

# Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

## Υλοποιητικό Πρότζεκτ

Ονοματεπώνυμο: Γκουργκούτας Βασίλειος

AM:1084667

Έτος:4ο

Email: [up1084667@ac.upatras.gr](mailto:up1084667@ac.upatras.gr)

## Καταγραφή του Περιβάλλοντος Υλοποίησης

Σαν περιβάλλον υλοποίησης χρησιμοποιήθηκε το Visual Studio Code στο οποίο αξιοποιήθηκε η γλώσσα προγραμματισμού python. Πιο συγκεκριμένα τα προγράμματα που υλοποιήθηκαν είναι της μορφής ipynb που υποδηλώνει την χρήση jupyter notebooks για την συγκεκριμένη εργασία. Για την απάντηση των ερωτημάτων χρησιμοποιήθηκαν ποικίλες βιβλιοθήκες της python.

Οι βιβλιοθήκες που αξιοποιήθηκαν είναι οι εξής:

**Pandas:** Αξιοποιήθηκε η βιβλιοθήκη pandas της python ώστε να μπορέσουμε να επεξεργαστούμε τα αρχεία csv. (πχ τροποποίηση, επεξεργασία στηλών)

Η εντολή που απαιτείται για την εγκατάσταση της συγκεκριμένης βιβλιοθήκης είναι η εξής:

```
pip install pandas
```

```
pip install pandas
✓ 1.9s

Requirement already satisfied: pandas in
Requirement already satisfied: numpy>=1.
Requirement already satisfied: python-da
Requirement already satisfied: pytz>=202
Requirement already satisfied: tzdata>=2
Requirement already satisfied: six>=1.5
```

**Matplotlib:** Η συγκεκριμένη βιβλιοθήκη της python αξιοποιήθηκε για την αναπαράσταση των αποτελεσμάτων της επεξεργασίας των csv των συμμετεχόντων με γραφήματα.

Η εγκατάσταση της απαιτεί την εξής εντολή:

pip install matplotlib

```
pip install matplotlib
✓ 1.8s

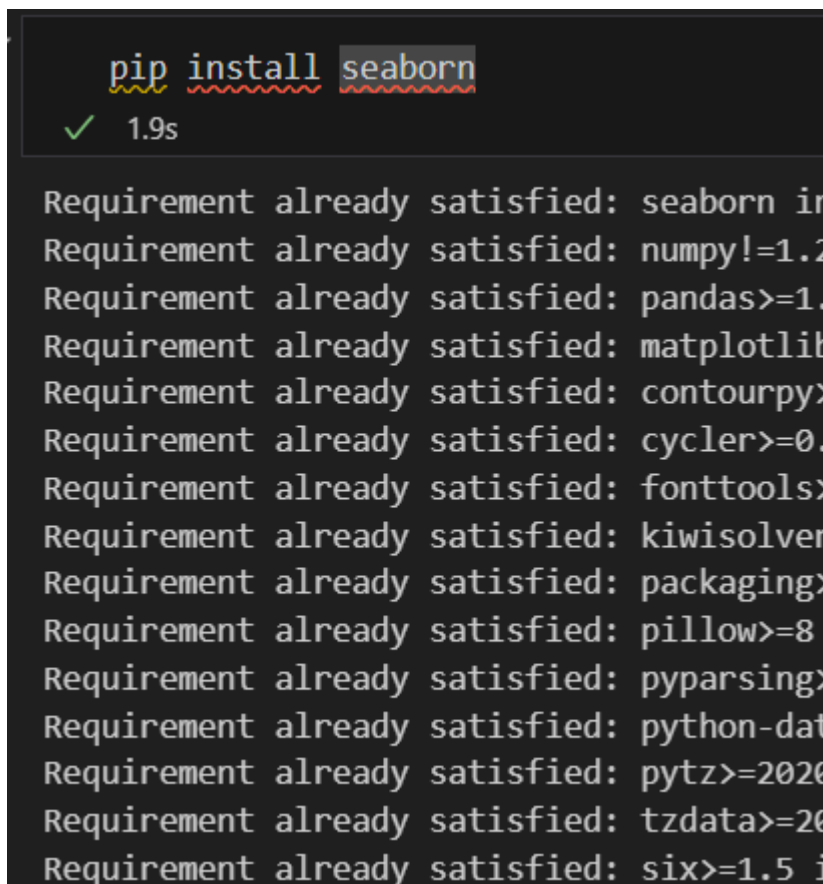
Requirement already satisfied: matplotlib i
Requirement already satisfied: contourpy>=1
Requirement already satisfied: cyclor>=0.16
Requirement already satisfied: fonttools>=4
Requirement already satisfied: kiwisolver>=
Requirement already satisfied: numpy>=1.21
Requirement already satisfied: packaging>=2
Requirement already satisfied: pillow>=8 in
Requirement already satisfied: pyparsing>=2
Requirement already satisfied: python-dateu
Requirement already satisfied: six>=1.5 in
```

**Seaborn:** Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται και αυτή για οπτικοποίηση των δεδομένων που βασίζεται στη βιβλιοθήκη

matplotlib. Ωστόσο παρέχει μια ανώτερη διεπαφή για τη δημιουργία σύνθετων γραφημάτων και την οπτικοποίηση στατιστικών δεδομένων.

Για την αξιοποίηση της βιβλιοθήκης απαιτείται η εγκατάστασή της με την εντολή:

```
pip install seaborn
```



```
pip install seaborn
✓ 1.9s

Requirement already satisfied: seaborn in
Requirement already satisfied: numpy!=1.2
Requirement already satisfied: pandas>=1.
Requirement already satisfied: matplotlib
Requirement already satisfied: contourpy>
Requirement already satisfied: cycler>=0.
Requirement already satisfied: fonttools>
Requirement already satisfied: kiwisolver
Requirement already satisfied: packaging>
Requirement already satisfied: pillow>=8
Requirement already satisfied: pyparsing>
Requirement already satisfied: python-dat
Requirement already satisfied: pytz>=2020
Requirement already satisfied: tzdata>=20
Requirement already satisfied: six>=1.5 i
```

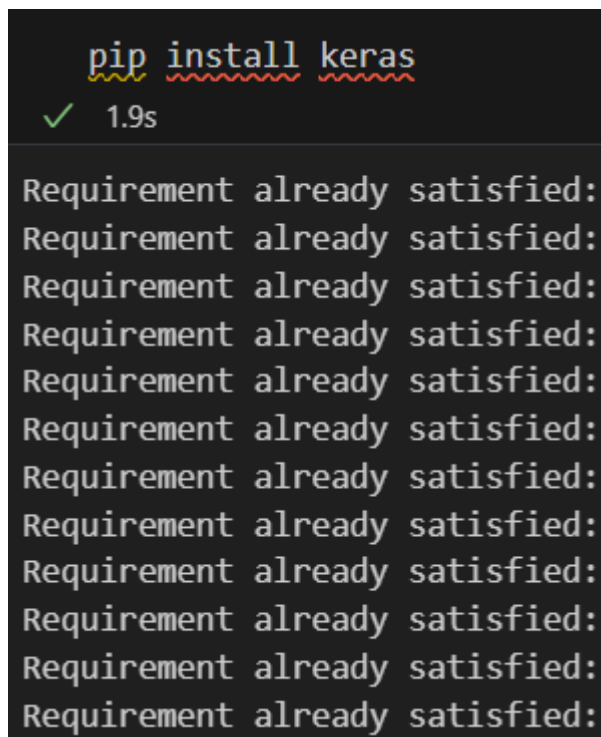
**os:** Η συγκεκριμένη βιβλιοθήκη της python δεν απαιτεί κάποια εγκατάσταση καθώς έρχεται προ εγκατεστημένη με την python. Παρέχει ένα τρόπο αλληλεπίδρασης με το λειτουργικό σύστημα. Περιέχει λειτουργίες για τη διαχείριση αρχείων και καταλόγων, τη λήψη πληροφοριών για το σύστημα, και άλλες λειτουργίες που σχετίζονται με το λειτουργικό σύστημα.

**glob:** Όπως και η βιβλιοθήκη os έτσι και η βιβλιοθήκη glob δεν απαιτεί εγκατάσταση. Χρησιμοποιείται για την εύρεση διαδρομών αρχείων και καταλόγων που ταιριάζουν σε ένα συγκεκριμένο μοτίβο. Χρησιμοποιεί wildcards (όπως το \*) για την αναζήτηση αρχείων και καταλόγων.

**keras:** Με την βιβλιοθήκη keras στην python μπορούμε να υλοποιήσουμε τον ταξινομητή με νευρωνικό δίκτυο που ζητείται στο δεύτερο ερώτημα.

Η εγκατάσταση της βιβλιοθήκης γίνεται με την εντολή:

`pip install keras`



```
pip install keras
✓ 1.9s
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
```

**Tensorflow:** Το tensorflow είναι και αυτή μία βιβλιοθήκη στην python που μαζί με το keras χρησιμοποιήθηκε για την υλοποίηση του Νευρωνικού δικτύου.

Η εγκατάστασή της γίνεται με την εντολή

✓ 2.1s

**Scikit-learn:** Η συγκεκριμένη βιβλιοθήκη στην Python ήταν πολύτιμη καθώς παρείχε τις περισσότερες υποβιβλιοθήκες για την υλοποίηση του πρότζεκτ. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι εξής: `accuracy_score`, `classification_report`,

confusion\_matrix, precision\_score, recall\_score, f1\_score,  
GaussianNB, train\_test\_split, StandardScaler, LabelEncoder,  
RandomForestClassifier, StandardScaler, PCA, KMeans,  
AgglomerativeClustering, silhouette\_score

Η εγκατάσταση γίνεται με την εντολή

```
pip install scikit-learn
✓ 2.0s
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
```

## Ερώτημα 1

**Περιγραφή του κώδικα που υλοποιήθηκε για το συγκεκριμένο ερώτημα:**

Αρχικά έχουμε τοποθετήσει όλα τα αρχεία csv στον φάκελο harth. Έπειτα διατρέχουμε με μια for όλα τα csv αρχεία ώστε να πραγματοποιήσουμε μια πρώτη ανάλυση των δεδομένων. Σε κάποια csv αρχεία παρατηρήθηκε ότι υπάρχουν επιπλέον στήλες

που δεν χρησιμοποιούνται όπως είναι η στήλη index και η στήλη η οποία αριθμεί την κάθε γραμμή και δεν φέρει κάποιο όνομα. Οι συγκεκριμένες στήλες αφαιρέθηκαν κατά την επεξεργασία των csv αρχείων με την χρήση pandas.

```
# Μετονομασία της πρώτης στήλης εάν είναι κενή ή "Unnamed"
if df.columns[0] == "" or "Unnamed" in df.columns[0]:
    df.rename(columns={df.columns[0]: 'unwanted'}, inplace=True) # Μετονομασία στήλης χωρίς όνομα

# Αφαίρεση περιττών στηλών
df.drop(columns=["unwanted", "index", "Unnamed: 0"], inplace=True, errors='ignore')
```

Έπειτα μετατράπηκε η στήλη που περιέχει τα timestamps για κάθε μέτρηση στην μορφή ημερομηνίας και ώρας με την εξής εντολή:

```
# Μετατροπή της στήλης 'timestamp' σε ημερομηνία/ώρα
df["timestamp"] = pd.to_datetime(df["timestamp"])
```

Στην συνέχεια για το κάθε csv αρχείο έγινε υπολογισμός κάποιων βασικών στατιστικών όπως είναι το min, το max, το std, το mean καθώς επίσης και τα ποσοστημόρια 25%, 50% και 75%.

Τα παραπάνω στατιστικά δίνονται με την χρήση της εξής εντολής:

```
# Υπολογισμός βασικών στατιστικών μεγεθών
summary_stats = df[columns_of_interest].describe()
print(f"\nΒασικά Συγκεντρωτικά Στατιστικά Μεγέθη για {csv_file}:")
print(summary_stats)
```

Έπειτα με την χρήση του κομματιού κώδικα:

```
# Χρονοσειρές για να δούμε μοτίβα
plt.figure(figsize=(12, 8))
for col in df.columns:
    if col != "timestamp":
        plt.plot(df["timestamp"], df[col], label=col) # Χωρίς ανεπιθύμητες στήλες
```

Δημιουργούμε χρονοσειρές σε γραφήματα όπου αποτυπώνονται τα χαρακτηριστικά με ετικέτες: back\_x, back\_y, back\_z, thigh\_x, thigh\_y, thigh\_z και label.

Με αυτόν τον τρόπο μπορούμε να δούμε με βάση την ετικέτα label και την κατανομή των δεδομένων στο γράφημα από την συλλογή των δεδομένων στο back και στο thigh τι είδους άσκηση έκανε ο κάθε συμμετέχοντας. Το είδος της άσκησης δίνεται από την ετικέτα label η οποία περιέχει νούμερα και το κάθε νούμερο αντιστοιχεί σε ένα είδος άσκησης:

The dataset contains the following annotated activities with the corresponding coding:

- 1: walking
- 2: running
- 3: shuffling
- 4: stairs (ascending)
- 5: stairs (descending)
- 6: standing
- 7: sitting
- 8: lying
- 13: cycling (sit)
- 14: cycling (stand)
- 130: cycling (sit, inactive)
- 140: cycling (stand, inactive)

Τα γραφήματα αποτυπώνονται με την εξής εκτέλεση κώδικα:

```
plt.title("Χρονοσειρά των Μετρήσεων")
plt.xlabel("Χρόνος")
plt.ylabel("Τιμή")
plt.legend()
plt.show()
```

Στην συνέχεια με την χρήση των εντολών:



```
# Απεικόνιση των συσχετίσεων
corr_matrix = df.iloc[:, 1:].corr() # Χωρίς τις ανεπιθύμητες στήλες
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title(f"Απεικόνιση των Συσχετίσεων για {csv_file}")
plt.show()
```

Απεικονίζουμε σε γράφημα τις συσχετίσεις μεταξύ των χαρακτηριστικών (θερμογραφική απεικόνιση). Επιπλέον οι τιμές των συσχετίσεων εκτυπώνονται κάθε φορά και σε έναν πίνακα με την εξής εντολή:

```
# Εμφάνιση του πίνακα συσχετίσεων
print(f"\nΠίνακας Συσχετίσεων για {csv_file}:")
print(corr_matrix)
```

Τέλος έχουμε υλοποιήσει ένα το κομμάτι κώδικα σύμφωνα με το οποίο θα εμφανίζονται ιστογράμματα. Το κάθε ιστόγραμμα αναφέρεται σε ένα χαρακτηριστικό των αρχείων csv. Τα χαρακτηριστικά που αποτυπώνονται στα ιστογράμματα είναι τα εξής: back\_x, back\_y, back\_z, thigh\_x, thigh\_y, thigh\_z, label.

Στα ιστογράμματα παρουσιάζονται οι τιμές που εμφανίζονται στο κάθε χαρακτηριστικό καθώς επίσης και το πλήθος των τιμών για το κάθε χαρακτηριστικό. Ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση των ιστογραμμάτων είναι ο εξής:

```
# Ιστογράμματα για επιλεγμένες στήλες
columns_of_interest = ["back_x", "back_y", "back_z", "thigh_x", "thigh_y", "thigh_z", "label"]
df[columns_of_interest].hist(figsize=(12, 10), bins=30)
plt.suptitle(f"Ιστογράμματα για {csv_file}")
plt.show()
```

## Τελικά Αποτελέσματα

Έπειτα από την εκτέλεση του κώδικα λαμβάνουμε αρχικά τις μετρήσεις των βασικών συγκεντρωτικών χαρακτηριστικών. Παρακάτω δίνεται ένα παράδειγμα από την εκτέλεση του κώδικα

για τα δύο πρώτα csv και όμοια αποτελέσματα προκύπτουν για τα υπόλοιπα 20 csv αρχεία των συμμετεχόντων.

Επεξεργασία αρχείου: harth\S006.csv

Βασικά Συγκεντρωτικά Στατιστικά Μεγέθη για harth\S006.csv:

	back_x	back_y	back_z	thigh_x \
count	408709.000000	408709.000000	408709.000000	408709.000000
mean	-0.802201	-0.000687	-0.274718	-0.370317
std	0.238347	0.189062	0.441805	0.506666
min	-3.542889	-3.016498	-1.024363	-6.844045
25%	-0.983647	0.001063	-0.702338	-0.952840
50%	-0.937195	0.033240	-0.277446	-0.277711
75%	-0.654541	0.074822	0.064811	0.068999
max	0.952109	2.569339	1.628023	3.898547

	thigh_y	thigh_z	label
count	408709.000000	408709.000000	408709.000000
mean	0.143471	0.617527	10.190187
std	0.213864	0.536430	20.328336
min	-5.757406	-4.884791	1.000000
25%	0.022534	0.144114	6.000000
50%	0.086248	0.924066	7.000000
75%	0.246292	1.001372	7.000000
max	4.602909	5.391660	130.000000

Επεξεργασία αρχείου: harth\S008.csv

Βασικά Συγκεντρωτικά Στατιστικά Μεγέθη για harth\S008.csv:

	back_x	back_y	back_z	thigh_x \
count	418989.000000	418989.000000	418989.000000	418989.000000
mean	-0.920351	0.040018	-0.326746	-0.332798
std	0.130877	0.107516	0.297591	0.463427
min	-3.066853	-1.209330	-0.960136	-5.922062
25%	-0.998823	-0.013310	-0.580225	-0.882999
50%	-0.972054	0.048302	-0.297335	-0.085027
75%	-0.826290	0.101082	-0.143575	-0.005905
max	0.873471	1.255642	1.872940	2.312848

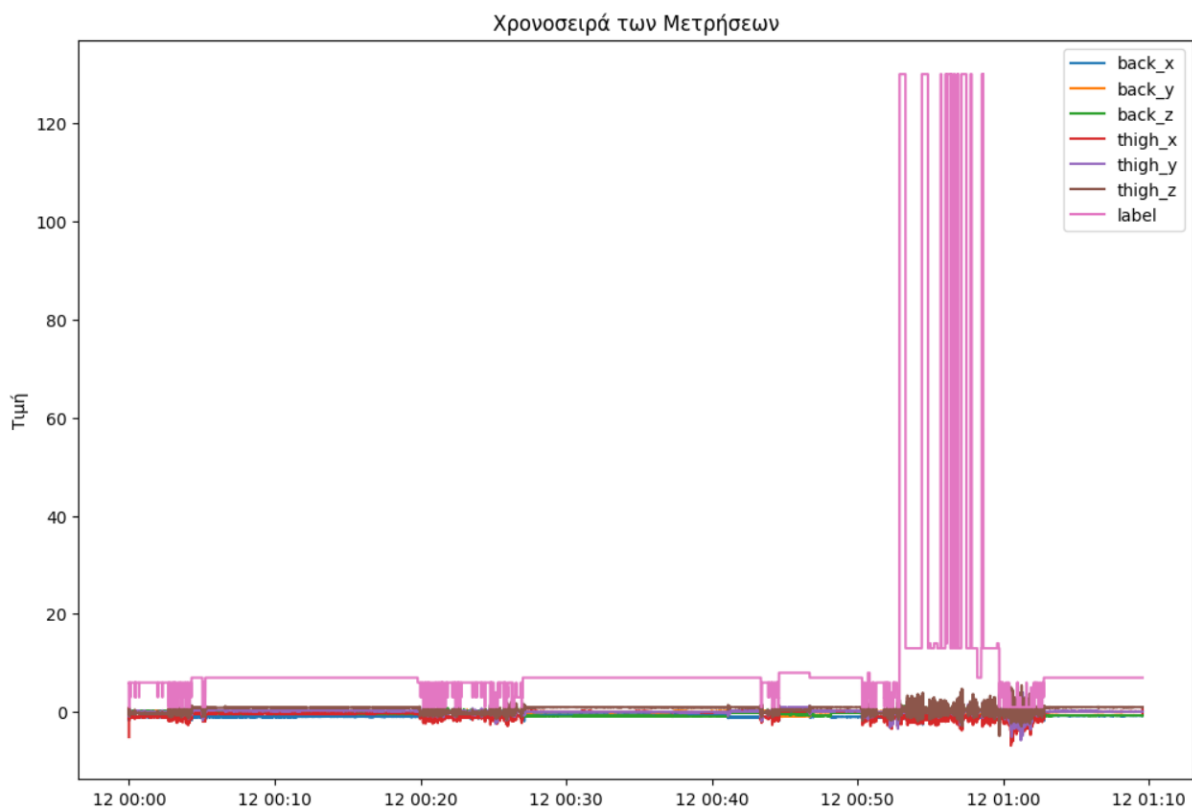
	thigh_y	thigh_z	label
count	418989.000000	418989.000000	418989.000000
mean	0.050361	0.656598	6.661819
std	0.218070	0.523197	6.445625
min	-2.857320	-4.233158	1.000000
25%	-0.088989	0.256630	6.000000
50%	0.068947	0.950087	7.000000
75%	0.166770	0.994429	7.000000
max	4.809320	5.324356	140.000000

Όπως προαναφέραμε για όλα τα csv αρχεία λαμβάνουμε συγκεντρωτικά χαρακτηριστικά με σημαντικές πληροφορίες ώστε να κατανοήσουμε καλύτερα την κατανομή των δεδομένων. Τα βασικά συγκεντρωτικά χαρακτηριστικά αποτελούνται από το πλήθος των συνολικών γραμμών του κάθε αρχείου, την μέση τιμή που προκύπτει για κάθε χαρακτηριστικό, την τυπική απόκλιση ώστε να δούμε πόσο απέχουν οι τιμές των δεδομένων κάθε χαρακτηριστικού από τον μέσο όρο τους. Επιπλέον, δίνεται η μικρότερη τιμή για κάθε χαρακτηριστικό με την τιμή min καθώς επίσης και η μεγαλύτερη τιμή με το max. Τέλος παρέχουμε τα ποσοστημόρια 25%,50% και 75%. Πιο συγκεκριμένα το 25% δείχνει την τιμή κάτω από την οποία βρίσκεται το 25% των δεδομένων. Όμοια το 50% δείχνει την τιμή κάτω από την οποία

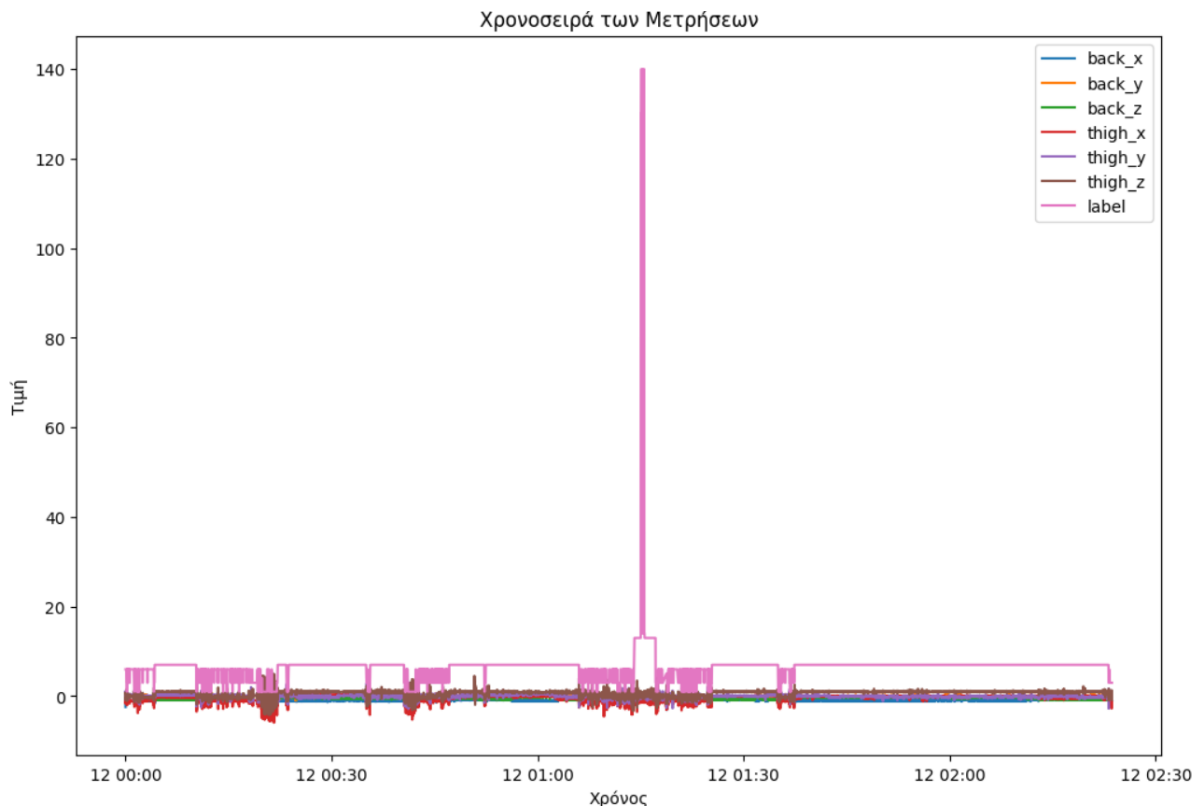
βρίσκεται το 50% των δεδομένων και το 75% δείχνει την τιμή κάτω από την οποία βρίσκεται το 75% των δεδομένων.

Στην συνέχεια δίνεται ένα γράφημα για κάθε csv το οποίο περιέχει τις μετρήσεις που λήφθηκαν από τον δεξί μηρό και από το κάτω μέρος της πλάτης μαζί με τις ετικέτες που δηλώνουν την κίνηση του κάθε συμμετέχοντα κάθε χρονική στιγμή. Με το συγκεκριμένο γράφημα έχουμε την δυνατότητα να παρατηρήσουμε με βάση τις μετρήσεις των χαρακτηριστικών back\_x, back\_y, back\_z, thigh\_x, thigh\_y, thigh\_z το είδος της κίνησης εκείνη την χρονική στιγμή από το χαρακτηριστικό label.

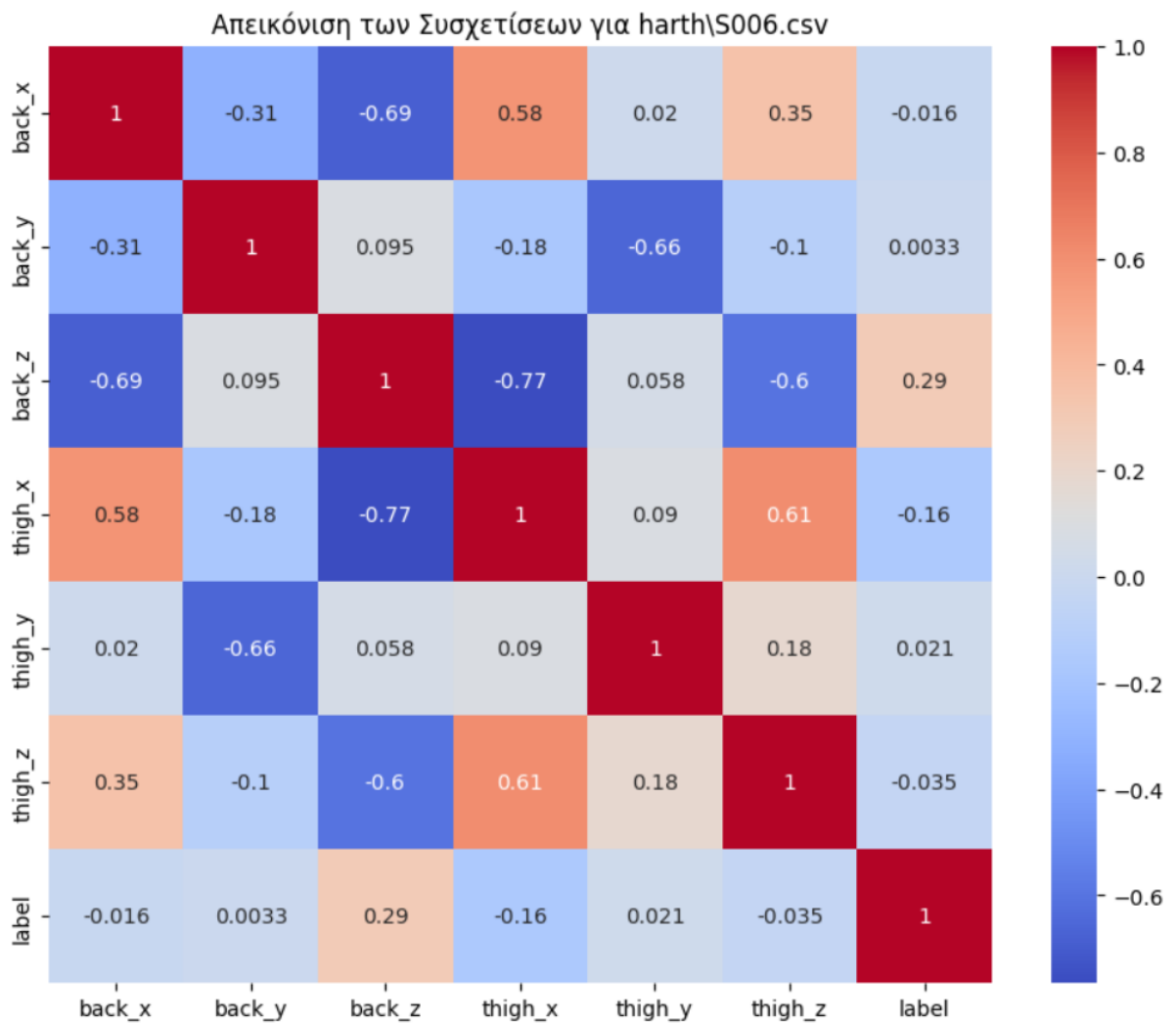
Για το csv006:



Για το csv008:

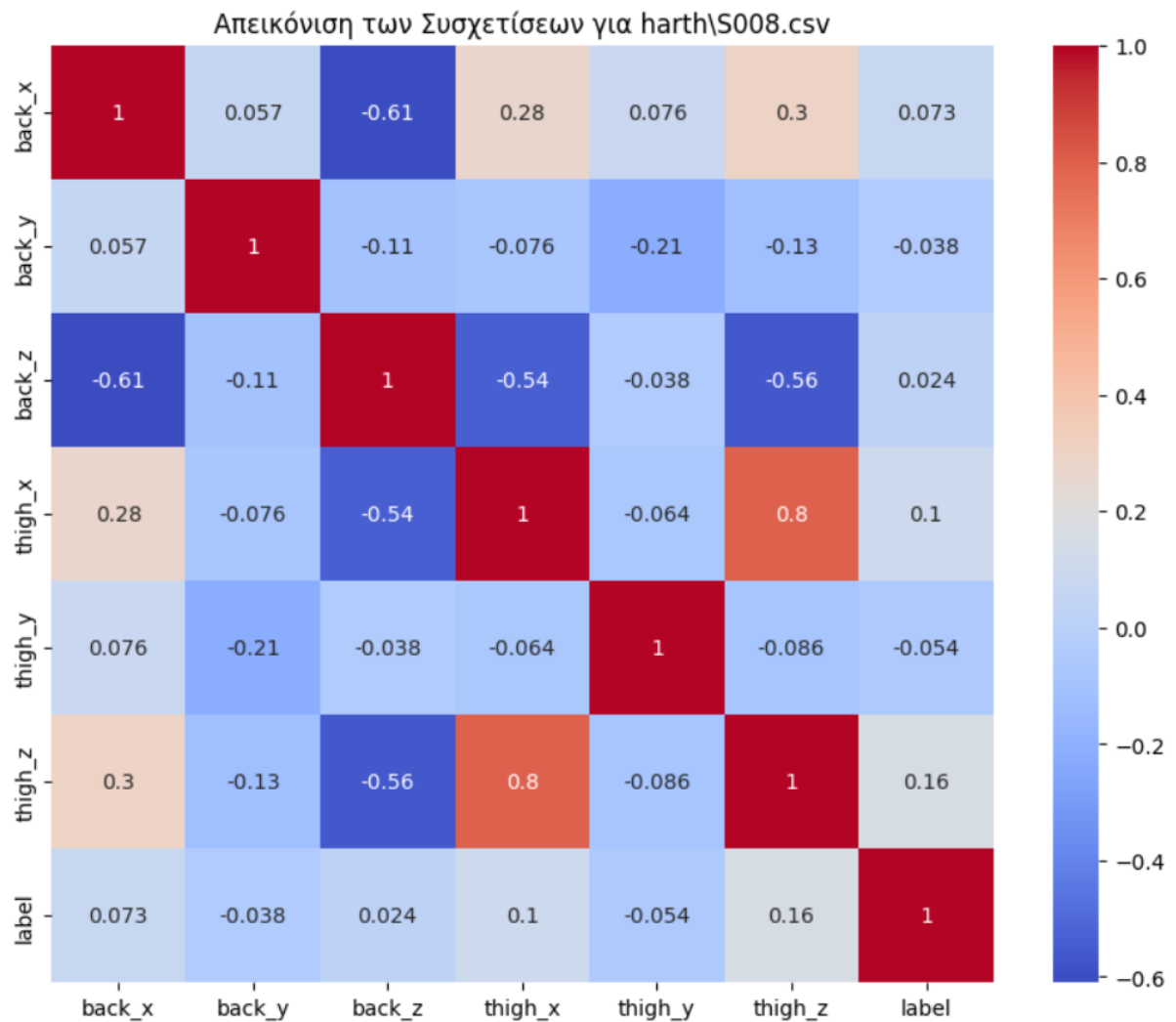


- Επιπρόσθετα δίνεται ένας πίνακας συσχετίσεων μεταξύ των χαρακτηριστικών καθώς επίσης και γραφική απεικόνιση των συσχετίσεων. Μέσα από τις συσχετίσεις μεταξύ των χαρακτηριστικών μπορούμε να εντοπίσουμε εάν δύο χαρακτηριστικά συσχετίζονται μεταξύ τους και σε τι βαθμό. Για παράδειγμα εάν οι τιμές των συσχετίσεων είναι κοντά στο 1 ή στο -1 τότε έχουμε ισχυρή συσχέτιση μεταξύ των χαρακτηριστικών, ενώ εάν η τιμή πλησιάζει στο μηδέν, τότε αυτό υποδεικνύει ότι δεν υπάρχει συσχέτιση μεταξύ των χαρακτηριστικών. Επομένως δεν θα πρέπει να αμελήσουμε ότι ο πίνακας συσχέτισης παρέχει σημαντική πληροφορία καθώς η ανάλυση αυτή μπορεί να βοηθήσει στην επιλογή των πιο σημαντικών μεταβλητών για την κατασκευή μοντέλων μηχανικής μάθησης, εξαιρώντας τις μεταβλητές που δεν συσχετίζονται ισχυρά με τη μεταβλητή στόχο. Παρακάτω δίνονται screenshot του πίνακα συσχετίσεων καθώς και γραφική απεικόνισή του.



Πίνακας Συσχετίσεων για harth\S006.csv:

	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
back_x	1.000000	-0.311357	-0.685972	0.584329	0.020321	0.348377	-0.015739
back_y	-0.311357	1.000000	0.095169	-0.179142	-0.658729	-0.104510	0.003336
back_z	-0.685972	0.095169	1.000000	-0.765473	0.058496	-0.601516	0.289784
thigh_x	0.584329	-0.179142	-0.765473	1.000000	0.090144	0.611128	-0.164220
thigh_y	0.020321	-0.658729	0.058496	0.090144	1.000000	0.183705	0.020728
thigh_z	0.348377	-0.104510	-0.601516	0.611128	0.183705	1.000000	-0.034696
label	-0.015739	0.003336	0.289784	-0.164220	0.020728	-0.034696	1.000000



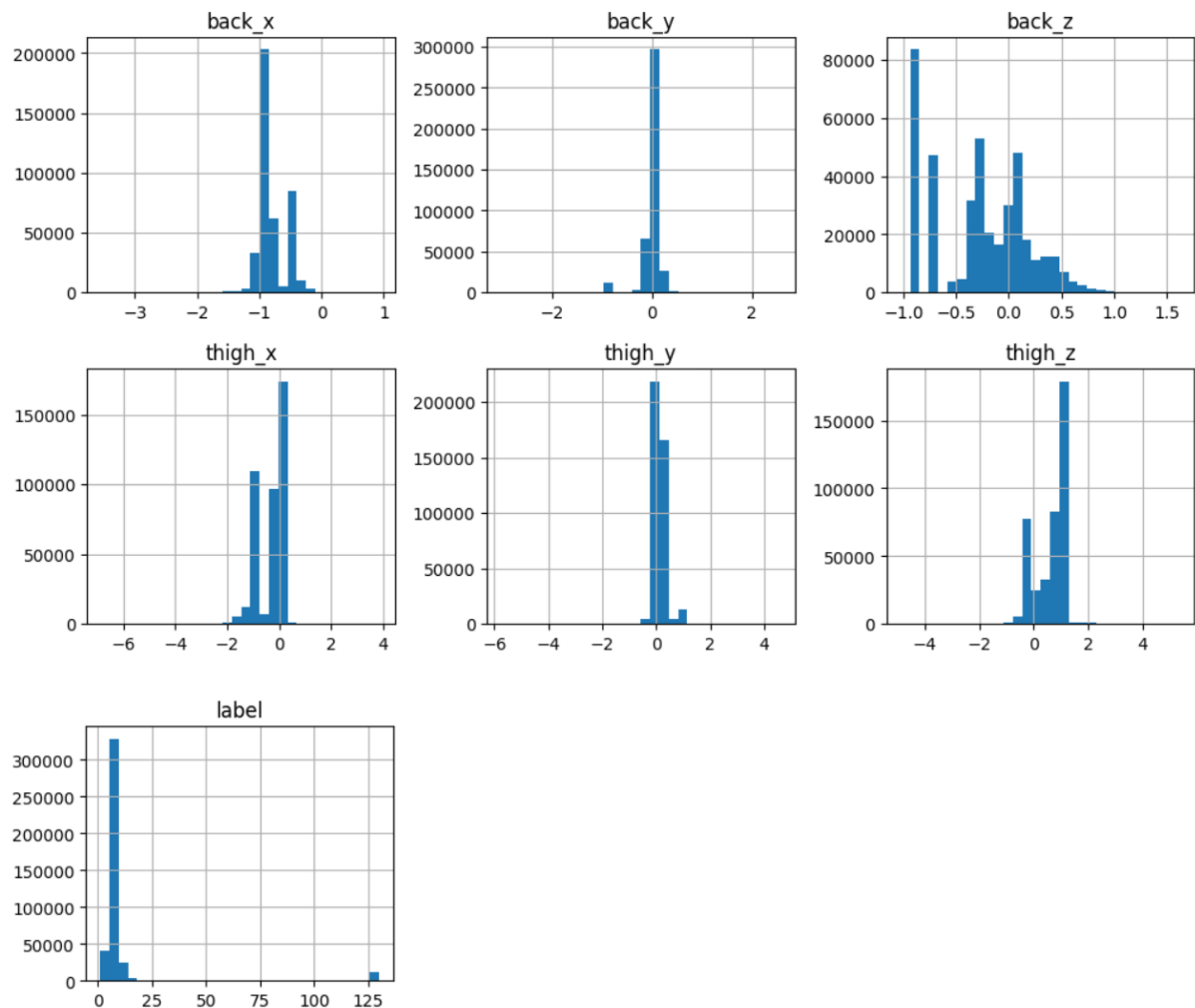
Πίνακας Συσχετίσεων για harth\S008.csv:

	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
back_x	1.000000	0.057371	-0.609736	0.275009	0.076364	0.301330	0.073391
back_y	0.057371	1.000000	-0.106374	-0.076440	-0.214233	-0.125404	-0.038467
back_z	-0.609736	-0.106374	1.000000	-0.536753	-0.038452	-0.562754	0.024301
thigh_x	0.275009	-0.076440	-0.536753	1.000000	-0.063749	0.796152	0.103219
thigh_y	0.076364	-0.214233	-0.038452	-0.063749	1.000000	-0.085586	-0.053859
thigh_z	0.301330	-0.125404	-0.562754	0.796152	-0.085586	1.000000	0.156191
label	0.073391	-0.038467	0.024301	0.103219	-0.053859	0.156191	1.000000

Τέλος δημιουργούμε και ένα ιστόγραμμα για κάθε csv αρχείο το οποίο μας παρέχει χρήσιμες πληροφορίες όσον αφορά το πλήθος των γραμμών που αποτελείται το κάθε csv. Επιπλέον με το ιστόγραμμα βλέπουμε τις τιμές που λαμβάνει ένα χαρακτηριστικό καθώς επίσης και το πλήθος των τιμών αυτών. Ουσιαστικά με την χρήση του ιστογράμματος κατανοούμε την

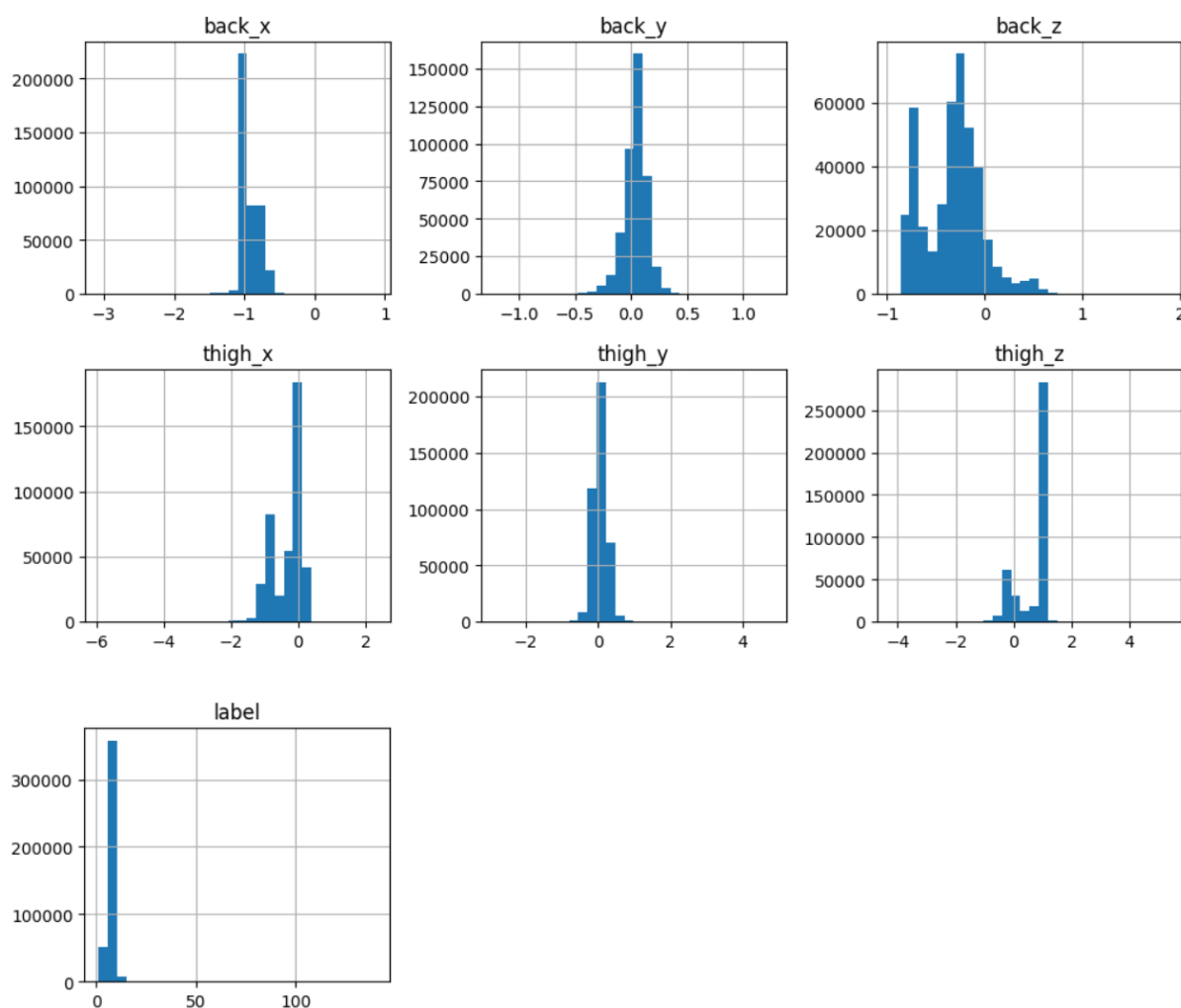
κατανομή των δεδομένων και μπορούμε να εντοπίσουμε κάποιου είδους ανωμαλία ή ακραίες τιμές. Παρακάτω δίνονται screenshots από τα ιστογράμματα κάποιων csv των συμμετεχόντων:

Ιστογράμματα για harth\S006.csv





Ιστογράμματα για harth\S008.csv



## Ερώτημα 2:

Στην συνέχεια υλοποιήθηκαν τρεις ταξινομητές: ένας σε Neural Networks, ένας σε Random Forest και ένας σε Bayesian Networks.

Πιο συγκεκριμένα για τους ταξινομητές που υλοποιήθηκαν, για τον ταξινομητή σε Νευρωνικό Δίκτυο, αξιοποιήθηκε η βιβλιοθήκη keras tensorflow της Python. Για τον ταξινομητή με Random Forest χρησιμοποιήθηκε η ήδη υλοποιημένη συνάρτηση RandomForestClassifier. Επιπλέον για τον

ταξινομητή Bayesian Network χρησιμοποιήθηκε η συνάρτηση GaussianNB.

Αρχικά με το διάβασμα των αρχείων κάναμε διαχωρισμό των δεδομένων σε δεδομένα εκπαίδευσης και δεδομένα ελέγχου.

```
# Διαχωρισμός σε σύνολα εκπαίδευσης και δοκιμής
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Έπειτα για καλύτερη αξιοποίηση των δεδομένων έγινε κανονικοποίησή τους και δημιουργήθηκαν ετικέτες εξόδου για την έξοδο y όσον αφορά τον ταξινομητή σε Νευρωνικό δίκτυο.

```
# Δημιουργία ενός LabelEncoder για τις ετικέτες εξόδου
label_encoder = LabelEncoder()

# Εφαρμογή του LabelEncoder στις ετικέτες y_train και y_test σε περίπτωση που τα δεδομένα μας στις ετικέτες δεν ήταν αριθμητικά
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

Στην συνέχεια υλοποιήθηκε το νευρωνικό δίκτυο το οποίο αποτελείται από ένα επίπεδο εισόδου, δύο κρυφά επίπεδα και ένα επίπεδο εξόδου.

```
# Αρχικοποίηση του μοντέλου και υλοποίηση του Νευρωνικού Δικτύου
model = Sequential()
model.add(Input(shape=(X_train_scaled.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(len(df['label'].unique()), activation='softmax'))
```

Όσον αφορά τον ταξινομητή σε RandomForest υλοποιήθηκε το εξής κομμάτι κώδικα

```
# Δημιουργία και εκπαίδευση του ταξινομητή Random Forests
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Όμοια για τον ταξινομητή σε Bayesian Network χρησιμοποιήθηκε το εξής κομμάτι κώδικα

```
# Δημιουργία και εκπαίδευση του ταξινομητή Gaussian Naive Bayes
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
```

Έπειτα από την εκπαίδευση του μοντέλου γίνεται πρόβλεψη του είδους της φυσικής δραστηριότητας.

Για τον ταξινομητή σε νευρωνικό δίκτυο

```
# Προβλέψεις για το σύνολο δοκιμής
y_pred_prob = model.predict(X_test_scaled)
y_pred = np.argmax(y_pred_prob, axis=1)
```

Για τον ταξινομητή σε Random Forest

```
# Προβλέψεις με το σύνολο δοκιμής
y_pred = rf_model.predict(X_test)
```

Για τον ταξινομητή σε Bayesian Networks

```
# Προβλέψεις με το σύνολο δοκιμής
y_pred = nb_model.predict(X_test)
```

Στην συνέχεια εκτυπώνονται οι μετρικές accuracy, precision, f1\_score και recall για να αξιολογήσουμε τα μοντέλα μας.

Για τον ταξινομητή σε Νευρωνικό Δίκτυο

```
# Εκτύπωση μετρικών precision, recall, f1-score, accuracy
precision = precision_score(y_test_encoded, y_pred, average='weighted')
recall = recall_score(y_test_encoded, y_pred, average='weighted')
f1 = f1_score(y_test_encoded, y_pred, average='weighted')
accuracy = accuracy_score(y_test_encoded, y_pred)

print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")
print(f"Accuracy: {accuracy}")
```

Για τον ταξινομητή σε Random Forest

```
# Υπολογισμός μετρικών
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

Για τον ταξινομητή σε Bayesian Networks

```
# Υπολογισμός μετρικών
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
```

Επιπλέον έχει χρησιμοποιηθεί και κομμάτι κώδικα το οποίο αναπαριστά γραφικά τις αναμενόμενες τιμές και τις προβλεπόμενες τιμές.

Για τον ταξινομητή σε Νευρωνικό Δίκτυο

```
# Δημιουργία γραφήματος με τις πραγματικές και τις προβλεπόμενες τιμές ώστε να συγκρίνουμε τα αποτελέσματα
plt.figure(figsize=(12, 6))
plt.stem(X_test.index, y_test, label='Πραγματικές Τιμές', linefmt='b-', markerfmt='bo', basefmt=' ')
plt.stem(X_test.index, y_pred, label='Προβλεπόμενες Τιμές', linefmt='r--', markerfmt='ro', basefmt=' ')
plt.xlabel("Δείκτης")
plt.ylabel("Ετικέτα Δραστηριότητας")
plt.title("Πραγματικές και Προβλεπόμενες Τιμές")
plt.legend()
plt.show()
```

Για τον ταξινομητή σε Random Forest

```
# Διάγραμμα βλαστών για προβλεπόμενες και πραγματικές τιμές
plt.figure(figsize=(12, 6))
plt.stem(X_test.index, y_test, label='Πραγματικές Τιμές', linefmt='b-', markerfmt='bo', basefmt=' ')
plt.stem(X_test.index, y_pred, label='Προβλεπόμενες Τιμές', linefmt='r--', markerfmt='ro', basefmt=' ')
plt.xlabel("Δείκτης")
plt.ylabel("Ετικέτα Δραστηριότητας")
plt.title("Πραγματικές και Προβλεπόμενες Τιμές με τη Μέθοδο Stem Plot")
plt.legend()
plt.show()
```

Για τον ταξινομητή σε Bayesian Networks

```
# Διάγραμμα βλαστών για προβλεπόμενες και πραγματικές τιμές
plt.figure(figsize=(12, 6))
plt.stem(range(len(y_test)), y_test, label='Πραγματικές Τιμές', linefmt='b-', markerfmt='bo', basefmt=' ')
plt.stem(range(len(y_test)), y_pred, label='Προβλεπόμενες Τιμές', linefmt='r--', markerfmt='ro', basefmt=' ')
plt.xlabel("Δείκτης")
plt.ylabel("Ετικέτα Δραστηριότητας")
plt.title("Πραγματικές και Προβλεπόμενες Τιμές με τη Μέθοδο Stem Plot")
plt.legend()
plt.show()
```

Επιπρόσθετα έχει υλοποιηθεί κώδικας για την αναπαράσταση πινάκων σύγχυσης έτσι ώστε να αναλύσουμε καλύτερα της απόδοση ενός ταξινομητή.

Για τον ταξινομητή σε Νευρωνικό Δίκτυο

```
# Δημιουργία πίνακα σύγχυσης
conf_matrix = confusion_matrix(y_test_encoded, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title(f"Πίνακας Σύγχυσης για {csv_file}")
plt.xlabel("Προβλεπόμενη")
plt.ylabel("Πραγματική")
plt.show()
```

Για τον ταξινομητή σε Random Forest

```
# Δημιουργία πίνακα σύγχυσης
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=y.unique(), yticklabels=y.unique())
plt.title(f"Πίνακας Σύγχυσης για {csv_file}")
plt.xlabel("Προβλεπόμενη")
plt.ylabel("Πραγματική")
plt.show()
```

Για τον ταξινομητή σε Bayesian Networks

```
# Δημιουργία πίνακα σύγχυσης
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.title(f"Πίνακας Σύγχυσης για {csv_file}")
plt.xlabel("Προβλεπόμενη")
plt.ylabel("Πραγματική")
plt.show()
```

Τέλος υπολογίζονται οι μέσοι όροι των μετρικών μαζί με μία τυπική απόκλιση έπειτα από την εκτέλεση του μοντέλου και για τα 22 csv αρχεία ώστε να συγκρίνουμε τα μοντέλα μας.

Για τον ταξινομητή σε Νευρωνικό Δίκτυο

```

# Υπολογισμός των μέσων όρων και των τυπικών αποκλίσεων για κάθε μετρική
mean_precision = np.mean(precisions)
mean_recall = np.mean(recalls)
mean_f1_score = np.mean(f1_scores)
mean_accuracy = np.mean(accuracies)

std_precision = np.std(precisions)
std_recall = np.std(recalls)
std_f1_score = np.std(f1_scores)
std_accuracy = np.std(accuracies)

# Εκτύπωση των μέσων όρων και των τυπικών αποκλίσεων των μετρικών
print("\nΜέσοι Όροι και Τυπικές Αποκλίσεις Μετρικών:")
print(f"Mean Precision: {mean_precision} (±{std_precision})")
print(f"Mean Recall: {mean_recall} (±{std_recall})")
print(f"Mean F1-Score: {mean_f1_score} (±{std_f1_score})")
print(f"Mean Accuracy: {mean_accuracy} (±{std_accuracy})")

```

Για τον ταξινομητή σε Random Forest

```

# Υπολογισμός και εκτύπωση των μέσων όρων και των τυπικών αποκλίσεων για τις μετρικές
mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)

mean_precision = np.mean(precisions)
std_precision = np.std(precisions)

mean_recall = np.mean(recalls)
std_recall = np.std(recalls)

mean_f1 = np.mean(f1_scores)
std_f1 = np.std(f1_scores)

# Εκτύπωση των μέσων όρων και των τυπικών αποκλίσεων των μετρικών
print("\nΜέσοι Όροι και Τυπικές Αποκλίσεις Μετρικών:")
print(f"Mean Precision: {mean_precision} (±{std_precision})")
print(f"Mean Recall: {mean_recall} (±{std_recall})")
print(f"Mean F1-Score: {mean_f1} (±{std_f1})")
print(f"Mean Accuracy: {mean_accuracy} (±{std_accuracy})")

```

Για τον ταξινομητή σε Bayesian Networks

```
# Υπολογισμός και εκτύπωση των μέσων όρων και των τυπικών αποκλίσεων για τις μετρικές
mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)

mean_precision = np.mean(precisions)
std_precision = np.std(precisions)

mean_recall = np.mean(recalls)
std_recall = np.std(recalls)

mean_f1 = np.mean(f1_scores)
std_f1 = np.std(f1_scores)

print(f"Μέσος Όρος Ακρίβειας: {mean_accuracy} (±{std_accuracy})")
print(f"Μέσος Όρος Precision: {mean_precision} (±{std_precision})")
print(f"Μέσος Όρος Recall: {mean_recall} (±{std_recall})")
print(f"Μέσος Όρος F1-Score: {mean_f1} (±{std_f1})")
```

## Τελικά Αποτελέσματα

Παρακάτω δίνονται κάποια screenshots από την εκτέλεση των υλοποιημένων ταξινομητών στην python. Παραθέτουμε τα δύο πρώτα αρχεία csv από τα 22 csv και ο κώδικας όμοια παράγει αποτελέσματα για τα υπόλοιπα 20 csv.

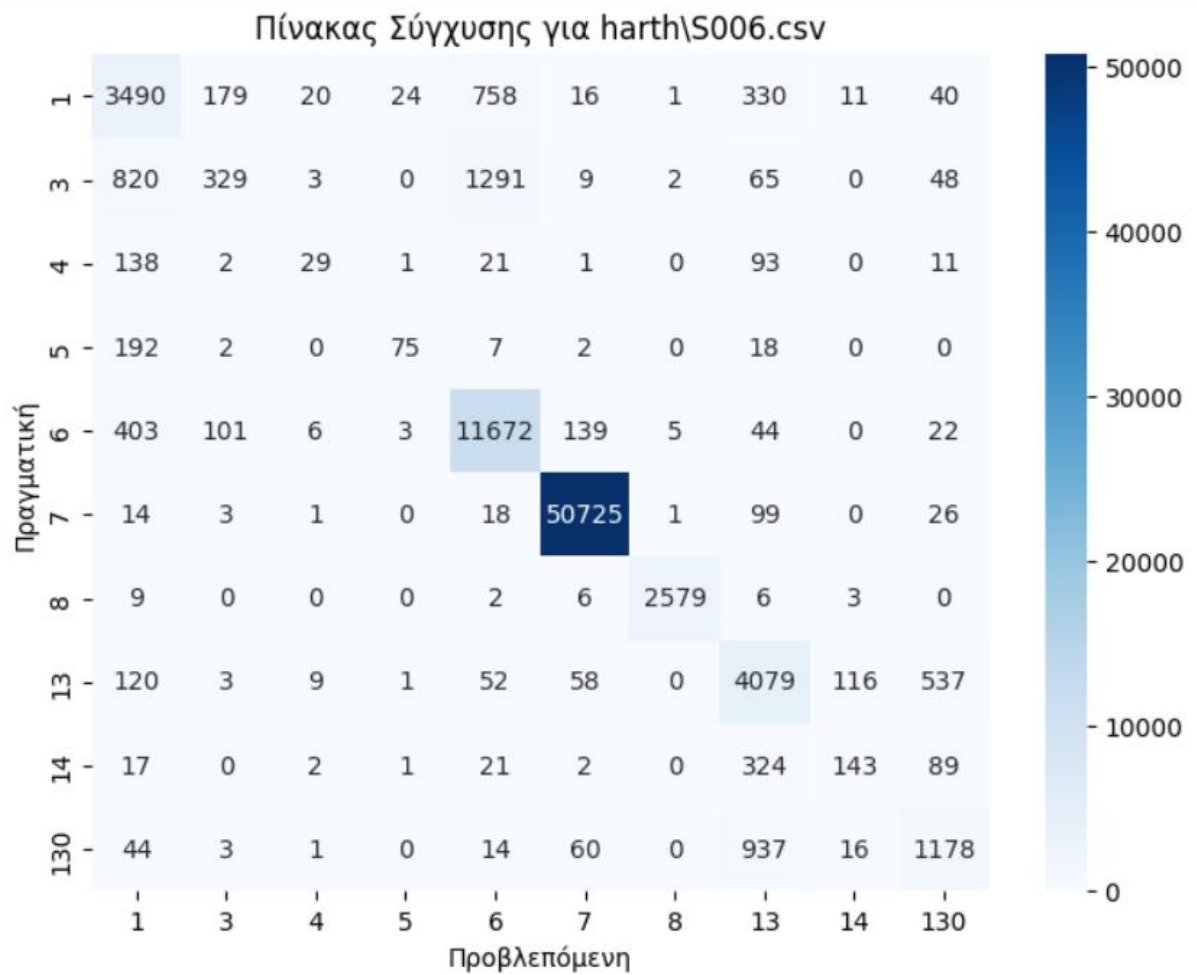
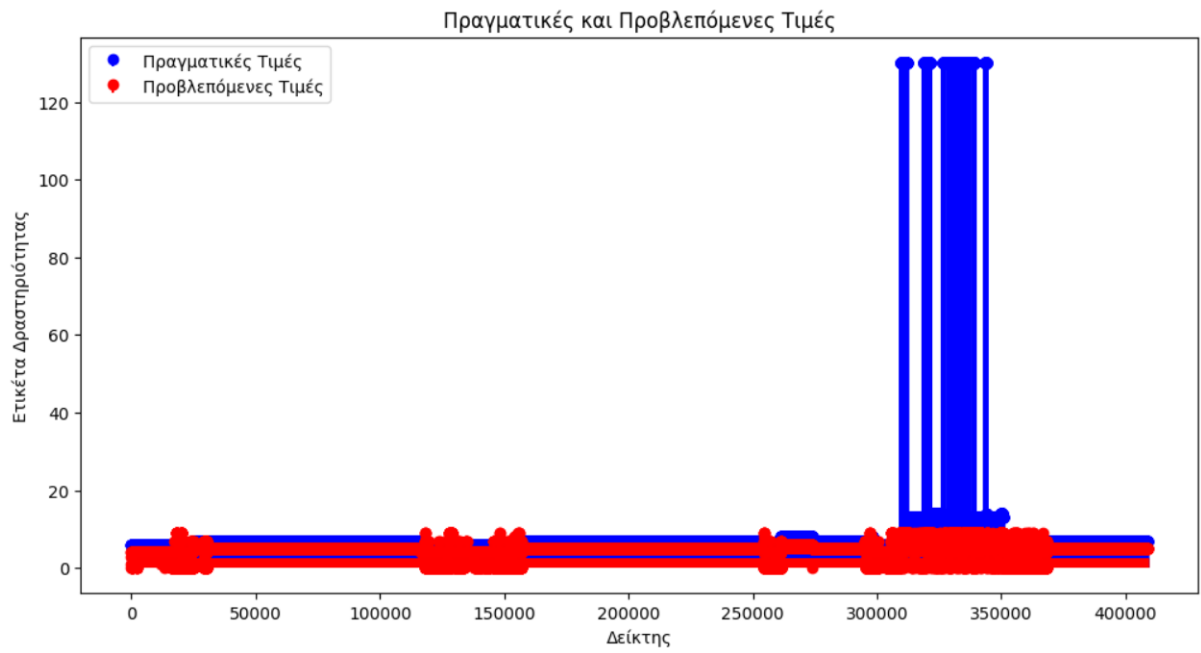
### Για τον ταξινομητή σε Νευρωνικό Δίκτυο

```
Επεξεργασία αρχείου: harth\S006.csv
Epoch 1/5
8175/8175 ————— 5s 586us/step - accuracy: 0.8674 - loss: 0.4292 - val_accuracy: 0.8946 - val_loss: 0.2960
Epoch 2/5
8175/8175 ————— 5s 557us/step - accuracy: 0.8994 - loss: 0.2823 - val_accuracy: 0.9010 - val_loss: 0.2788
Epoch 3/5
8175/8175 ————— 5s 555us/step - accuracy: 0.9030 - loss: 0.2680 - val_accuracy: 0.9049 - val_loss: 0.2686
Epoch 4/5
8175/8175 ————— 4s 547us/step - accuracy: 0.9039 - loss: 0.2637 - val_accuracy: 0.9036 - val_loss: 0.2686
Epoch 5/5
8175/8175 ————— 4s 539us/step - accuracy: 0.9060 - loss: 0.2576 - val_accuracy: 0.9081 - val_loss: 0.2567
2555/2555 ————— 1s 471us/step

Αναφορά Ταξινόμησης:
      precision    recall  f1-score   support

0         0.67       0.72       0.69       4869
1         0.53       0.13       0.21       2567
2         0.41       0.10       0.16        296
3         0.71       0.25       0.37        296
4         0.84       0.94       0.89      12395
5         0.99       1.00       1.00     50887
6         1.00       0.99       0.99       2605
7         0.68       0.82       0.74       4975
8         0.49       0.24       0.32        599

...
Precision: 0.9004232643588014
Recall: 0.9089452178806489
F1-Score: 0.8988238423932591
Accuracy: 0.9089452178806489
```





```

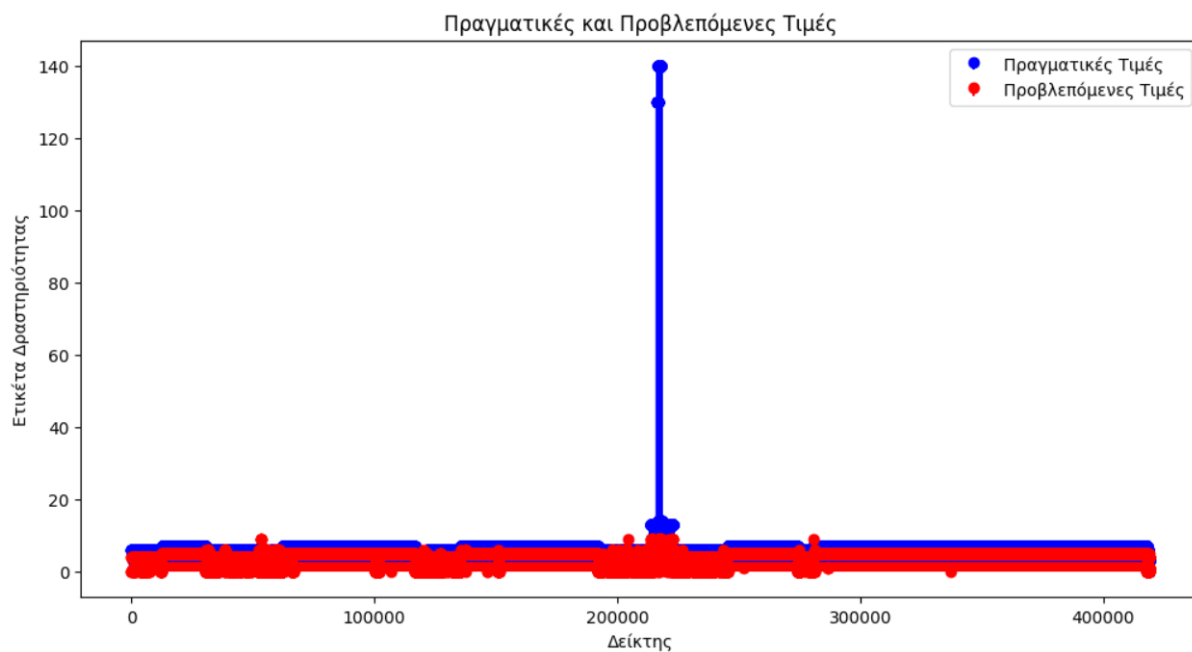
Επεξεργασία αρχείου: harth\S008.csv
Epoch 1/5
8380/8380 ————— 5s 576us/step - accuracy: 0.8908 - loss: 0.3199 - val_accuracy: 0.9210 - val_loss: 0.1985
Epoch 2/5
8380/8380 ————— 5s 549us/step - accuracy: 0.9253 - loss: 0.1911 - val_accuracy: 0.9243 - val_loss: 0.1894
Epoch 3/5
8380/8380 ————— 5s 554us/step - accuracy: 0.9273 - loss: 0.1848 - val_accuracy: 0.9261 - val_loss: 0.1855
Epoch 4/5
8380/8380 ————— 5s 543us/step - accuracy: 0.9297 - loss: 0.1789 - val_accuracy: 0.9258 - val_loss: 0.1900
Epoch 5/5
8380/8380 ————— 5s 535us/step - accuracy: 0.9301 - loss: 0.1774 - val_accuracy: 0.9287 - val_loss: 0.1791
2619/2619 ————— 1s 450us/step

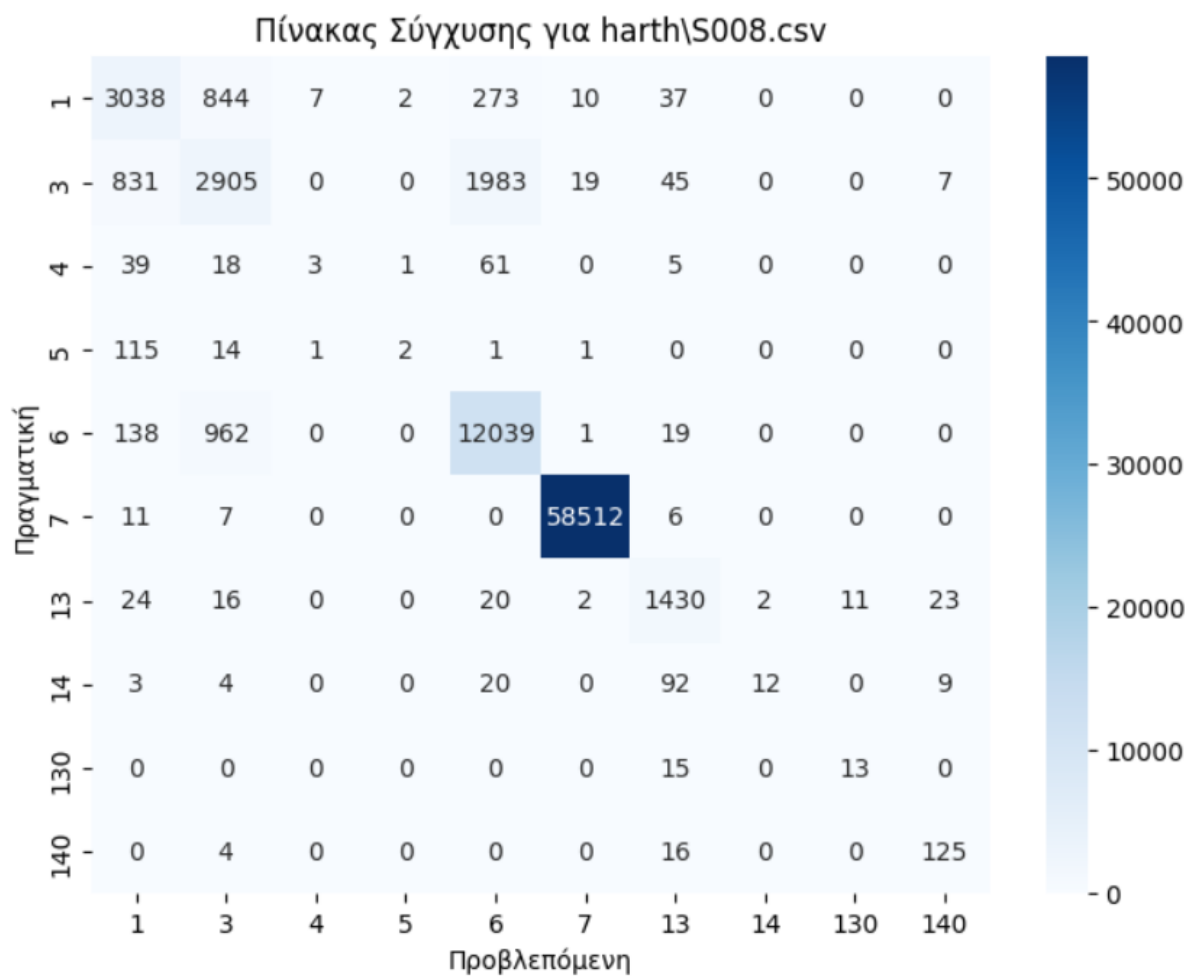
Αναφορά Ταξινόμησης:
      precision    recall  f1-score   support

0         0.72       0.72       0.72      4211
1         0.61       0.50       0.55      5790
2         0.27       0.02       0.04        127
3         0.40       0.01       0.03        134
4         0.84       0.91       0.87     13159
5         1.00       1.00       1.00    58536
6         0.86       0.94       0.90     1528
7         0.86       0.09       0.16        140
8         0.54       0.46       0.50         28

...
Precision: 0.9275036398406268
Recall: 0.9317525477935035
F1-Score: 0.9279876464271168
Accuracy: 0.9317525477935035

```



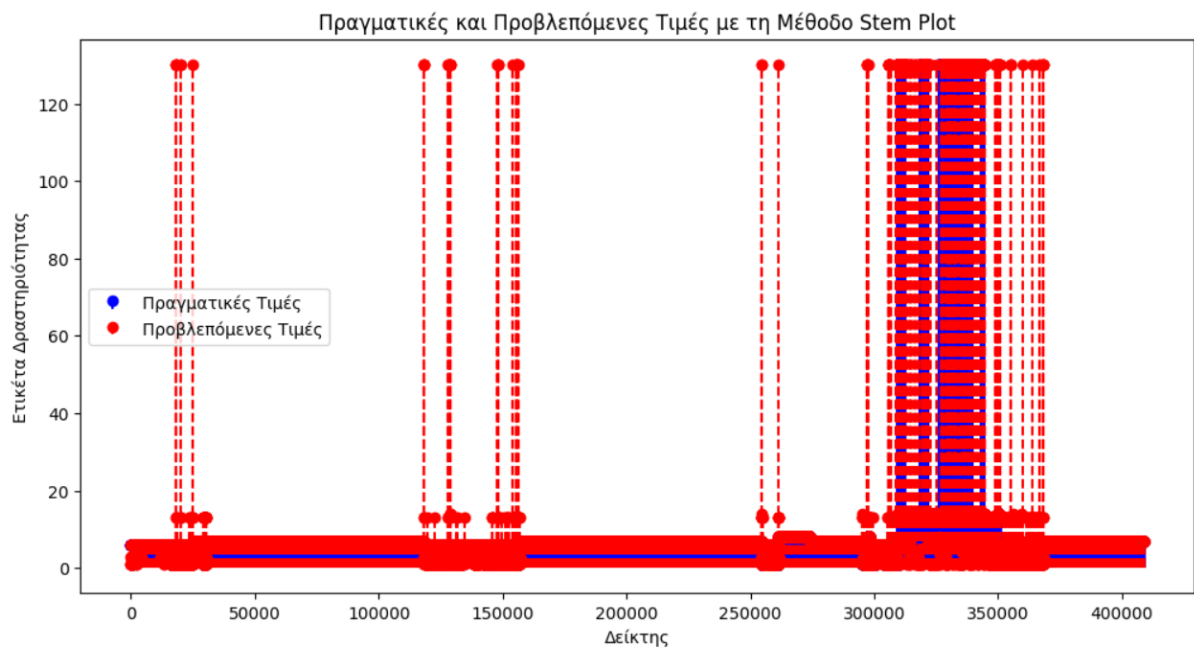


Για τον ταξινομητή σε Random Forest

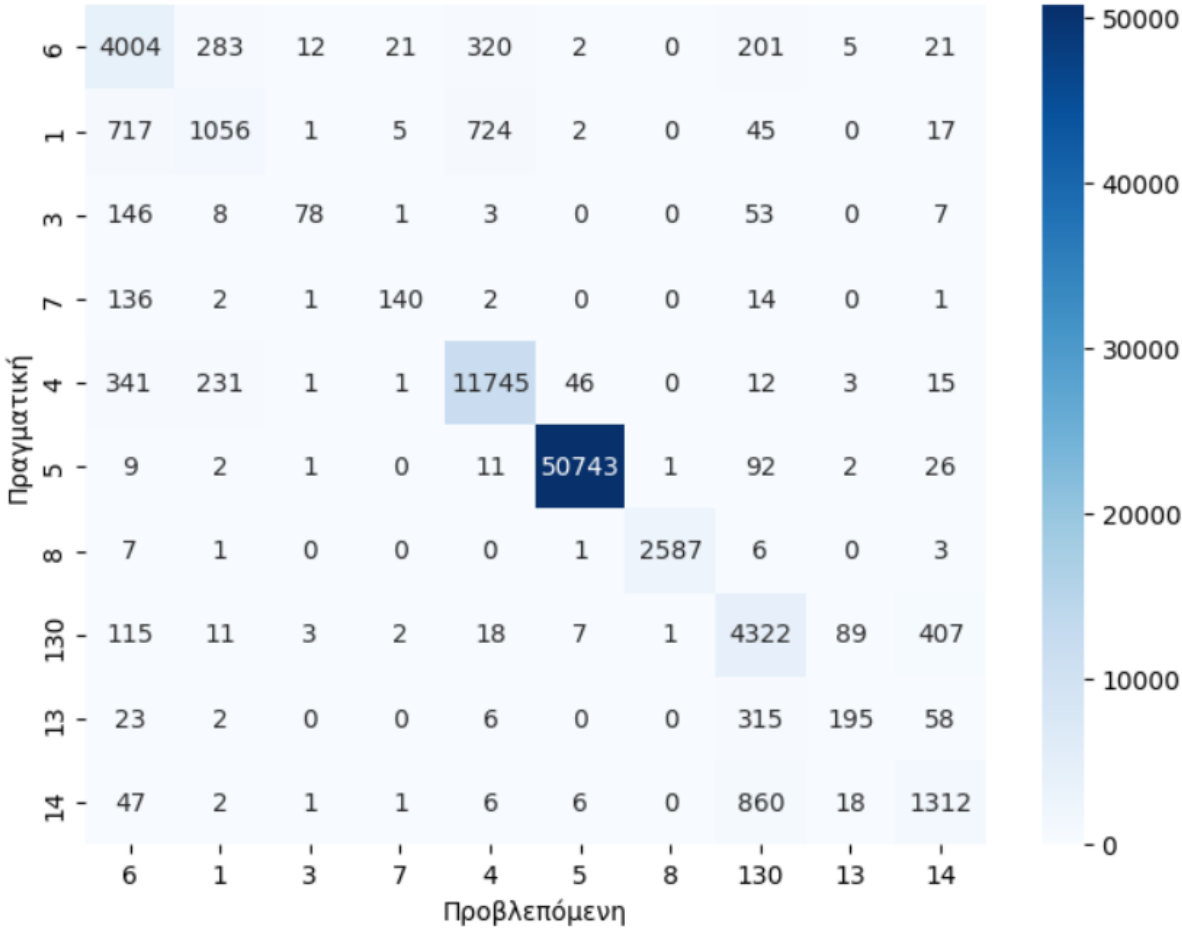
Επεξεργασία αρχείου: harth\S006.csv  
Ακρίβεια του μοντέλου: 0.9319811113014118  
Precision: 0.9303431073180567  
Recall: 0.9319811113014118  
F1-Score: 0.9284942614928385

#### Αναφορά Ταξινόμησης:

	precision	recall	f1-score	support
1	0.72	0.82	0.77	4869
3	0.66	0.41	0.51	2567
4	0.80	0.26	0.40	296
5	0.82	0.47	0.60	296
6	0.92	0.95	0.93	12395
7	1.00	1.00	1.00	50887
8	1.00	0.99	1.00	2605
13	0.73	0.87	0.79	4975
14	0.62	0.33	0.43	599
130	0.70	0.58	0.64	2253
accuracy			0.93	81742
macro avg	0.80	0.67	0.71	81742
weighted avg	0.93	0.93	0.93	81742



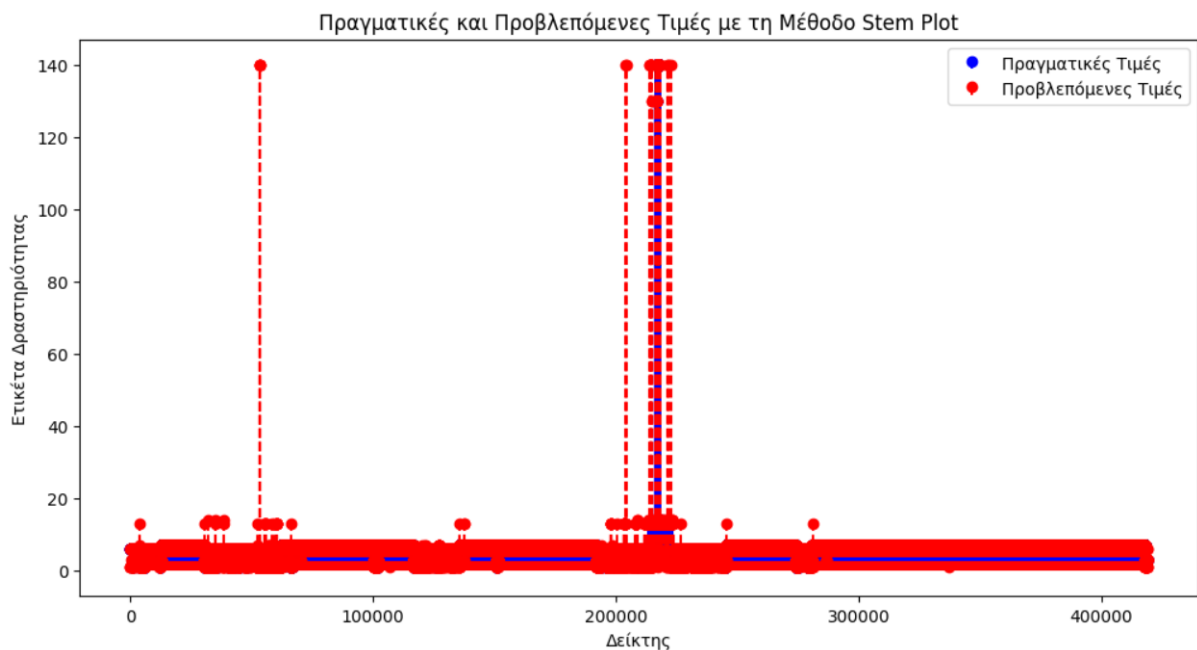
Πίνακας Σύγκρισης για harth\S006.csv

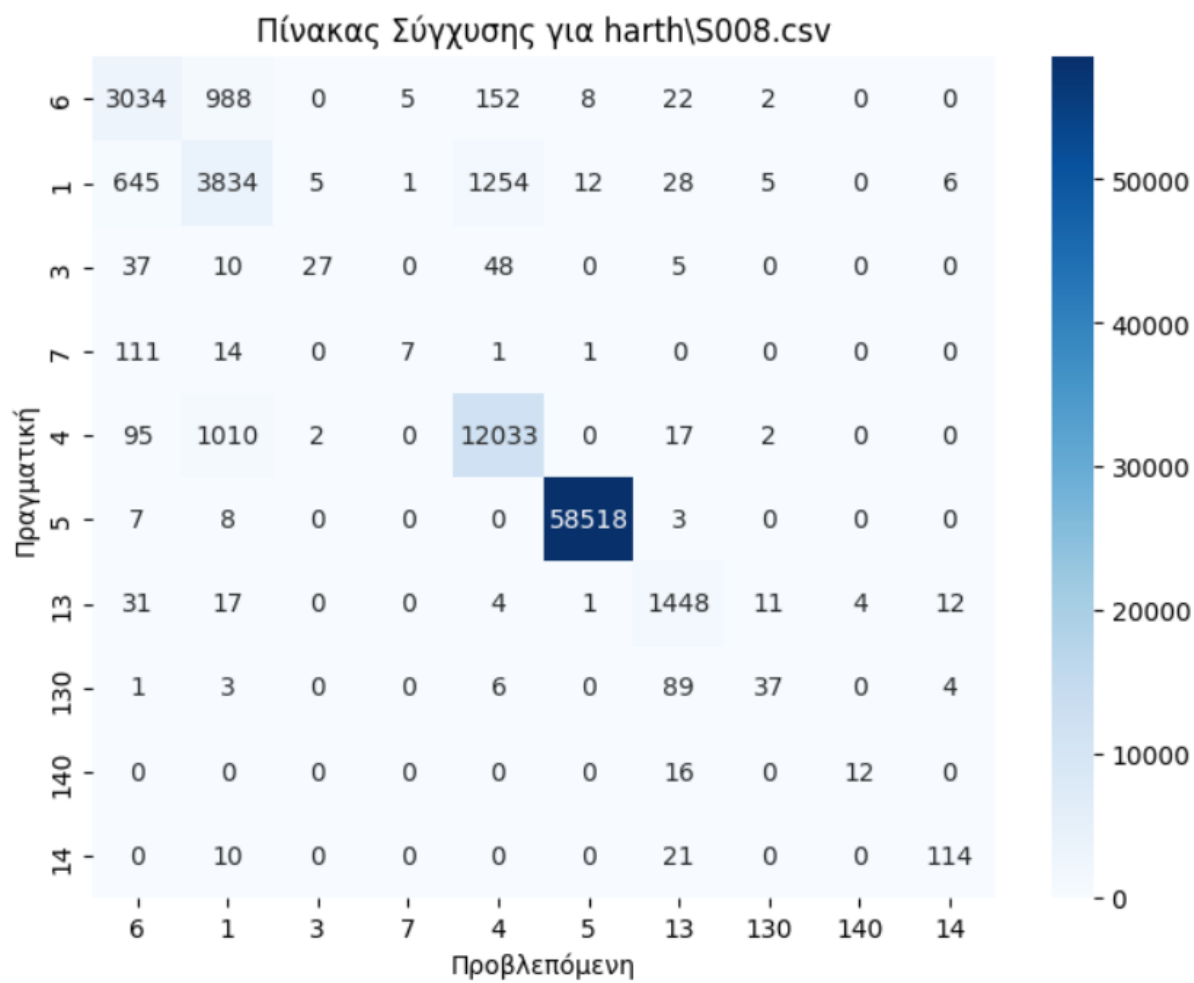


Επεξεργασία αρχείου: harth\S008.csv  
Ακρίβεια του μοντέλου: 0.943507004940452  
Precision: 0.9425620258880093  
Recall: 0.943507004940452  
F1-Score: 0.9422228223461365

#### Αναφορά Ταξινόμησης:

	precision	recall	f1-score	support
1	0.77	0.72	0.74	4211
3	0.65	0.66	0.66	5790
4	0.79	0.21	0.34	127
5	0.54	0.05	0.10	134
6	0.89	0.91	0.90	13159
7	1.00	1.00	1.00	58536
13	0.88	0.95	0.91	1528
14	0.65	0.26	0.38	140
130	0.75	0.43	0.55	28
140	0.84	0.79	0.81	145
accuracy			0.94	83798
macro avg	0.78	0.60	0.64	83798
weighted avg	0.94	0.94	0.94	83798





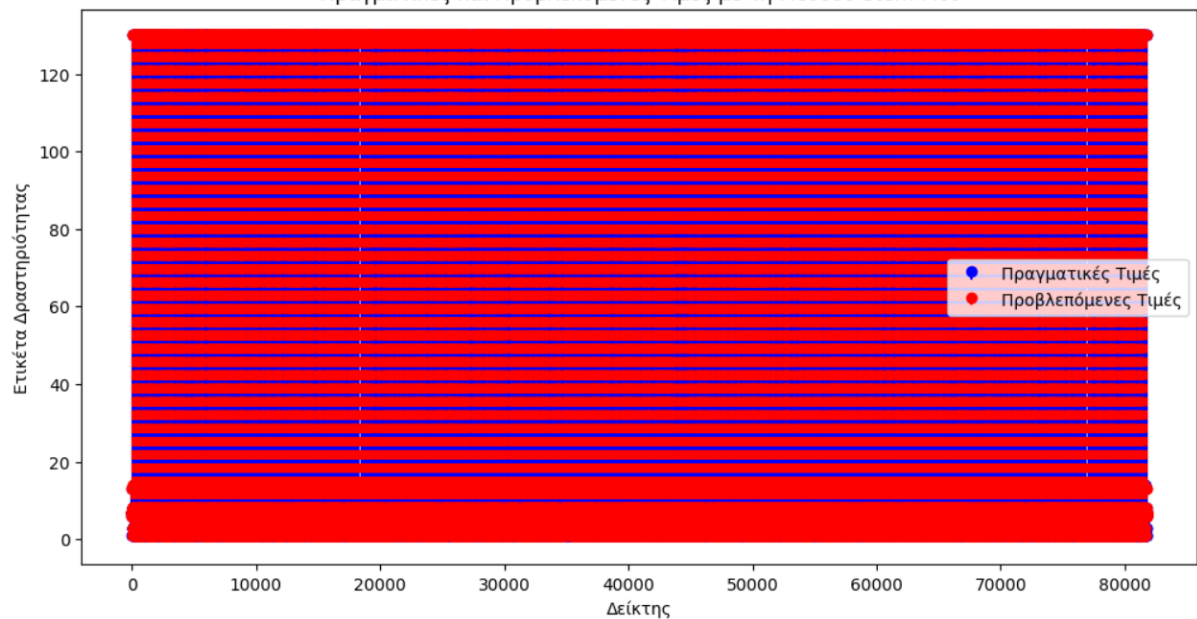
Για τον ταξινομητή σε Bayesian Networks

Επεξεργασία αρχείου: harth\S006.csv  
Ακρίβεια του μοντέλου: 0.8720975752978884  
Precision: 0.8625345783529715  
Recall: 0.8720975752978884  
F1-Score: 0.8627604905779106

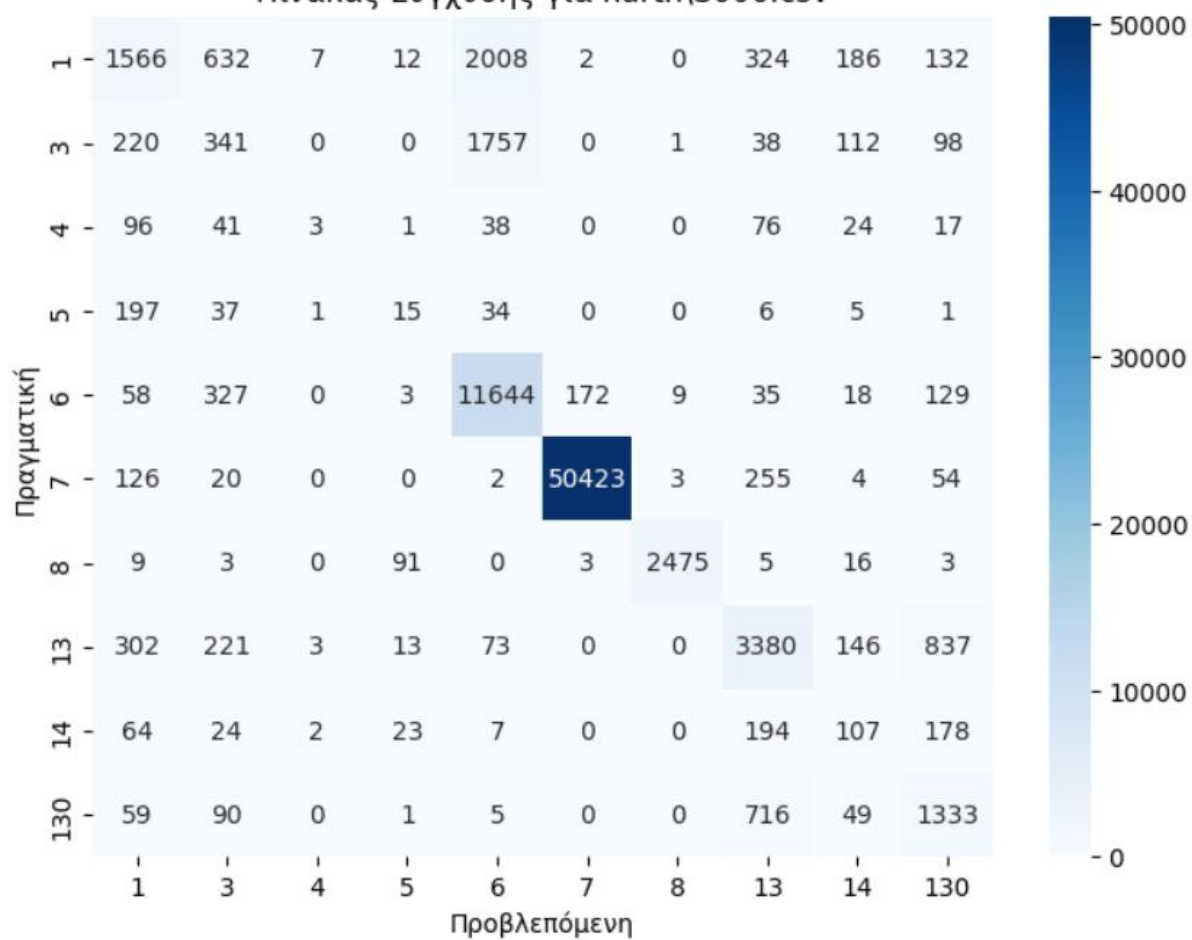
#### Αναφορά Ταξινόμησης:

	precision	recall	f1-score	support
1	0.58	0.32	0.41	4869
3	0.20	0.13	0.16	2567
4	0.19	0.01	0.02	296
5	0.09	0.05	0.07	296
6	0.75	0.94	0.83	12395
7	1.00	0.99	0.99	50887
8	0.99	0.95	0.97	2605
13	0.67	0.68	0.68	4975
14	0.16	0.18	0.17	599
130	0.48	0.59	0.53	2253
accuracy			0.87	81742
macro avg	0.51	0.48	0.48	81742
weighted avg	0.86	0.87	0.86	81742

Πραγματικές και Προβλεπόμενες Τιμές με τη Μέθοδο Stem Plot



Πίνακας Σύγκρισης για harth\S006.csv

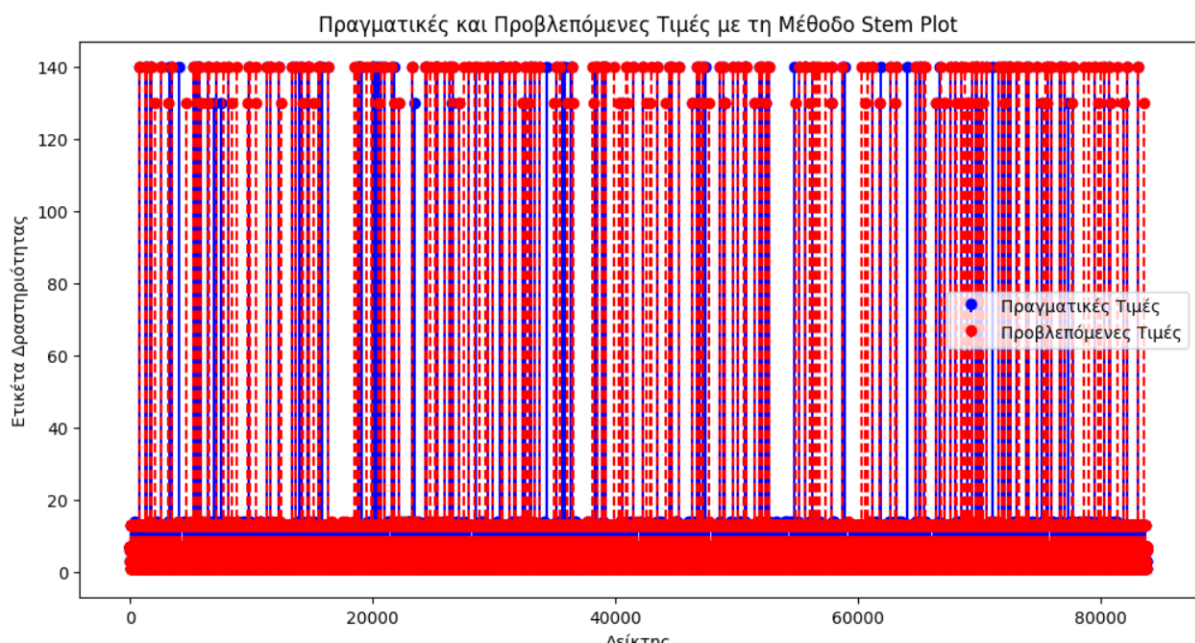


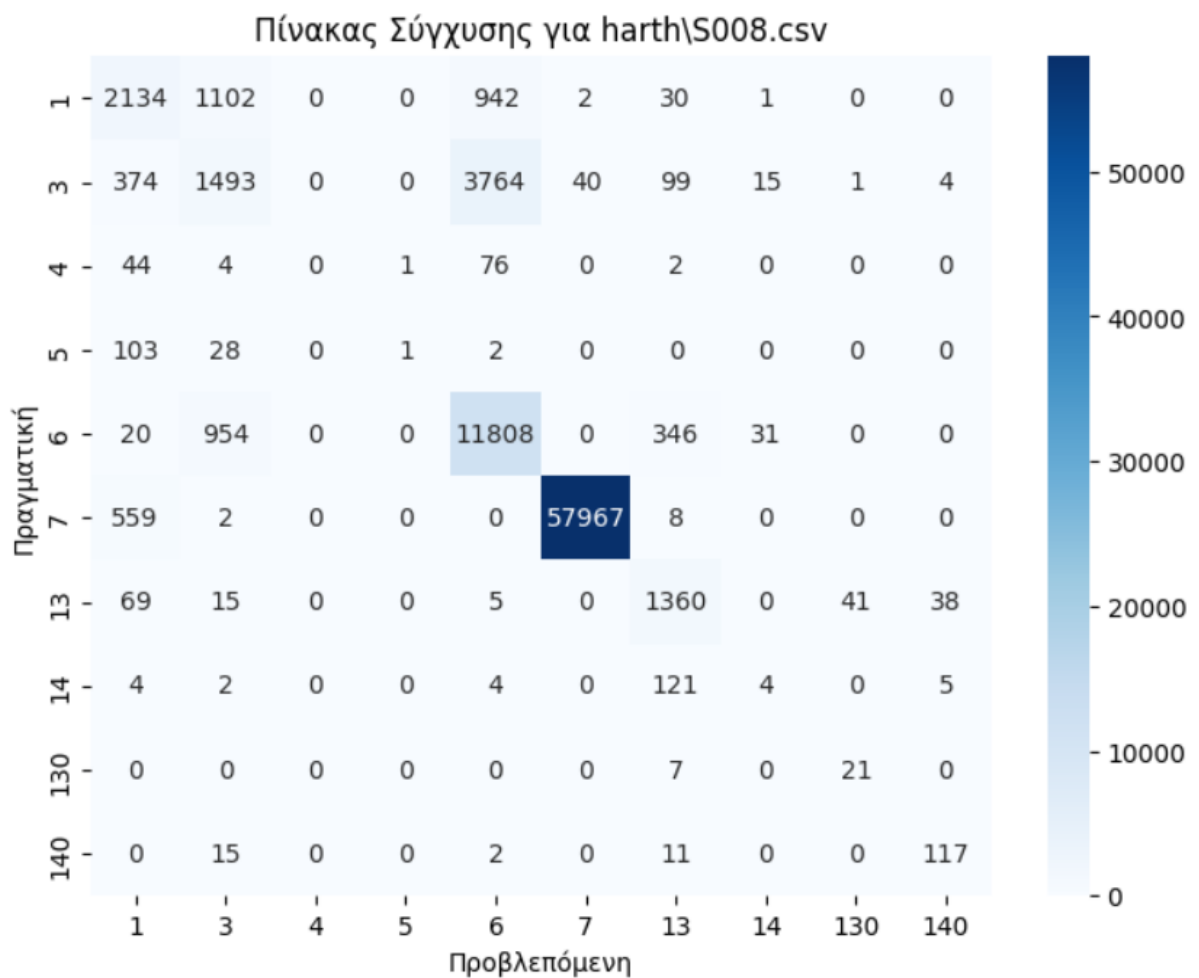


Επεξεργασία αρχείου: harth\S008.csv  
Ακρίβεια του μοντέλου: 0.8938757488245543  
Precision: 0.8854514693264818  
Recall: 0.8938757488245543  
F1-Score: 0.8856246416257915

Αναφορά Ταξινόμησης:

	precision	recall	f1-score	support
1	0.65	0.51	0.57	4211
3	0.41	0.26	0.32	5790
4	0.00	0.00	0.00	127
5	0.50	0.01	0.01	134
6	0.71	0.90	0.79	13159
7	1.00	0.99	0.99	58536
13	0.69	0.89	0.77	1528
14	0.08	0.03	0.04	140
130	0.33	0.75	0.46	28
140	0.71	0.81	0.76	145
accuracy			0.89	83798
macro avg	0.51	0.51	0.47	83798
weighted avg	0.89	0.89	0.89	83798





Παρατηρούμε ότι όσον αφορά το αρχείο csv με τον συμμετέχοντα S006 ο ταξινομητής Random Forest παρείχε την καλύτερη ακρίβεια από όλους τους ταξινομητές και για όλες τις μετρικές που χρησιμοποιήθηκαν. Στην δεύτερη θέση έρχεται ο ταξινομητής σε Νευρωνικό Δίκτυο ενώ τρίτος στην σωστή πρόβλεψη τιμών έρχεται ο ταξινομητής σε Bayesian Networks.

Όσον αφορά το αρχείο csv του συμμετέχοντα S008 παρατηρούμε και πάλι ότι ο ταξινομητής Random Forest πέτυχε το υψηλότερο σκορ για όλες τις μετρικές. Στην δεύτερη θέση έρχεται και πάλι ο ταξινομητής σε Neural Networks και τελευταίος στην πρόβλεψη έρχεται και πάλι ο ταξινομητής σε Bayesian Networks.

Παρακάτω δίνονται οι μέσοι όροι των μετρικών για κάθε ταξινομητή:

Για τον ταξινομητή σε Νευρωνικό Δίκτυο

```
Μέσοι Όροι και Τυπικές Αποκλίσεις Μετρικών:  
Mean Precision: 0.9085823001946466 (±0.04117574041651999)  
Mean Recall: 0.9161867005289057 (±0.03816923398331726)  
Mean F1-Score: 0.9077745957666769 (±0.04203332479517551)  
Mean Accuracy: 0.9161867005289057 (±0.03816923398331726)
```

Για τον ταξινομητή σε Random Forest

```
Μέσοι Όροι και Τυπικές Αποκλίσεις Μετρικών:  
Mean Precision: 0.9343491128254476 (±0.03751509897230887)  
Mean Recall: 0.9366920715132898 (±0.03527474574624393)  
Mean F1-Score: 0.932867813020075 (±0.03786140840472489)  
Mean Accuracy: 0.9366920715132898 (±0.03527474574624393)
```

Για τον ταξινομητή σε Bayesian Networks

```
Μέσος Όρος Ακρίβειας: 0.8121290790509256 (±0.10728139628550333)  
Μέσος Όρος Precision: 0.7961126794434996 (±0.137699030990149)  
Μέσος Όρος Recall: 0.8121290790509256 (±0.10728139628550333)  
Μέσος Όρος F1-Score: 0.7889236790864057 (±0.13978042345829575)
```

Παρατηρούμε από τους μέσους όρους που παρατίθενται για κάθε μετρική ότι ο ταξινομητής Random Forest φαίνεται να ταξινομεί και να προβλέπει καλύτερα όλα τα αρχεία csv σε αντίθεση με τους άλλους δύο ταξινομητές. Στην δεύτερη θέση έρχεται ο ταξινομητής σε Νευρωνικό Δίκτυο με ελάχιστα μικρότερο σκορ από τον ταξινομητή Random Forest για όλες τις μετρικές. Τέλος με αρκετά μικρότερο σκορ σε όλες τις μετρικές έρχεται ο ταξινομητής σε Bayesian Networks. Επιπλέον από τις τυπικές αποκλίσεις παρατηρούμε και πάλι ότι ο ταξινομητής

Random Forest έχει την μικρότερη τυπική απόκλιση που σημαίνει ότι οι τιμές δεν μεταβάλλονται πολύ από τον μέσο όρο των μετρικών. Έπειτα στην δεύτερη θέση έρχεται και πάλι ο ταξινομητής σε Νευρωνικό δίκτυο με παρόμοια τυπική απόκλιση και παρατηρούμε πως ο ταξινομητής σε Bayesian Networks παρουσιάζει την μεγαλύτερη τυπική απόκλιση που σημαίνει ότι η ακρίβεια του δεν είναι όσο ισχυρή είναι των άλλων δύο ταξινομητών.

Επομένως αντιλαμβανόμαστε ότι ο Random Forest που χρησιμοποιεί πολλαπλά decision trees για να βελτιώσει την ακρίβεια και την ανθεκτικότητα των προβλέψεων έχει την καλύτερη ακρίβεια σε σχέση με όλα τα μοντέλα που υλοποιήθηκαν. Ωστόσο ο Gaussian Naïve Bayes παρόλο που είναι απλός και αποτελεσματικός στην υλοποίηση, επειδή υποθέτει ότι οι τιμές των χαρακτηριστικών ακολουθούν την κανονική κατανομή ίσως φέρει την λιγότερη ακρίβεια από όλους τους ταξινομητές.

### **Ερώτημα 3:**

Για την συσταδοποίηση χρησιμοποιήθηκαν δύο αλγόριθμοι συσταδοποίησης. Ο πρώτος αλγόριθμος συσταδοποίησης που χρησιμοποιήθηκε είναι ο K- Means και ο δεύτερος αλγόριθμος είναι ένας ιεραρχικός αλγόριθμος συσταδοποίησης ο Agglomerative Clustering. Στον κώδικα που υλοποιήθηκε για το συγκεκριμένο ερώτημα αρχικά συνενώσαμε όλα τα αρχεία csv σε ένα ενιαίο. Προστέθηκε μία στήλη participant\_id η οποία περιέχει τα id του κάθε συμμετέχοντα. Στην συνέχεια χρησιμοποιούμε σαν είσοδο για την ομαδοποίηση από τα

δεδομένα back και thigh τις μέσες τιμές, τις τυπικές αποκλίσεις, τις μέγιστες και ελάχιστες τιμές ώστε να κάνουμε ομαδοποίηση των συμμετεχόντων ανάλογα με την δραστηριότητα τους. Στην συνέχεια έγινε μια πρόβλεψη και για τους δύο αλγορίθμους στην οποία υπολογίστηκε για ένα εύρος clusters (από 2 έως 10) το Silhouette Score και ο αριθμός των clusters που μας δίνει το μεγαλύτερο σκορ θα χρησιμοποιηθεί για την ομαδοποίηση των συμμετεχόντων.

Παρακάτω τίθενται κομμάτια του κώδικα που υλοποιούν τα παραπάνω.

```
# Συγκέντρωση όλων των δεδομένων σε ένα DataFrame
dataframes = []
for csv_file in csv_files:
    print(f"Επεξεργασία αρχείου: {csv_file}")
    df = pd.read_csv(csv_file)
    df["timestamp"] = pd.to_datetime(df["timestamp"])
    df["participant_id"] = os.path.basename(csv_file).split('.')[0] # Δημιουργούμε μία στήλη με τα ονόματα των αρχείων csv χωρίς την κατάληξη .csv
    dataframes.append(df)
```

```
# Διαχωρισμός χαρακτηριστικών και στόχου
X = data.drop(["timestamp", "label"], axis=1, errors='ignore')
participant_ids = data["participant_id"]
```

```
# Ομαδοποίηση δεδομένων ανά συμμετέχοντα και δραστηριότητα με βάση την μέση τιμή, την τυπική απόκλιση, την μέγιστη και την ελάχιστη τιμή
grouped = X.groupby("participant_id").agg(['mean', 'std', 'max', 'min']).reset_index()
```

```
# Βρίσκουμε ποιά Silhouette Score είναι το καλύτερο για τον αλγόριθμο συσταδοποίησης K-Means
for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters_kmeans = kmeans.fit_predict(X_pca)
    silhouette_avg = silhouette_score(X_pca, clusters_kmeans)
    print(f"K-means - Clusters: {n_clusters}, Silhouette Score: {silhouette_avg}")
    if silhouette_avg > best_silhouette_kmeans:
        best_silhouette_kmeans = silhouette_avg
        best_clusters_kmeans = clusters_kmeans
    optimal_clusters = n_clusters

# Προσθήκη των clusters στο αρχικό DataFrame για τον K-means
grouped["cluster_kmeans"] = best_clusters_kmeans
```

```
# Βρίσκουμε ποιά Silhouette Score είναι το καλύτερο για τον αλγόριθμο συσταδοποίησης Agglomerative Clustering
for n_clusters in range(2, 11):
    hierarchical_clustering = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward')
    clusters_hierarchical = hierarchical_clustering.fit_predict(X_pca)
    silhouette_avg = silhouette_score(X_pca, clusters_hierarchical)
    print(f"Hierarchical Clustering - Clusters: {n_clusters}, Silhouette Score: {silhouette_avg}")
    if silhouette_avg > best_silhouette_hierarchical:
        best_silhouette_hierarchical = silhouette_avg
        best_clusters_hierarchical = clusters_hierarchical

# Προσθήκη των clusters στο αρχικό DataFrame για το Hierarchical Clustering
grouped["cluster_hierarchical"] = best_clusters_hierarchical
```

```
# Απεικόνιση των clusters για τον K-means
pca_df["cluster"] = best_clusters_kmeans
pca_df["participant_id"] = grouped["participant_id"].values

plt.figure(figsize=(10, 6))
sns.scatterplot(data=pca_df, x="PC1", y="PC2", hue="cluster", palette="Set1")
plt.title(f"K-means Clustering of Participants (Optimal Clusters: {optimal_clusters})")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.show()

# Απεικόνιση των clusters για το Agglomerative Clustering
pca_df["cluster"] = best_clusters_hierarchical

plt.figure(figsize=(10, 6))
sns.scatterplot(data=pca_df, x="PC1", y="PC2", hue="cluster", palette="Set1")
plt.title(f"Hierarchical Clustering of Participants (Optimal Clusters: {optimal_clusters})")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.show()
```

```
# Υπολογισμός των μετρικών αξιολόγησης για τον K-means
silhouette_avg_kmeans = silhouette_score(X_pca, best_clusters_kmeans)
print(f"K-means Best Silhouette Score: {silhouette_avg_kmeans}")

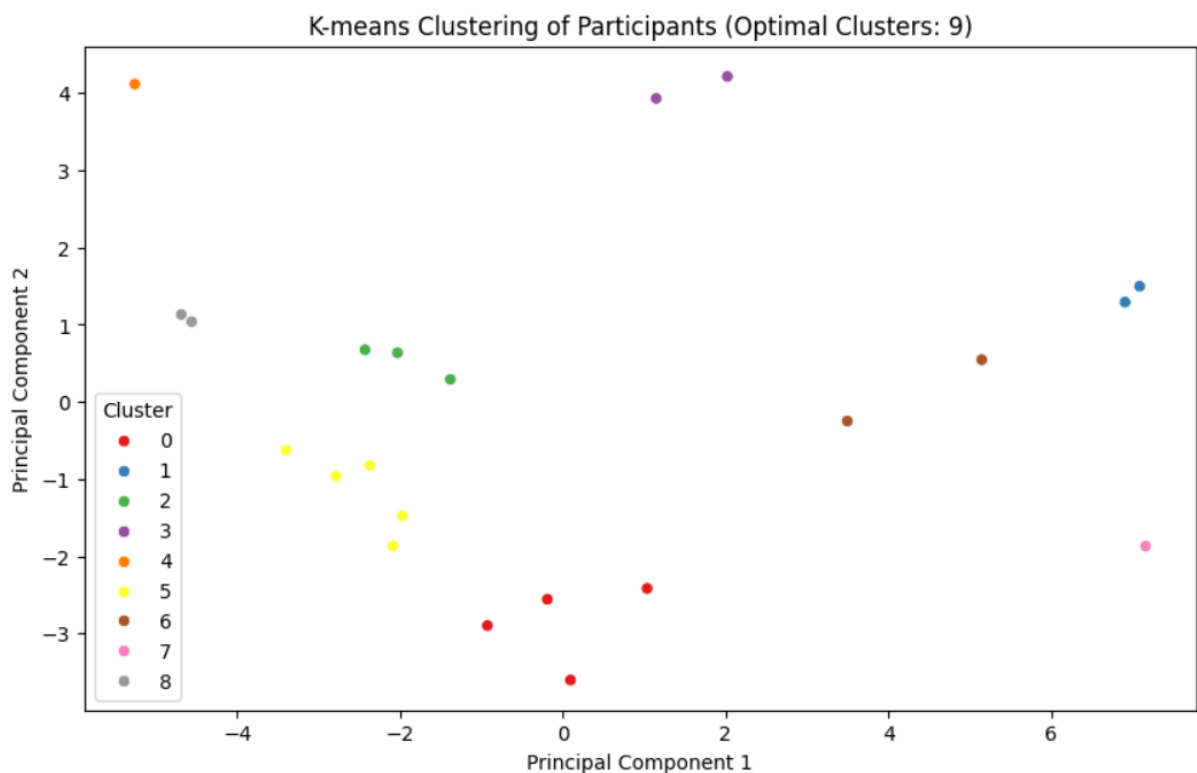
# Υπολογισμός των μετρικών αξιολόγησης για το Hierarchical Clustering
silhouette_avg_hierarchical = silhouette_score(X_pca, best_clusters_hierarchical)
print(f"Hierarchical Clustering Best Silhouette Score: {silhouette_avg_hierarchical}")
```

```
# Ανάλυση των clusters
for cluster in pca_df["cluster"].unique():
    print(f"\nCluster {cluster}:")
    participants_in_cluster = pca_df[pca_df["cluster"] == cluster]["participant_id"].unique()
    print("Participants in this cluster:")
    print(participants_in_cluster)
```

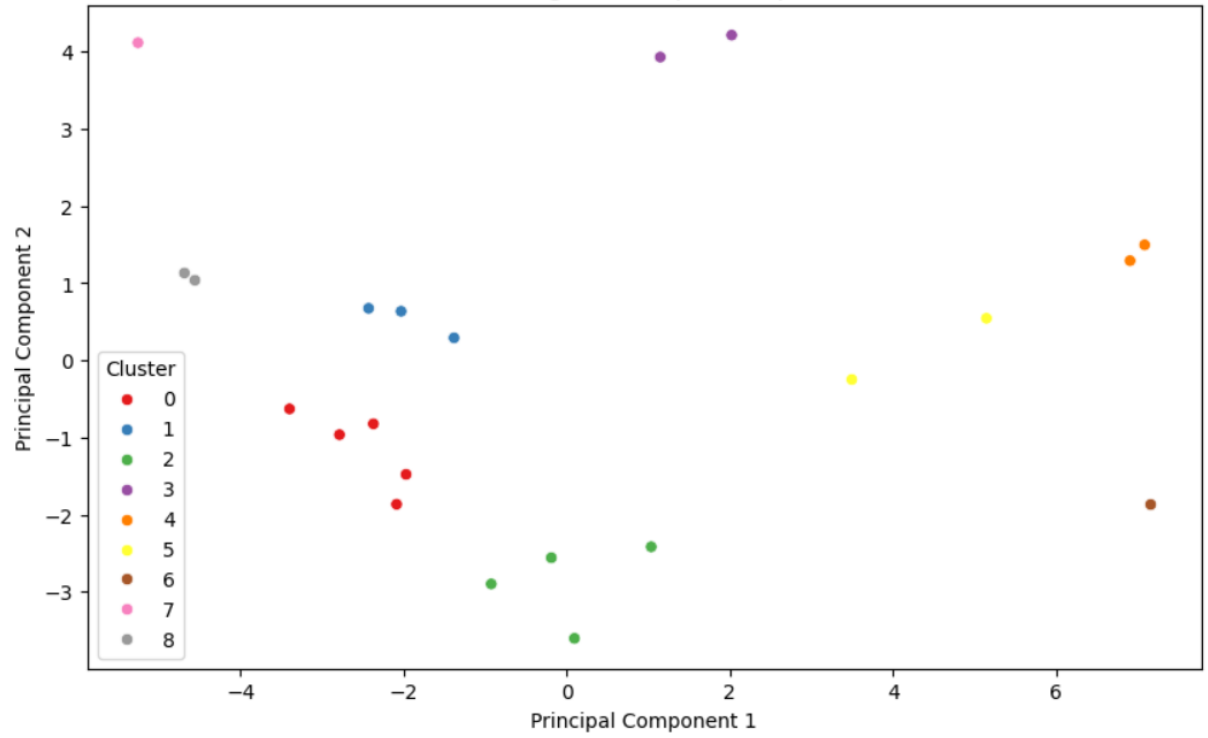
## Τελικά Αποτελέσματα

Έπειτα από την εκτέλεση του κώδικα λαμβάνουμε τα εξής αποτελέσματα:

```
K-means - Clusters: 2, Silhouette Score: 0.5413182604908673
K-means - Clusters: 3, Silhouette Score: 0.3664348416047517
K-means - Clusters: 4, Silhouette Score: 0.47845795052904716
K-means - Clusters: 5, Silhouette Score: 0.4392170464415941
K-means - Clusters: 6, Silhouette Score: 0.41715506759212584
K-means - Clusters: 7, Silhouette Score: 0.5084911484334717
K-means - Clusters: 8, Silhouette Score: 0.42298954361897717
K-means - Clusters: 9, Silhouette Score: 0.5644952704011358
K-means - Clusters: 10, Silhouette Score: 0.5291030494024854
Hierarchical Clustering - Clusters: 2, Silhouette Score: 0.5413182604908673
Hierarchical Clustering - Clusters: 3, Silhouette Score: 0.42605623640191226
Hierarchical Clustering - Clusters: 4, Silhouette Score: 0.5013093824219459
Hierarchical Clustering - Clusters: 5, Silhouette Score: 0.547279389577562
Hierarchical Clustering - Clusters: 6, Silhouette Score: 0.5053196725533857
Hierarchical Clustering - Clusters: 7, Silhouette Score: 0.5410414675000618
Hierarchical Clustering - Clusters: 8, Silhouette Score: 0.5352680552079665
Hierarchical Clustering - Clusters: 9, Silhouette Score: 0.5644952704011358
Hierarchical Clustering - Clusters: 10, Silhouette Score: 0.5384536609310132
```



Hierarchical Clustering of Participants (Optimal Clusters: 9)





```
K-means Best Silhouette Score: 0.5644952704011358  
Hierarchical Clustering Best Silhouette Score: 0.5644952704011358
```

```
Cluster 0:  
Participants in this cluster:  
['s006' 's014' 's015' 's019' 's021']
```

```
Cluster 8:  
Participants in this cluster:  
['s008' 's012']
```

```
Cluster 1:  
Participants in this cluster:  
['s009' 's013' 's022']
```

```
Cluster 7:  
Participants in this cluster:  
['s010']
```

```
Cluster 2:  
Participants in this cluster:  
['s016' 's017' 's018' 's020']
```

```
Cluster 5:  
Participants in this cluster:  
['s023' 's025']
```

```
Cluster 6:  
Participants in this cluster:  
['s024']
```

```
Cluster 3:  
Participants in this cluster:  
['s026' 's028']
```

```
Cluster 4:  
Participants in this cluster:  
['s027' 's029']
```

Από την παρατήρηση των αποτελεσμάτων αντιλαμβανόμαστε ότι ο Agglomerative Clustering Αλγόριθμος παρουσιάζει έστω

και ελάχιστα μεγαλύτερο σκορ σε όλα τα clusters που προβλέπονται σε αντίθεση με τον αλγόριθμο K-Means. Ο βασικός στόχος του Agglomerative Clustering που είναι ένας bottom up αλγόριθμος είναι να συνδέσει τα δεδομένα σε μία ιεραρχία συστάδων όπου κάθε δεδομένο αρχικά θεωρείται ως μια ξεχωριστή συστάδα και στην συνέχεια ενώνεται σταδιακά με άλλα δεδομένα έτσι ώστε να δημιουργήσει μεγαλύτερες συστάδες με βάση κάποια μέτρα ομοιότητας μεταξύ των δεδομένων. Ο αλγόριθμος συσταδοποίησης K-Means έχει ως σκοπό τα στοιχεία που βρίσκονται στην ίδια κλάση να είναι όσο το δυνατόν πιο παρόμοια μεταξύ τους και ταυτόχρονα όσο το δυνατόν πιο διαφορετικά από τα στοιχεία που βρίσκονται σε άλλες κλάσεις. Στόχος του είναι να ελαχιστοποιηθεί η αθροιστική εσωτερική διακύμανση, δηλαδή η απόσταση των σημείων από τα κέντρα των ομάδων τους. Παρατηρούμε και για τους δύο αλγορίθμους συσταδοποίησης ότι ο βέλτιστος αριθμός clusters είναι ο αριθμός 9. Επιπλέον παρατηρούμε ότι και οι δύο αλγόριθμοι δίνουν σαν αποτέλεσμα για την χρήση του βέλτιστου αριθμού clusters δηλαδή το 9, το ίδιο ακριβώς Silhouette Score. Επομένως και οι δύο αλγόριθμοι συσταδοποίησης είναι αποδοτικοί για 9 clusters.