Πανεπιστήμιο Μακεδονίας
Π.Μ.Σ. στην Εφαρμοσμένη Πληροφορική (ειδίκευση Ανάπτυξη Λογισμικού και Νέφος)
Τμήμα Εφαρμοσμένης Πληροφορικής

Μέθοδοι και Εργαλεία Τεχνητής Νοημοσύνης

2η Εργασία (Μάθηση Χωρίς Επίβλεψη – Συσταδοποίηση)

Φοιτητής: Καλαϊτζίδης Δημήτριος
ΑΜ: mai24018

Εκθεση για την Ανάλυση και Αξιολόγηση Τεχνικών Συσταδοποίησης στο Σύνολο Δεδομένων Fashion MNIST

Εισαγωγή

Στην παρούσα εργασία, μελετήθηκε η αποτελεσματικότητα διαφόρων τεχνικών συσταδοποίησης (**clustering**) στο σύνολο δεδομένων Fashion MNIST χρησιμοποιώντας τεχνικές μείωσης διαστάσεων. Συγκεκριμένα, εξετάστηκαν η **Principal Component Analysis (PCA**) και ο **Stacked Autoencoder (SAE)**, σε συνδυασμό με αλγορίθμους clustering όπως ο **MiniBatchKMeans** και η **Gaussian Mixture Model (GMM)**.

1. **Φόρτωση και Προεπεξεργασία Δεδομένων:**

Αρχικά, τα δεδομένα εισόδου φορτώνονται και κανονικοποιούνται για να είναι στο διάστημα **[0, 1]**. Η κανονικοποίηση των δεδομένων είναι απαραίτητη για τη βελτιστοποίηση της απόδοσης των μεθόδων μη επιβλεπόμενης μάθησης που θα εφαρμόσουμε στη συνέχεια.

Στη συνέχεια, τα δεδομένα ανασχηματίζονται ώστε να είναι σε μορφή που να μπορούν να χρησιμοποιηθούν από τους αλγορίθμους μας. Οι εικόνες των δεδομένων ανασχηματίζονται σε έναν πίνακα με διαστάσεις **(28x28)** για κάθε εικόνα, και στη συνέχεια επίπεδα των εικόνων αναδιαμορφώνονται σε έναν πίνακα με διαστάσεις **(784,)** για κάθε εικόνα, ώστε να είναι συμβατό με τους αλγορίθμους.

Τέλος, τα δεδομένα διαχωρίζονται σε σύνολα εκπαίδευσης και επικύρωσης με τη χρήση της συνάρτησης **train_test_split**. Το 10% των δεδομένων χρησιμοποιείται για validation, ενώ το υπόλοιπο 90% χρησιμοποιείται για την εκπαίδευση των μοντέλων. Αυτός ο διαχωρισμός είναι απαραίτητος για την εκπαίδευση και την αξιολόγηση των μοντέλων συσταδοποίησης.

```
# Normalize data
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# Reshape Data
X_train = X_train.reshape(-1, 784)
X_test = X_test.reshape(-1, 784)

# 2) Split the data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=42)
```

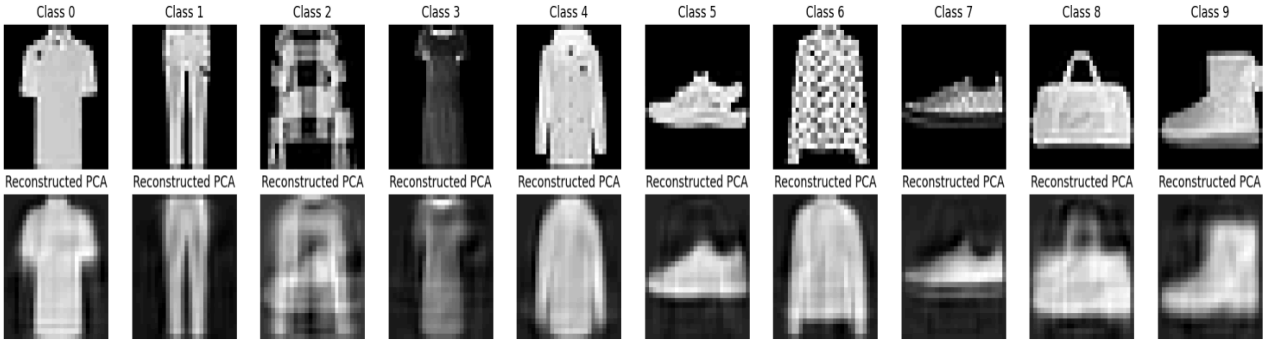
2. Τεχνικές Μείωσης Διαστάσεων:

Στο επόμενο βήμα, χρησιμοποιήσαμε τεχνικές μείωσης διαστάσεων για την επεξεργασία των δεδομένων μας πριν από την εφαρμογή των αλγορίθμων συσταδοποίησης. Ειδικότερα, χρησιμοποιήθηκαν οι παρακάτω μέθοδοι:

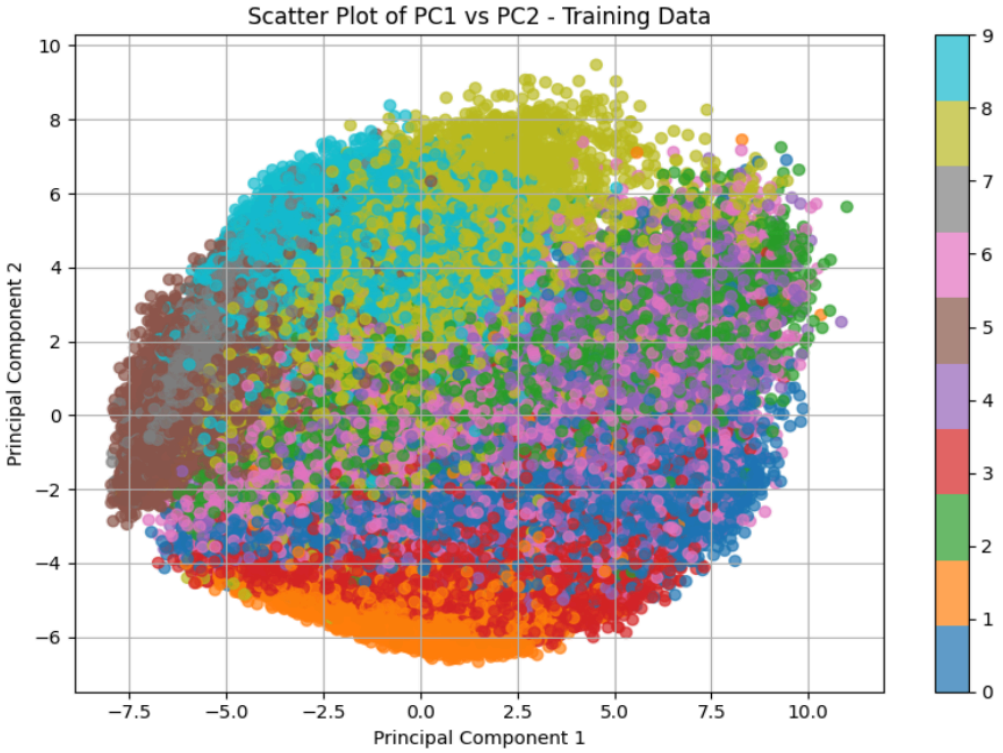
- **PCA (Principal Component Analysis)**: Χρησιμοποιήσαμε τη μέθοδο PCA για τη μείωση της διαστατικότητας των δεδομένων μας. Αυτό επιτρέπει την αναπαράσταση των δεδομένων με λιγότερες διαστάσεις, κρατώντας ταυτόχρονα τις βασικές τους χαρακτηριστικές ιδιότητες. Τα πρώτα δύο κύρια components (principal components) χρησιμοποιήθηκαν για την απεικόνιση των δεδομένων σε ένα διαστηματικό χώρο μειωμένων διαστάσεων, ενώ τα δεδομένα συμπιέστηκαν σε 50 components, δηλαδή σε 50 διαστάσεις από τις 784 (28x28).

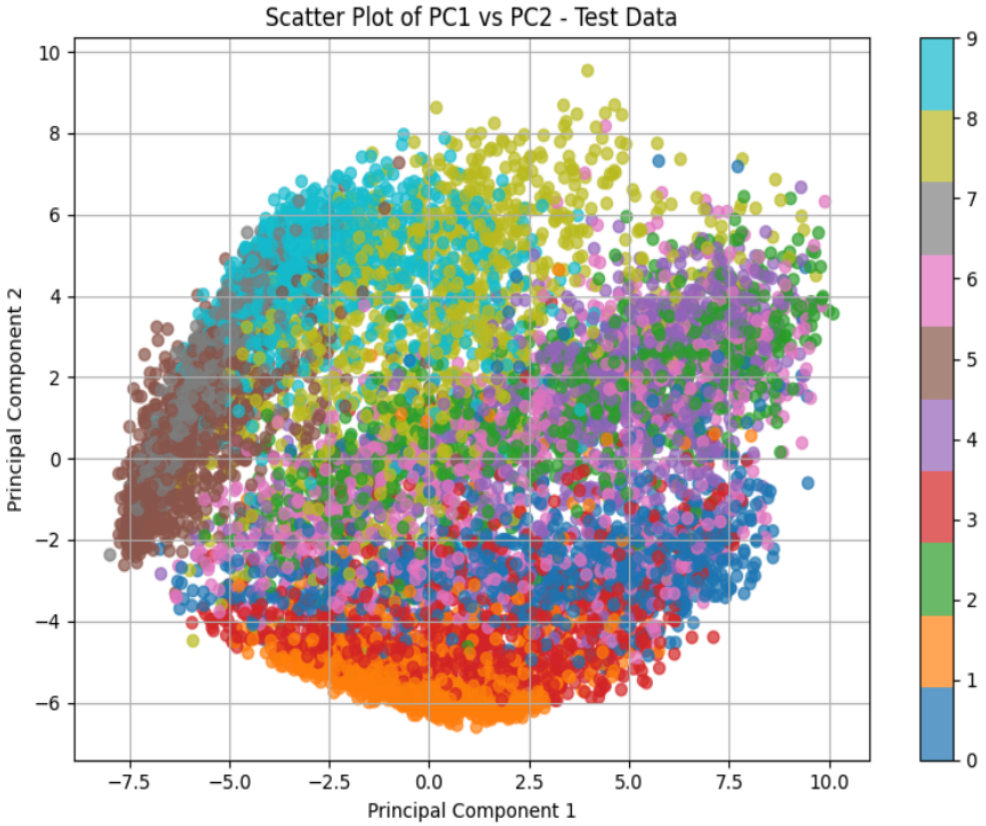
```
# 3, 6) Principal Component Analysis (PCA)
def apply_pca(X_train, X_val, X_test, n_components=50):
    pca = PCA(n_components=n_components)
    X_train_pca = pca.fit_transform(X_train)
    X_val_pca = pca.transform(X_val)
    X_test_pca = pca.transform(X_test)
    return X_train_pca, X_val_pca, X_test_pca, pca
```

Για κάθε κλάση, απεικονίσαμε μια εικόνα στην αρχική της έκδοση και στην αντίστοιχη μορφή της μετά την εφαρμογή του PCA:



Απεικονίσαμε γραφικά τα δεδομένα στον χώρο των δύο πρώτων κύριων συνιστωσών (principal components), προκειμένου να δούμε πώς απεικονίζονται οι εικόνες σε έναν χώρο με μειωμένη διαστατικότητα.



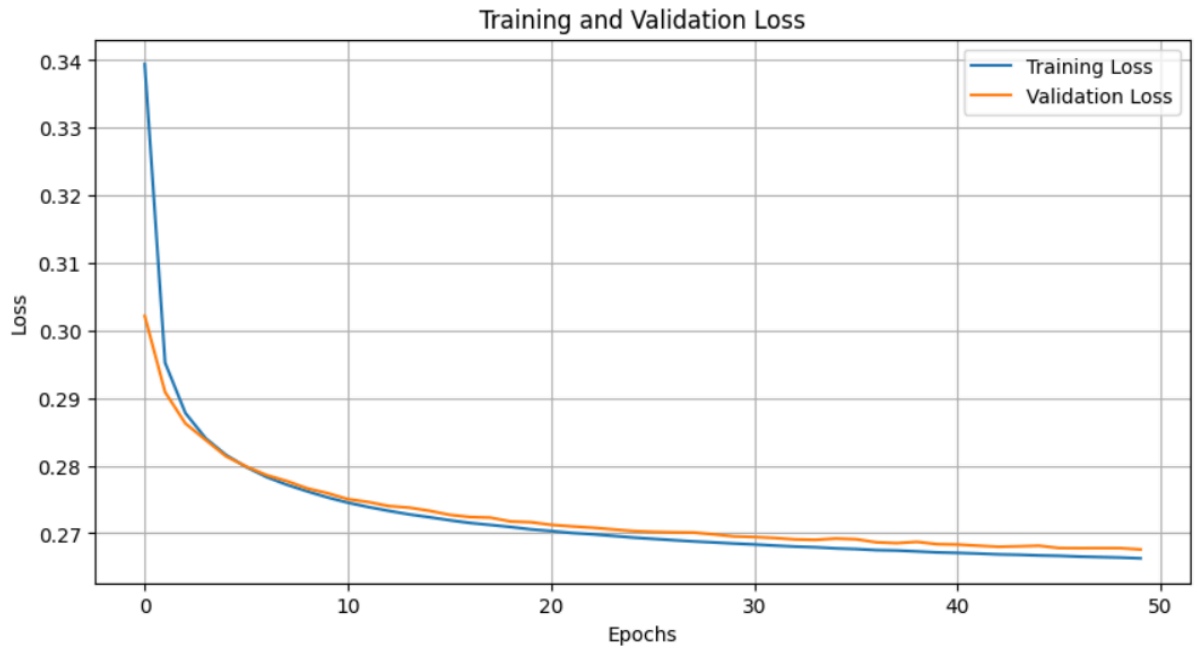


- **SAE (Stacked Autoencoders)**: Η τεχνική SAE επιτρέπει την αυτόματη εξαγωγή συνεκτικών αναπαραστάσεων των δεδομένων, με μια ενσωματωμένη διαδικασία μάθησης για την αναγνώριση των σημαντικών χαρακτηριστικών. Χρησιμοποιήσαμε την SAE για τη μείωση της διαστατικότητας των δεδομένων και τη δημιουργία μιας πιο συμπαγούς αναπαράστασης των εικόνων, η οποία χρησιμοποιήθηκε στη συνέχεια για την εφαρμογή των αλγορίθμων συσταδοποίησης.

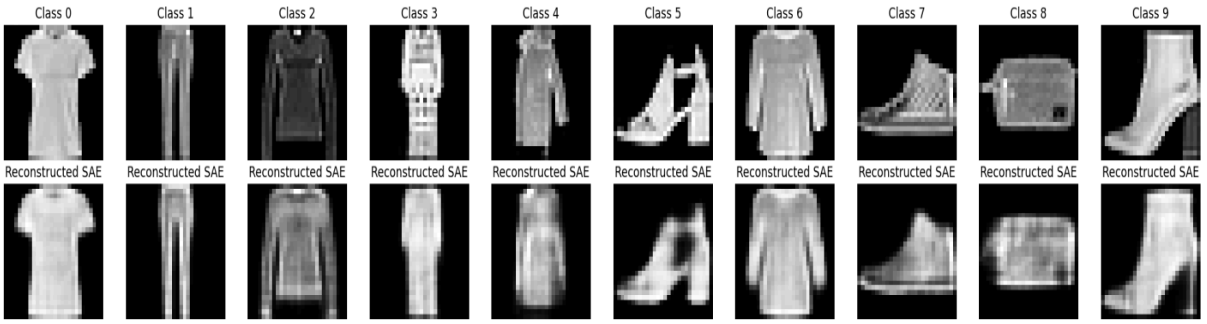
```python
# 3, 6) Stacked Autoencoder (SAE)
def build_autoencoder(input_dim, encoding_dim):
    input_img = Input(shape=(input_dim,))
    encoded = Dense(256, activation='relu')(input_img)
    encoded = Dense(encoding_dim, activation='relu', activity_regularizer=regularizers.l1(10e-5))(encoded)

    decoded = Dense(256, activation='relu')(encoded)
    decoded = Dense(input_dim, activation='sigmoid')(decoded)

    autoencoder = Model(input_img, decoded)
    encoder = Model(input_img, encoded)

    autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
    return autoencoder, encoder

def apply_sae(X_train, X_val, X_test, encoding_dim=50, epochs=50, batch_size=128):
    input_dim = X_train.shape[1]
    autoencoder, encoder = build_autoencoder(input_dim, encoding_dim)
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
    history = autoencoder.fit(X_train, X_train,
                              epochs=epochs,
                              batch_size=batch_size,
                              shuffle=True,
                              validation_data=(X_val, X_val),
                              callbacks=[early_stopping])
    X_train_sae = encoder.predict(X_train)
    X_val_sae = encoder.predict(X_val)
    X_test_sae = encoder.predict(X_test)
    return X_train_sae, X_val_sae, X_test_sae, autoencoder, encoder, history
```

Στο συγκεκριμένο μοντέλο SAE που αναπτύχθηκε, η είσοδος περνά μέσω δύο κρυφών στρωμάτων (ή "stacked layers") από πλήρως συνδεδεμένους νευρώνες. Αρχικά, χρησιμοποιείται ένα στρώμα κωδικοποιητή που αποτελείται από 256 νευρώνες με την συνάρτηση ενεργοποίησης ReLU, το οποίο συνδέεται με το κύριο στρώμα **encoder** που χρησιμοποιεί την ίδια συνάρτηση ενεργοποίησης, αλλά προσθέτει έναν τύπο περιορισμού δραστηριότητας για να ελέγχει την αποδοτικότητα του δικτύου. Στη συνέχεια, ένα άλλο στρώμα **decoder** αποτελείται από 256 νευρώνες και χρησιμοποιεί την συνάρτηση ενεργοποίησης Sigmoid για να αποκρυπτογραφήσει τα δεδομένα πριν από την αρχική είσοδο. Το νευρωνικό δίκτυο εκπαιδεύεται για 50 epochs, με batch size 128 και σε 50 διαστάσεις συμπιεσμένης αναπαράστασης των δεδομένων μετά τον encoder.

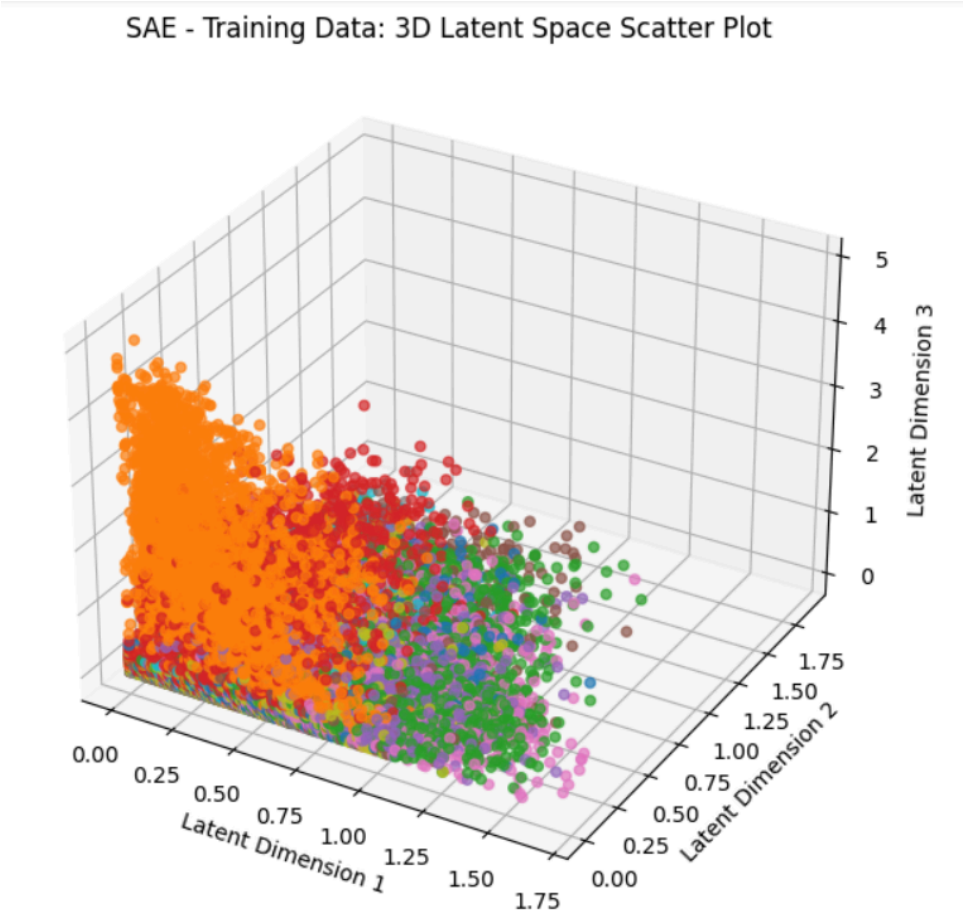
Κατά την εκπαίδευση του μοντέλου, παρατηρήσαμε ότι το training loss και το validation loss κινούνται παρόμοια και συγκλίνουν. Αυτό υποδεικνύει ότι το μοντέλο δεν παρουσιάζει **overfitting.** Η παρόμοια πορεία των δύο μετρικών δείχνει ότι το μοντέλο έχει καλή ικανότητα γενίκευσης σε νέα δεδομένα, ενώ οι παράμετροι εκπαίδευσης έχουν ρυθμιστεί κατάλληλα.

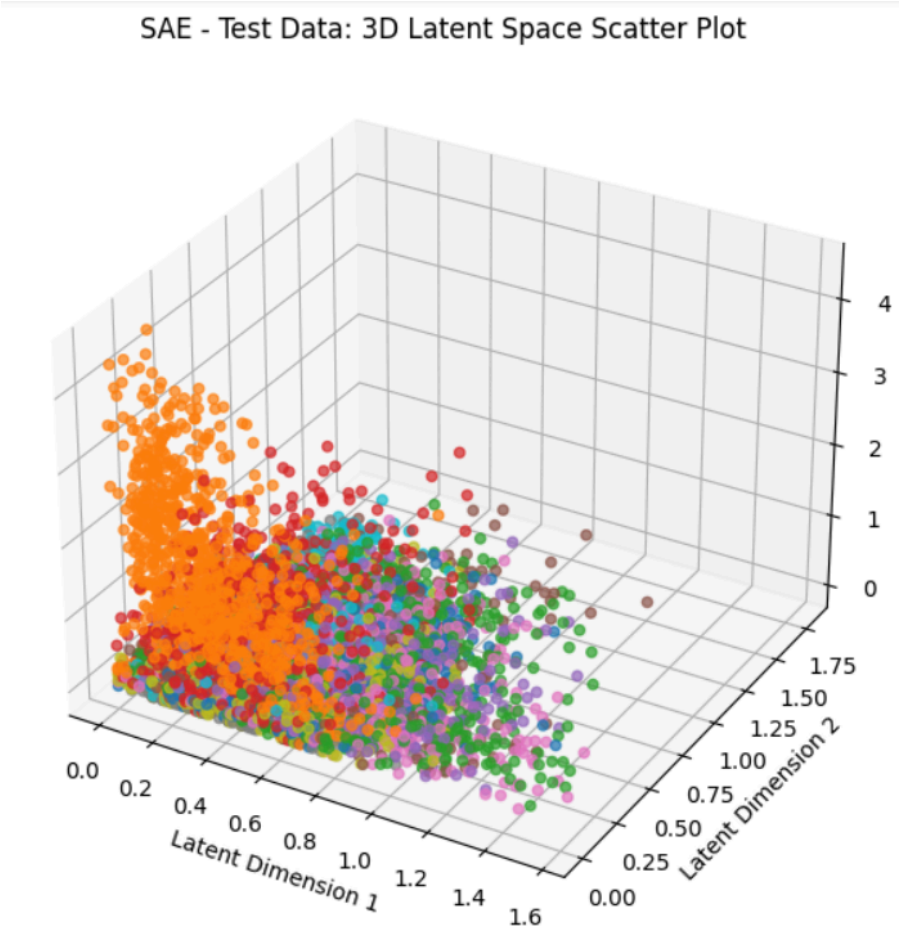


Για κάθε κλάση, απεικονίσαμε μια εικόνα στην αρχική της έκδοση και στην αντίστοιχη μορφή της μετά την εφαρμογή του SAE:



Αναπαραστήσαμε γραφικά τις εικόνες στον χώρο των τριών πρώτων διαστάσεων του **Latent Space**, όπως προέκυψαν μετά την εφαρμογή της SAE.





Αυτές οι τεχνικές βοήθησαν στη βελτιστοποίηση της απόδοσης των αλγορίθμων συσταδοποίησης, επιτρέποντας ταυτόχρονα την οπτικοποίηση των δεδομένων με μειωμένη διαστατικότητα.

3. **Αλγόριθμοι Clustering:**

Για την συσταδοποίηση των δεδομένων, χρησιμοποιήθηκε η παρακάτω συνάρτηση:

```
def apply_clustering(algorithm, X):
    if algorithm == 'MiniBatchKMeans':
      clustering = MiniBatchKMeans(n_clusters=10, random_state=42, n_init=3)
    elif algorithm == 'GMM':
      clustering = GaussianMixture(n_components=10, random_state=42)
    else:
      raise ValueError("Unknown algorithm: " + algorithm)
    start_time = time.time()
    y_pred = clustering.fit_predict(X)
    end_time = time.time()
    execution_time = end_time - start_time
    return y_pred, execution_time
```

Η συνάρτηση **apply_clustering** χρησιμοποιείται για την εφαρμογή δύο διαφορετικών αλγορίθμων συσταδοποίησης σε ένα σύνολο δεδομένων X. Ανάλογα με την τιμή της παραμέτρου algorithm, αρχικοποιεί το κατάλληλο μοντέλο συσταδοποίησης, υπολογίζει τα αποτελέσματα της συσταδοποίησης και μετρά τον χρόνο εκτέλεσης.

➔ **Παράμετροι Εισόδου:**

- **algorithm**: Αυτή η παράμετρος καθορίζει ποιον αλγόριθμο συσταδοποίησης θα χρησιμοποιηθεί. Υποστηρίζονται δύο επιλογές:
  - **'MiniBatchKMeans'**: Χρησιμοποιεί τον αλγόριθμο Mini-Batch K-Means για συσταδοποίηση.
  - **'GMM'**: Χρησιμοποιεί Gaussian Mixture Model για συσταδοποίηση.
  - Αν δοθεί οποιαδήποτε άλλη τιμή, εκτοξεύεται μια εξαίρεση ValueError.

➔ **Αρχικοποίηση της Συσταδοποίησης:**

- ◆ Αν η παράμετρος algorithm είναι **'MiniBatchKMeans'**, η συνάρτηση αρχικοποιεί ένα μοντέλο συσταδοποίησης **MiniBatchKMeans** με τις εξής παραμέτρους:
  - **n_clusters**: Ο αριθμός των συστάδων ορίζεται στο 10.
  - **random_state**: Seed για την παραγωγή τυχαίων αριθμών, προκειμένου να επιτευχθεί η επαναληπτικότητα.
  - **n_init**: Ο αριθμός των αρχικοποιήσεων που πραγματοποιούνται. Η προεπιλεγμένη τιμή είναι 3.

- ◆ Αν η παράμετρος algorithm είναι **'GMM'**, αρχικοποιεί ένα μοντέλο συσταδοποίησης **GaussianMixture** με τις παραμέτρους:
  - **n_components**: Ο αριθμός των συσταδών ορίζεται στο 10.
  - **random_state**: Seed για την παραγωγή τυχαίων αριθμών, προκειμένου να επιτευχθεί η επαναληπτικότητα.

➔ **Επιστροφή Αποτελεσμάτων:**

- ◆ Εκτελεί τον αλγόριθμο συσταδοποίησης στα δεδομένα εισόδου **X** και επιστρέφει τις προβλεπόμενες ετικέτες των συστάδων **y_pred**.
- ◆ Υπολογίζει τον χρόνο εκτέλεσης του αλγορίθμου και τον επιστρέφει ως **execution_time**.

4. **Αξιολόγηση και Σύγκριση Αποτελεσμάτων:**

Τα αποτελέσματα της ανάλυσης συσταδοποίησης έχουν αποθηκευτεί σε ένα DataFrame για περαιτέρω ανάλυση και παρουσίαση. Το DataFrame περιλαμβάνει τις ακόλουθες μετρικές αξιολόγησης για διάφορους συνδυασμούς τεχνικών μειωμένης διάστασης και αλγορίθμων συσταδοποίησης:

- Ο δείκτης Calinski-Harabasz
- Ο δείκτης Davies-Bouldin
- Η μετρική Silhouette

Τα δεδομένα αυτά έχουν αποθηκευτεί επίσης σε ένα αρχείο CSV για ευκολότερη πρόσβαση και επεξεργασία. Αυτή η διαδικασία επιτρέπει τη σύγκριση των αλγορίθμων συσταδοποίησης και των τεχνικών μειωμένης διάστασης βάσει ποιότητας και επίδοσης, προσφέροντας έτσι διευκόλυνση για την επιλογή του βέλτιστου συνδυασμού για την συγκεκριμένη ανάλυση δεδομένων.

Τα αποτελέσματα που βρέθηκαν είναι τα παρακάτω:

	Dimensionality Reduction	Clustering Algorithm	Training Time	Execution Time	\
0	Raw	MiniBatchKMeans	0	2.065742	
1	Raw	GMM	0	215.203379	
2	PCA	MiniBatchKMeans	4.781806	0.264118	
3	PCA	GMM	4.781806	6.288243	
4	SAE	MiniBatchKMeans	149.730679	0.168318	
5	SAE	GMM	149.730679	6.282729	

	Clusters	Calinski-Harabasz Index	Davies-Bouldin Index	Silhouette Score
0	10	1183.382937	2.134079	0.123747
1	10	1029.288628	2.192798	0.105612
2	10	1679.317172	1.813288	0.182173
3	10	1092.375733	2.589049	0.126703
4	10	1081.975362	1.909082	0.147528
5	10	979.348376	2.226180	0.126767

Αξιολόγηση:

Από τα αποτελέσματα που παρουσιάστηκαν, παρατηρούμε τα εξής:

- **Χρόνος Εκπαίδευσης και Εκτέλεσης:** Η χρήση της μείωσης διαστάσεων PCA συνήθως μειώνει τόσο τον χρόνο εκπαίδευσης όσο και τον χρόνο εκτέλεσης σε σχέση με τη χρήση των αρχικών δεδομένων ή την εφαρμογή SAE. Αυτό οφείλεται στη μείωση της διαστατικότητας των δεδομένων, που επιτρέπει στους αλγορίθμους συσταδοποίησης να λειτουργούν πιο γρήγορα και αποδοτικά.

  Χρόνοι Εκτέλεσης:

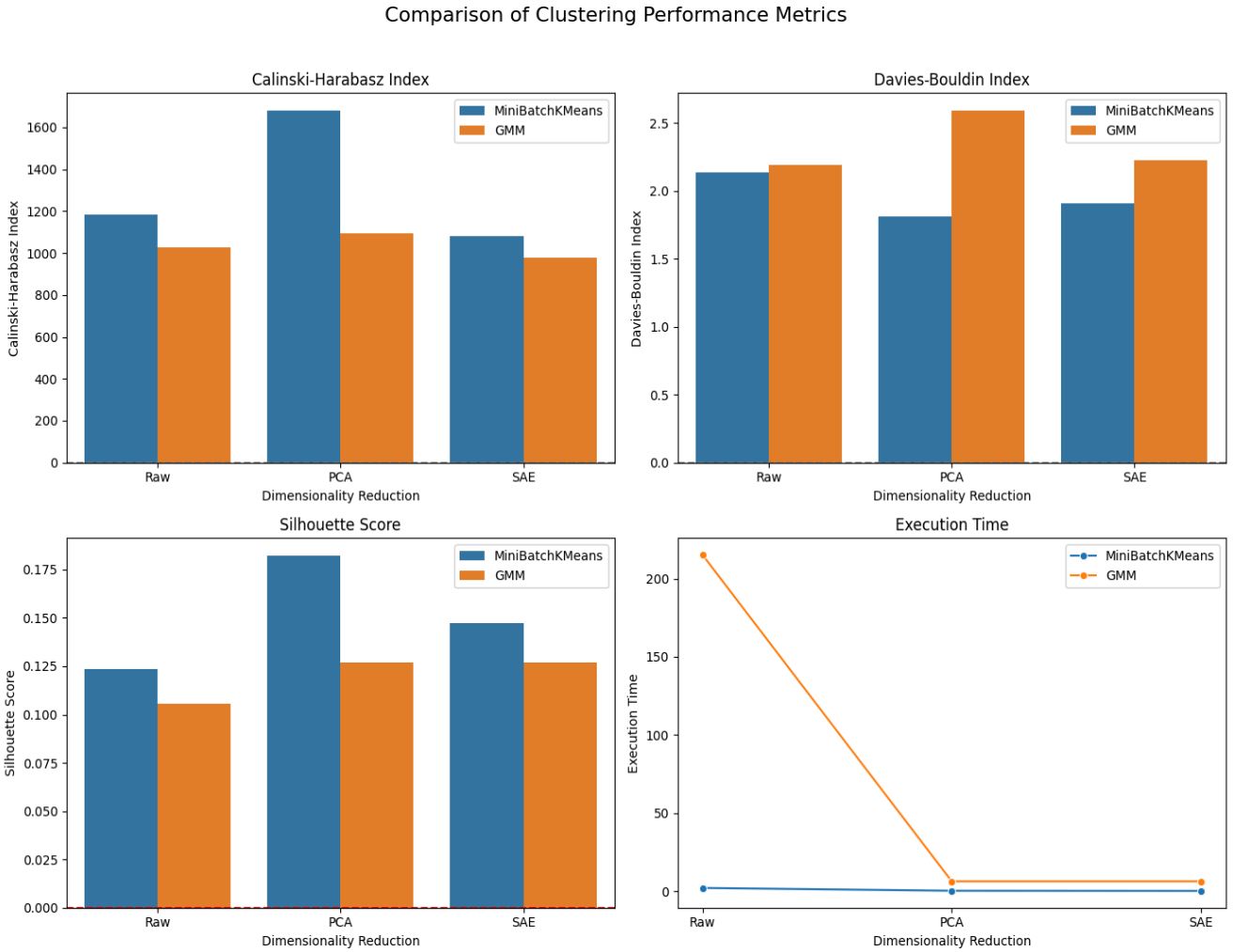
  - **PCA με MiniBatchKMeans:** Χρόνος εκπαίδευσης: περίπου 4.78 δευτερόλεπτα, Χρόνος εκτέλεσης: περίπου 0.26 δευτερόλεπτα.
  - **PCA με GMM:** Χρόνος εκπαίδευσης: περίπου 4.78 δευτερόλεπτα, Χρόνος εκτέλεσης: περίπου 6.29 δευτερόλεπτα.
  - **SAE με MiniBatchKMeans:** Χρόνος εκπαίδευσης: περίπου 149.73 δευτερόλεπτα, Χρόνος εκτέλεσης: περίπου 0.17 δευτερόλεπτα.
  - **SAE με GMM:** Χρόνος εκπαίδευσης: περίπου 149.73 δευτερόλεπτα, Χρόνος εκτέλεσης: περίπου 6.28 δευτερόλεπτα.

- **Ποιότητα Συσταδοποίησης:** Η μείωση της διαστατικότητας μπορεί να έχει επίσης θετική επίδραση στην ποιότητα της συσταδοποίησης, όπως φαίνεται από τις μετρικές Calinski-Harabasz, Davies-Bouldin και Silhouette. Συγκεκριμένα, η PCA παράγει συνήθως καλύτερες μετρικές σε σχέση με τα αρχικά δεδομένα και την SAE, ενώ η SAE επιτυγχάνει επίσης αξιοπρεπείς επιδόσεις.

  - **PCA με MiniBatchKMeans:** Καλύτερη απόδοση στις μετρικές Calinski-Harabasz και Davies-Bouldin συγκριτικά με τον GMM. Υψηλότερος βαθμός Silhouette.
  - **PCA με GMM:** Χειρότερη απόδοση στις μετρικές Calinski-Harabasz και Davies-Bouldin συγκριτικά με τον MiniBatchKMeans. Μέτριος βαθμός Silhouette.
  - **SAE με MiniBatchKMeans:** Αξιοπρεπής απόδοση στις μετρικές Calinski-Harabasz και Davies-Bouldin. Υψηλός βαθμός Silhouette.

- **SAE με GMM:** Χαμηλότερη απόδοση στις μετρικές Calinski-Harabasz και Davies-Bouldin συγκριτικά με τον MiniBatchKMeans. Μέτριος βαθμός Silhouette.

Παρακάτω βλέπουμε σε γραφήματα τις συγκρίσεις μεταξύ των αλγορίθμων συσταδοποίησης και τεχνικών μείωσης διαστάσεων:



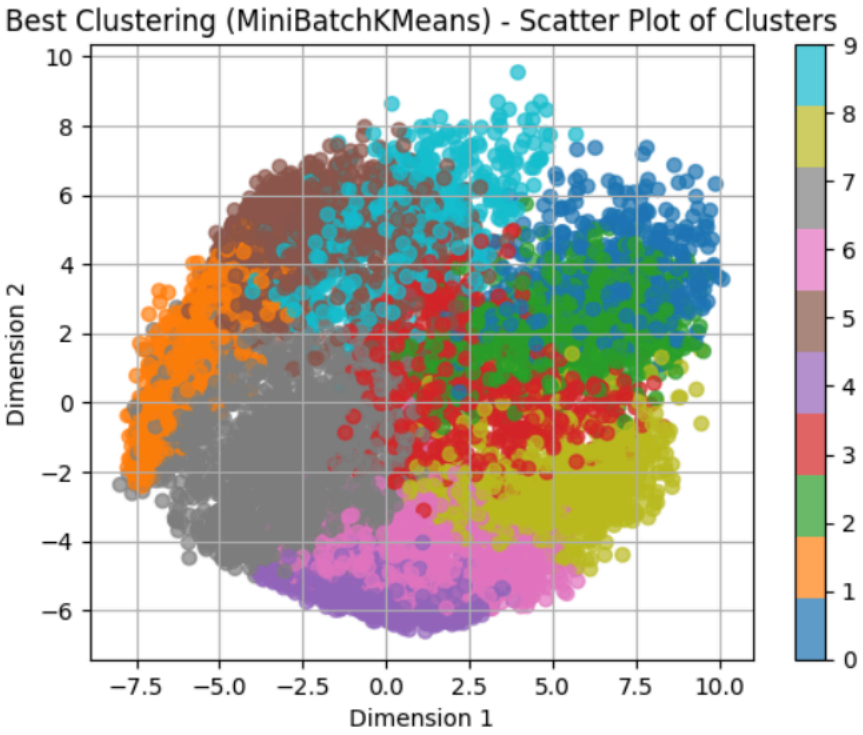
5. **Βέλτιστο Μοντέλο:**

Το βέλτιστο μοντέλο συσταδοποίησης που επιλέχθηκε χρησιμοποιεί τη μέθοδο μείωσης διαστάσεων PCA και τον αλγόριθμο MiniBatchKMeans για την ομαδοποίηση. Ακολουθεί μια περιγραφή των χαρακτηριστικών του:

- **Μέθοδος Μείωσης Διαστάσεων:** PCA (Principal Component Analysis)
- **Αλγόριθμος Συσταδοποίησης:** MiniBatchKMeans
- **Χρόνος Εκπαίδευσης:** Περίπου 4.78 δευτερόλεπτα
- **Χρόνος Εκτέλεσης:** Περίπου 0.26 δευτερόλεπτα
- **Αριθμός Συστάδων:** 10
- **Calinski-Harabasz Index:** Περίπου 1679.32
- **Davies-Bouldin Index:** Περίπου 1.81
- **Silhouette Score:** Περίπου 0.18

Το μοντέλο αυτό επιλέχθηκε για την ικανότητά του να παρέχει καλή ποιότητα συσταδοποίησης (υψηλότεροι δείκτες Calinski-Harabasz και Silhouette, χαμηλότερος δείκτης Davies-Bouldin) με σχετικά χαμηλό χρόνο εκτέλεσης.

Παρακάτω βλέπουμε και ένα γράφημα, στο οποίο βλέπουμε την απόδοση αυτού του μοντέλου.



Παρατηρούμε ότι οι συστάδες είναι αρκετά καλά ομαδοποιημένες και είναι εύκολη η παρατήρηση και ο διαχωρισμός τους. Παρόλα αυτά, παρατηρούμε ότι υπάρχει μερική επικάλυψη ορισμένων συστάδων. Ωστόσο, σε γενικές γραμμές, ο MiniBatchKMeans με εφαρμοσμένο PCA στα δεδομένα, οδηγεί σε ένα ικανοποιητικό αποτέλεσμα συσταδοποίησης.