Analysis of Project 3 Machine Learning

By Megh Raval and Shubh Patel

Distribution of Work:

Each of us selected a dataset we were interested in. Then we wrote the code for the datasets, but each of us applied alternate algorithms on the dataset chosen by the other group member to ensure that we understood both datasets and contributed to the code equally. So, we were able to implement each algorithm once in either dataset. Preprocessing and visualizations were also handled in the same way, so if one of us did preprocessing and EDA for one dataset the other one did it for the second one so that we could learn all the steps. This kind of work distribution, albeit a little inconvenient helped us learn all the steps while being familiar with both the datasets. Both of us wrote up all the steps we did while doing it and then merged it into this writeup. So, this writeup is also a combined effort with titbits of work done by the both of us.

Introduction

In this report, we have presented the analysis of Clustering (Kmeans and Hierarchical) and Dimensionality Reduction(PCA and UMAP) applied to two different datasets after which we have used 2 ensemble methods (AdaBoost and Random Forests). The primary objective is to evaluate and analyze each step and evaluate their effectiveness. The two datasets under consideration are as follows:

1. **Dataset 1**: Patient Survival Detection

Link for Dataset 1: https://www.kaggle.com/datasets/mitishaagarwal/patient/data

The dataset, sourced from Kaggle, comprises patient survival data featuring 85 features and around 131,000 data points. Its extensive feature set and ample sample size make it conducive for dimensionality reduction exploration due to its complexity. It includes diverse patient attributes such as ethnicity, gender, age, and various clinical measurements.
The classification task aims to predict hospital outcomes, with "hospital_death" as the binary target variable. A value of 1 indicates patient mortality during hospitalization, while 0 signifies survival. This dataset enables analysis to forecast patient survival based on provided features, aiding healthcare decisions and treatment strategies. With its rich data, it offers valuable insights into predictive modeling in healthcare and factors influencing patient outcomes during hospitalization.

2. **Dataset 2**: Customer Segmentation Dataset

Link to dataset: https://www.kaggle.com/datasets/kaushiksuresh147/customer-segmentation?rvi=1&select=Train.csv

We sourced this dataset from Kaggle, provided by an automobile company planning market expansions. Named "Train.csv," it holds customer attributes for segmentation prediction—targeted as A, B, C, or D segments—making it a multiclass classification task.
Motivation: Our interest stemmed from its potential for both clustering and supervised learning due to its customer segmentation focus with labeled segments.
Dataset Overview: It encompasses diverse customer attributes like demographics, marital status, age, education, profession, work experience, spending score, family size, and an anonymized category variable. The target variable "Segmentation" denotes the customer segment.

EDA and Preprocessing

Dataset 1:

Exploratory Data Analysis and Preprocessing
Upon loading the dataset and conducting initial exploration, we performed several preprocessing steps to prepare the data for further analysis:

1. **Data Cleaning:**
   - We dropped irrelevant columns such as "encounter_id", "patient_id", "hospital_id", and "Unnamed: 83" as they did not contribute to the analysis.
   - We filtered the dataset to include only patients of Caucasian ethnicity for the sake of this analysis (To reduce the number of datapoints – 77% of the total data included)
2. **Handling Missing Values:**
   - For numerical features, we imputed missing values using the median of each column to maintain robustness against outliers.
   - Categorical features were imputed using the mode (most frequent value) of each column to preserve the distribution of categorical data.
3. **Encoding Categorical Features:**
   - We employed count encoding from the **category_encoders** library to encode categorical features, converting them into numerical representations suitable for machine learning algorithms.
4. **Data Sampling:**
   - To reduce computational overhead and optimize memory usage, we sampled a fraction (20%) of the dataset for analysis.
5. **Feature Scaling:**
   - We utilized RobustScaler from **sklearn.preprocessing** to scale the numerical features, ensuring that they have similar scales and minimizing the influence of outliers.

Dataset 2:

Exploratory Data Analysis and Preprocessing

1. **Handling Missing Values:**
   - Missing values in columns such as 'Ever_Married', 'Work_Experience', 'Family_Size', 'Graduated', and 'Profession' were imputed using the mode (most frequent value) of each respective column to ensure data completeness.
2. **Encoding Categorical Features:**
   - Categorical features were transformed into numerical representations using label encoding.
   - 'Gender', 'Ever_Married', and 'Graduated' were encoded as binary values (1 for 'Yes', 0 for 'No').
   - 'Spending_Score' was label encoded using a numeric scale.
   - 'Profession' was label encoded to represent different professions numerically.
3. **Drop Unnecessary Columns:**
   - The 'ID' column was dropped as it does not contribute to the analysis.
   - 'Var_1' column was also dropped.
4. **Exploring Target Variable Distribution:**
   - The distribution of the target variable 'Segmentation' was visualized using a count plot to understand the distribution of segments within the dataset.
5. **Splitting Data and Feature Scaling:**
   - The dataset was split into features (X) and the target variable (y_encoded) after label encoding.

- RobustScaler from **sklearn.preprocessing** was applied to scale the numerical features to ensure consistency and robustness in the presence of outliers.

Dimensionality Reduction

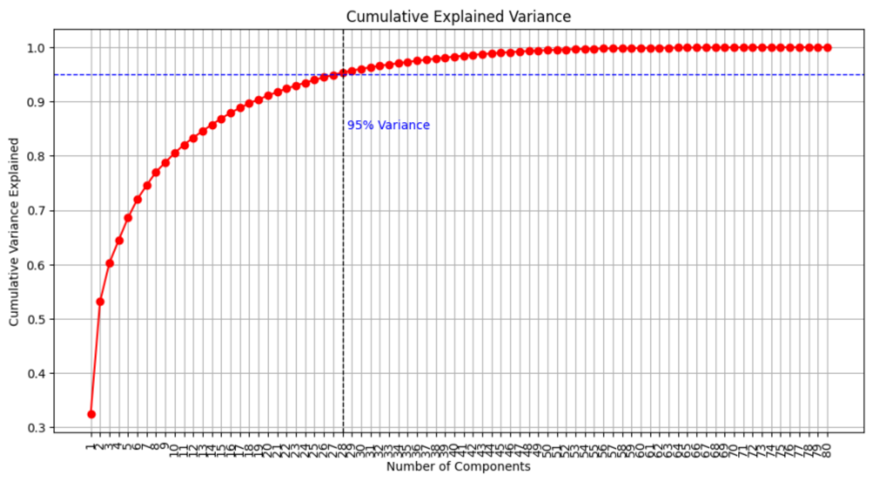
Dataset 1:

Steps:

1. **Principal Component Analysis (PCA):**
   - We used PCA to condense our dataset's dimensionality by transforming features into principal components capturing maximum variance.
   - Initially, we created a PCA object without specifying the number of components.
   - Through an elbow plot, we visualized explained variance ratios to determine the optimal number of components, considering a 95% threshold for cumulative explained variance.
   - A new PCA object with the chosen components was then created and applied to the scaled training data.
2. **Uniform Manifold Approximation and Projection (UMAP):**
   - UMAP, offering nonlinear manifold learning, was explored as an alternative dimensionality reduction method.
   - We defined a parameter grid for manual tuning, experimenting with combinations of hyperparameters like **n_neighbors** and **min_dist**.
   - Each parameter combination was trained on the scaled training data, and trustworthiness scores were evaluated to identify the best parameters.
   - Finally, the optimal UMAP model was used to transform both training and test sets for reduced-dimensional representations.

Analyses:

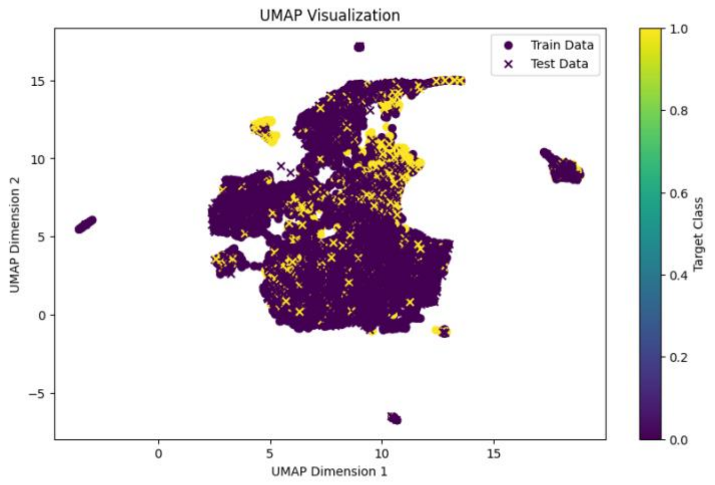
1. **Principal Component Analysis (PCA):**
   - Upon applying PCA to our dataset, we found that 28 components were required to explain 95% of the total variance present in the data. This suggests that the original high-dimensional feature space could be effectively represented using a reduced set of principal components.



   - By visualizing the scree plot of explained variance ratio, we observed a steep decline in the explained variance after the first few components, indicating that these components capture the majority of the variance within the data.

- The reduced-dimensional representation obtained through PCA facilitates enhanced interpretability and visualization of the dataset, enabling us to focus on the most relevant features for analysis and modeling.

2. **Uniform Manifold Approximation and Projection (UMAP):**
   - Through manual tuning of UMAP hyperparameters, we identified the optimal parameter configuration yielding the highest trustworthiness score of 0.877.
   - The best UMAP parameters include a minimum distance (**min_dist**) of 0.25 and a number of neighbors (**n_neighbors**) set to 5. These parameters influence the local connectivity of the UMAP embedding and affect the preservation of the neighborhood structure in the reduced-dimensional space.
   - Despite reducing the dimensionality to only 2 components, UMAP achieves a high level of trustworthiness, indicating its effectiveness in preserving the intrinsic structure of the dataset in a lower-dimensional space.



Clustering (Kmeans And Agglomerative)

Steps:

In our exploration of clustering algorithms, namely K-means and Hierarchical Clustering, we conducted tuning efforts to optimize their performance and enhance the quality of clustering results.

1. **K-means Clustering:**
   - **Number of Clusters (n_clusters):**
     - Explored values ranging from 3 to 5.
     - Utilized silhouette score to assess cluster quality and determine the optimal number of clusters for each dataset variant.
     - Iteratively evaluated different cluster counts to maximize the silhouette score, indicating better cluster separation.
2. **Agglomerative Clustering:**
   - **Number of Clusters (n_clusters) and Linkage Method (linkage):**
     - Evaluated cluster counts from 3 to 5.
     - Examined different linkage methods including 'ward' and 'single'.
     - Utilized silhouette score to determine the best combination of cluster count and linkage method.
     - Iteratively tuned both parameters to enhance the clustering performance across original, PCA-reduced, and UMAP-reduced datasets.

Analyses and Results:

Silhouette Scores:

	KMeans	Agglomerative
Original	0.524437	0.69832
PCA Reduced	0.53921	0.708846
UMAP Reduced	0.448763	0.459891

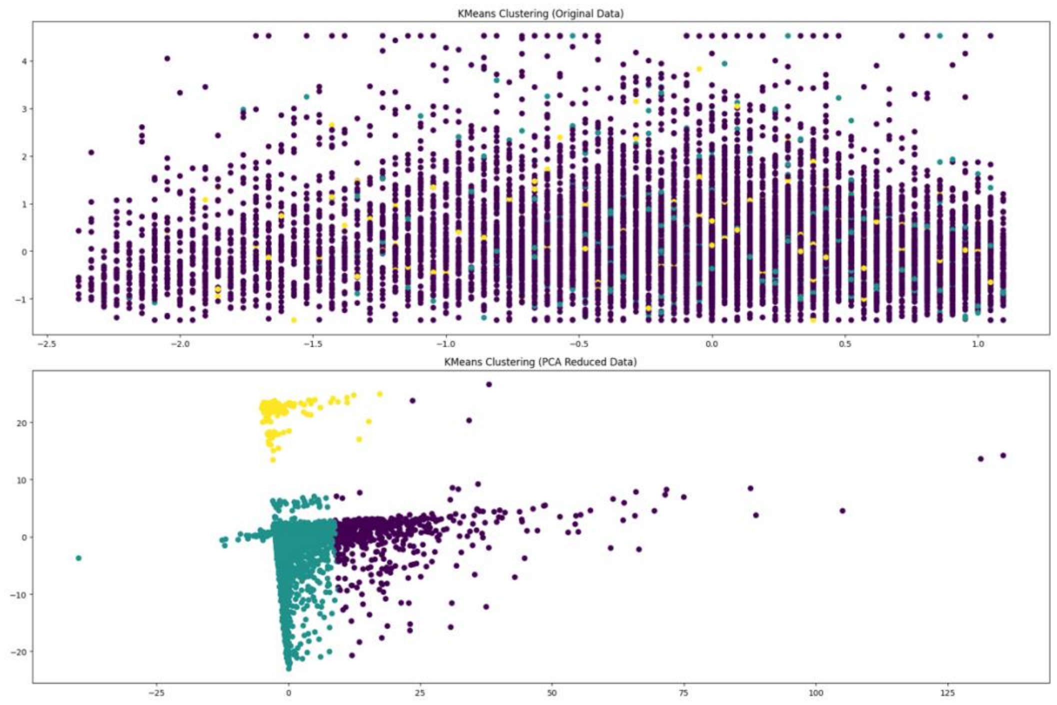
Best number of clusters for KMeans:

Original: 3
PCA Reduced: 3
UMAP Reduced: 3

Best number of clusters for Agglomerative Clustering:

Original: 3 clusters with linkage single
PCA Reduced: 3 clusters with linkage single
UMAP Reduced: 5 clusters with linkage ward

1. **KMeans vs. Agglomerative Clustering:**
    - Across all dataset variants (Original, PCA Reduced, UMAP Reduced), we observed that Agglomerative Clustering consistently outperformed KMeans in terms of silhouette scores.
    - It's interesting to note that Agglomerative Clustering achieved higher silhouette scores, indicating better cluster quality and separation compared to KMeans.
2. **Effect of Dimensionality Reduction:**
    - When we looked at both PCA Reduced and UMAP Reduced datasets, we noticed improved silhouette scores compared to the Original dataset for both clustering algorithms.
    - This suggests that our dimensionality reduction techniques contributed to enhancing cluster quality by capturing essential features and reducing noise.
3. **Impact of Clustering Algorithm on Dimensionality Reduction Techniques:**
    - For the Original dataset, Agglomerative Clustering achieved notably higher silhouette scores than KMeans, indicating its effectiveness in capturing complex data structures.
    - However, for the PCA Reduced and UMAP Reduced datasets, we observed that the performance gap between KMeans and Agglomerative Clustering narrowed, suggesting that dimensionality reduction mitigated the advantage of Agglomerative Clustering to some extent.
4. **Optimal Number of Clusters:**
    - Regardless of the data's dimensionality or the clustering algorithm used, we found that the optimal number of clusters was consistently identified as 3 for all dataset variants.
    - This implies that our dataset tends to exhibit three distinct clusters, regardless of whether we've applied dimensionality reduction or not.
5. **Effect of Linkage Method in Agglomerative Clustering:**
    - We also noticed that the choice of linkage method in Agglomerative Clustering had a notable impact on clustering performance.
    - Specifically, the 'single' linkage method consistently yielded better silhouette scores compared to 'ward' for both the Original and PCA Reduced datasets.
    - However, for the UMAP Reduced dataset, we observed that the 'ward' linkage method with 5 clusters outperformed 'single' linkage with 3 clusters, indicating different data structures and cluster formations in the reduced dimensional space.



The above image shows how kmeans clustering along with PCA uncovers the hidden patterns in data, segregating different clusters. We obtained this using 2 features from our dataset.

Ensemble Methods:

1. **AdaBoost (Adaptive Boosting)**:
   - Utilized AdaBoostClassifier from scikit-learn.
   - Defined hyperparameter grids for tuning, including the number of estimators (n_estimators) and learning rate (learning_rate).
   - Employed GridSearchCV for hyperparameter tuning with 3-fold cross-validation.
   - Evaluated the best AdaBoost model's performance using accuracy as the scoring metric.
2. **Random Forest**:
   - Employed RandomForestClassifier from scikit-learn.
   - Specified hyperparameter grids for tuning, encompassing the number of estimators (n_estimators) and maximum depth of trees (max_depth).
   - Utilized GridSearchCV for hyperparameter optimization with 3-fold cross-validation.
   - Assessed the accuracy of the best Random Forest model to evaluate its performance.

- **AdaBoost and Random Forest Hyperparameter Tuning**:
  - We experimented with different combinations of hyperparameters for AdaBoost and Random Forest to optimize their performance.
  - For AdaBoost, we varied the number of estimators (50, 100, and 150) and the learning rate (0.5 and 1.0).
  - With Random Forest, we explored different numbers of estimators (100, 200, and 300) and maximum depths (10 and 20) for the decision trees.
  - The goal was to identify the parameter combinations that would maximize the predictive accuracy of each ensemble method.

Analyses and Results:

	AdaBoost	Random Forest
Original Dataset	0.925	0.924
PCA Reduced	0.923	0.924
UMAP Reduced	0.911	0.916

1. **Best Parameters for AdaBoost and Random Forest**:
   - For the original dataset:
     - AdaBoost: We found the best parameters to be {'learning_rate': 0.5, 'n_estimators': 150}.
     - Random Forest: The optimal parameters were {'max_depth': 20, 'n_estimators': 300}.
   - When using PCA reduced data:
     - AdaBoost: The best parameters remained the same as the original dataset.
     - Random Forest: Similarly, the parameters {'max_depth': 20, 'n_estimators': 100} were identified as optimal.
   - With UMAP reduced data:
     - AdaBoost: Again, the best parameters matched those of the original and PCA reduced datasets.
     - Random Forest: The optimal parameters differed slightly, with {'max_depth': 10, 'n_estimators': 200}.
2. **Accuracy Comparison**:
   - AdaBoost:
     - Original: Achieved an accuracy of 92.54%.
     - PCA Reduced: Slightly lower accuracy of 92.29%.
     - UMAP Reduced: Showed the lowest accuracy of 91.12%.
   - Random Forest:
     - Original: Attained an accuracy of 92.43%.
     - PCA Reduced: Maintained a similar accuracy of 92.36%.
     - UMAP Reduced: Demonstrated an accuracy improvement compared to AdaBoost, achieving 91.62%.

Overall, both AdaBoost and Random Forest performed consistently across different dimensionality reduction techniques. However, the original dataset and PCA reduced data consistently outperformed UMAP reduced data in terms of accuracy for both algorithms. Despite slight variations in hyperparameters, the models achieved competitive accuracies across all scenarios, indicating robustness and generalizability.

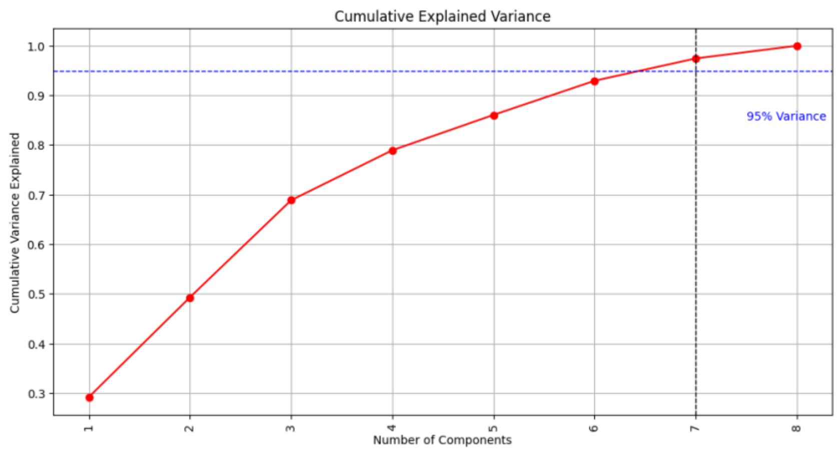
Dimensionality Reduction:

Steps:

The same steps as Dataset 1 were followed in this case.

Analysis:

1. **Principal Component Analysis (PCA)**:
   - Applying PCA revealed that 7 components were necessary to capture 95% of the variance within the dataset, suggesting effective dimensionality reduction.
   - Visualization of the scree plot showcased a sharp decline in explained variance after the initial components, emphasizing their significance in representing the dataset.



- The resulting reduced-dimensional representation facilitated enhanced interpretability and visualization, focusing on the most informative features.

2. **Uniform Manifold Approximation and Projection (UMAP)**:
   - Manual tuning of UMAP hyperparameters identified optimal settings, yielding a trustworthiness score of 0.998.
   - The best UMAP parameters, including a minimum distance (min_dist) of 0.25 and 10 nearest neighbors (n_neighbors), were crucial in preserving the neighborhood structure.
   - Despite reducing dimensionality to 2 components, UMAP maintained high trustworthiness, underscoring its effectiveness in preserving intrinsic data structure in lower-dimensional space.

Clustering:

Steps:

Same steps as the first dataset were used here.

Analyses and results:

Silhouette Scores:

	KMeans	Agglomerative
Original	0.251214	0.343222
PCA Reduced	0.2589	0.301006
UMAP Reduced	0.466836	0.42756

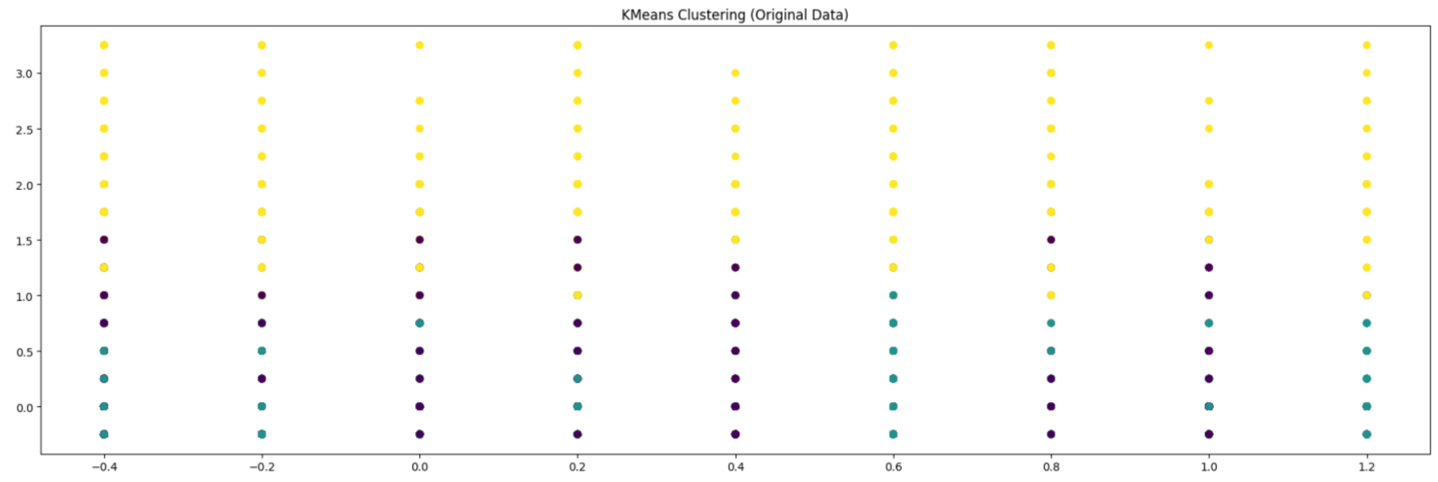
Best number of clusters for KMeans:

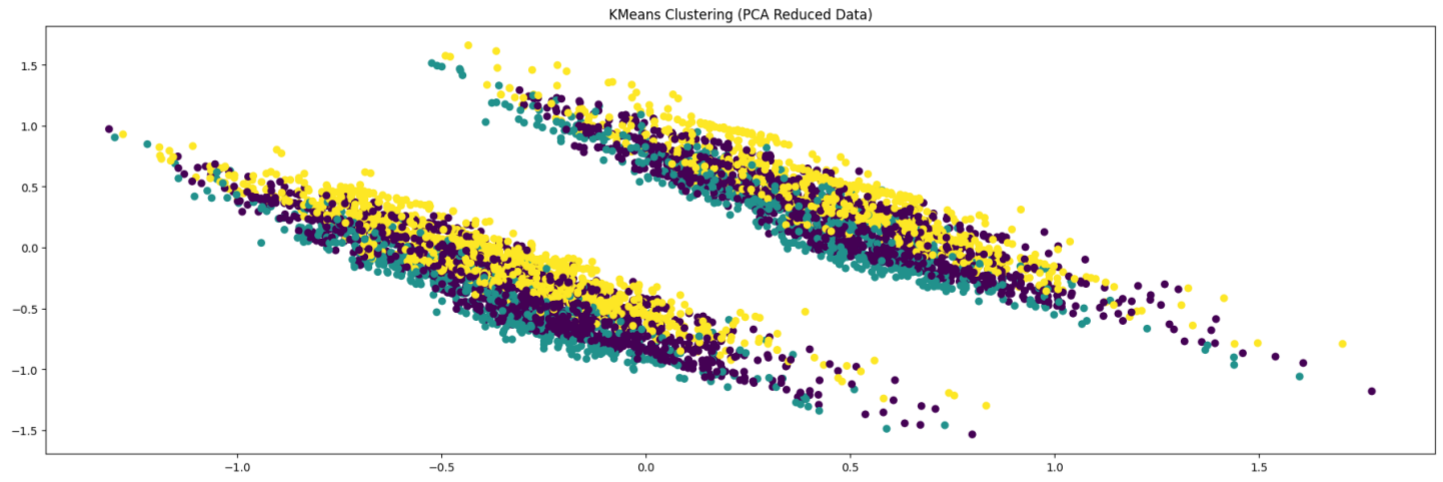
Original: 3
PCA Reduced: 3
UMAP Reduced: 5

Best number of clusters for Agglomerative Clustering:

Original: 3 clusters with linkage single
PCA Reduced: 3 clusters with linkage single
UMAP Reduced: 4 clusters with linkage ward

1. **Silhouette Scores Analysis**:
   - The silhouette scores provide insights into the quality and separation of clusters. Higher scores indicate better-defined clusters.
   - For the original dataset, KMeans achieved a silhouette score of 0.251, while Agglomerative clustering scored 0.343.
   - Upon applying PCA for dimensionality reduction, silhouette scores improved slightly. KMeans scored 0.259, and Agglomerative clustering scored 0.301.
   - UMAP reduction showed the highest silhouette scores, with KMeans at 0.467 and Agglomerative clustering at 0.428. This suggests superior cluster quality and separation.
2. **Optimal Number of Clusters**:
   - KMeans identified 3 clusters as optimal for all datasets, indicating distinct cluster structures.
   - Agglomerative clustering also favored 3 clusters for the original and PCA reduced data.
   - However, for UMAP reduced data, KMeans suggested 5 clusters, implying a more intricate clustering pattern.
   - Agglomerative clustering with UMAP reduction determined 4 clusters as optimal, showcasing a nuanced cluster structure captured by this combination.





Ensemble Methods:

Same steps as before were followed.

Analyses:

	AdaBoost	Random Forest
Original Dataset	0.464	0.501

PCA Reduced	0.440	0.489
UMAP Reduced	0.403	0.460

1. **Best Parameters Analysis**:
   - For the original dataset, AdaBoost achieved the best accuracy of approximately 46% with parameters: {'algorithm': 'SAMME.R', 'learning_rate': 0.5, 'n_estimators': 150}.
   - Random Forest attained the highest accuracy of around 50% with parameters: {'max_depth': 10, 'n_estimators': 300}.
   - After applying PCA for dimensionality reduction, AdaBoost's accuracy decreased slightly to about 44% with similar parameters.
   - PCA Reduced Random Forest also saw a decrease in accuracy to approximately 49%, with parameters similar to those of the original Random Forest.
   - When UMAP reduction was applied, both AdaBoost and Random Forest classifiers exhibited lower accuracies of around 40% and 46%, respectively, with different parameter configurations compared to the original dataset.
2. **Model Performance Analysis**:
   - The performance of ensemble methods varied based on the dimensionality reduction technique applied.
   - Random Forest consistently outperformed AdaBoost in terms of accuracy across all datasets.
   - Dimensionality reduction, particularly through UMAP, resulted in a noticeable decrease in model performance compared to the original dataset, indicating a loss of important information during reduction.
   - Despite the reduction in accuracy, ensemble methods still demonstrate predictive capability, suggesting potential for further optimization or exploration of alternative techniques.

References:

1. GFG page for Clustering
2. Blog for overview on understanding coding part of clustering
3. Kaggle Notebook for Sample Classification of Customer Segmentation
4. Kaggle Notebook for Sample Classification of Patient Survival
5. tutorials point practice page for clustering
6. Blog for Dimensionality Reduction
7. Evaluation of features post dimensionality reduction