Instacart Market Basket Analysis - Αναλυτικη Αναφορα

Φοιτητής: Μουχταρόπουλος Πέτρος (ΑΕΜ 4099)

Εισαγωγή

Η παρούσα εργασία έχει ως στόχο **την ανάλυση των δεδομένων από το dataset *Instacart Market Basket Analysis***, με σκοπό την **εξαγωγή χρήσιμων εμπορικών και λειτουργικών συμπερασμάτων** σχετικά με τη συμπεριφορά των καταναλωτών κατά τη διάρκεια των *online* παραγγελιών τροφίμων.
Προτείνω, μέσω του GitHub μου, να τρέξετε το *InstaCartAnalysis* στο Google Colab, για να έχετε πρόσβαση σε όλα τα *visualizations*.
https://github.com/pmouchtar/InstacartAnalysis.git

Φόρτωση και Συγχώνευση Δεδομένων

Αρχικά, φορτώνουμε τα απαραίτητα CSV αρχεία μέσω του *kagglehub*, και συγκεκριμένα:

orders.csv
aisles.csv
products.csv
departments.csv
order_products__prior.csv
order_products__train.csv

Για λόγους απόδοσης, γίνεται **δειγματοληψία** 500.000 εγγραφών από το *order_products__prior.csv*. Αυτά τα δεδομένα συνενώνονται με τις υπόλοιπες πληροφορίες (προϊόντα, τμήματα, κατηγορίες, παραγγελίες) μέσω διαδοχικών merge.

Εξερευνητική Ανάλυση Δεδομένων (EDA)

Περιγραφικά Στατιστικά

Το merged.describe() μας παρέχει βασικά στατιστικά για κάθε αριθμητική στήλη,συμπεριλαμβανομένων των order_id, add_to_cart_order, reordered,order_hour_of_day, κ.α

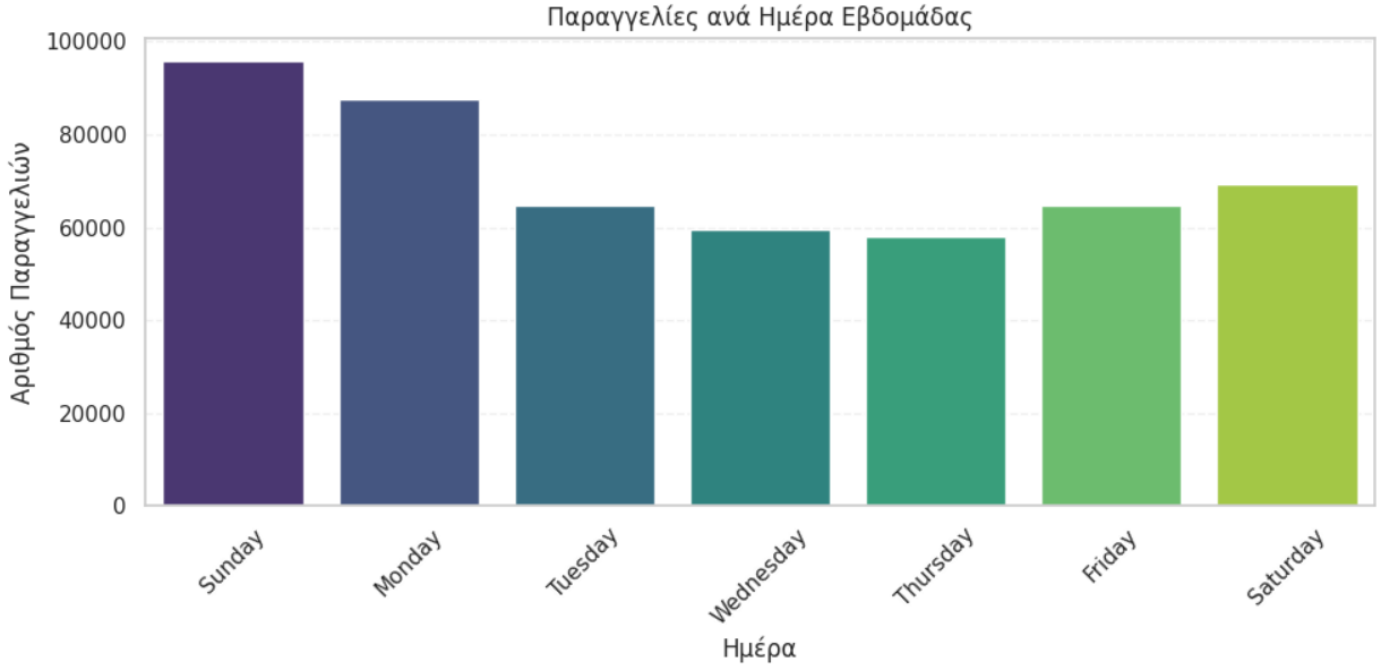
```
merged.describe()
```

	order_id	product_id	add_to_cart_order	reordered	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	aisle_id	department_id
count	5.000000e+05	500000.000000	500000.000000	500000.000000	500000.000000	500000.000000	500000.000000	500000.000000	467945.000000	500000.000000	500000.000000
mean	1.712130e+06	25565.318266	8.363160	0.589302	102932.557446	17.115786	2.735148	13.434576	11.110252	71.184980	9.926668
std	9.870904e+05	14104.361196	7.145807	0.491961	59455.164781	17.472609	2.089889	4.249730	8.776726	38.196697	6.278136
min	1.800000e+01	1.000000	1.000000	0.000000	2.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000
25%	8.581280e+05	13517.000000	3.000000	0.000000	51496.500000	5.000000	1.000000	10.000000	5.000000	31.000000	4.000000
50%	1.713476e+06	25197.000000	6.000000	1.000000	102520.500000	11.000000	3.000000	13.000000	8.000000	83.000000	9.000000
75%	2.565028e+06	37919.000000	11.000000	1.000000	154482.000000	24.000000	5.000000	16.000000	15.000000	107.000000	16.000000
max	3.421081e+06	49688.000000	136.000000	1.000000	206209.000000	99.000000	6.000000	23.000000	30.000000	134.000000	21.000000

Παρατηρούμε ότι ο μέσος όρος επαναπαραγγελιών είναι περίπου 59%.

Οι περισσότερες παραγγελίες γίνονται κατά μέσο όρο μετά από 11 μέρες από την προηγούμενη.

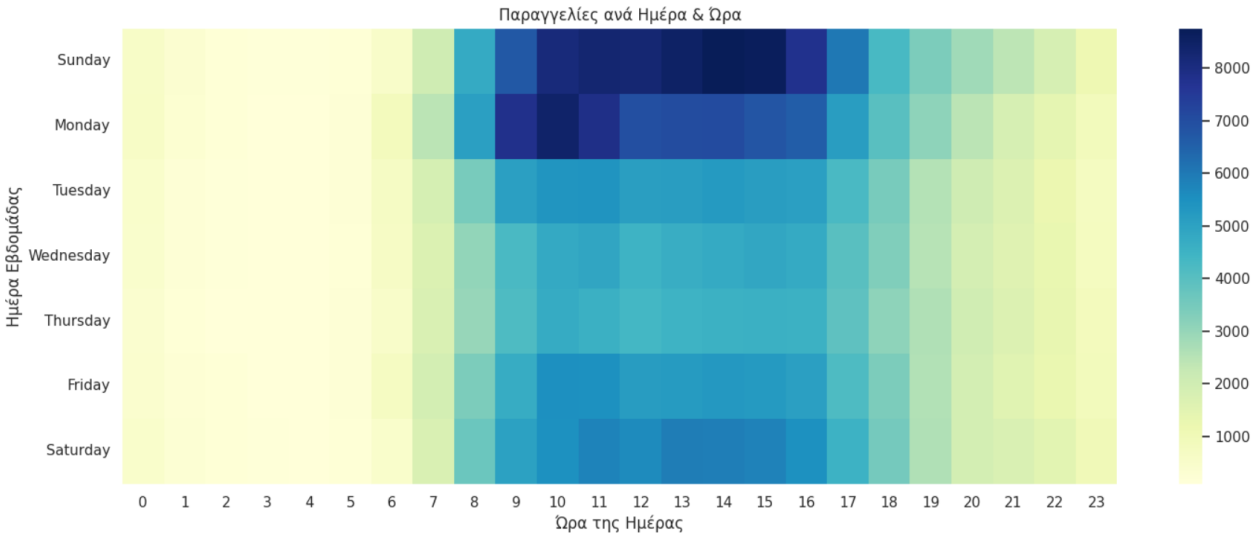
Παραγγελίες ανά Ημέρα Εβδομάδας:



Το γράφημα δείχνει ότι οι περισσότερες παραγγελίες γίνονται την Κυριακή και Δευτέρα.

Οι λιγότερες γίνονται Τετάρτη και Πέμπτη.

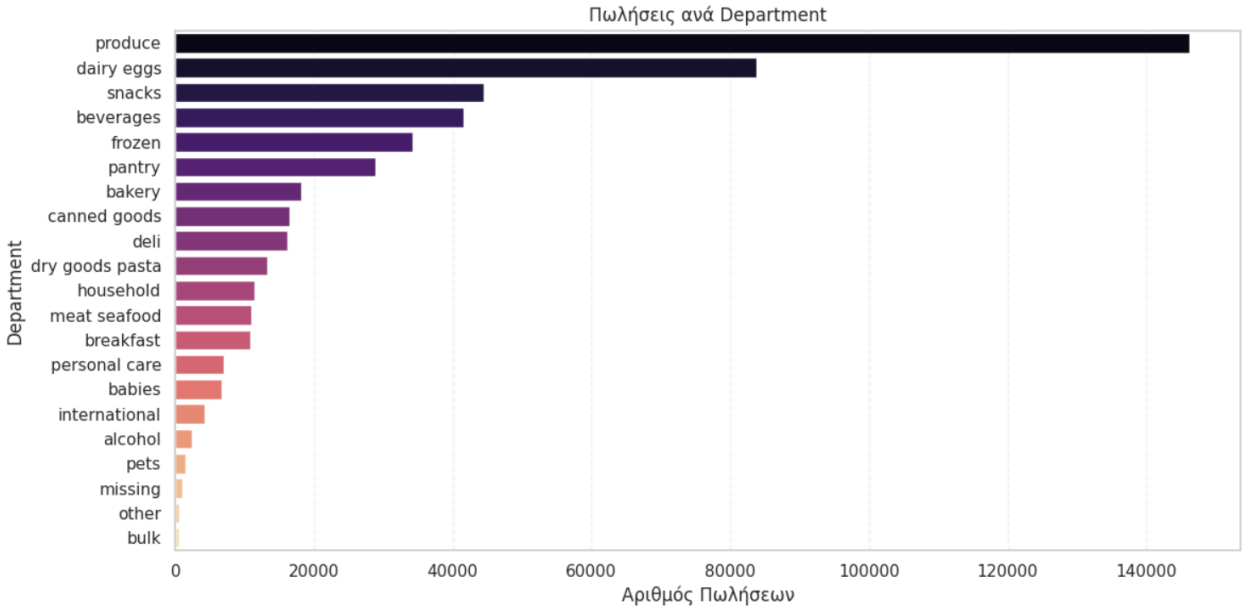
Το heatmap:



Έντονη κίνηση ανάμεσα στις 10:00 - 16:00.

Οι ώρες αιχμής είναι πιο έντονες τα Σαββατοκύριακα.

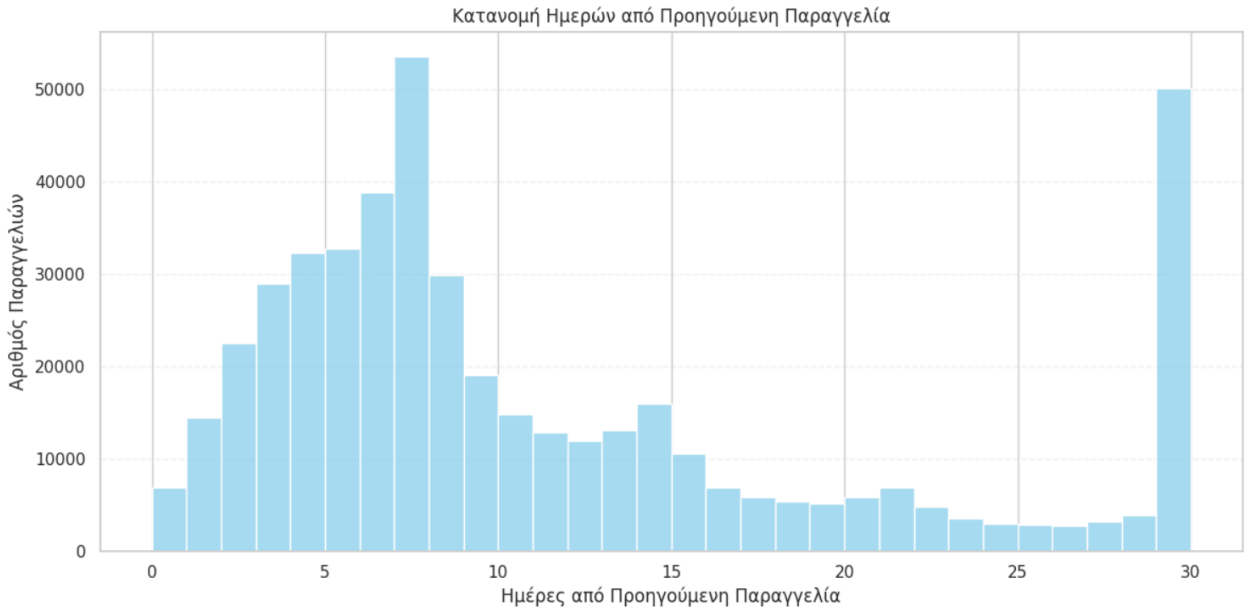
Πωλήσεις ανά Department:



Οι δημοφιλέστερες κατηγορίες είναι:
produce (φρούτα και λαχανικά)

dairy eggs
snacks
Αυτό αντανακλά βασικά και συχνά αγοραζόμενα αγαθά.

Χρόνος από Προηγούμενη Παραγγελία:



Η πλειοψηφία των χρηστών παραγγέλνει μετά από 6–10 μέρες, ενώ υπάρχει κορυφή και στις 30 ημέρες, δείχνοντας περιοδικότητα ή μηνιαίες αγορές.

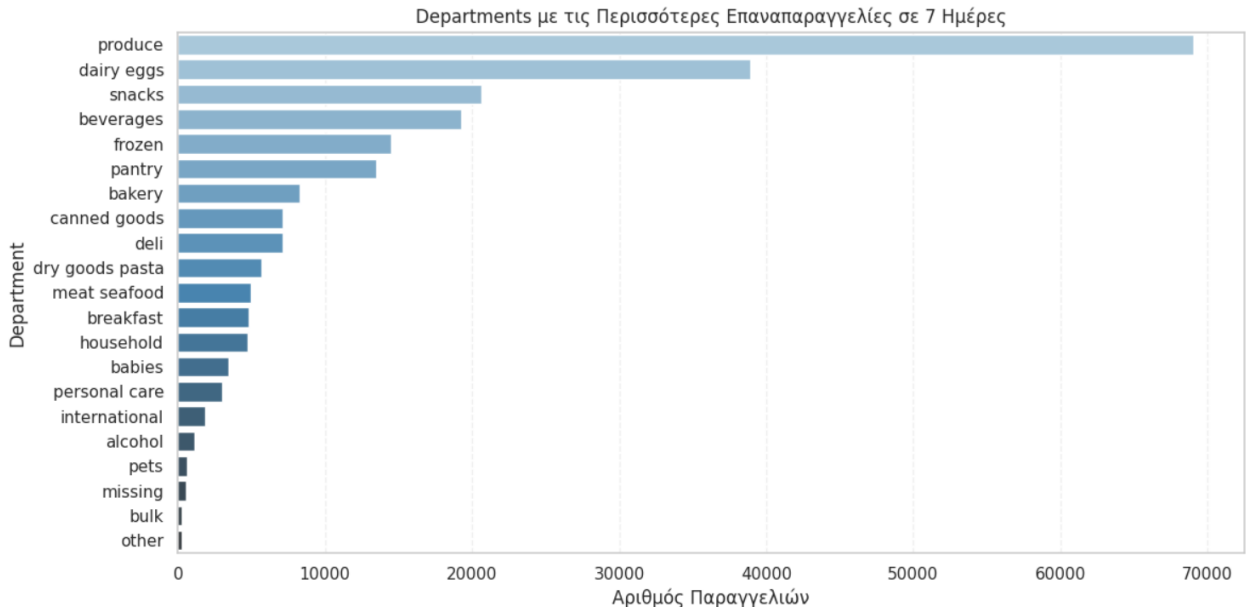
Ανάλυση Επαναπαραγγελιών Ανά Department

Για να εντοπίσουμε ποια τμήματα (departments) έχουν τις περισσότερες επαναπαραγγελίες σε σύντομο χρονικό διάστημα (1-7 ημέρες), φιλτράραμε το dataset merged και ομαδοποιήσαμε τα αποτελέσματα:

short_repeat_orders = merged[merged['days_since_prior_order'] <= 7]

department_repeat_sales = short_repeat_orders.groupby('department')['product_id'].count().sort_values(ascending=False)

Οπτικοποιήσαμε τα αποτελέσματα με barplot, αποκαλύπτοντας ποια departments έχουν τη μεγαλύτερη συχνότητα σύντομων επαναπαραγγελιών:



Predictive Modeling - Αναμενόμενη Επαναπαραγγελία

Στόχος: Πρόβλεψη της Μεταβλητής reordered

Η μεταβλητή στόχος είναι αν ένα προϊόν θα επαναπαραγγελθεί (reordered). Ξεκινήσαμε με βασικά χαρακτηριστικά, κάνοντας drop τα λιγότερο χρήσιμα:

Βασική Εκπαίδευση Μοντέλων

```
X = merged.drop(columns=['order_dow_name', 'reordered', 'order_id', 'product_id', 'user_id', 'product_name', 'eval_set', 'aisle', 'department'])
y = merged['reordered']
```

Κατόπιν κανονικοποιήσαμε και κάναμε imputation με mean, εκπαιδεύσαμε:

- Random Forest
- Logistic Regression

Αποτελέσματα:

Random Forest - Accuracy: 0.7157, Precision: 0.7285, Recall: 0.8224, F1: 0.7726

Logistic Regression - Accuracy: 0.7003, Precision: 0.7202, Recall: 0.8007, F1: 0.7583

Προσθήκη Χαρακτηριστικών Frequency (Product/User)

Μετά προσθέσαμε δύο χαρακτηριστικά:

- product_reorder_freq: πόσες φορές έχει επαναπαραγγελθεί ένα προϊόν

- user_reorder_freq: πόσες φορές ο χρήστης έχει επαναπαραγγείλει συνολικά

Αποτελέσματα:

Random Forest - Accuracy: 0.7947, Precision: 0.7789, Recall: 0.9081, F1: 0.8385

Logistic Regression - Accuracy: 0.7256, Precision: 0.7582, Recall: 0.7820, F1: 0.7699

Επιπλέον Χαρακτηριστικά

Ενισχύσαμε το μοντέλο με νέα features:

- days_since_prior_order_diff: διαφορά στις ημέρες μεταξύ των παραγγελιών
- num_products_in_cart: πλήθος προϊόντων στο καλάθι
- order_dow_hour_interaction: αλληλεπίδραση ημέρας και ώρας παραγγελίας

Αποτελέσματα:

Random Forest - Accuracy: 0.8030, Precision: 0.7833, Recall: 0.9184, F1: 0.8455

Logistic Regression - Accuracy: 0.7266, Precision: 0.7583, Recall: 0.7843, F1: 0.7711

Τελική Έκδοση με Encoding Κατηγοριών

Κάναμε encoding στα πεδία department και aisle, και προσθέσαμε όλα τα παραπάνω χαρακτηριστικά στο τελικό σύνολο.

Αποτελέσματα:

Random Forest - Accuracy: 0.7994, Precision: 0.7804, Recall: 0.9161, F1: 0.8428

Logistic Regression - Accuracy: 0.7342, Precision: 0.7582, Recall: 0.8036, F1: 0.7803

Συμπεράσματα:

Πλεονεκτήματα Random Forest:

Πολύ ισχυρό Recall: μεγαλύτερη ακρίβεια, ιδανικό για σύστημα που θέλει να προβλέψει τι θα ξαναγοραστεί.

Αναβαθμίζεται ξεκάθαρα με παραπάνω features: σημαντική άνοδος σε όλους τους δείκτες, ιδίως με τα frequency-based features και interaction terms.

Ανθεκτικός σε μη-γραμμικές σχέσεις, δεν χρειάζεται scaling, δουλεύει με μεγάλες ποσότητες χαρακτηριστικών.

Μειονεκτήματα:

Πιο αργός σε εκπαίδευση/πρόβλεψη, και δύσκολος στο να εξηγηθεί (black box).

Κίνδυνος overfitting, αν και εδώ δεν φαίνεται να συμβαίνει σοβαρά.

Πλεονεκτήματα Logistic Regression:

Απλό, γρήγορο και ερμηνεύσιμο μοντέλο — καλό για baseline.

Αντέδρασε καλά σε παραπάνω χαρακτηριστικά, αλλά χωρίς μεγάλες βελτιώσεις.

Μειονεκτήματα:

Χαμηλότερο Recall => χάνει πολλά reorders (σημαντικό πρόβλημα για recommender σύστημα).

Λιγότερο αποτελεσματικό σε μη-γραμμικές σχέσεις, ακόμα και με engineered features.

Τελική Επιλογή: Random Forest

Εξαιρετικά καλύτερη F1 score και Recall (πιο κρίσιμο για προβλέψεις reorders).

Ανταποκρίνεται καλύτερα στην προσθήκη νέων χαρακτηριστικών.

Είναι πιο κατάλληλο για προβλήματα αυτού του τύπου, όπου υπάρχει πολυπλοκότητα, συσχετίσεις, και μεγάλες ποσότητες δεδομένων.

Customer Segmentation

Το customer segmentation αποσκοπεί στην κατανόηση της αγοραστικής συμπεριφοράς και στην ανάπτυξη στοχευμένων στρατηγικών marketing. Η βασική μεταβλητή που χρησιμοποιήθηκε ήταν ο μέσος χρόνος επαναπαραγγελίας (days_since_prior_order).

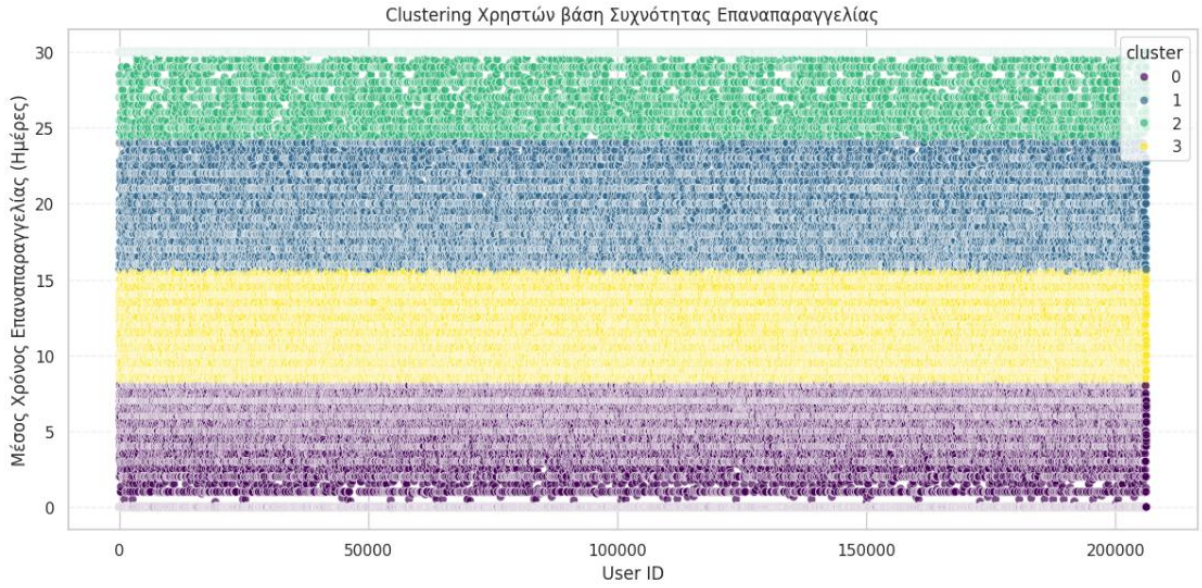
Μέθοδοι Κατηγοριοποίησης

KMeans Clustering (n_clusters = 4)

Χρησιμοποιήθηκε ο μέσος χρόνος επαναπαραγγελίας ανά χρήστη για clustering.

Αξιολόγηση αποτελεσμάτων:

- **Silhouette Score**: 0.5734 (θεωρείται καλό, υποδηλώνει καθαρό διαχωρισμό)
- **Davies-Bouldin Index**: 0.5094 (κάτω από 1 = χαμηλή επικάλυψη)

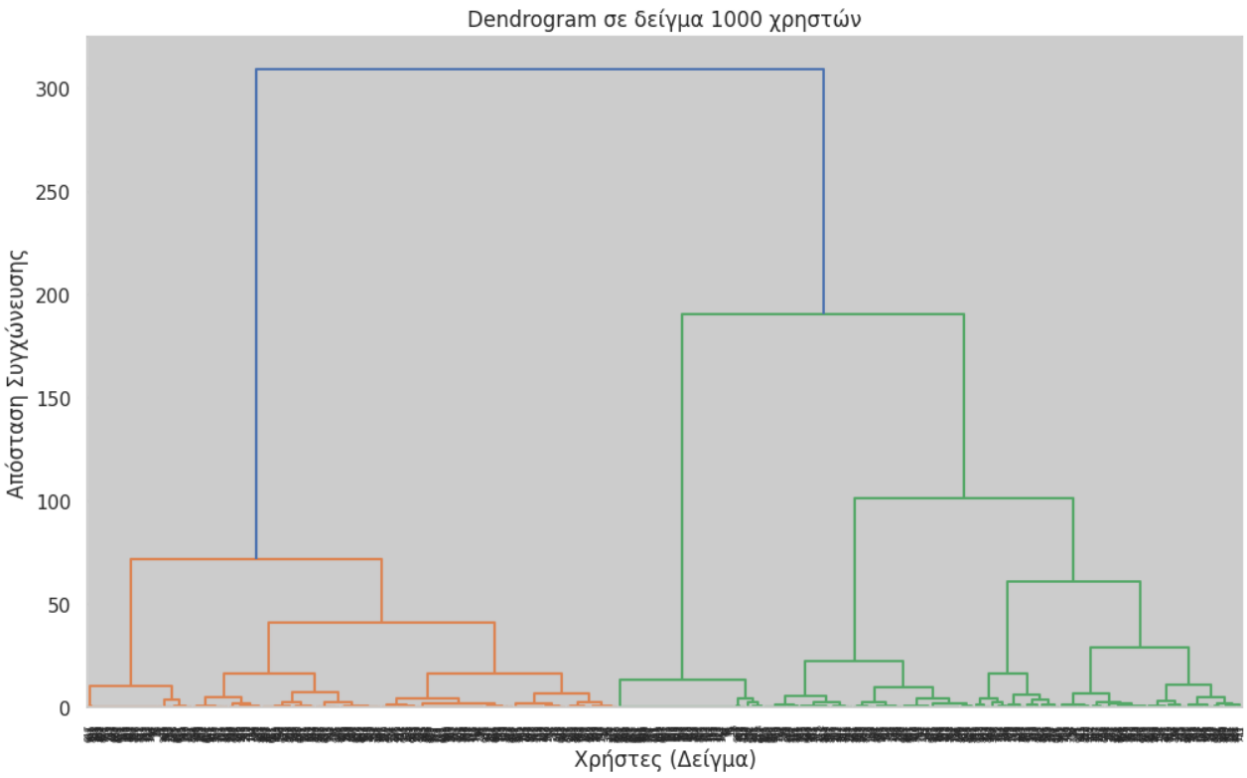


Hierarchical Clustering

Εφαρμόστηκε σε δείγμα 1.000 χρηστών με βάση μόνο το days_since_prior_order.

- **Silhouette Score**: 0.5048
- **Davies-Bouldin Index**: 0.4995

Τα αποτελέσματα ήταν ελαφρώς λιγότερο καθαρά σε σχέση με το KMeans αλλά επιβεβαίωσαν τη βασική κατανομή.



Ερμηνεία των αποτελεσμάτων και προτεινόμενες στρατηγικές

1. **cluster = 0**

- Χαμηλή Μέση Ημέρα Επαναγοράς (1-7 μέρες)
- Πολύ ενεργοί χρήστες
- Loyalty campaigns, points system

2. **cluster = 1**

- Μέτρια Μέση Ημέρα Επαναγοράς (8-15 μέρες)
- Συστηματικοί, αλλά όχι καθημερινοί
- Personalized offers, bundles

3. **cluster = 2**

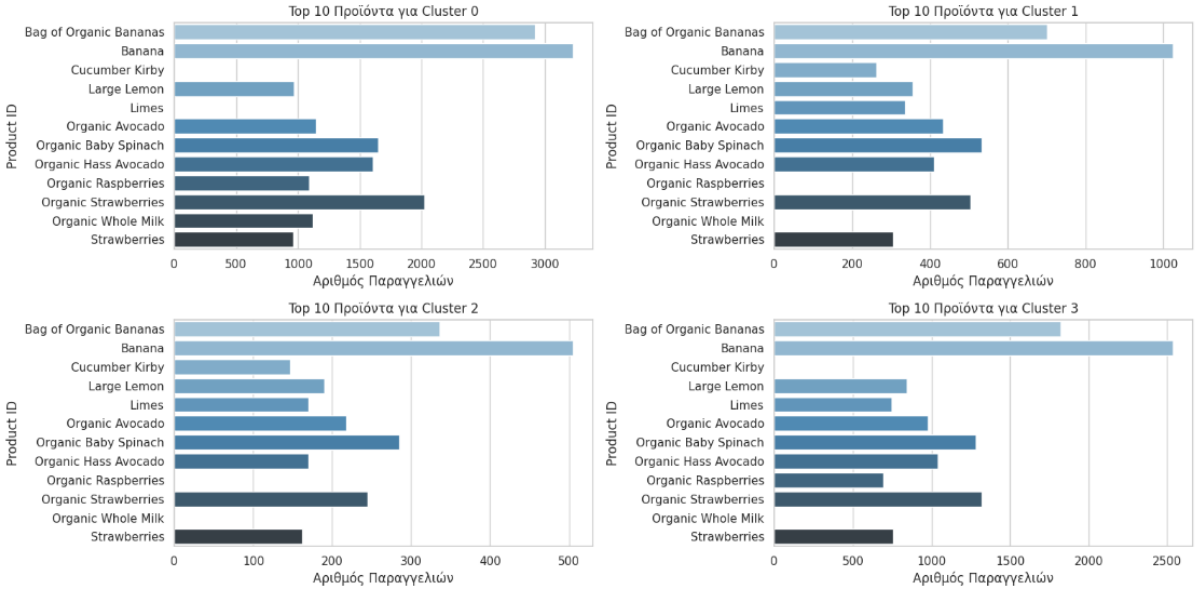
- Υψηλή Μέση Ημέρα Επαναγοράς (16-25 μέρες)
- Αραιοί αγοραστές

- Reminder emails

4. **cluster = 3**

- Πολύ υψηλή Μέση Ημέρα Επαναγοράς (>25 μέρες)
- Σχεδόν ανενεργοί χρήστες
- Reactivation campaigns, special promotions

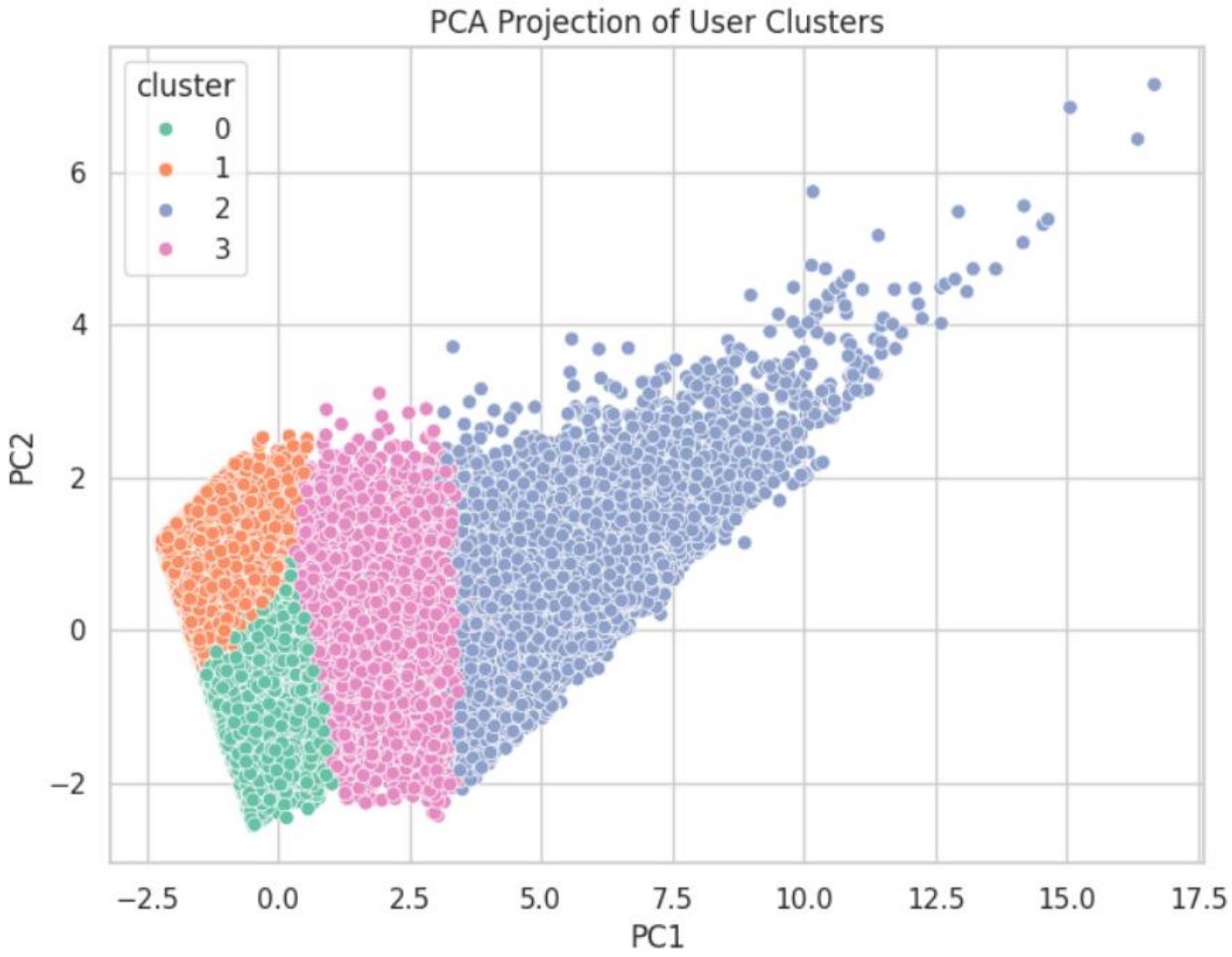
Επιπλέον, οπτικοποιήθηκαν τα Top 10 προϊόντα για κάθε cluster, με φρούτα και γαλακτοκομικά να εμφανίζονται έντονα σε όλα τα τμήματα (π.χ. "Organic Bananas", "Avocados", "Spinach").



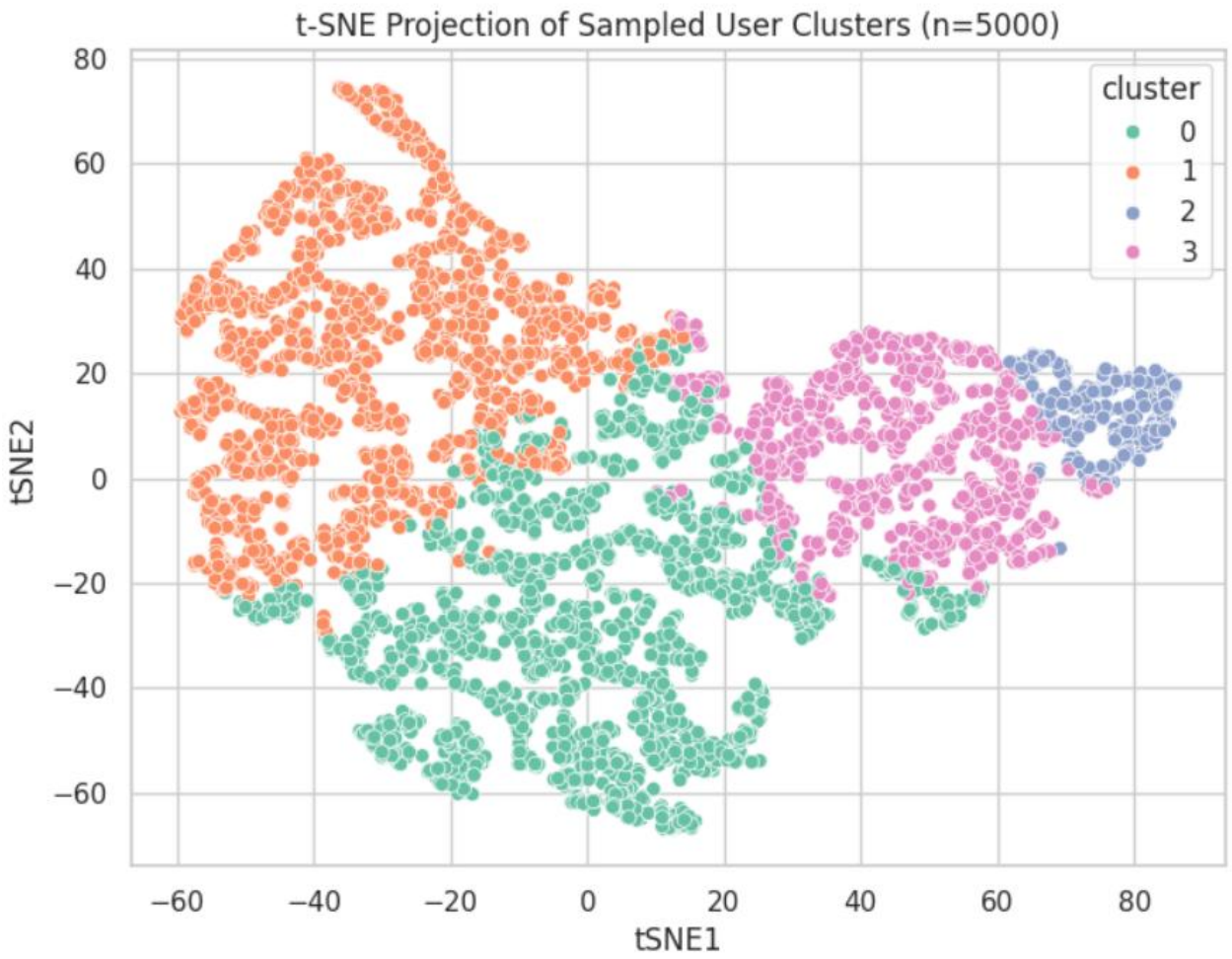
Για περαιτέρω ανάλυση, δημιουργήθηκαν πρόσθετα χαρακτηριστικά ανά χρήστη (total_orders, avg_days_between_orders, total_products, distinct_products)

Αφού έγινε **κανονικοποίηση** και εφαρμογή **KMeans (4 clusters)**, χρησιμοποιήθηκαν:

PCA: για προβολή σε 2D χώρο, με ευδιάκριτο διαχωρισμό των clusters.



t-SNE: για πιο "μη γραμμική" χαρτογράφηση σε 2D, αποκαλύπτοντας την εσωτερική δομή των clusters σε δείγμα 5.000 χρηστών.



Βελτιστοποίηση Μεταφοράς Προϊόντων (Transportation Problem)

Έστω ότι η **Instacart** επιδιώκει να βελτιστοποιήσει τη μεταφορά προϊόντων από τις αποθήκες της προς διάφορες περιοχές εξυπηρέτησης, με στόχο την ελαχιστοποίηση του κόστους μεταφοράς, ενώ ταυτόχρονα ικανοποιείται η ζήτηση των πελατών και δεν παραβιάζονται οι φυσικοί περιορισμοί χωρητικότητας.

Το πρόβλημα μοντελοποιείται ως **πρόβλημα γραμμικού προγραμματισμού** (Transportation Problem).

Οντότητες:

- **Suppliers (Πηγές):** Οι αποθήκες που διαθέτουν αποθέματα προϊόντων (π.χ. Αποθήκη1, Αποθήκη2).

- **Consumers (Καταναλωτές):** Οι περιοχές που έχουν ζήτηση (π.χ. Περιοχή1, Περιοχή2, Περιοχή3).
- **Supply:** Η ποσότητα προϊόντων που διατίθεται από κάθε αποθήκη.
- **Demand:** Η ζήτηση κάθε περιοχής.
- **Max Capacity:** Ο μέγιστος αριθμός μονάδων που μπορούν να μεταφερθούν από κάθε αποθήκη σε κάθε περιοχή.
- **Cost Matrix:** Το μοναδιαίο κόστος μεταφοράς προϊόντων από κάθε αποθήκη σε κάθε περιοχή.

```
# Προμήθεια (supply) από κάθε αποθήκη
supply = [100, 120]  # Αποθήκη 1, Αποθήκη 2

# Ζήτηση (demand) σε κάθε περιοχή
demand = [80, 70, 70]  # Περιοχή 1, 2, 3

# Κόστος μεταφοράς ανά μονάδα
costs = [
    [4, 6, 9],  # Από Αποθήκη 1
    [5, 4, 7],  # Από Αποθήκη 2
]

# Ανώτατη χωρητικότητα μεταφοράς από κάθε αποθήκη σε κάθε περιοχή
max_capacity = [
    [70, 30, 50],  # Από Αποθήκη 1
    [60, 50, 70],  # Από Αποθήκη 2
]
```

Εφαρμογή του Μοντέλου

Χρησιμοποιήθηκε η βιβλιοθήκη **scipy.optimize.linprog** για την επίλυση του προβλήματος μέσω της μεθόδου **HiGHS**. Οι περιορισμοί του μοντέλου περιλάμβαναν:

- Την ικανοποίηση της συνολικής **ζήτησης** κάθε περιοχής.
- Τη μη υπέρβαση της **προσφοράς** κάθε αποθήκης.
- Την τήρηση του **ανώτατου ορίου χωρητικότητας** στις μεταφορές.
- **Μη αρνητικές μεταβλητές**.

Βέλτιστη Λύση

Η κατανομή των ροών μεταφοράς που ελαχιστοποιεί το συνολικό κόστος παρουσιάζεται παρακάτω:

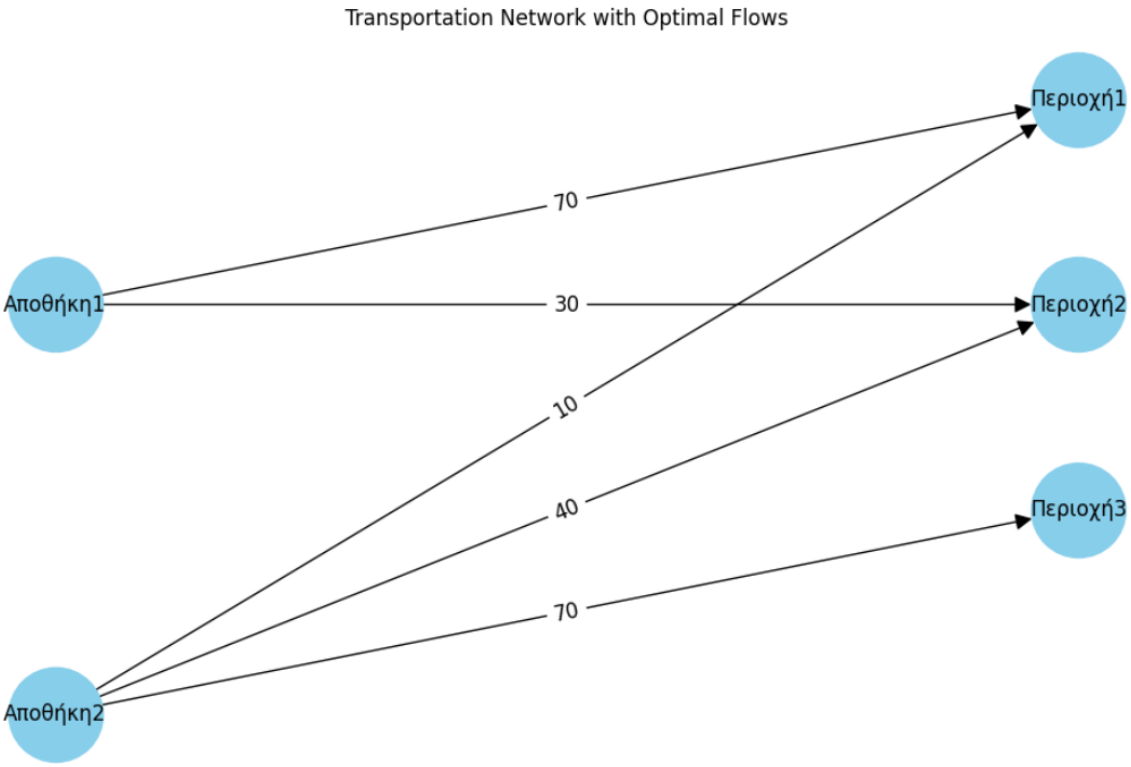
	Περιοχή1	**Περιοχή2**	**Περιοχή3**
Αποθήκη 1	70	30	0
Αποθήκη 2	10	40	70

Συνολικό Κόστος: 1160 μονάδες

Αξιοσημείωτο είναι ότι η Αποθήκη1 εξαντλεί τη χωρητικότητά της για τις Περιοχές 1 και 2, ενώ η Περιοχή3 εξυπηρετείται εξ ολοκλήρου από την Αποθήκη2, γεγονός που οφείλεται στο χαμηλότερο κόστος και την επάρκεια αποθέματος.

Οπτικοποίηση της Λύσης

Για να καταστεί πιο κατανοητή η βέλτιστη ροή προϊόντων, δημιουργήθηκε ένα **κατευθυνόμενο γράφημα (directed graph)**, όπου κάθε κόμβος αντιπροσωπεύει αποθήκες και περιοχές και κάθε ακμή αντιστοιχεί σε ροή προϊόντων.



Συμπεράσματα

Η προσέγγιση αυτή παρέχει ένα αποτελεσματικό πλαίσιο για τη λήψη αποφάσεων γύρω από τη **βελτιστοποίηση logistics** σε κατανεμημένα συστήματα αποθήκευσης και διανομής, όπως το δίκτυο της Instacart. Επεκτάσεις του μοντέλου μπορούν να περιλαμβάνουν δυναμικές παραμέτρους (π.χ. κυκλοφοριακά δεδομένα, εποχικότητα, real-time demand), καθιστώντας το ακόμη πιο χρήσιμο σε πραγματικές επιχειρησιακές συνθήκες.