-LEARNING TO LABEL – FROM CLUSTERING TO CLASSIFICATION

Venu Makka

Mridul Sharma

Akhilesh Sayana

Pawan Kumar Jha

CONTENTS

EXECUTIVE SUMMARY

- **Overview**

This project addresses a common real-world challenge in machine learning: building effective classification systems using **unlabeled data**.

The central objective is to develop a robust two-step pipeline that can **automatically generate labels** from raw datasets in the first step through unsupervised techniques **(clustering)** and subsequently train accurate **classifiers** based on those labels in the next step.

- **Context**

In many industry scenarios, clean, labeled datasets are scarce or expensive to obtain. This project simulates such conditions by providing two large, unlabeled datasets—one consisting of images (a subset of handwritten digits from the MNIST dataset) and another of text data(a set of 1500 text documents). The aim is to transform these datasets into labeled formats using intelligent machine learning methods, then build high-performing classifiers from the newly labeled data.

EXECUTIVE SUMMARY – TEXT DATASET

- Key steps:

  Preprocessing → Feature Extraction → Model Training → Evaluation.

- Data loading, cleaning, and preprocessing using techniques like tokenization, stopword removal, lemmatization and Stemming. We used symspellpy library to correct spelling mistakes in the data to keep consistency of words.
- Feature engineering

  Done using Count, TF-IDF, Word2Vec and Doc2Vec Vectorization methods.
- PCA

  Performed on all the vectors to determine the number of principal components required to capture 95% of cumulative sum variance. For Word2Vec, only two components were able to capture more than 95% of cumulative variance.
- Clustering Techniques

  Used two clustering techniques to evaluate the above Vectorization methods: KMeans and Agglomerative Clustering techniques.
- Classifier Training and Evaluation

  Multiple machine learning algorithms—Logistic Regression, Naive Bayes, Support Vector Machines, and Random Forest—were trained and evaluated using metrics such as accuracy, precision, recall, and F1-score.

EXECUTIVE SUMMARY – IMAGE DATASET

1. **Label Generation via Clustering:**
   Applied unsupervised learning techniques (e.g., k-means, DBSCAN, hierarchical clustering) with UMAP to identify patterns and groupings in the data, thereby generating provisional labels.
2. **Label Validation & Refinement:**
   Evaluate and refine the clustering-based labels using methods such as dimensionality reduction (e.g., PCA, t-SNE), cluster purity, or semi-supervised learning if applicable.
3. **Classifier Training:**
   With the generated labels, train supervised learning models (e.g., Logistic Regression, Random Forest, and XGBoost classifiers) to accurately classify new, unseen data.
4. **Performance Evaluation:**
   Compared models based on accuracy, precision, recall, and F1-score.

Expected Outcome:

A **fully functional classification system** built without pre-existing labels.

A documented methodology for **labeling unstructured data** that can be generalized to other domains.

Insights into the challenges and effectiveness of unsupervised and semi-supervised labeling techniques.

TEXT DATASET

1. DATA DESCRIPTION

- The data appears to be a list of raw text strings, likely email-style posts, containing informal language, email addresses, new line characters, tabs/unnecessary spaces. Eg:

```
In article <93089.204431GRV101@psuvm.psu.edu> Callec Dradja <GRV101@psuvm.psu.edu> writes:
>I am a bit nervous about posting this beacause it is begining to
>stray fron the topic of space but then again that doesn't seem to
>stop alot of other people. :-)
>
>With all of this talk about breathing at high pressures, I began
>to think about the movie Abyss. If you remember, in that movie one
>of the characters dove to great depths by wearing a suit that used
>a fluid that carries oxegen as opposed to some sort of gas.  Now I
```

```
In article <1993Apr19.131102.7843@rchland.ibm.com>
xzz0280@rchland.vnet.ibm.com (R. J. Traff) writes:
>|> In article <C5MuIw.AqC@mailer.cc.fsu.edu>
>|> dlecoint@garnet.acns.fsu.edu (Darius_Lecointe) writes:
>|> >question is "On what authority do we proclaim that the requirements of the
>|>               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
>|> >fourth commandment are no longer relevant to modern Christians?"  Please
>|> ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
>I don't believe most Christians believe they are *above* the Law.  However,
>we are not saved by adherence to the Law.  The Law exists to tell us what
```

2. PREPROCESSING TECHNIQUES

- Cleaning text using regular expression:
  - Remove new lines, emails, URLs, non-alphanumeric characters and extra white spaces.
  - Convert all text into lower case.
- Correct spelling mistakes using symspellpy:
  - Here we are using a frequency dictionary available online on github: frequency_dictionary_en_82_765.txt
- Considering **text only after** some keywords like " **writes**:" and " **wrote**:" as in the documents, acutal text is being written after these keywords.
- Now that the spelling has been corrected, we performe:
  - Lemmatizing
  - Tokenization
  - PorterStemmer
  - Removal of stop words.
- After processing text, **drop documents with less than 5 words.**

2. PREPROCESSING TECHNIQUES CONTD.

- Unprocessed data looks like this:

```
In article <93089.204431GRV101@psuvm.psu.edu> Callec Dradja <GRV101@psuvm.psu.edu> writes:
>I am a bit nervous about posting this beacause it is begining to
>stray fron the topic of space but then again that doesn't seem to
>stop alot of other people. :-)
>
>With all of this talk about breathing at high pressures, I began
>to think about the movie Abyss. If you remember, in that movie one
>of the characters dove to great depths by wearing a suit that used
```

- Processed data looks like below:

```
['bit nervou post begin stray topic space doesnt seem stop lot peopl talk breath high pressur began think movi abyss rememb movi
suit use fluid carri oxygen oppos sort ga heard mous breath fluid reason human unabl anyon know detail gregson aux believ reason
simpli stop breath minut vehicl readi go better put hold els rememb liquid sever time dens ga natur think depend ga liquid compar
 'question author proclaim requir fourth command longer relev modern christian pleas dont believ christian believ law howev save
law case refer christian jew believ sin aton blood sacrific anim describ old testament last time heard jewish anim sacrific chris
jesu blood sacrific innoc mani make law irrelev dont christian follow dont even follow ten command break law sin avoid sin way si
gandhi hello note im theologian gist sever sermon ive heard late bibl studi ive ever wonder someon perhap great deceiv pull leg',
 'get littl put heavi filter deep purpl front ordinari flashlight bulb brightest get father use setup like law enforc work circa
fluoresc bulb id proceed get cheap batterypow fluoresc light go electr suppli hous find bulb would fit michael covington associ r
program univers georgia phone athen georgia usa amateur radio ntmi'
```

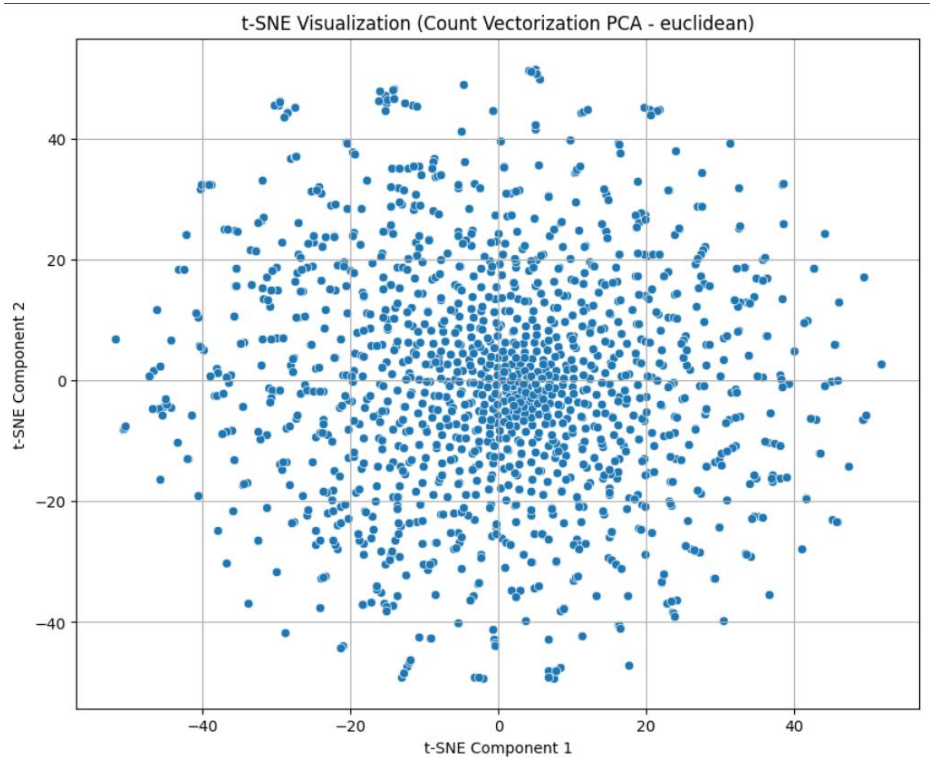
3. FEATURE ENGINEERING

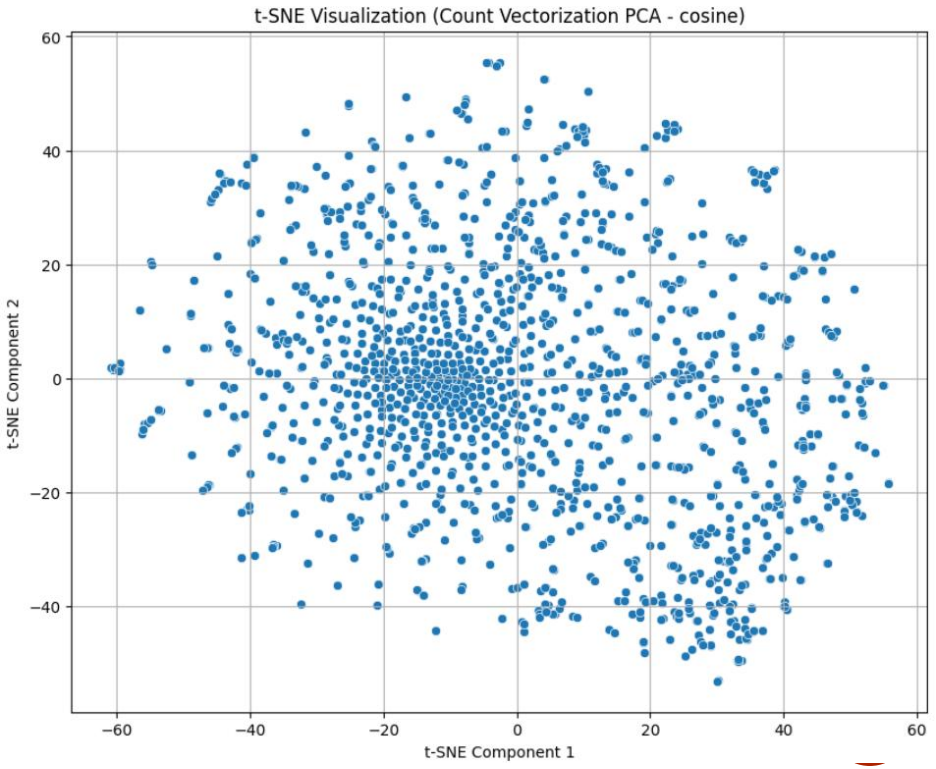
- Following techniques are compared for feature engineering:
  - Count Vectorization
  - TF-IDF Vectorization
  - Word2Vec Embedding
  - Doc2Vec Embedding
- Once all the documents are vectorized/embedded, we have used PCA to identify number of components required to capture 95% of the cumulative variance. Below are the results for the same:

```
Number of components for Count Vectorization PCA (95% variance): 507
Number of components for TF-IDF Vectorization PCA (95% variance): 1062
Number of components for Word2Vec PCA (95% variance): 2
Number of components for Doc2Vec PCA (95% variance): 90
```

4. VISUALIZE THE VECTORS BY PLOTTING PCA WITH THE HELP OF TSNE

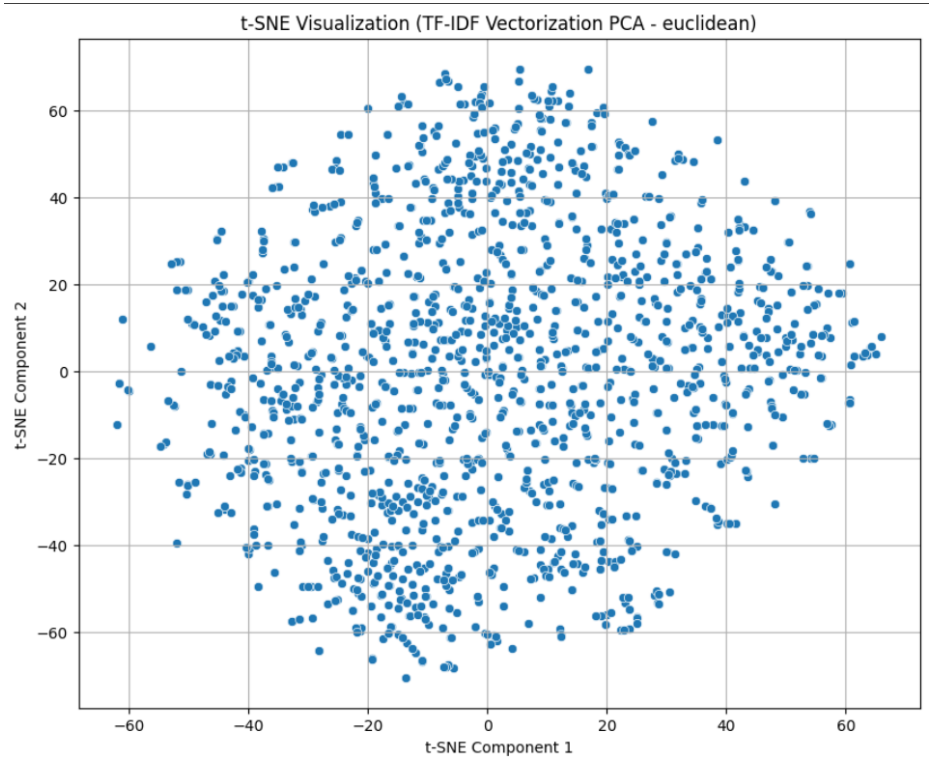
- Following are the tSNE visualization for Euclidean and Cosine distances for each of the components:
- 1. Count Vectorization:

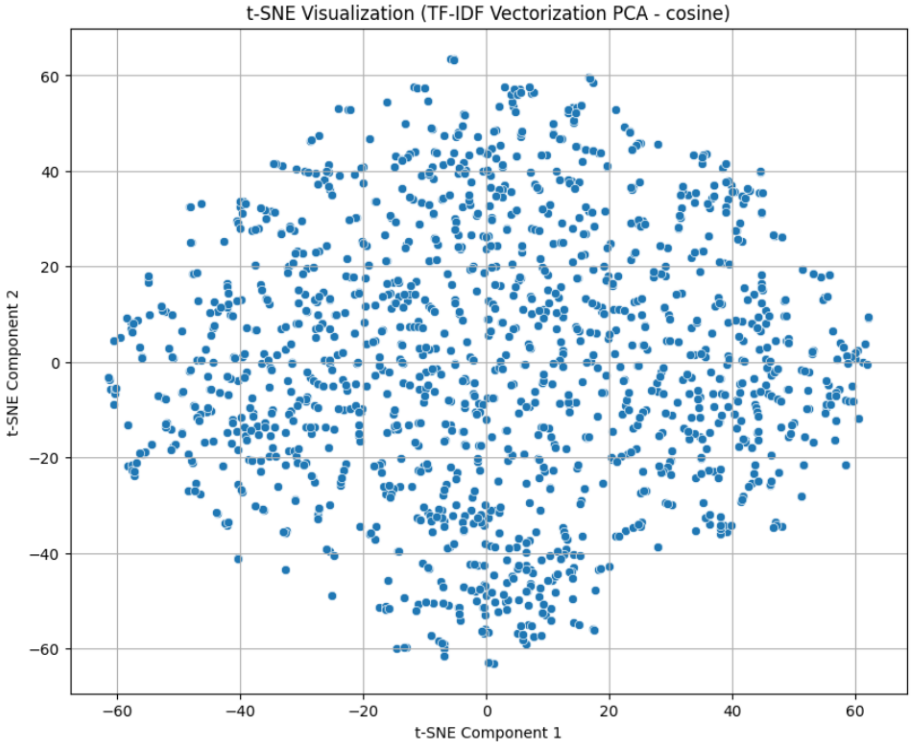




4. VISUALIZE THE VECTORS BY PLOTTING PCA WITH THE HELP OF TSNE

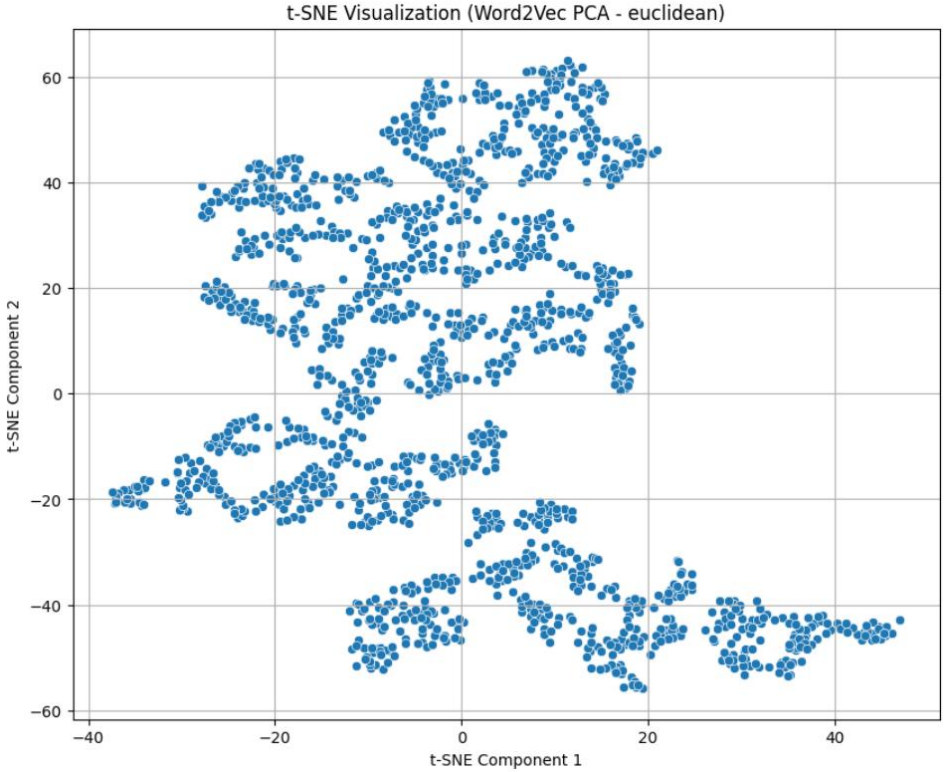
- 2. TF-IDF:

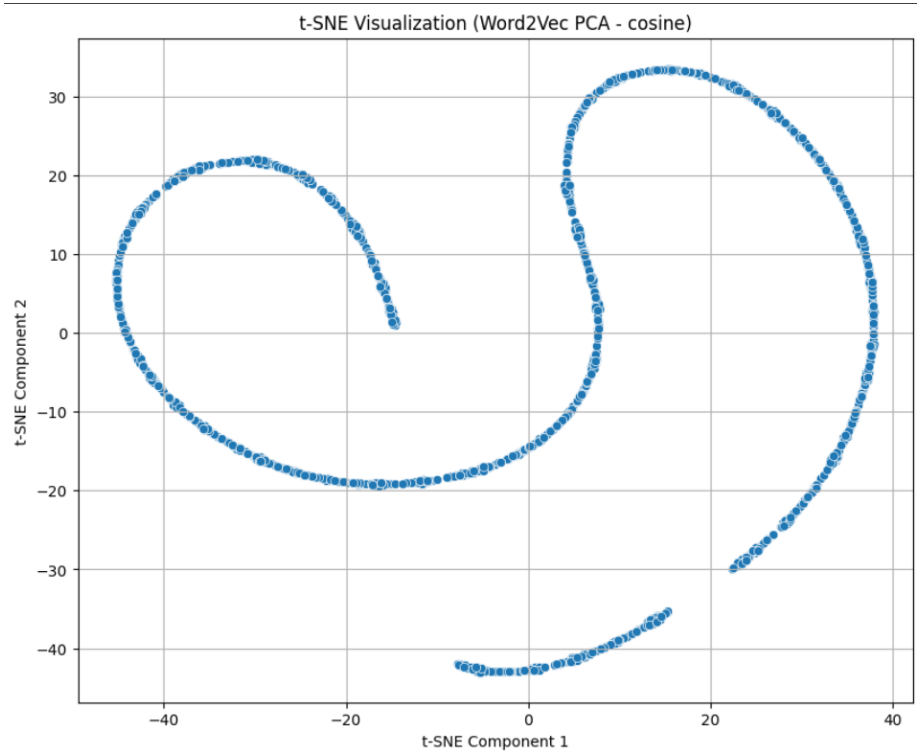




4. VISUALIZE THE VECTORS BY PLOTTING PCA WITH THE HELP OF TSNE

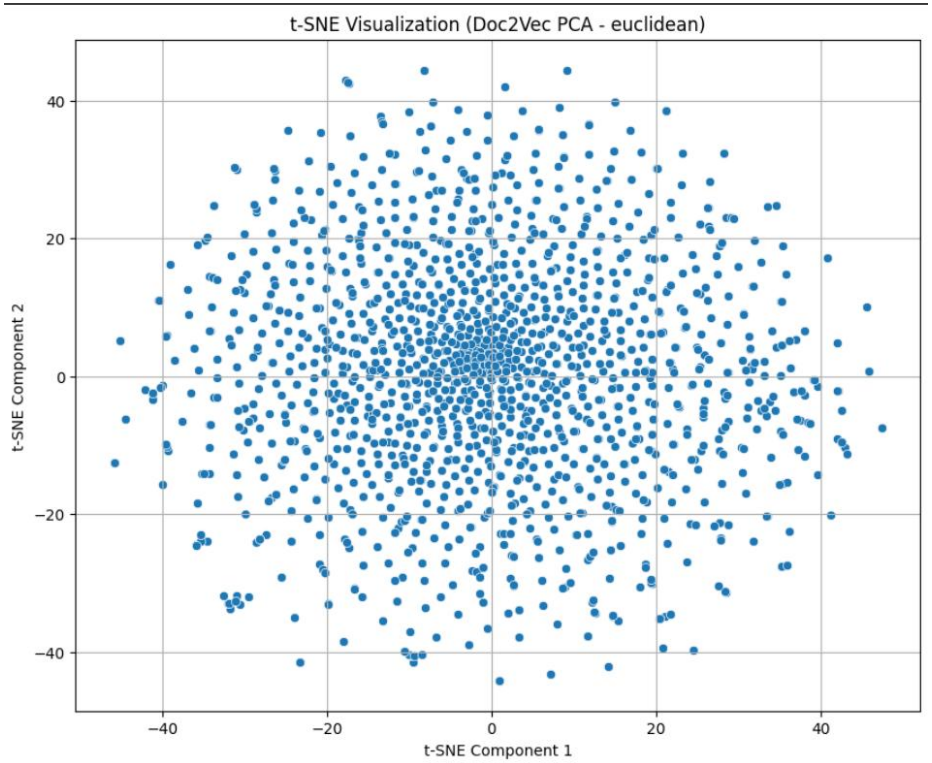
- 3. Word2Vec Embedding:

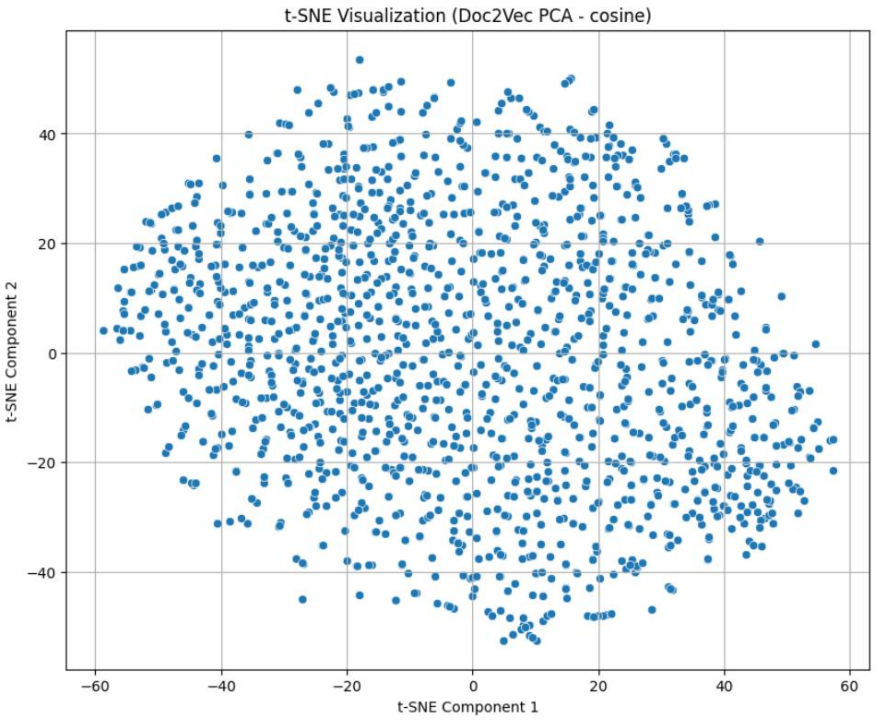




4. VISUALIZE THE VECTORS BY PLOTTING PCA WITH THE HELP OF TSNE

- 4. Doc2Vec Embedding:





4 VISUALIZE THE VECTORS BY PLOTTING PCA WITH THE HELP OF TSNE

- Interpretation:
  - Only Word2Vec is forming some sort of clusters/pattern when PCA are visualized.
  - The data points for Word2Vec cosine distance form a **non-linear, curved "S" or spiral shape.** This suggests that **t-SNE successfully captured non-linear relationships** in the high-dimensional word embeddings.
  - There's no obvious hard clustering, like tight balls of points: The embedding space is **continuous and non-linear**, not clearly clustered.
  - Rest of the Vectors does not form any pattern when visualized.

5. APPLY KMEANS AND AGGLOMERATIVE CLUSTERING TECHNIQUES

- We applied KMeans and Agglomerative clustering techniques and we can see the corresponding silhouette score. We can see that KMeans perform better with Word2Vec as compared to Agglomerative Clustering when their silhouette scores are compared(0.3488 vs 0.3102).

```
Applying KMeans to Count Vectorization PCA with n_clusters=5
Document counts per cluster:
  Cluster 0: 12
  Cluster 1: 13
  Cluster 2: 1
  Cluster 3: 1468
  Cluster 4: 1
Cannot calculate silhouette score: Only one cluster or too few

Applying KMeans to TF-IDF Vectorization PCA with n_clusters=5
Document counts per cluster:
  Cluster 0: 116
  Cluster 1: 166
  Cluster 2: 193
  Cluster 3: 235
  Cluster 4: 785
Silhouette Score: 0.0048

Applying KMeans to Word2Vec PCA with n_clusters=5
Document counts per cluster:
  Cluster 0: 141
  Cluster 1: 507
  Cluster 2: 219
  Cluster 3: 314
  Cluster 4: 314
Silhouette Score: 0.3488

Applying KMeans to Doc2Vec PCA with n_clusters=5
Document counts per cluster:
  Cluster 0: 2
  Cluster 1: 153
  Cluster 2: 977
  Cluster 3: 235
  Cluster 4: 128
Silhouette Score: 0.0833
```

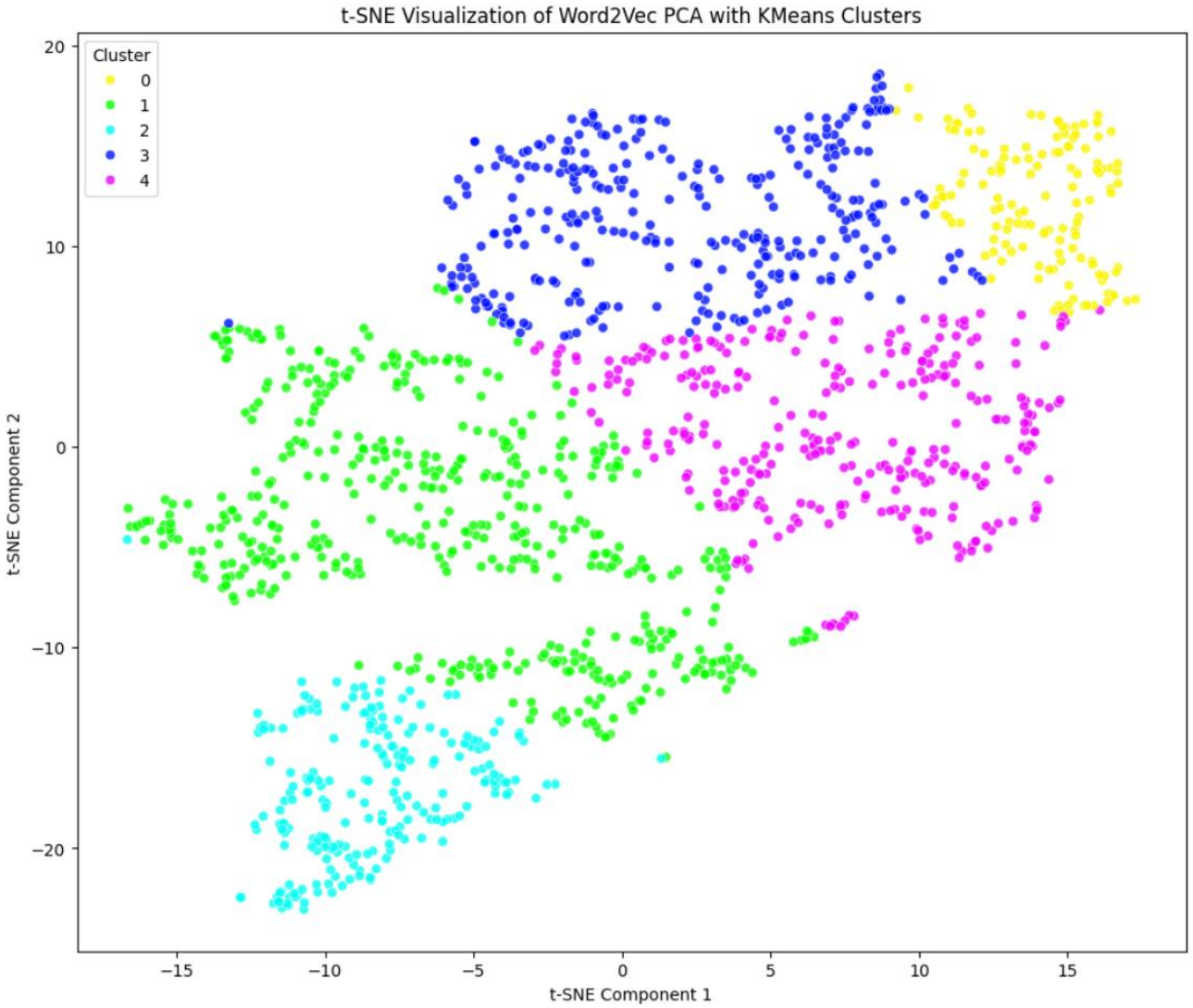
```
Applying AgglomerativeClustering to Count Vectorization PCA with n_cl
Document counts per cluster:
  Cluster 0: 682
  Cluster 1: 337
  Cluster 2: 203
  Cluster 3: 38
  Cluster 4: 235
Silhouette Score: -0.3136

Applying AgglomerativeClustering to TF-IDF Vectorization PCA with n_c
Document counts per cluster:
  Cluster 0: 682
  Cluster 1: 337
  Cluster 2: 203
  Cluster 3: 38
  Cluster 4: 235
Silhouette Score: -0.0033

Applying AgglomerativeClustering to Word2Vec PCA with n_clusters=5
Document counts per cluster:
  Cluster 0: 682
  Cluster 1: 337
  Cluster 2: 203
  Cluster 3: 38
  Cluster 4: 235
Silhouette Score: 0.3102

Applying AgglomerativeClustering to Doc2Vec PCA with n_clusters=5
Document counts per cluster:
  Cluster 0: 682
  Cluster 1: 337
  Cluster 2: 203
  Cluster 3: 38
  Cluster 4: 235
Silhouette Score: -0.1491
```

6. VISUALIZATION OF CLUSTERS IN COLORS



7. IDENTIFY LABELS FOR EACH CLUSTER

- KMeans algorithm has created 5 clusters numbered 0 to 4.
  We have retrieved 40 most common words per clusters:

```
Most frequent words in each cluster:

Cluster 0:
  Top 40 words: [('peopl', 165), ('think', 124), ('would', 123), ('ms', 118), ('myer', 118), ('dont', 115), ('say', 112), ('one', 112), ('work', 96), ('know', 95), ('moral', 91), ('go', 89), ('god', 87), ('jo

Cluster 1:
  Top 40 words: [('one', 413), ('would', 376), ('use', 355), ('space', 326), ('like', 287), ('get', 259), ('time', 242), ('go', 213), ('peopl', 209), ('new', 196), ('work', 186), ('know', 185), ('dont', 185),

Cluster 2:
  Top 40 words: [('one', 106), ('use', 80), ('like', 77), ('would', 77), ('get', 68), ('time', 64), ('go', 58), ('bike', 55), ('may', 55), ('make', 53), ('new', 52), ('know', 45), ('ride', 44), ('work', 42),

Cluster 3:
  Top 40 words: [('would', 319), ('one', 281), ('go', 274), ('use', 236), ('get', 227), ('govern', 202), ('like', 201), ('think', 196), ('work', 177), ('year', 175), ('know', 172), ('make', 167), ('peopl', 164

Cluster 4:
  Top 40 words: [('one', 409), ('peopl', 384), ('would', 355), ('dont', 332), ('god', 289), ('christian', 273), ('like', 272), ('jesu', 257), ('know', 254), ('say', 250), ('think', 227), ('get', 225), ('want',
```

- Thus we have identified below categories for each cluster:
  - 0: Social, Religious and Community discussion
  - 1: General and Technical Topics
  - 2: Motor and vehicle discussion
  - 3: Technology and Software
  - 4: Aerospace and Aviation

8. TRAINING MULTIPLE CLASSIFIERS

- We treated newly created clusters as our target variable and Word2Vec PCA as our variable.
- Data is split into 80:20 ratio and then we have trained below classifiers:
  - Logistic Regression
  - Decision Tree
  - K-Nearest Neighbors
  - Support Vector Machine
  - Naive Bayes
  - Random Forest
- Performance metric used are Accuracy, Precision, Recall and F1 score

8. TRAINING MULTIPLE CLASSIFIERS >> PERFORMANCE METRIC

Classifier	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.8261	0.8563	0.8261	0.8219
Decision Tree	0.9766	0.9766	0.9766	0.9765
K-Nearest Neighbors	0.9933	0.9935	0.9933	0.9933
Support Vector Machine	0.9967	0.9967	0.9967	0.9967
Naive Bayes	0.9732	0.9740	0.9732	0.9734
Random Forest	0.9900	0.9900	0.9900	0.9899

8. CLASSIFIERS PERFORMANCE

- Best Performing Model: **Support Vector Machine (SVM)**
  - **SVM achieved the highest scores** across all metrics (Accuracy, Precision, Recall, F1-Score: 0.9967).
- Other High Performers
  - **K-Nearest Neighbors (KNN)** and **Random Forest** also performed exceptionally well (F1-scores of 0.9933 and 0.9899 respectively).
- **Decision Tree**
  - Though it performed well (F1-score: 0.9765), it's often prone to **overfitting**, which should be tested further on unseen data.
- **Logistic Regression**
  - This model had **moderate performance** (F1-score: 0.8219).

9. TRAINING CLASSIFIERS ON MANUALLY UPDATED TARGET VARIABLES

- We have updated a few of the target variables to match with the clusters we have defined earlier, let us see what performance difference does it make. (manually updated file: clustering_results_word2vec_pca_manual_update.csv)

9. TRAINING CLASSIFIERS ON MANUALLY UPDATED TARGET VARIABLES

Classifier	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.7960	0.8273	0.7960	0.7868
Decision Tree	0.8930	0.8966	0.8930	0.8930
K-Nearest Neighbors	0.9331	0.9351	0.9331	0.9326
Support Vector Machine	0.9465	0.9490	0.9465	0.9454
Naive Bayes	0.9030	0.9118	0.9030	0.9023
Random Forest	0.9398	0.9419	0.9398	0.9392

9. TRAINING CLASSIFIERS ON MANUALLY UPDATED TARGET VARIABLES

- Manually updating the target variable has reduced the classifiers performance by upto 6%.
- **Top Performer: Support Vector Machine (SVM)**
  - Continues to outperform all other models with **F1-Score: 0.9454**.
  - **Random Forest (F1: 0.9392)** and **KNN (F1: 0.9326)** also show **strong adaptability** to updated data.
- **Naive Bayes** and **Decision Tree** show solid but comparatively lower performance.
- **Lowest Performer**
  - **Logistic Regression's drop in F1-score (to 0.7868)** suggests it may struggle with more complex patterns.

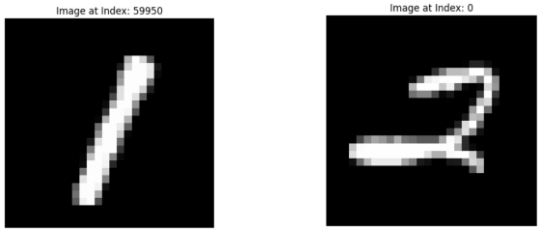
FINAL SUMMARY – TEXT DATASET

- **Support Vector Machine (SVM)** gave the best performance across all metrics (accuracy, precision, recall, and F1-score)
- **K-Nearest Neighbors (KNN)** and **Random Forest** also performed exceptionally well.
- **Logistic Regression** struggle with complex patterns.
- **Naive Bayes** and **Decision Tree** show solid but comparatively lower performance.
- The labeling approach using Word2Vec + KMeans clustering proved to be a highly effective self-supervised method for bootstrapping labels from unlabelled data.

IMAGE DATASET

PRE-PROCESSING

- The dataset contains 60000 images each with 3-dimensions and shape 1 x 28 x 28. The first dimension is the value 1 representing the image to be a grayscale with 28 pixels high and 28 pixels wide.



- Each of the images are then flattened and min-max scaled with the resulting image feature vector with dimension 1 x 784

```
# Remove singleton dimension (channels) and flatten images
X = X.squeeze()
n_samples, height, width = X.shape
X_flattened = X.reshape(n_samples, height * width)
print("Flattened shape:", X_flattened.shape)

# Normalize data to [0,1] - better for ML algorithms
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X_flattened)
```

DIMENSIONALITY REDUCTION

Dimensionality Reduction

- **What is PCA?**
  Principal Component Analysis, is a statistical method used to reduce the dimensionality of a dataset while preserving as much of the original data's variance as possible. It transforms a large set of variables into a smaller set of new, orthogonal variables called principal components. These capture the most significant variations in the original data.

- **Purpose of PCA**
  Dimensionality Reduction: PCA aims to reduce the number of variables in a dataset without losing much information. This is useful for simplifying data, improving visualization, and speeding up machine learning models.

  Data Compression: By identifying the principal components, PCA effectively compresses the data while preserving the most important patterns.

  Data Exploration: PCA helps in understanding the underlying structure of the data and identifying relationships between variables.

Dimensionality Reduction

- **How PCA works?**

  1. Standardization: Ensures each variable has mean 0 and standard deviation of 1.

  2. Covariance matrix calculation: PCA calculates the covariance matrix of the standardized data, which represents the relationships between the variables.

  3. Eigenvalue and Eigenvector Calculation: The covariance matrix is then used to find its eigen values and eigenvectors.

  4. Principal Component Selection: The eigenvectors corresponding to the highest eigenvalues are selected as the principal components. These eigenvectors represent the directions of maximum variance in the data.

  5. Data Transformation: The original data is transformed into the new coordinate system defined by the principal components.

Dimensionality Reduction

- **Why PCA is chosen?**

1. Simplified Data: Reduced dimensionality makes it easier to visualize and analyze high-dimensional data.

2. Improved Model Performance: By reducing irrelevant features, PCA can improve the performance of machine learning models hence in this use-case improving the quality of the resulting clusters.

3. Feature Selection: PCA can be used to identify the most important features in a dataset.

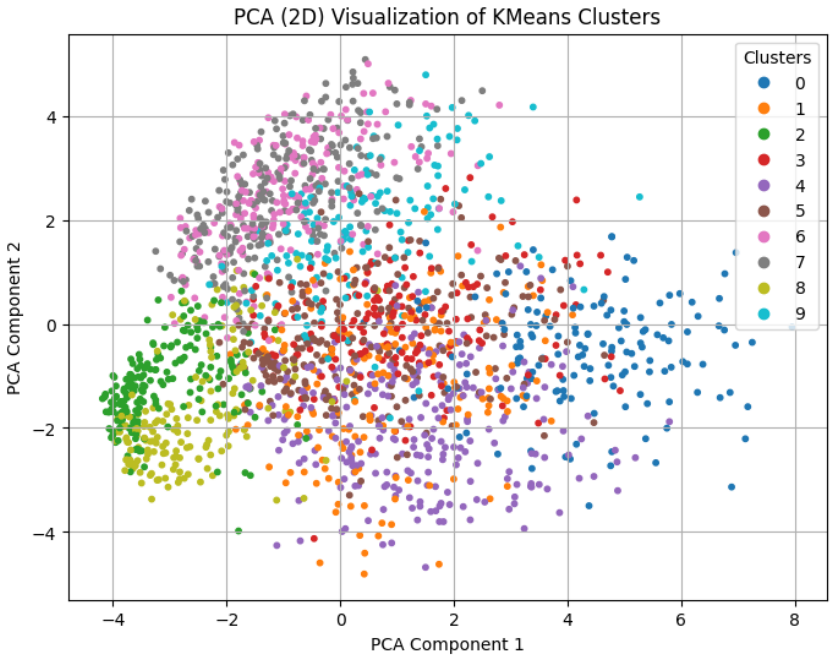
CLUSTERING

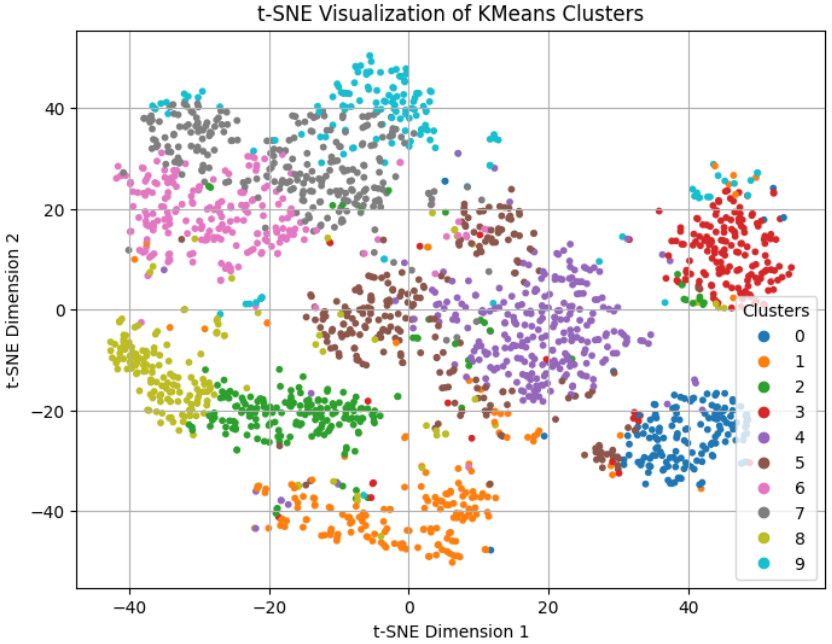
K-Means Clustering

- **What is K-Means Clustering?**
  K-means clustering is an unsupervised machine learning algorithm used to group similar data points into clusters. It aims to partition a dataset into a specified number of clusters, denoted as 'k', where each data point belongs to the cluster with the nearest mean (centroid). The algorithm iteratively minimizes the sum of distances between data points and their respective cluster centroids.

- **Key Characteristics of K-Means Clustering:**
  1. **Unsupervised:** It doesn't require labeled data; it discovers patterns within the data itself.
  2. **Iterative:** It repeatedly refines the cluster assignments and centroid positions.
  3. **Centroid-based:** Each cluster is represented by its centroid, which is the mean of the data points in that cluster.
  4. **Distance-based:** The algorithm relies on a distance metric to determine the closeness of data points to centroids.
  5. **Requires 'k':** The number of clusters (k) needs to be specified beforehand.

Cluster Visualization





- PCA visualization: clusters are overlapping significantly, suggesting PCA alone isn't effective in clearly separating the digits.
- t-SNE visualization: excellent clear separation among clusters, indicating that data contains well-separated groups if processed appropriately.

GMM Clustering

- **What is GMM Clustering?**
  Gaussian Mixture Model is a probabilistic model that assumes all data points are generated from a mixture of several Gaussian distributions with unknown parameters. Unlike hard clustering methods like K-Means that assigns each data point to one cluster based on the closest centroid, GMMs perform soft clustering meaning each data point belongs to multiple clusters with certain probabilities.

The probability of data point coming from Gaussian cluster k is expressed as

$$P(z_n = k \mid x_n) = \frac{\pi_k \cdot \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n|\mu_k,\Sigma_k)}$$

where:

•z_n=k is a latent variable indicating which Gaussian the point belongs to.

•π_k is the mixing probability of the k-th Gaussian.

•N(z_n= x_n | μ_k, Σ_k) is the Gaussian distribution with mean μkμk and covariance Σk

Agglomerative Clustering

- **What is Agglomerative Clustering?**
  Agglomerative clustering is a hierarchical clustering technique that follows a bottom-up approach, starting with each data point as its own cluster and iteratively merging the most similar clusters until a single cluster containing all data points is formed. This process builds a hierarchy of clusters, allowing for a nuanced exploration of relationships within the data.

- **Advantages:**
  1. Agglomerative clustering doesn't require a pre-defined number of clusters.
  2. It can identify clusters of varying shapes and sizes.
  3. It's relatively easy to visualize the clustering process using a dendrogram.

- **Disadvantages:**
  1. It can be computationally expensive for large datasets
  2. It's not suitable for datasets with many outliers
  3. The choice of linkage method can significantly impact the results.

DBSCAN Clustering

- **What is DBSCAN?**

  DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an ML algorithm used for clustering data points based on their density. Unlike some algorithms that require specifying the number of clusters beforehand, DBSCAN automatically identifies clusters of arbitrary shapes and sizes, and can also detect outliers (noise). It identifies core points (centers of dense regions), then expands clusters by recursively finding all the points that are directly density-reachable from a core point and are also core points themselves and finally points that are not part of any cluster are marked as noise.

- **Advantages:**

  1. It can identify clusters of arbitrary shape.
  2. It can effectively identify and remove outliers(noise) from the dataset.
  3. It doesn't require specifying the number of clusters, making it useful for EDA.

- **Disadvantages:**

  1. The performance of DBSCAN is sensitive to the choice of epsilon and MinPts parameters.
  2. It may struggle with datasets where the density varies significantly across different regions.
  3. DBSCAN can be computationally expensive for very large datasets.

CLUSTERING EVALUATION

Evaluation Metric

- **What metric is used for selecting the clustering approach?**
  Silhouette score is used as a metric to select the clustering approach.
- **What is Silhouette Score?**
  The Silhouette Score is a metric used to evaluate the quality of clustering, specifically how well data points are grouped into clusters. It measures the similarity of each data point to its own cluster compared to other clusters, with values ranging from -1 to 1.

- **How it works:**
  1. **Cohesion** (a): Measures how close a data point is to other points in the same cluster.
  2. **Separation** (b): Measures how far a data point is from the points in the nearest cluster.
  3. Silhouette Score: Calculated as (b - a) / max(a, b).

- **Interpretation:**
  1. Higher average Silhouette Score indicates better clustering quality.
  2. Negative values suggest that the sample might have been assigned to the wrong cluster.
  3. Values around 0 indicate that the sample is on or very close to the decision boundary between two clusters while values near +1 indicate a sample well-matched to its own cluster.

Clustering Results

- **Clustering Results with multiple setups.**

Setup	n_components	Silhouette Score
GMM with PCA	50	0.0404
K-Means with PCA	50	0.0818
K-Means with PCA	100	0.0676
Agglomerative Clustering with PCA	50	0.0503
DBSCAN with PCA	50	-0.2634
Kmeans t-SNE embeddings	2	0.4632

Conclusion:

•t-SNE embeddings + K-Means clustering gave clearly the best performance.

•PCA (50) + K-Means was the next best option but significantly lower than t-SNE embeddings.

•Since the t-SNE embeddings + K-Means method performed best by far, let's proceed with this method for creating your manual labeling set.

•It should be able to find to t-SNE here as our primary intent is to label the data.

CLUSTERING APPROACH SELECTION

UMAP

Based on the evaluation in the previous slides t-SNE embeddings + K-Means clustering gave clearly the best performance.
However, running t-SNE on 60,000 samples directly will take a long time. To overcome this, we can use a faster but equally effective alternative: UMAP.

- **What is UMAP?**
  UMAP (Uniform Manifold Approximation and Projection) is a dimension reduction technique used in machine learning and data visualization. It helps to visualize high-dimensional data in a lower dimension (typically 2 or 3) while preserving the global structure of the data.

- **Why UMAP?**
  - Similar visual quality as t-SNE
  - Scalable to large datasets (60k samples feasible on your hardware)
  - Produces embeddings quickly and clearly separates clusters (like t-SNE)

UMAP

Setup	n_components	Silhouette Score
GMM with PCA	50	0.0404
K-Means with PCA	50	0.0818
K-Means with PCA	100	0.0676
Agglomerative Clustering with PCA	50	0.0503
DBSCAN with PCA	50	-0.2634
Kmeans t-SNE embeddings	2	0.4632
K-Means with UMAP	2	0.619

UMAP Resulted better clustering with silhouette score of **0.6190**

Few Observations

While manually labelling the data with clustering outputs:

- The cluster corresponding to the digit '4' contains a few samples that resemble the digit '9'.
- The cluster assigned to the digit '9' includes some images that visually resemble the digit '7'.
- There is a cluster where the symbols appear more like a '/' or are ambiguous—not clearly any digit from 0 to 9.
- Since a distinct cluster for the digit '7' has not formed elsewhere and the shapes in this cluster are closer to '7', I have assigned this cluster to label '7'.

CLASSIFICATION

Classification Approach

We have trained and evaluated multiple ML classifiers (Logistic Regression, Random Forest, XGBoost) on your labeled data and compare their performance using accuracy, precision, recall, and F1-score.

Step-by-step Plan:

- Load the labeled Excel file with True_Label column
- Map the index to original image data using the Index column
- Split the dataset into train and test sets
- Train classifiers (Logistic Regression, Random Forest, XGBoost)
- Evaluate models using accuracy, precision, recall, and F1-score
- Generate observations from model metrics

Final Summary

```
--- Logistic Regression ---
              precision    recall  f1-score   support

           0     0.9688    0.9631    0.9660      1194
           1     0.9610    0.9623    0.9617       717
           2     0.9297    0.9182    0.9239      1137
           3     0.9112    0.9195    0.9153      1305
           4     0.9380    0.9411    0.9396      2090
           5     0.8984    0.8882    0.8932      1055
           6     0.9527    0.9511    0.9519      1227
           7     0.9729    0.9531    0.9629       640
           8     0.8922    0.9058    0.8990      1051
           9     0.9273    0.9343    0.9308      1584

    accuracy                         0.9331     12000
   macro avg     0.9352    0.9337    0.9344     12000
weighted avg     0.9332    0.9331    0.9331     12000
```

Final Summary

```
--- Random Forest ---
              precision    recall  f1-score   support

           0     0.9833    0.9891    0.9862      1194
           1     0.9772    0.9568    0.9669       717
           2     0.9579    0.9815    0.9696      1137
           3     0.9611    0.9663    0.9637      1305
           4     0.9564    0.9766    0.9664      2090
           5     0.9681    0.9488    0.9584      1055
           6     0.9901    0.9829    0.9865      1227
           7     0.9902    0.9469    0.9681       640
           8     0.9493    0.9619    0.9556      1051
           9     0.9685    0.9520    0.9602      1584

    accuracy                         0.9681     12000
   macro avg     0.9702    0.9663    0.9681     12000
weighted avg     0.9682    0.9681    0.9681     12000
```

Final Summary

--- XGBoost ---

	precision	recall	f1-score	support
0	0.9891	0.9891	0.9891	1194
1	0.9776	0.9721	0.9748	717
2	0.9764	0.9833	0.9798	1137
3	0.9688	0.9739	0.9713	1305
4	0.9727	0.9732	0.9730	2090
5	0.9702	0.9564	0.9632	1055
6	0.9861	0.9861	0.9861	1227
7	0.9854	0.9484	0.9666	640
8	0.9546	0.9800	0.9671	1051
9	0.9678	0.9672	0.9675	1584
accuracy			0.9741	12000
macro avg	0.9749	0.9730	0.9739	12000
weighted avg	0.9741	0.9741	0.9741	12000

Final Summary

Observations and Analysis

- XGBoost gave the best performance across all metrics (accuracy, precision, recall, and F1-score), achieving 97.41% accuracy. It handled class boundaries and subtle differences between digits very well, even with just flattened pixel features.
- Random Forest performed nearly as well as XGBoost, with a strong 96.81% accuracy. It's a good non-linear model for this task and was fast to train with no special installation requirements.
- Logistic Regression gave respectable results (93.31% accuracy), but struggled slightly with more complex digit shapes like '5', '8', and '3'. This suggests linear models are less effective for handwritten digit classification without deeper features.
- Class-wise performance was balanced for all three models—there were no major drops in any specific digit class, which validates the quality of manual labeling and the consistency of the dataset.
- The labeling approach using UMAP + KMeans clustering proved to be a highly effective self-supervised method for bootstrapping labels from unlabelled data. The high accuracy from all models confirms that the cluster-to-label mapping was meaningful.

THANK YOU