🚚 Vehicle Type Clustering with Autoencoders & Unsupervised Learning

1. 📦 Project Objective

The goal of this project is to cluster vehicles into meaningful categories (bus, car, van) using **unsupervised learning techniques**. We applied traditional clustering methods directly on features, then improved performance by compressing data with an **autoencoder** before clustering on the learned latent space.

2. 📊 Data Overview

- **Dataset**: Tabular dataset of vehicle features with an associated class label (bus, car, van) used only for evaluation.
- **Shape**: (X samples, Y features) – exact dimensions not specified.
- **Attributes**: Numerical features (e.g., size, shape, acceleration, etc.)
- **Labels**: class – used only for post-clustering evaluation (unsupervised setting).

3. 🔍 Exploratory Data Analysis (EDA)

- **Descriptive statistics**: Distribution, mean, variance of features.
- **Feature Scaling**: Applied MinMaxScaler or StandardScaler to normalize all features to a similar range.
- **Class Balance**: Approximately balanced distribution among bus, car, and van.
- **Insights**:
    - Many features show overlapping distributions across classes.
    - Traditional clustering on raw features resulted in poor separation.

4. 🧠 Dimensionality Reduction with Autoencoder

We trained a deep autoencoder to compress the feature space:

Architecture:

- **Input Layer**: Same size as original feature count
- **Encoder Layers**: 2-3 dense layers reducing down to encoding_dim = 10
- **Decoder Layers**: Mirror structure to reconstruct input
- **Loss Function**: Mean Squared Error
- **Optimizer**: Adam

Training:

python

```
autoencoder.fit(X_scaled, X_scaled,
            epochs=..., batch_size=...,
            validation_split=0.2)
```

- Training and validation loss dropped consistently.
- Latent features (encoded_df) were extracted using:

python

```
encoded_features = encoder.predict(X_scaled)
```

5. 📈 Clustering Approaches

We applied multiple clustering techniques before and after feature compression:

A. DBSCAN on Raw & Encoded Data

- **Raw Features**: Detected mostly one cluster + noise
- **Encoded Features**: Slight improvement with tuned eps, but still struggled to detect clear separation.
- Tuning eps revealed potential second cluster—but many points remained noise.

B. KMeans on Encoded Data

python

```
kmeans = KMeans(n_clusters=3)
labels_kmeans = kmeans.fit_predict(encoded_df)
```

- Much better separation of vehicle types.
- **Silhouette Score ≈ 0.83**, indicating strong intra-cluster cohesion.

C. Agglomerative Clustering

python

```
agg = AgglomerativeClustering(n_clusters=3)
labels_agg = agg.fit_predict(encoded_df)
```

- **Best alignment with actual labels**: buses, cars, and vans separated proportionally.
- Minimal misclassification, especially for cars.

6. 🧠 Feature Space Visualization

t-SNE Projection:

python

```
from sklearn.manifold import TSNE

encoded_tsne = TSNE(...).fit_transform(encoded_df)
```

- Cleanly separated clusters in 2D space.
- Confirmed that latent space learned by autoencoder captured **nonlinear separability** better than raw features.

Scatter Plot:

Each class formed a visually distinct cluster using true labels as hue.

7. 📊 Evaluation Metrics

Silhouette Scores:

- DBSCAN (raw): ~0.5–0.6
- DBSCAN (encoded): improved marginally
- KMeans (encoded): **0.4565**
- Agglomerative (encoded): close structural match with labels

Logistic Regression (Supervised Check)

Using the encoded features:

Accuracy: 83%

Class-wise F1:

- Bus: 0.83

- Car: 0.84

- Van: 0.79

8. ✅ Key Takeaways

- **Autoencoders are effective** at learning meaningful, compressed representations for clustering.
- **Agglomerative Clustering** performed the best when paired with autoencoded features.
- **t-SNE revealed clear separability**, validating that your encoder learned class-specific structure.

- Clustering accuracy was comparable to a **fully supervised classifier**.