2^{Η} ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Ακαδημαϊκό έτος 2024-25

Ομαδοποίηση δεδομένων με μείωση διάστασης

ΟΜΑΔΑ

ΧΡΙΣΤΟΔΟΥΛΟΥ ΧΡΗΣΤΟΣ: 5392

ΛΑΛΟΥΤΣΟΣ ΝΙΚΟΛΑΟΣ: 5266

1) ΜΕΙΩΣΗ ΔΙΑΣΤΑΣΕΩΝ

Pca

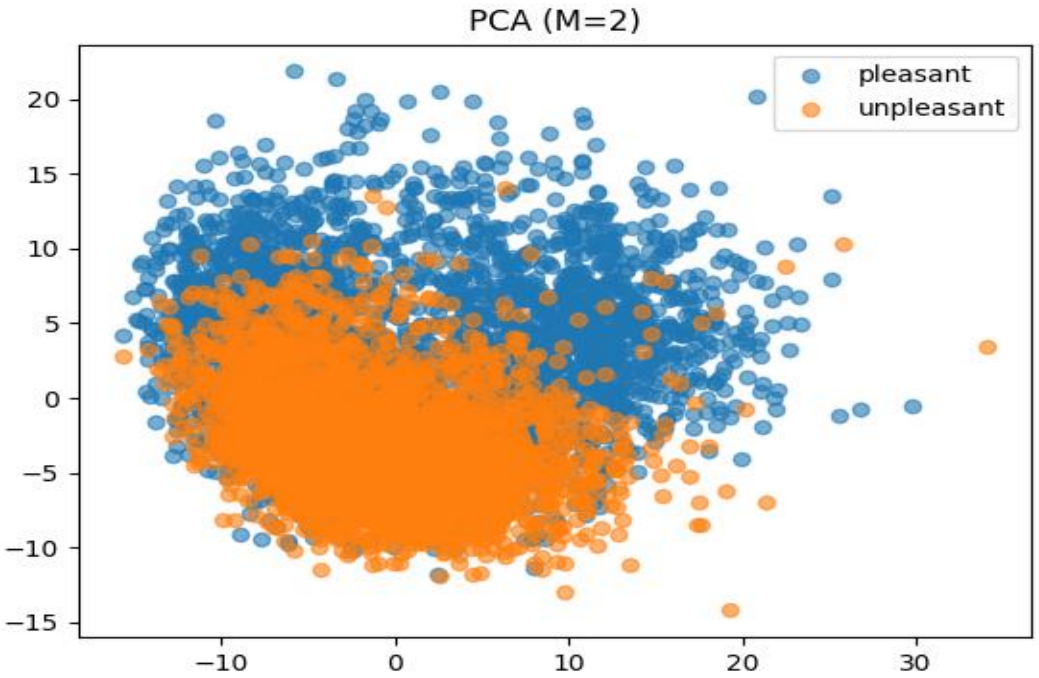
Το PCA είναι μια γραμμική μέθοδος μείωσης διαστάσεων που μετατρέπει το σύνολο των δεδομένων σε ένα νέο σύνολο αξόνων (called **κύριες συνιστώσες**) έτσι ώστε:

- Η πρώτη συνιστώσα να έχει τη μέγιστη διασπορά (variance).
- Η δεύτερη να είναι κάθετη στην πρώτη και να έχει τη μέγιστη δυνατή διασπορά, κ.ο.κ.

Υλοποιηση στο κωδικα μας

Μεσα στη μεθοδο **def apply_and_plot(x,y,method_name)** για διαφορες τιμες του m→[2,3,5,10,20]

- model = PCA(n_components=m)
- X_m = model.fit_transform(X)
- Κανουμε plot →m=2 ,m=3



Τι βλέπουμε: (PCA-2D)

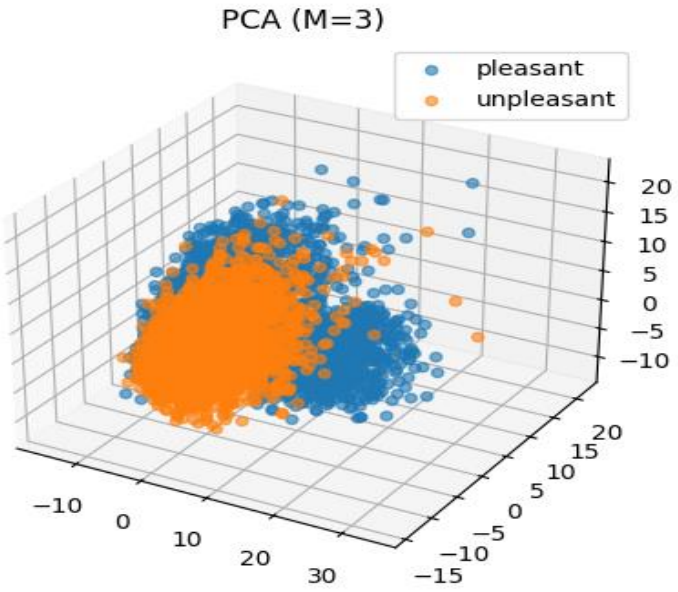
- Το PCA έχει συμπιέσει τα αρχικά δεδομένα σε 2 διαστάσεις.
- Η κάθε κουκκίδα είναι ένα δείγμα.
- Μπλε = pleasant, Πορτοκαλί = unpleasant.

Ερμηνεία:

- Υπάρχει μια **σχετική τάση διαχωρισμού**:
    - Τα πορτοκαλί σημεία (unpleasant) συγκεντρώνονται πιο **χαμηλά** στον κατακόρυφο άξονα (κάτω μέρος).
    - Τα μπλε σημεία (pleasant) εξαπλώνονται **ψηλότερα** και πιο ευρέως.
- Ωστόσο:
    - Υπάρχει σημαντική **επικάλυψη** στο κέντρο του γραφήματος.
    - Δηλαδή, σε αυτόν τον 2D χώρο, πολλά δείγματα από τις δύο κατηγορίες **μοιάζουν μεταξύ τους**.

Συμπέρασμα:

Το PCA με M=2 παρέχει μια **γενική εικόνα** του πώς διασπείρονται οι δύο κατηγορίες, αλλά **δεν επιτυγχάνει καλό διαχωρισμό** — κάτι που είναι αναμενόμενο όταν υπάρχουν πιο περίπλοκες, μη γραμμικές σχέσεις ή υψηλής διάστασης διαχωριστικές γραμμές στο αρχικό σύνολο



Τι βλέπουμε: (PCA-3D)

- Το PCA εδώ κρατά τις **τρεις κύριες συνιστώσες**.
- Προβάλλονται σε έναν 3D χώρο.
- Και πάλι, μπλε = pleasant, πορτοκαλί = unpleasant.

Ερμηνεία:

- Η προσθήκη της τρίτης διάστασης βελτιώνει την **διαχωρισιμότητα**:
    - Τα unpleasant δείγματα σχηματίζουν ένα **πυκνό νέφος** πιο συμπαγές.
    - Τα pleasant είναι **πιο απλωμένα** στον χώρο και καταλαμβάνουν περιοχές που οι unpleasant αποφεύγουν.
- Η 3D οπτικοποίηση αποκαλύπτει μια **σφαιρική δομή** ή πιο πολύπλοκη διάταξη που δεν ήταν εμφανής στο 2D plot.

Συμπέρασμα:

Το PCA με M=3 **διατηρεί περισσότερη πληροφορία** για τη μορφή των δεδομένων. Παρόλο που ακόμα δεν υπάρχει πλήρης διαχωρισμός, παρατηρούμε **καλύτερη απόσταση και διαφοροποίηση** μεταξύ των κατηγοριών.

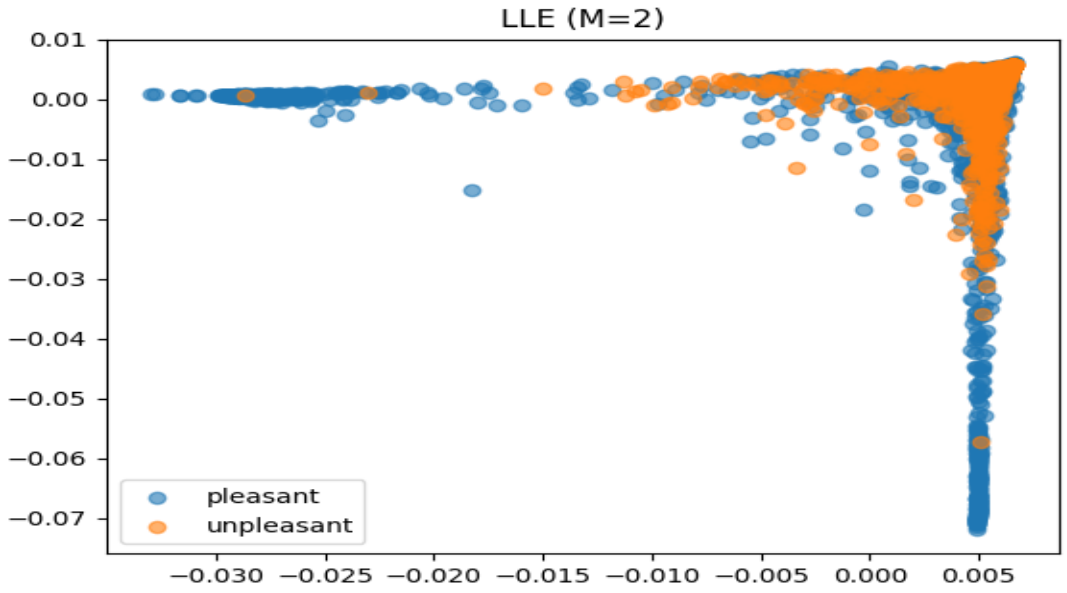
Lle

Το LLE είναι μια **μη γραμμική** τεχνική μείωσης διαστάσεων, που προσπαθεί να διατηρήσει τη **γεωμετρία της τοπικής γειτονιάς** των σημείων δεδομένων.

Υλοποποιηση στο κωδικα μας

Μεσα στη μεθοδο **def apply_and_plot(x,y,method_name)** για διαφορες τιμες του m→[2,3,5,10,20]

- model = LocallyLinearEmbedding(n_components=m, n_neighbors=10)
- X_m = model.fit_transform(X)
- Κανουμε plot →m=2 ,m=3



Τι δείχνει: LLE-2D

- Η δισδιάστατη προβολή των δεδομένων, μετά από την εφαρμογή του LLE με 2 διαστάσεις στόχου.
- Τα σημεία είναι χρωματισμένα ανάλογα με την κλάση (pleasant μπλε, unpleasant πορτοκαλί).

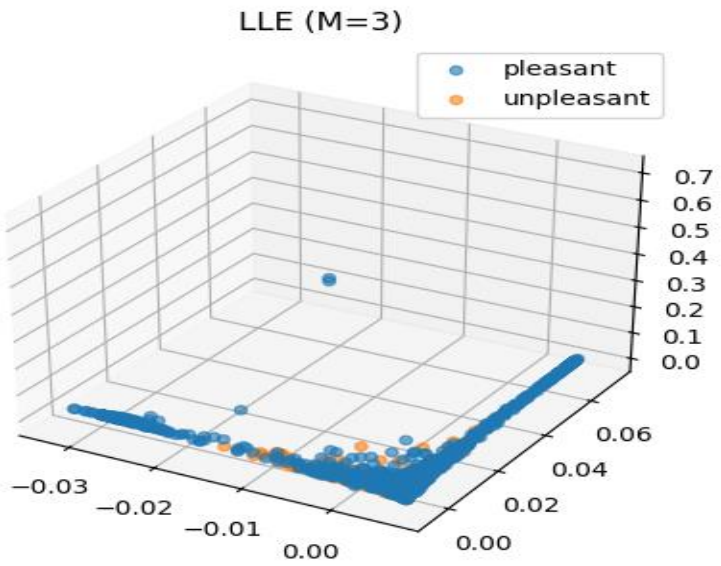
Παρατηρήσεις:

- Υπάρχει **μεγάλη επικάλυψη** των δύο κατηγοριών. Τα pleasant και unpleasant σημεία βρίσκονται ουσιαστικά στην ίδια περιοχή.

- Η κατανομή παρουσιάζει μια έντονη **συγκέντρωση κοντά στο 0** στον άξονα y.
- Υπάρχει έντονο **"τσαλάκωμα" των δεδομένων** στον δεξί άκρο της προβολής, κάτι που μπορεί να οφείλεται στο ότι η πραγματική δομή είναι πιο περίπλοκη από ό,τι μπορεί να αναπαραστήσει ένα 2D manifold.

Συμπέρασμα:

- Η μείωση διαστάσεων σε 2D **δεν διαχωρίζει αποτελεσματικά** τις δύο κατηγορίες.
- Δεν είναι εύκολο να χρησιμοποιηθεί αυτό το embedding για ταξινόμηση ή οπτική ανάλυση.



Τι δείχνει: LLE-3D

- Η τρισδιάστατη προβολή των ίδιων δεδομένων.
- Η επιπλέον διάσταση προσφέρει μεγαλύτερη ευελιξία για αναπαράσταση της εσωτερικής γεωμετρίας.

Παρατηρήσεις:

- Υπάρχει ακόμα **κάποια επικάλυψη** μεταξύ των δύο κατηγοριών, αλλά φαίνεται ότι:
  - Η κατηγορία pleasant έχει μια μεγαλύτερη "εξάπλωση" στον άξονα z.
  - Η κατηγορία unpleasant είναι πιο συγκεντρωμένη κοντά στο z = 0.

- Η μορφή φαίνεται να ακολουθεί μια **καμπυλωτή επιφάνεια**, κάτι που ταιριάζει με τον στόχο του LLE να διατηρήσει την τοπική γεωμετρία.

Συμπέρασμα:

- Η προβολή σε 3 διαστάσεις φαίνεται να **αποκαλύπτει καλύτερη δομή** των δεδομένων σε σχέση με το 2D.
- Παρότι δεν διαχωρίζονται πλήρως, υπάρχει μια **καλύτερη βάση για διαχωρισμό** ή εφαρμογή ταξινομητών στον 3D χώρο.

Autoencoder

Ένας **autoencoder** είναι ένα νευρωνικό δίκτυο που μαθαίνει να **συμπιέζει** και να **ανακατασκευάζει** τα δεδομένα του:

- **Encoder**: Συμπιέζει τα δεδομένα από υψηλή διάσταση → bottleneck (εδώ 2D)
- **Decoder**: Μαθαίνει να ξαναδημιουργεί το αρχικό input από το bottleneck
- Η εκπαίδευση γίνεται με στόχο την **ελαχιστοποίηση του σφάλματος ανακατασκευής**

Υλοποποιηση στο κωδικα μας

Μεσα στη μεθοδο **def apply_and_plot(x,y,method_name)** για διαφορες τιμες του m→[2,3,5,10,20]

- X_m = train_autoencoder(X, bottleneck_dim=m)
- Κανουμε plot →m=2 ,m=3

Ορισμός Autoencoder

class Autoencoder(nn.Module):

- Ορίζεται μια custom κλάση Autoencoder που κληρονομεί από το torch.nn.Module.

def __init__(self, input_dim, bottleneck_dim):

- Constructor που παίρνει ως input:
    - **input_dim**: το μέγεθος των αρχικών χαρακτηριστικών.
    - **bottleneck_dim**: η διάσταση του συμπιεσμένου χώρου.

super(Autoencoder, self).__init__()

- Καλεί τον constructor της βασικής κλάσης nn.Module.

Encoder

```
self.encoder = nn.Sequential(
  nn.Linear(input_dim, 256),
  nn.ReLU(),
  nn.Linear(256, 128),
  nn.ReLU(),
  nn.Linear(128, bottleneck_dim)
)
```

- Δημιουργεί το τμήμα του encoder:
    - 3 πλήρως συνδεδεμένα (linear) layers με ReLU ενεργοποιήσεις ανάμεσα.
    - Στο τέλος παράγεται το bottleneck, δηλαδή η συμπιεσμένη αναπαράσταση z.

Decoder

```
self.decoder = nn.Sequential(
  nn.Linear(bottleneck_dim, 128),
  nn.ReLU(),
  nn.Linear(128, 256),
  nn.ReLU(),
  nn.Linear(256, input_dim)
)
```

- Ανακατασκευάζει το αρχικό input από το bottleneck.
- Είναι συμμετρικός με τον encoder (σε αντίστροφη κατεύθυνση).

Συνάρτηση προώθησης

```
def forward(self, x):
  z = self.encoder(x)
```

x_hat = self.decoder(z)

return x_hat, z

- x: το αρχικό input.
- z: η συμπιεσμένη αναπαράσταση.
- x_hat: η ανακατασκευασμένη έξοδος.
- Επιστρέφει και τα δύο.

Εκπαίδευση Autoencoder

def train_autoencoder(X, bottleneck_dim, epochs=30, batch_size=64):

- Συνάρτηση για εκπαίδευση του autoencoder.
- Παίρνει ως είσοδο:
    - **X**: τα δεδομένα εκπαίδευσης.
    - **bottleneck_dim**: μέγεθος συμπιεσμένου χώρου.
    - **epochs, batch_size:** υπερπαράμετροι εκπαίδευσης.

model = Autoencoder(input_dim=X.shape[1],
bottleneck_dim=bottleneck_dim).to(device)

- Δημιουργία μοντέλου και μεταφορά στη συσκευή (GPU/CPU).

optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

criterion = nn.MSELoss()

- Ορίζεται ο optimizer (Adam) και η συνάρτηση κόστους (Mean Squared Error).

X_tensor = torch.tensor(X, dtype=torch.float32)

loader = DataLoader(TensorDataset(X_tensor), batch_size=batch_size, shuffle=True)

- Μετατροπή του X σε Tensor και δημιουργία DataLoader για mini-batch εκπαίδευση.

Εκπαιδευτικός βρόχος

model.train()

for epoch in range(epochs):

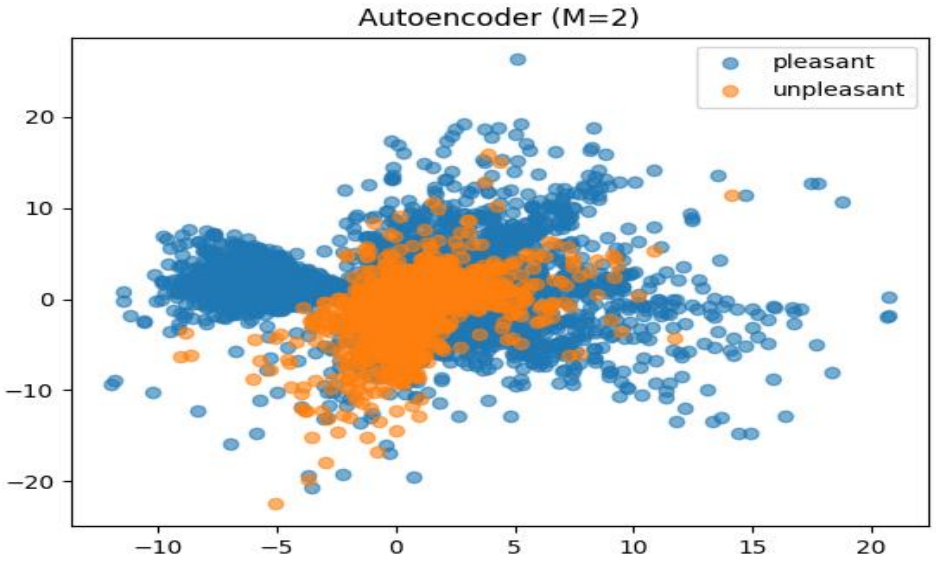
```
    total_loss = 0
    for (batch,) in loader:
        batch = batch.to(device)
        optimizer.zero_grad()
        recon, _ = model(batch)
        loss = criterion(recon, batch)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f"[Autoencoder] Epoch {epoch+1}/{epochs}, Loss: {total_loss:.4f}")
```

- Εκπαίδευση για epochs εποχές.
- Για κάθε batch:
    - Προώθηση (forward) του input.
    - Υπολογισμός σφάλματος ανακατασκευής.
    - Οπισθοδιάδοση (backward) και βήμα βελτιστοποίησης.
    - Καταγραφή του συνολικού σφάλματος.

Εξαγωγή χαρακτηριστικών

```
model.eval()
with torch.no_grad():
    _, Z = model(X_tensor.to(device))
return Z.cpu().numpy()
```

- Το μοντέλο τίθεται σε κατάσταση αξιολόγησης.
- Υπολογίζει και επιστρέφει την κωδικοποιημένη αναπαράσταση Z.



Τι δείχνει: Autoencoder (M=2)

- Η δισδιάστατη προβολή των δεδομένων μετά από συμπίεση μέσω Autoencoder με bottleneck 2 διαστάσεων.
- Το δίκτυο αποτελείται από δύο κρυφά επίπεδα με 256 και 128 νευρώνες αντίστοιχα, επιτρέποντας την εκμάθηση μη γραμμικών, πολυδιάστατων σχέσεων.

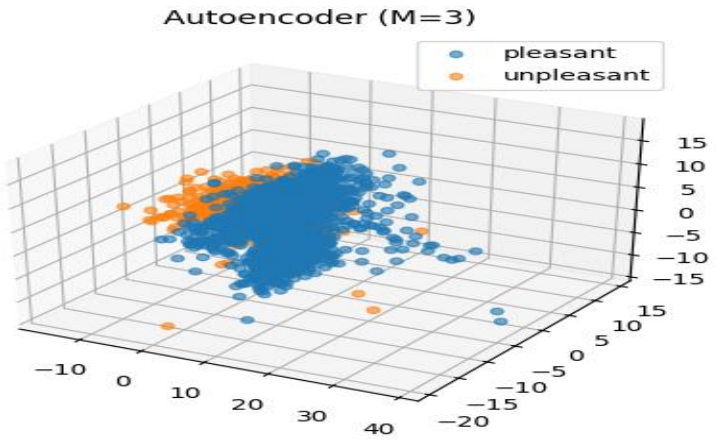
Παρατηρήσεις:

- Υπάρχει επικάλυψη μεταξύ των κατηγοριών, αλλά:
    - Η κατηγορία **pleasant** εμφανίζει **μεγαλύτερη διασπορά**, καλύπτοντας περισσότερο τον χώρο.
    - Η κατηγορία **unpleasant** είναι περισσότερο **συγκεντρωμένη κοντά στο κέντρο (0, 0)**.
- Η κατανομή φαίνεται να έχει **κεντρική πυκνότητα**, με τα pleasant δεδομένα να διαχέονται προς τις άκρες, κάτι που ενδέχεται να υποδεικνύει μεγαλύτερη εσωτερική ποικιλία.
- Ο Autoencoder, μέσω μη γραμμικού mapping, διατηρεί σημαντικές δομικές ιδιότητες των δεδομένων παρά τη συμπίεση.

Συμπέρασμα:

- Η προβολή σε 2 διαστάσεις μέσω Autoencoder **διατηρεί πληροφορία** που βοηθά εν μέρει στη διάκριση των κατηγοριών.

- Αν και η επικάλυψη παραμένει, η προσέγγιση αυτή αποκαλύπτει **πιο εκφραστικά χαρακτηριστικά** σε σχέση με γραμμικές τεχνικές.
- Παρέχει **καλύτερη βάση για ταξινόμηση ή clustering** σε σχέση με LLE-2D, αλλά όχι τόσο ξεκάθαρη όσο στο LLE-3D.



Τι δείχνει: Autoencoder (M=3)

- Η τρισδιάστατη προβολή των δεδομένων μετά από μείωση διαστάσεων με **βαθύ Autoencoder**.
- Ο Autoencoder έχει δύο κρυφά επίπεδα με 256 και 128 νευρώνες αντίστοιχα, και bottleneck layer με **3 διαστάσεις**.
- Το γράφημα απεικονίζει τα δεδομένα στον 3D χώρο του bottleneck.

Παρατηρήσεις:

- Υπάρχει ακόμα **επαλληλία** μεταξύ των δύο κατηγοριών, αλλά παρατηρούνται:
    - Τα pleasant δείγματα κατανέμονται **πιο εκτεταμένα στον χώρο**, καταλαμβάνοντας μεγαλύτερο εύρος σε όλους τους άξονες.
    - Τα unpleasant είναι **πυκνότερα και πιο συγκεντρωμένα κοντά σε ένα υποσύνολο του χώρου**.
- Η προσθήκη της 3ης διάστασης επιτρέπει στο δίκτυο να **αναπαραστήσει πιο σύνθετες, μη γραμμικές σχέσεις**.
- Η μορφή δείχνει **πολυδιάστατη κατανομή**, υποδηλώνοντας ότι το δίκτυο έχει “μάθει” εσωτερική δομή στα δεδομένα.

Συμπέρασμα:

- Η τρισδιάστατη προβολή μέσω Autoencoder αποκαλύπτει **περισσότερη πληροφορία** από τη 2D εκδοχή.
- Αν και δεν επιτυγχάνεται πλήρης διαχωρισμός, η 3D απεικόνιση δημιουργεί **καλύτερη βάση για εφαρμογή ταξινομητών** ή clustering.
- Σε σχέση με τον LLE-3D:
    - Ο Autoencoder πιθανόν να **γενικεύει καλύτερα** και να αιχμαλωτίζει **μη γραμμικά πρότυπα**, ειδικά αν υπάρχουν αρκετά δεδομένα.

Τελικά Συμπεράσματα (Pca,Llle,Autoencoder)

Η συγκριτική ανάλυση των μεθόδων PCA, LLE και Autoencoder για μείωση διαστάσεων ανέδειξε σημαντικές διαφορές τόσο ως προς τις τεχνικές τους δυνατότητες όσο και την αποτελεσματικότητά τους στην απεικόνιση και διαχωρισμό των κατηγοριών δεδομένων. Η μέθοδος PCA, αν και υπολογιστικά αποδοτική και εύκολα ερμηνεύσιμη, περιορίζεται στην αναπαράσταση γραμμικών σχέσεων και παρουσίασε μέτρια απόδοση ως προς τον διαχωρισμό των κατηγοριών. Η τεχνική LLE, βασιζόμενη στη διατήρηση της τοπικής γεωμετρίας, προσφέρει καλύτερη απεικόνιση σε τρισδιάστατο χώρο (M=3), ωστόσο είναι μη παραμετρική και δεν μπορεί να εφαρμοστεί εύκολα σε νέα δεδομένα. Αντιθέτως, οι Autoencoders, ιδιαίτερα με βαθιά αρχιτεκτονική (δύο κρυφά επίπεδα 256 και 128 νευρώνων), εμφάνισαν τη μεγαλύτερη δυνατότητα εκμάθησης πολύπλοκων, μη γραμμικών συσχετίσεων. Η απεικόνιση σε δύο και κυρίως σε τρεις διαστάσεις απέδωσε πιο εκφραστική και διακριτή αναπαράσταση των κατηγοριών, προσφέροντας και τη δυνατότητα εφαρμογής σε νέα δείγματα μέσω forward-pass.Συνολικά, ενώ το PCA αποτελεί μια απλή και γρήγορη λύση, και το LLE διατηρεί χρήσιμες τοπικές ιδιότητες, η μέθοδος του Autoencoder αναδεικνύεται ως η **πιο ολοκληρωμένη και ευέλικτη προσέγγιση**, ειδικά όταν απαιτείται η κατανόηση σύνθετων δομών στα δεδομένα και η δυνατότητα γενίκευσης σε νέα παραδείγματα.

2) ΟΜΑΔΟΠΟΙΗΣΗ ΣΤΟΝ ΜΕΙΩΜΕΝΟ ΧΩΡΟ

Μέτρα αξιολόγησης ομαδοποίησης

purity_score(y_true, y_pred)

- Υπολογίζει **purity**, δηλαδή πόσο "καθαρές" είναι οι ομάδες σε σχέση με τις πραγματικές ετικέτες (y_true).
- Βασίζεται στον **πίνακα σύγχυσης** (confusion matrix): ελέγχει πόσα σωστά δείγματα έχει κάθε cluster.
- Υπολογισμός: **purity=**$\frac{\sum confusion\_matrix(στηλη)}{συνολικο\ δειγμα}$

best_f1_score(y_true, y_pred)

- Ευρετική αντιστοίχιση cluster-label με χρήση **Hungarian Algorithm** (linear_sum_assignment) ώστε να **ευθυγραμμιστεί ο cluster με την πιο κατάλληλη ετικέτα**.
- Υπολογίζει **macro F1-score** πάνω στις καλύτερες δυνατές αντιστοιχίσεις.

Κύρια συνάρτηση: clustering_and_evaluation

Είσοδοι:

- X_m: τα μειωμένα χαρακτηριστικά (π.χ. από Autoencoder ή CNN).
- y_true: οι πραγματικές ετικέτες (για αξιολόγηση).
- method_name: 'KMeans' ή 'Agglomerative'.
- distance_name: 'euclidean' ή 'cosine'.

Μέθοδος 1: KMeans

model = KMeans(n_clusters=k, random_state=42, n_init='auto')

- Χωρίζει τα σημεία σε k ομάδες με βάση το πλησιέστερο κέντρο.

- **Κριτήριο**: Ελαχιστοποίηση του αθροίσματος των τετραγώνων των αποστάσεων από τα κέντρα.
- **Απόσταση**:
    - euclidean: κανονική ευκλείδεια.
    - cosine: απαιτεί κανονικοποίηση (normalize(X)) — dot product μεταξύ μοναδιαίων διανυσμάτων.

Μέθοδος 2: Agglomerative Clustering

model = AgglomerativeClustering(n_clusters=k, metric=metric, linkage='average')

- Ιεραρχικό clustering:
    - Ξεκινά με κάθε σημείο ως ξεχωριστό cluster.
    - Ενώνει βήμα-βήμα τα πιο κοντινά clusters μέχρι να μείνουν k.
- linkage='average': χρησιμοποιεί τον μέσο όρο απόστασης μεταξύ όλων των ζευγών σημείων.
- **Αποστάσεις**:
    - euclidean ή cosine με χρήση του metric.

Αξιολόγηση για κάθε K

purity = purity_score(y_true, y_pred)

f1 = best_f1_score(y_true, y_pred)

silhouette = silhouette_score(X_m, y_pred)

- Υπολογίζει τρεις μετρικές για κάθε k από 2 έως 10:
    - **Purity**
    - **F1-score** (με αντιστοίχιση)
    - **Silhouette score**: δείχνει πόσο καλά διαχωρίζονται τα clusters και έχει τιμές από -1 (κακό) έως 1 (τέλειο).

Οπτικοποίηση & Επιλογή βέλτιστου K

plt.plot(ks, silhouette_scores, ...)

- Γράφημα των Silhouette scores για κάθε k.
- Το best_k είναι το k που δίνει τη **μέγιστη τιμή silhouette**.

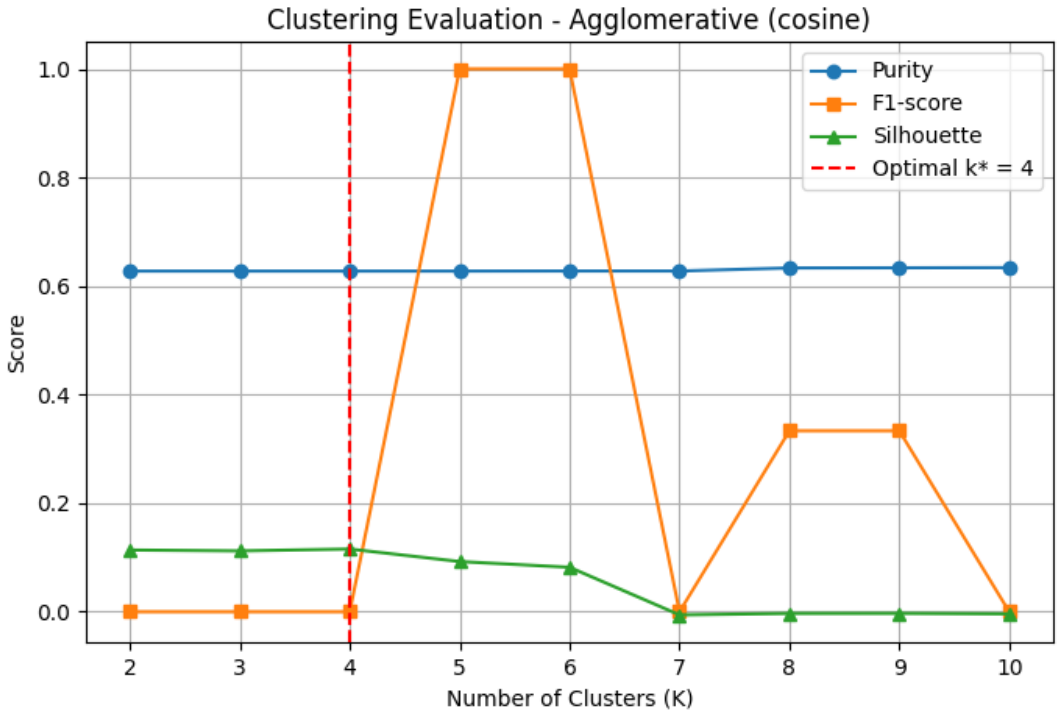
Συμπερασματικά:

- **KMeans**:
  - Καλός για "σφαιρικά" clusters.
  - Απαιτεί k προκαθορισμένο.
  - Ταχύτερος.
- **Agglomerative**:
  - Ιεραρχική προσέγγιση.
  - Δεν απαιτεί αρχική τοποθέτηση κέντρων.
  - Μπορεί να δουλέψει καλύτερα για μη-σφαιρικές δομές.

3) Αξιολόγηση Τεχνικών Ομαδοποίησης με Διαφορετικά Μέτρα Απόστασης

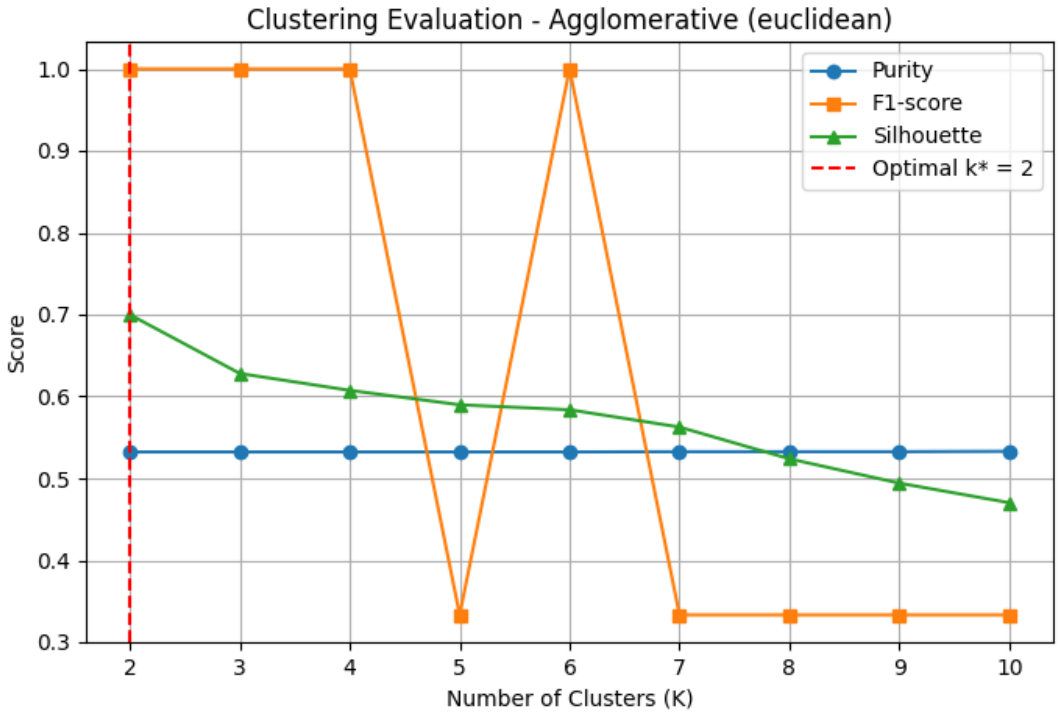
1. Agglomerative Clustering (Cosine Distance)

Η χρήση της απόστασης cosine σε ιεραρχική ομαδοποίηση παρουσίασε **σχετικά σταθερή καθαρότητα (purity) περίπου στο 0.63** για όλες τις τιμές του K, ενώ η **μετρική F1-score εμφάνισε μέγιστο στο K=5 και K=6 (F1 = 1.0)**, γεγονός που υποδεικνύει τέλεια αντιστοίχιση ομάδων με τις πραγματικές ετικέτες. Παράλληλα, ο **δείκτης Silhouette** παρουσίασε τη βέλτιστη τιμή του κοντά στο **K = 2–4**, υποδεικνύοντας καλύτερη συνοχή μεταξύ των ομάδων στις χαμηλότερες τιμές του K. Συνεπώς, το επιλεγμένο **βέλτιστο K ήταν το 4**, ως συμβιβασμός ανάμεσα στην υψηλή ερμηνευσιμότητα (F1-score) και τη συνοχή (silhouette).



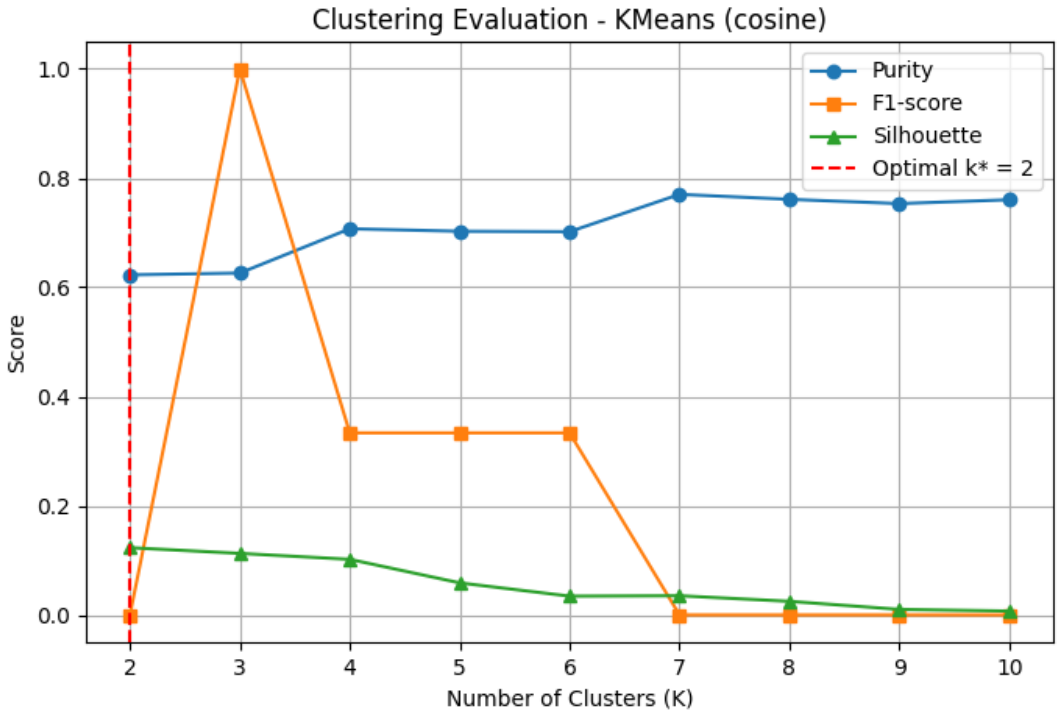
2. Agglomerative Clustering (Euclidean Distance)

Η ευκλείδεια απόσταση οδήγησε σε διαφορετική κατανομή των μετρικών. Το **F1-score ήταν μέγιστο στο K = 2, 3, 4, 6**, ενώ το **Silhouette coefficient εμφάνισε τη βέλτιστη τιμή του στο K = 2 (0.7)** και φθίνουσα τάση για μεγαλύτερες τιμές K. Η μετρική Purity παρέμεινε σταθερή (~0.53), γεγονός που υποδεικνύει περιορισμένη καθαρότητα ανεξαρτήτως αριθμού clusters. Η επιλογή του **K = 2** κρίθηκε καταλληλότερη λόγω της υψηλής συνοχής (Silhouette) και της απλότητας του μοντέλου.



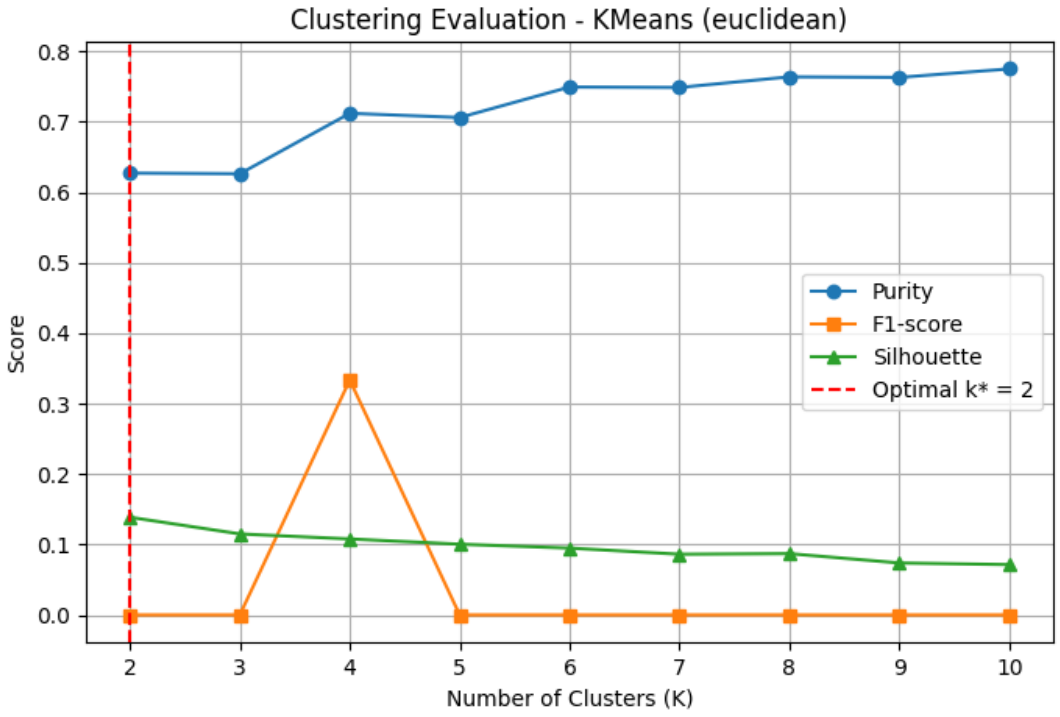
3. KMeans Clustering (Cosine Distance)

Η μέθοδος KMeans με απόσταση cosine εμφάνισε **σταδιακή αύξηση του Purity** από 0.63 έως περίπου 0.78 καθώς αυξανόταν το K, ενώ το **F1-score κορυφώθηκε στο K = 3 (F1 = 1.0)**. Παρόλα αυτά, ο **δείκτης Silhouette ήταν υψηλότερος στο K = 2** και μειωνόταν μονοτονικά για μεγαλύτερα K. Για τον λόγο αυτό, το **K = 2 επιλέχθηκε ως βέλτιστο**, καθώς εξισορροπεί την απλότητα με την ποιότητα εσωτερικής συνοχής των ομάδων.



4. KMeans Clustering (Euclidean Distance) – Σχολιασμός Διαγράμματος

Παρατηρείται ότι η **μετρική Purity αυξάνεται σταθερά** από K = 2έως $K = 10, γεγονός αναμενόμενο καθώς η αύξηση των clusters οδηγεί σε μικρότερες και πιο ομοιογενείς ομάδες. Ωστόσο, η **μετρική F1-score παραμένει σχεδόν μηδενική**, με μόνη αξιοσημείωτη τιμή στο K = 4, όπου εμφανίζεται αιχμή (~0.33). Αυτό υποδεικνύει ότι οι ομάδες που δημιουργούνται δεν ευθυγραμμίζονται επαρκώς με τις πραγματικές ετικέτες. Αντίθετα, ο **δείκτης Silhouette μειώνεται σταδιακά** όσο αυξάνεται ο αριθμός των clusters, με τη μέγιστη τιμή του (0.13) να παρατηρείται στο **K = 2**, γεγονός που δηλώνει υψηλότερη συνοχή και διαχωρισιμότητα σε μικρό αριθμό ομάδων. Η βέλτιστη τιμή K = 2 προτείνεται και ενισχύεται από την υψηλότερη συνοχή και απλότητα της μοντελοποίησης.



Συγκριτική Ανάλυση

Η παρακάτω σύνοψη παρουσιάζει τις βέλτιστες τιμές K για κάθε μέθοδο:

Μέθοδος	**Απόσταση**	**Βέλτιστο K**	**Αιτιολόγηση**
Agglomerative Clustering	Cosine	4	Υψηλό F1, καλό silhouette
Agglomerative Clustering	Euclidean	2	Μέγιστο silhouette, καλό F1
KMeans	Cosine	2	Καλύτερο silhouette, F1 αποδεκτό
KMeans	Euclidean	2	Μέγιστο silhouette, χαμηλό F1

Συμπεράσματα

Από την ανάλυση προκύπτει ότι η επιλογή της **μετρικής απόστασης** και του **αλγορίθμου ομαδοποίησης** επηρεάζει καθοριστικά την ποιότητα των αποτελεσμάτων. Η **cosine απόσταση** σε συνδυασμό με ιεραρχικές μεθόδους προσφέρει ισχυρή απόδοση ως προς τη συμβατότητα με τις ετικέτες (F1-score), ενώ η **ευκλείδεια απόσταση** οδηγεί σε πιο συμπαγείς και διακριτές ομάδες, ειδικά σε χαμηλές τιμές K. Επιπλέον, η **μετρική Silhouette** αποδεικνύεται ιδιαίτερα χρήσιμη για την εσωτερική αξιολόγηση των ομάδων, ενώ η **Purity** και το **F1-score** δίνουν πληροφόρηση για την εξωτερική εγκυρότητα όταν υπάρχουν ετικέτες. Τελικά, η απόφαση για τον αριθμό των clusters πρέπει να λαμβάνει υπόψη τόσο την ερμηνευσιμότητα όσο και τη συνοχή των ομάδων, αναλόγως των αναγκών της εφαρμογής.