι υπηρεσιών νέφους αναλαμβάνουν αυτές τις ευθύνες, διευκολύνοντας την εστίαση των επιχειρήσεων στις πυρηνικές τους λειτουργίες.

6. Βιβλιογραφία

[1] Evangelos Sakkopoulos, Dimitris Kanellopoulos, Athanasios K. Tsakalidis:

Semantic mining and web service discovery techniques for media resources management. Int. J. Metadata Semant. Ontologies 1(1): 66-75 (2006)

[2] Shekh A. "Web Services vs API: Top 10+ Difference between API and Web Service." 6 Ιανουαρίου 2023. Φωτογραφία. Διαθέσιμο από: https://www.shekhali.com/web-api-vs-web-service-differences/.

[3] Jugger E. "What's the Difference between API and Web Services?" https://www.abstractapi.com/. 4 Αυγούστου 2023. Διαθέσιμο από: https://www.abstractapi.com/guides/api-vs-web-services.

[4] Lane K. "SOAP API 101: What Is a SOAP API and How Does It Work?" https://www.postman.com/. 28 Ιουνίου 2020. Διαθέσιμο από: https://blog.postman.com/soap-api-definition/.

[5] "SAP and SOA the Business Process Platform | SAP Blogs." Technology Blogs by Members. 4 Φεβρουαρίου 2014. Φωτογραφία. Διαθέσιμο από: https://community.sap.com/t5/technology-blogs-by-members/sap-and-soa-the-business-process-platform/ba-p/13268693.

[6] Konieczny E., Ashcraft R., Cunningham D. Maripuri S. "Establishing Presence within the Service-Oriented Environment." 2009 IEEE Aerospace conference, Big Sky, MT, USA, Απρίλιος 2009. pp. 3. Διαθέσιμο από: https://doi.org/10.1109/AERO.2009.4839647.

[7] Ulvi M, Eken S, Sayar A. "Service Oriented Visual Interpretation Tool for Time Series Data." ANADOLU UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY –A Applied Sciences and Engineering. Ιανουάριος 2013;14:194. Φωτογραφία.

[8] Eleni Papadopoulou, Evangelos Sakkopoulos:

Semantic Cataloging of Public Services Using Basic Government Vocabularies and the Data Catalog Vocabulary for a Unified European Digital Market. IISA 2023: 1-4

[9] John D. Garofalakis, Yannis Panagis, Evangelos Sakkopoulos, Athanasios K. Tsakalidis:

Contemporary Web Service Discovery Mechanisms. J. Web Eng. 5(3): 265-290 (2006)

[10] Tilkov S. "UDDI R.I.P. [Stefan Tilkov's Blog]." https://www.innoq.com/. 25 Μαρτίου 2010. Διαθέσιμο από: https://innoq.com/blog/st/2010/03/uddi-r.i.p/.

[11] Vassiliki Diamadopoulou, Christos Makris, Yannis Panagis, Evangelos Sakkopoulos:

Techniques to support Web Service selection and consumption with QoS characteristics. J. Netw. Comput. Appl. 31(2): 108-130 (2008)

[12] Christos Makris, Yannis Panagis, Evangelos Sakkopoulos, Athanasios K. Tsakalidis:

Efficient and adaptive discovery techniques of Web Services handling large data sets. J. Syst. Softw. 79(4): 480-495 (2006)

[13] Vassilios S. Verykios, Elias C. Stavropoulos, Panteleimon Krasadakis, Evangelos Sakkopoulos:

Frequent itemset hiding revisited: pushing hiding constraints into mining. Appl. Intell. 52(3): 2539-2555 (2022)

[14] Roy D., Dutta M. "A Systematic Review and Research Perspective on Recommender Systems." Journal of Big Data. 3 Μαίου 2022;9(1). Available from: https://doi.org/10.1186/s40537-022-00592-5.

[15] Bhat, T. "Recommendation System — Basics and Use Cases." Medium, 9 Αυγούστου 2021, https://medium.com/@tabindabhat/recommendation-system-basics-and-use-cases-6c5a8b26b98. Φωτογραφία.

[16] Valentine L, D'Alfonso S, Lederman R. "Recommender systems for mental health apps: advantages and ethical challenges." AI & Soc. 2023;38:1627-1638. Διαθέσιμο από: https://doi.org/10.1007/s00146-021-01322-w.

[17] Hamid RA, Albahri AS, Alwan JK, Al-qaysi ZT, Albahri OS, Zaidan AA, Alnoor A, Alamoodi AH, Zaidan BB. "How smart is e-tourism? A systematic review of smart tourism recommendation system applying data management". Computer Science Review,. 2021;39:100337. Διαθέσιμο από: https://doi.org/10.1016/j.cosrev.2020.100337.

[18] Kaur R, Gupta D, Madhukar M, Singh A, Abdelhaq M, Alsaqour R, Breñosa J, Goyal N. "E-Learning Environment Based Intelligent Profiling System for Enhancing User Adaptation." Electronics. 2022;11(20):3354. Διαθέσιμο από: https://doi.org/10.3390/electronics11203354.

[19] Rabahallah K, Mahdaoui L, Azouaou F. "MOOCs Recommender System using Ontology and Memory-based Collaborative Filtering." 2018. p. 635-641. Διαθέσιμο από: https://doi.org/10.5220/0006786006350641.

[20] Aggarwal, Charu C. "Recommender Systems." Cham: Springer International Publishing; 2016. p. 9, 14-15. Διαθέσιμο από: https://doi.org/10.1007/978-3-319-29659-3.

[21] Turing. "A Guide to Content-Based Filtering in Recommender Systems." Turing. Διαθέσιμο από: https://turing.com/kb/content-based-filtering-in-recommender-systems. Φωτογραφία.

[22] Aggarwal, Charu C. "Recommender Systems." Cham: Springer International Publishing; 2016. pp. 8-13. Διαθέσιμο από: https://doi.org/10.1007/978-3-319-29659-3.

[23] Do MPT, Nguyen DV, Nguyen L. "Model-based approach for collaborative filtering." In: 6th International Conference on Information Technology for Education; 2010 Aug; pp. 217-228.

[24] Bartholomew DJ. "The Analysis and Interpretation of Multivariate Data for Social Scientists." Boca Raton, FL: Chapman & Hall/CRC Press; 2002. p. 235. Διαθέσιμο από: https://doi.org/10.1201/9781420057454.

[25] Berg E. "Markov decision process." Διαθέσιμο από: https://optimization.cbe.cornell.edu/index.php?title=Markov_decision_process. Φωτογραφία.

[26] Sarwar B, Karypis G, Konstan J, Riedl J. "Item-based Collaborative Filtering Recommendation Algorithms." Proceedings of ACM World Wide Web Conference. 2001;1. Διαθέσιμο από: https://doi.org/10.1145/371920.372071.

[27] Burke R. "Hybrid Recommender Systems: Survey and Experiments. User Model User-Adap Interact." 2002;12(4):331-370. Διαθέσιμο από: https://doi.org/10.1023/a:1021240730564.

[28] Tokarchuk L. "Digging Friendship: Paper Recommendation in Social Network." Jour. 2014;1(1). Διαθέσιμο από: https://www.eecs.qmul.ac.uk/~laurissa/Laurissas_Pages/Publications_files/DongMaTokarchukNAEC2009.pdf

[29] Nadeem. "Introduction to K-Nearest Neighbor (KNN) Algorithm." Διαθέσιμο από: https://nadeemm.medium.com/introduction-to-k-nearest-neighbor-knn-algorithm-f2c6bef73d58. Φωτογραφία.

[30] McCaffrey J. "Test Run - Understanding K-NN Classification Using C#." MSDN Magazine. 2019 Jan 14;32(12). Διαθέσιμο από: https://learn.microsoft.com/en-us/archive/msdn-magazine/2017/december/test-run-understanding-k-nn-classification-using-csharp.

[31] Al-Masri, E., and Mahmoud Q. H., "Investigating web services on the world wide web", ACM 17th international conference on World Wide Web (WWW '08), pp.795-804. Διαθέσιμο από: https://qwsdata.github.io/#section4

[32] Adobe Stock φωτογραφίες. Διαθέσιμες από: https://stock.adobe.com/search?k=recommendation+engine&asset_id=139655390