**Data Collection:**

The dataset used in this study is sourced from “parkinsons\_updrs.data.” The dataset comprises of biomedical voice recordings from 42 individuals with early stage Parkinson’s disease who participated in a six-month trial of a telemonitoring device for remote system progression monitoring. The voice measurements were automatically collected in the patients’ homes resulting in 5,875 records in total.

The dataset includes various columns such as subject number, subject age , subject gender , the time interval from baseline recruitment data, motor UPDRS, total UPDRS and 16 biomedical voice measures. Each row in the dataset corresponds to a specific voice recordings from one of the 42 recordings.

The primary objective of the data is to predict the motor and total UPDRS scores (‘motor\_UPDRS’ and ‘total\_UPDRS’) based on the 16 voice measures. The UPDRS scores are commonly used in assessing the severity of Parkinson’s disease symptoms, with motor UPDRS focusing on motor-related symptoms and total UPDRS considering both motor and non-motor symptoms.

The main aim behind using this dataset is to predict the motor and total UPDRS scores using 16 biomedical voice measures. This is valuable in context of Parkinson’s disease progression monitoring, which is crucial for clinicians to assess and manage the condition effectively.

**Data Normalization:**

After we have the dataset ready, we go for the normalization step. Normalization is a preprocessing technique used to scale numeric features within a specific range, typically between 0 to 1. The main aim behind normalizing data it guarantees that the data is appropriately scaled for machine learning algorithms, improving the accuracy and effectiveness of the subsequent analytical steps.

In our project, we have used Min-Max Normalization for scaling of the data. Also, to get the meaningful data we have excluded the first 3 columns from our consideration. This adjustment ensures that the resulting dataset only contains the relevant, normalized numeric features.

This step is crucial for preparing the data for subsequent analysis, improving the performance of clustering and regression algorithms.

**Clustering methods and Visualization Clustering Results**:

**1. K-Means:**

Initialization: In K-Means, the algorithm starts with an initial guess for the cluster centroids, typically chosen randomly from the data points. Random initialization, however, can result in suboptimal convergence and sensitivity to the initial placement of centroids.

Algorithm Steps:

1. Randomly initialize K centroids.

2. Assign each data point to the nearest centroid.

3. Update the centroids based on the mean of the data points assigned to each cluster.

4. Repeat steps 2 and 3 until convergence.

**2. K-Means++:**

Initialization: K-Means++ improves upon the initialization step by selecting centroids that are more likely to be distant from each other, resulting in better convergence and more robust solutions.

Algorithm Steps:

1. Randomly select the first centroid from the data points.

2. For each subsequent centroid, choose the next one from the remaining data points with probability proportional to the square of the distance from the point to the nearest existing centroid.

3. Assign data points to the nearest centroid.

4. Update centroids based on the mean of the data points assigned to each cluster.

5. Repeat steps 3 and 4 until convergence.

Comparison:

K-Means++ tends to converge faster and provides more accurate and stable results compared to K-Means, especially when the number of clusters (K) is relatively large.

K-Means++ reduces the risk of converging to a local minimum, making it more robust for various datasets.

While K-Means++ is generally preferred, K-Means might still be suitable for smaller datasets or when computational resources are limited.

**3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**

DBSCAN is a density-based clustering algorithm that divides a dataset into groups based on the density of data points in the feature space. Unlike partitioning methods like K-Means, DBSCAN does not require the number of clusters as an input and can discover clusters of arbitrary shapes. It is particularly effective in identifying clusters embedded in noise and handling varying cluster shapes and sizes.

Algorithm Steps:

1. Parameter Definition:

Epsilon(ɛ): The maximum distance between two data points for one to be considered in the neighbourhood of the other.

MinPts: The minimum number of data points required to form a dense region (including the data point itself).

2. Initialization:

Randomly select a data point that has not been visited.

3. Density-Based Exploration:

If the number of data points in the (ɛ) - neighborhood of the selected point is greater than or equal to MinPts, form a dense region and label all points within the (ɛ) - neighborhood as part of the same cluster.

If a point has fewer than MinPts neighbors but is within the (ɛ) - neighborhood of another point that is part of a dense region, the point is considered a border point and assigned to that cluster.

If a point has fewer than MinPts neighbors and is not within the (ɛ) - neighborhood of any dense region point, it is labelled as noise.

4.Expand Clusters:

Repeat the process for unvisited data points until all data points have been visited.

Output:

Core Points: Points with at least MinPts data points in their (ɛ) – neighborhood.

Border Points: Points with fewer than MinPts neighbors but within the (ɛ) - neighborhood of a core point.

Noise Points: Points that are neither core nor border points.

In summary, DBSCAN is a powerful clustering algorithm that is particularly useful for datasets with irregularly shaped clusters and varying densities. Its ability to identify noise and outliers makes it a valuable tool in exploratory data analysis and pattern recognition.

**4. OPTICS Algorithm (Ordering Points To Identify the Clustering Structure):**

Introduction:

OPTICS is a density-based clustering algorithm that extends the concept of DBSCAN. It aims to identify clusters of arbitrary shapes and sizes in a dataset while addressing some limitations of DBSCAN. OPTICS introduces the concept of reachability and produces an "ordered" list of points, providing a more flexible representation of the cluster structure.

Algorithm Steps:

1. Parameter Definition:

Epsilon(ɛ): The maximum distance between two data points for one to be considered in the neighborhood of the other.

MinPts: The minimum number of data points required to form a dense region (similar to DBSCAN).

2.Initialization:

Initialize an empty priority queue to store the reachability distances.

3. Core Distance:

For each data point, calculate the core distance, which is the distance to the MinPts-th nearest neighbor. This reflects the local density around the point.

4.Reachability Distance:

For each data point, calculate the reachability distance for every point in its (ɛ)-neighborhood. The reachability distance is the maximum of the core distance of the data point and the distance between the data point and the considered neighbor.

5. Building Reachability Plot:

Populate the priority queue with the reachability distances, creating a reachability plot that reflects the density-based structure of the dataset.

6. Cluster Extraction:

Traverse the reachability plot, identifying valleys in the plot as potential cluster boundaries.

Determine clusters based on the reachability distances and core distances.

**Classification :**

**SVM Classification on KMeans Clustered Data:**

The SVM classifier is trained on the features (`motor\_UPDRS` and `total\_UPDRS`) and labels (`Cluster`) obtained from KMeans clustering.

The accuracy and confusion matrix are computed to evaluate the performance of the SVM classifier.

A scatter plot is generated with the SVM decision boundary to visualize the classification results.

**SVM Classification on KMeans++ Clustered Data:**

Similar to the first case, this performs SVM classification on data clustered using KMeans++.

**Random Forest Classification for DBSCAN:**

Random Forest is an ensemble learning algorithm that operates by constructing a multitude of decision trees during training and outputting the mode of the classes for classification problems. It builds each tree using a random subset of the features and introduces randomness in both the data and the feature selection, leading to a more robust and accurate model. Random Forest is capable of handling complex relationships in data and is less prone to overfitting.

The provided code implements Random Forest Classification within distinct clusters identified by DBSCAN. Initially, the dataset is divided into subsets corresponding to unique cluster labels assigned by DBSCAN. Subsequently, within each cluster, a binary target variable is created by applying a threshold to the 'motor\_UPDRS' variable specific to that cluster. Features and the binary target variable are then meticulously prepared, excluding any irrelevant columns. The dataset is further partitioned into training and testing sets to facilitate model training and subsequent evaluation. Following this, a Random Forest Classifier is instantiated and trained on the designated training set within each individual cluster. The trained model is then employed to make predictions on the corresponding test set, and the resulting classification accuracy is computed and printed for each distinct cluster. This approach allows for a tailored assessment of the classification model within the context of different clusters, offering valuable insights into the model's performance across various subsets of the data.

**SVM Classification on OPTICS Clustered Data:**

We used Support Vector Regression (SVR) to predict 'motor\_UPDRS' values based on a set of features. SVR is a type of Support Vector Machine adapted for regression tasks, and in this context, it seeks to find a regression function that best fits the data. The model is trained on a subset of the data, and its performance is evaluated using metrics such as Mean Squared Error and R-squared. SVR is particularly useful when dealing with non-linear relationships and complex patterns in the data. Adjusting the kernel type and regularization parameter allows for customization to different data characteristics.