CS F320: Foundations of Data Science

Project Report

Anirudh Anand (2021B3A70981P)

Akshit Phophaliya (2021B3A71738P)

Vidit Benjwal (2021A7PS0004P)

Vishnu Hari (2022A7TS0094P)

BITS Pilani

20 Nov 2024

Abstract

This report evaluates multiple machine learning models for regression, classification, and clustering on a diabetes-related dataset, incorporating feature scaling and dimensionality reduction techniques. Models were assessed using metrics such as RMSE, MAE, precision, recall, F1-score, and clustering-specific metrics like Hubert Statistic and Dunn's Index. Our findings highlight the superior performance of NGBoost for regression, Neural Networks for classification, and K-medoids for clustering on Z-score-transformed data. The report further provides an in-depth assessment of model and metric selection and data preprocessing in analysis of this dataset.

Introduction

In this report, we explore the application of various machine learning techniques to analyse data using a structured workflow. The study begins with preprocessing to ensure the data is clean, consistent, and ready for analysis. This is followed by the implementation of regression, classification, and clustering models, each tailored to extract specific insights from the data. We further analyse the various parameters used in each model, as well as their impact on the results.

Dataset

We have chosen the diabetes prediction dataset, an aggregation of patient data containing information about each patient's medical status as well as whether they have diabetes. The dataset contains 9 columns and 2,500 entries. It includes features such as age, gender, BMI, hypertension, heart disease, smoking history, HbA1c level, and blood glucose level. By using this dataset, we hope to derive key insights that help us better understand the link between these factors and the likelihood of developing diabetes.

Pre-processing

In the pre-processing phase, we convert the raw dataset into a format better suited for processing by various models. Since there are no null or missing entries in our chosen dataset, we do not need to define a strategy for dealing with these. Hence, in the pre-processing phase, we first select only the columns in the dataset corresponding to numerical features, namely *age, bmi, HbA1c_level*, and *blood_glucose_level*. Our target, *diabetes*, is a categorical variable, hence it is removed by this stage of processing. The updated target variable used after this stage is *blood_glucose_level*. Having selected the numerical features, we now normalise them to unit variance and zero mean using a StandardScaler from the scikit-learn library. This gives us the pre-processed dataset. Finally, we run principal component analysis (PCA) on the pre-processed dataset using the scikit-learn library to obtain the three principal components with greatest explained variance. The normalisation stage is important to ensure no feature has undue influence on the derived principal components, as PCA is sensitive to scale. We notice that the three most important principal components have an explained variance ratio of 44.81%, 22.30%, and 16.66% respectively for a total explained variance ratio of 83.77%. We also join these principal components with the categorical features we earlier removed to obtain a final pre-processed dataset, ‘data_pca’.

Regression Analysis

In this phase, we perform regression analysis by preparing the dataset, training multiple models, and evaluating their performance. We choose *age, bmi*, and *HbA1c_level* as predictors and *age, bmi*, and *HbA1c_level* as the target variable, since *diabetes* is not a numeric feature. The dataset is split into training and testing sets, allocating 80% for training and 20% for testing, and ensures reproducibility by setting a fixed random state. To evaluate the models, we define a function that fits each model to the training data, predicts the test set, and calculates two key metrics: Root Mean Squared Error (RMSE), which places greater emphasis on outliers, and Mean Absolute Error (MAE), which averages absolute prediction errors. We analyse six models—Linear Regression, Random Forest, Extra Trees, AdaBoost, XGBoost, and NGBoost. The parameters we chose are as follows -

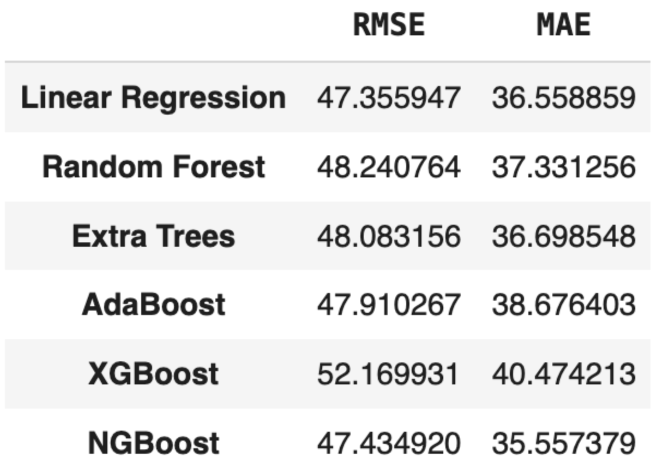
- **Linear Regression**: No additional parameters.
- **Random Forest:** random_state=42, max_depth=10, n_estimators=100.
- **Extra Trees:** random_state=42, max_depth=10, n_estimators=100.
- **AdaBoost:** random_state=42, n_estimators=50.
- **XGBoost:** objective='reg:squarederror', random_state=42, max_depth=5, n_estimators=100.
- **NGBoost:** random_state=42, n_estimators=100, verbose=False

Metrics

- **RMSE (Root Mean Square Error)**: Represents the square root of the average squared differences between predicted and actual values, penalizing large errors more heavily. It is sensitive to outliers.
- **MAE (Mean Absolute Error)**: Measures the average of the absolute differences between predicted and actual values, treating all errors equally regardless of their magnitude.

Results

Finally, we store their performance metrics in a structured format and present them in a table for comparison. This approach allows us to identify the most effective model for predicting blood glucose levels based on RMSE and MAE. We see that **Linear Regression** and **NGBoost** show competitive performance with respect to RMSE and MAE, respectively, while **XGBoost**



	RMSE	MAE
Linear Regression	47.355947	36.558859
Random Forest	48.240764	37.331256
Extra Trees	48.083156	36.698548
AdaBoost	47.910267	38.676403
XGBoost	52.169931	40.474213
NGBoost	47.434920	35.557379

performs poorly across the board. This suggests that NGBoost might perform better when focusing on reducing absolute errors, while Linear Regression is more effective at reducing squared errors.

Classification

For our classification task, we classify the *diabetes* column based on numerical attributes and compare classifier performance on both the original and PCA-transformed datasets. First, we extract the independent variables (*age, bmi, HbA1c_level, blood_glucose_level*) and the dependent variable (*diabetes*). We then split the dataset into training and testing sets, allocating 80% for training and 20% for testing while ensuring consistent splits with a fixed random state.

Next, we scale the independent variables using StandardScaler from the scikit-learn library to normalise the features. We fit the scaler on the training data and apply the same transformation to the test set, ensuring that both are scaled consistently. A PCA is then fit on the training data and applied to the test data as well.

We define a set of classifier models to evaluate, including Logistic Regression, Naive Bayes, K-Nearest Neighbors, Linear and Kernel Support Vector Machines, Decision Trees, and Neural Networks. Each classifier is tested using a custom evaluation function that calculates Precision, Recall, and F1-score by fitting the model to the training data, predicting on the test set, and comparing predictions to actual labels. The parameters of the same are as follows:

- **Logistic Regression**: No additional parameters
- **Naive Bayes**: No additional parameters
- **K-Nearest Neighbors**: No additional parameters
- **Linear SVM**: kernel='linear'
- **Kernel SVM**: kernel='rbf'
- **Decision Trees**: No additional parameters
- **Neural Networks**: max_iter=1000

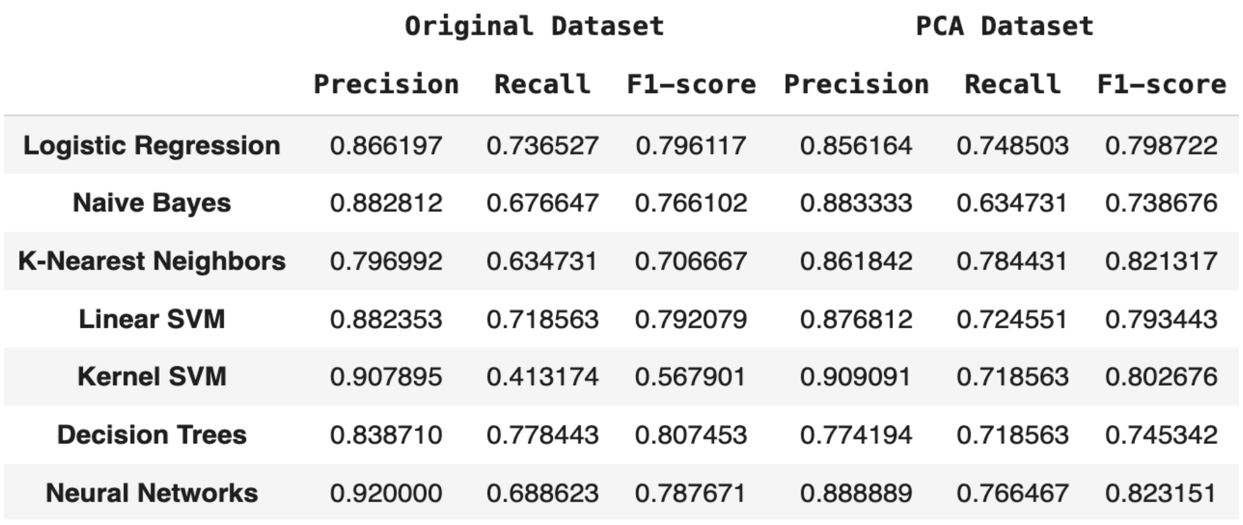
Metrics

- **Precision**: Measures the proportion of true positive predictions out of all positive predictions made by the model. High precision indicates fewer false positives.
- **Recall**: Evaluates the proportion of true positive predictions out of all actual positive cases. High recall means the model misses fewer true positives.
- **F1 Score**: Provides a harmonic mean of precision and recall, balancing their trade-off. It is especially useful when the dataset is imbalanced.

Results

As specified in the task, we evaluate the classifiers on both the original scaled dataset and the PCA-transformed dataset, storing their performance metrics (Precision, Recall, and F1-score) in structured dataframes. Finally, we compile a comparative table as shown below of results for the two datasets.

We see that on the original dataset, **Neural Networks** have the greatest precision, while



	Original Dataset			PCA Dataset		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Logistic Regression	0.866197	0.736527	0.796117	0.856164	0.748503	0.798722
Naive Bayes	0.882812	0.676647	0.766102	0.883333	0.634731	0.738676
K-Nearest Neighbors	0.796992	0.634731	0.706667	0.861842	0.784431	0.821317
Linear SVM	0.882353	0.718563	0.792079	0.876812	0.724551	0.793443
Kernel SVM	0.907895	0.413174	0.567901	0.909091	0.718563	0.802676
Decision Trees	0.838710	0.778443	0.807453	0.774194	0.718563	0.745342
Neural Networks	0.920000	0.688623	0.787671	0.888889	0.766467	0.823151

Decision Trees have the best recall and F1-Score and hence are better for diabetes prediction, as the priority in medical predictive treatment is to avoid false negatives. On the PCA dataset, however, **Kernel SVM** has the best precision, while **K-Nearest Neighbors** has the best recall and **Neural Networks** have the best F1-Score (slightly outperforming **K-Nearest Neighbors**). This discrepancy between the PCA and original datasets is likely because the original dataset retains the complete feature information, which aids Decision Trees in effectively capturing complex patterns for high recall and F1-Score. On the PCA dataset, dimensionality reduction emphasises dominant variance directions, which may benefit Kernel SVM and Neural Networks due to their ability to generalise well in transformed spaces, while K-Nearest Neighbors excels in identifying local patterns which are more pronounced under PCA.

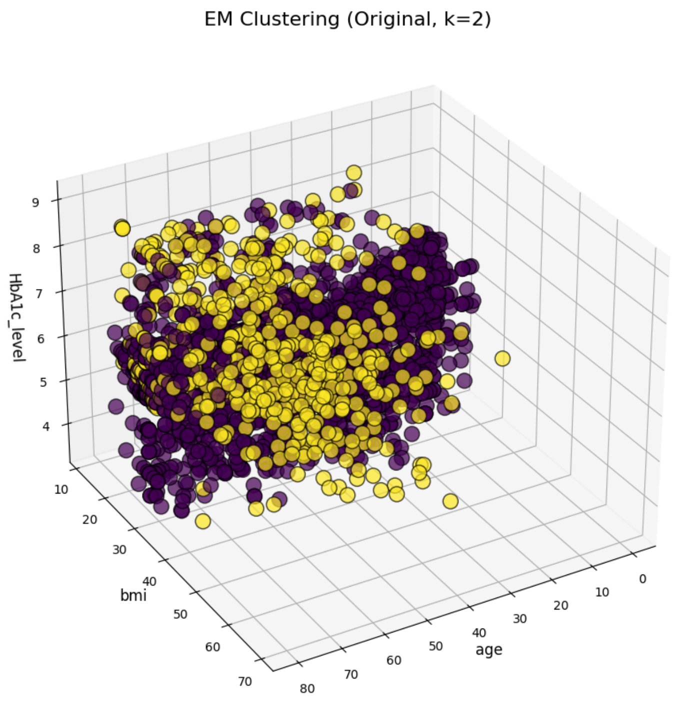
Clustering

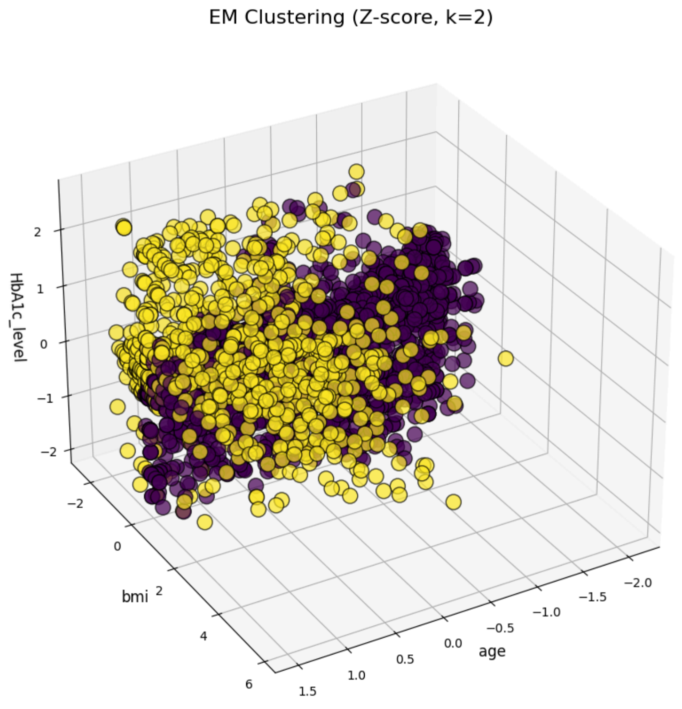
Clustering analyses are conducted on the dataset using three methods: K-means, Expectation Maximisation (EM), and K-medoids. We begin by preparing the data, extracting key numerical features (*age, bmi, HbA1c_level*, and *blood_glucose_level*), and scaling these features using Z-score normalisation to ensure all attributes have equal influence during clustering. A fixed cluster count of k = 2 is chosen, given the binary nature of the *diabetes* label, and each clustering algorithm is applied to both the original and scaled datasets. The Z-score transform standardises features by scaling them to have a mean of 0 and a standard deviation of 1, ensuring all variables contribute equally to clustering. The parameters used for the clustering algorithms are as follows:

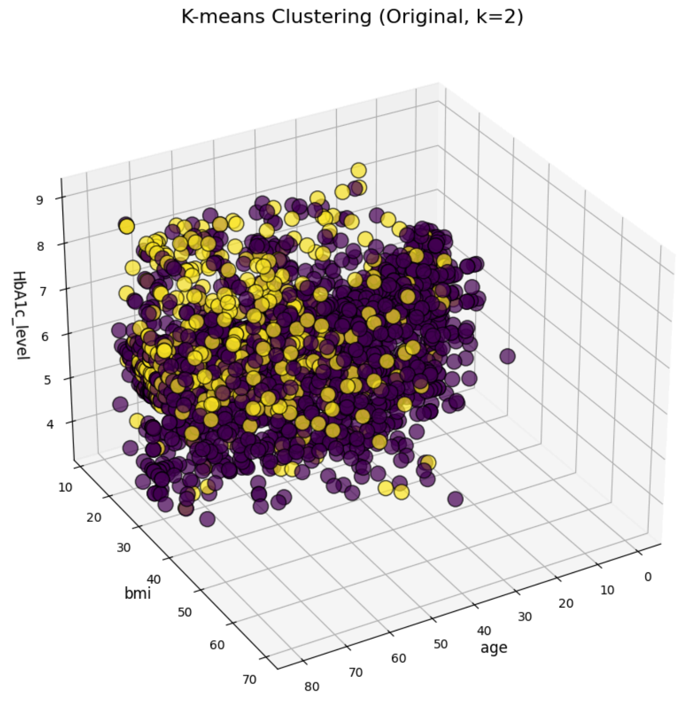
- **K-means Clustering**: n_clusters=2, random_state=42
- **Expectation Maximisation (Gaussian Mixture)**: n_components=2, random_state=42
- **K-medoids Clustering**: n_clusters=2, init='k-medoids++', random_state=42

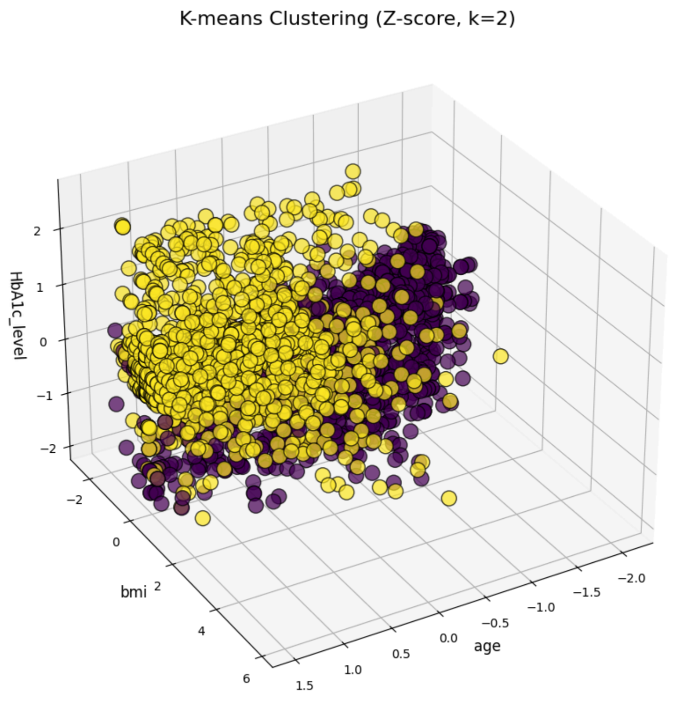
For evaluation, we compute metrics including Silhouette Scores (measuring intra-cluster cohesion and inter-cluster separation), Dunn's Index (assessing compactness and separation of clusters), BetaCV (comparing inter- and intra-cluster distances), Hubert Statistic (using Adjusted Rand Index to match clustering labels with true labels), and Sum of Squared Errors (SSE, specific to K-means). These metrics are computed for clusters generated on both datasets, allowing a straightforward comparison of clustering effectiveness pre- and post-scaling.

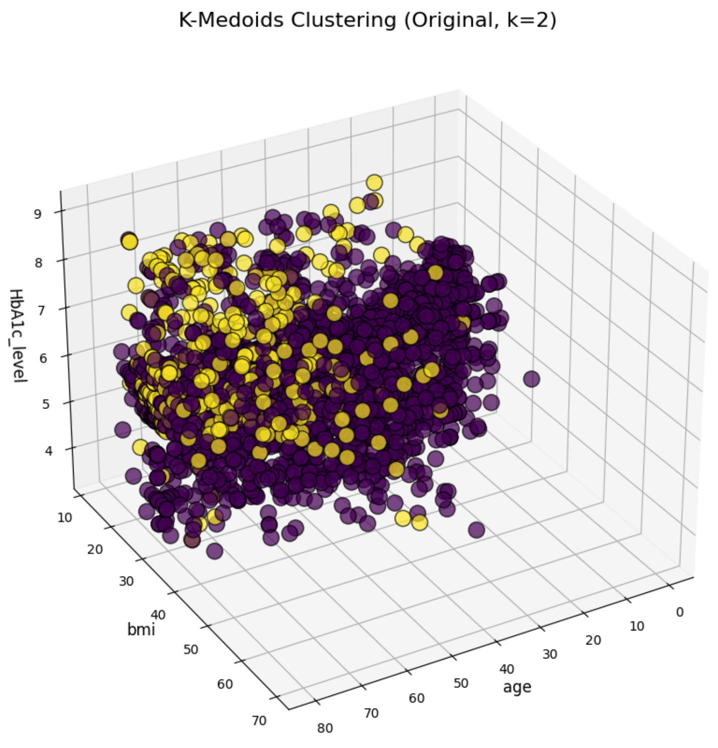
We generate a 3D visualisation to inspect clustering quality intuitively. Using the first three features (age, bmi, HbA1c_level), clusters are plotted for both datasets across all three methods. Illustrations for all the clustering methods are given below. Yellow dots represent data points in one cluster, and purple represents the other.

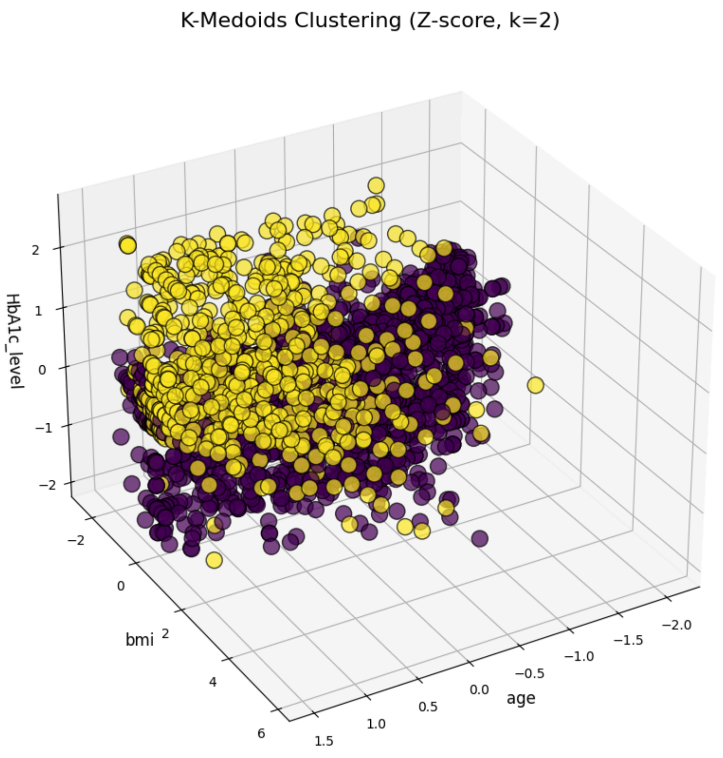












Metrics

- **Silhouette Score**: Measures the quality of clustering by evaluating how similar each point is to its own cluster compared to others. Higher scores indicate well-defined clusters.
- **Dunn's Index**: Assesses the compactness and separation of clusters by calculating the ratio of minimum inter-cluster distance to maximum intra-cluster distance. Higher values signify better clustering.
- **BetaCV**: Compares the mean inter-cluster distances to the mean intra-cluster distances. Lower values indicate better-defined and more cohesive clusters.
- **Hubert Statistic (Adjusted Rand Index)**: Quantifies the similarity between true labels and predicted clusters, accounting for chance. Values range from -1 (poor match) to 1 (perfect match).

- **SSE (Sum of Squared Errors)**: Used only for K-means, it calculates the sum of squared distances between points and their cluster centres. Lower SSE indicates tighter clusters.

Results

Finally, the clustering results are compiled into a comprehensive table for side-by-side comparison. We can see an approximately 2-fold reduction in both BetaCV and Silhouette Scores for all models when using the Z-score normalised data over the original dataset. Further, we notice that in the case of K-Means clustering, the SSE is reduced extremely by ***500 times.***

	Clustering Method	Silhouette Score	Dunn's Index	BetaCV	Hubert Statistic	SSE
0	K-means (Original)	0.563312	0.308604	2.283016	0.264006	3569228.444712
1	EM (Original)	0.387285	0.001843	1.517056	0.366783	N/A
2	K-medoids (Original)	0.570413	0.018577	2.493087	0.321107	N/A
3	K-means (Z-score)	0.256037	0.014794	1.372868	0.368408	7096.868923
4	EM (Z-score)	0.250315	0.009912	1.294957	0.394090	N/A
5	K-medoids (Z-score)	0.277701	0.009980	1.381920	0.572792	N/A

For the purposes of our evaluation, the best metric to consider is the **Hubert Statistic** as it is unaffected by the scaling applied by Z-score and measures the correspondence between clusters and true labels. For the Z-score dataset, **K-medoids clustering** outperforms other methods, indicating its robustness to feature scaling and ability to handle varied data distributions. On the original dataset, the **GMM EM** algorithm excels, likely due to its probabilistic nature, which captures underlying data patterns better. **K-means clustering** performs the worst in both cases, possibly because of its sensitivity to initial centroid placement and inability to handle complex cluster shapes effectively.

Conclusion

Through this analysis, we demonstrated the critical role of preprocessing techniques, including Z-score transformation and PCA, in enhancing model performance. NGBoost proved most effective for regression. Neural Networks, k-Nearest Neighbours, and Kernel SVM had the highest F1-scores, but Decision Trees balanced precision and recall for classification. K-medoids excelled in clustering on normalised data, while EM Clustering showed the best result on the non-normalised data. These insights underscore the need to align model and preprocessing strategies with dataset characteristics to achieve robust and accurate predictions in medical analytics.