# Data Preparation and Normalisation

**Data Preparation and Scaling Techniques**

## Overview of the Code

This Python function preprocesses data for a machine learning project by using certain procedures. A dataset must be loaded, missing values must be handled, categorical variables must be encoded, training and testing sets must be divided, numerical features must be scaled, and data distributions both before and after scaling must be visually shown.

## Libraries Used

**Pandas**: For analysis and data manipulation.   
• **numpy:** Offers assistance with numerical operations.   
• **Sklearn:** A machine learning package for scaling, preprocessing, and model selection.   
o **train\_test\_split:** Divides the data into sets for testing and training.   
o **StandardScaler:** This tool is used to standardise (scale features).   
o **LabelEncoder:** Provides numerical values based on categorical data.   
• **Seaborn:** For visualising statistical data.   
• **Matplotlib:** Data visualisation and plot creation tools.   
• **style:** A matplotlib module that allows plot styles to be specified.

## Step-by-Step Explanation of the Code

### Load the Dataset

*df = pd.read\_csv('Dataset (ATS).csv')*

Pandas is used to read the dataset from a CSV file. Presumably, the dataset includes features and customer data such as contract type, phone and internet service providers, gender, dependents, and a target variable (churn).

### Display Basic Information About the Dataset

print(df.info())

print(df.describe())

**• info():** Shows the dataset's structure, including the number of non-null items, data types, and column names. **• describe():** Offers summary statistics (such as mean, standard deviation, min, and max values) for numerical columns.

### Data Cleaning: Handling Missing Values

print(df.isnull().sum())

df.fillna(method='ffill', inplace=True) print(df.isnull().sum())

* **isnull().sum()**: Identifies missing values in each column.
* **fillna(method='ffill')**: Fills missing values using the forward fill technique, which propagates the last valid observation forward to fill gaps.
* **inplace=True**: Modifies the dataframe in place, meaning no new dataframe is created.

### Encode Categorical Variables

categorical\_cols = ['gender', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'Contract', 'Churn']

for col in categorical\_cols:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

* **Label Encoding**: Converts categorical variables into numeric values. For example, "Male" and "Female" in the gender column might be encoded as 0 and 1, respectively.
* This technique is necessary because machine learning models often work better with numeric data than categorical data.

### Split the Data Into Training and Testing Sets

X = df.drop('Churn', axis=1) y = df['Churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

* **Feature and Target Separation**: X contains all the features (independent variables), and y contains the target variable (Churn), which we aim to predict.
* **train\_test\_split()**: Splits the dataset into training (80%) and testing (20%) sets. This ensures that the model is trained on one portion of the data and evaluated on another to avoid overfitting.

## Emphasis on Scaling Techniques

### Standard Scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) X\_test\_scaled = scaler.transform(X\_test)

* **Scaling**: In this step, numerical features are normalized using **StandardScaler**, a method that ensures that the data has **mean of 0** and a **standard deviation of 1.**

#### Why Scaling is Important:

* + Some machine learning algorithms (like SVM, K-Means, and neural networks) perform better when the data is scaled because they are sensitive to the magnitude of features.
  + Scaling prevents features with large values from dominating those with smaller values.

#### How StandardScaler Works

* + - * **StandardScaler** transforms the data by:

where:

* + - * + **X** is the feature,

𝑍 =

(𝑋 − μ) σ

* + - * + **μ** is the mean of the feature values in the training data,
        + **σ** is the standard deviation.
      * The scaler first **fits** the X\_train data (calculates the mean and standard deviation), and then applies the transformation (fit\_transform).
      * For X\_test, the scaler uses the same mean and standard deviation values from X\_train (using transform), ensuring consistent scaling between training and

testing sets.

### Visualization of Distributions Before and After Scaling

style.use('seaborn-whitegrid') plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

sns.histplot(df['MonthlyCharges'], kde=True, color='skyblue') plt.title('MonthlyCharges Distribution Before Scaling') plt.xlabel('MonthlyCharges')

plt.subplot(1, 2, 2)

sns.histplot(X\_train\_scaled[:, X.columns.get\_loc('MonthlyCharges')], kde=True, color='salmon')

plt.title('MonthlyCharges Distribution After Scaling') plt.xlabel('MonthlyCharges (scaled)')

plt.tight\_layout() plt.show()

* This section visualizes the distribution of one specific feature (MonthlyCharges) before and after scaling.
* **Before Scaling**: The distribution of MonthlyCharges is shown as it appears in the original dataset. The values may have a wide range.
* **After Scaling**: The same feature (MonthlyCharges) is plotted after scaling. As expected, the distribution is now centered around 0, with values standardized around the mean of 0 and standard deviation of 1.

## Summary

This code snippet covers several key steps in preparing a dataset for machine learning, with a particular focus on scaling techniques. The following preprocessing steps are performed:

* **Data Loading**: Reads the dataset into a DataFrame using pandas.
* **Data Cleaning**: Handles missing values by forward filling.
* **Label Encoding**: Converts categorical variables to numeric form using LabelEncoder.
* **Train-Test Split**: Splits the dataset into training and testing sets.
* **Scaling**: Uses StandardScaler to normalize numerical features, ensuring that all features have a mean of 0 and a standard deviation of 1.

The scaling step is essential for many machine learning algorithms because it ensures that features are on the same scale, improving performance and stability of the models.

By visualizing data before and after scaling, you can clearly see the effect of normalization on feature distributions.

# Clustering Analysis Results and Findings

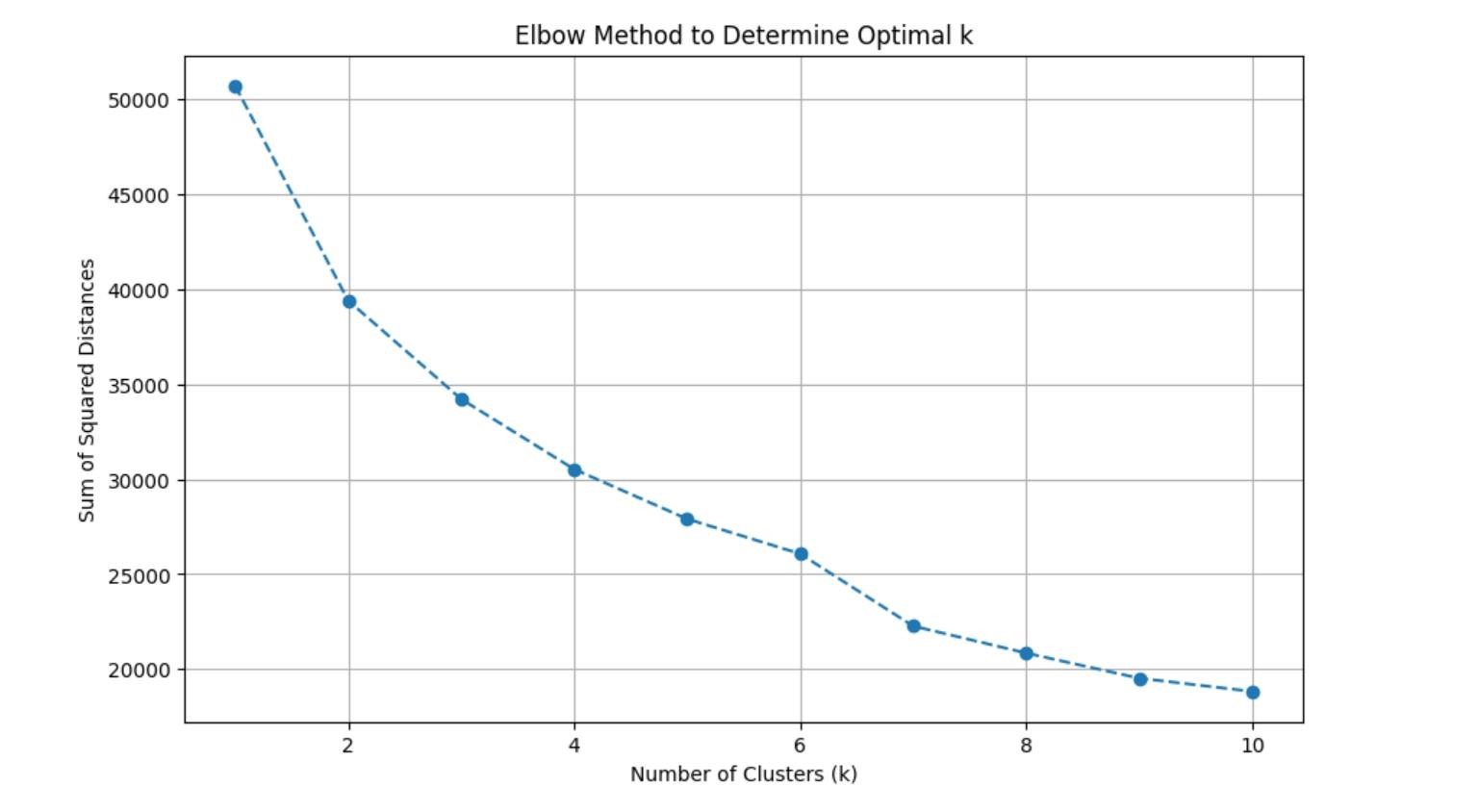
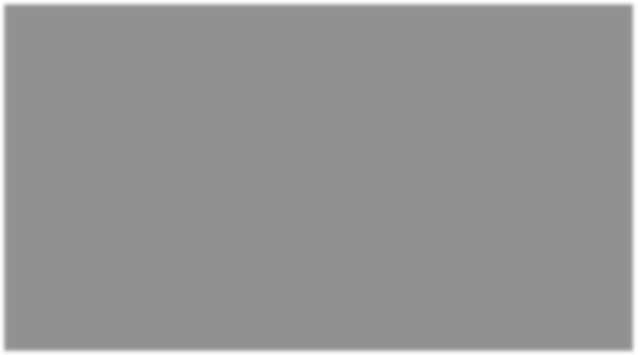
## Overview:

Four ideal clusters for consumer segmentation were found using the Elbow Method; beyond k = 4, a dramatic decline in SSE suggests decreasing profitability. The clusters were visualised using PCA, which showed separate but somewhat overlapped consumer segments. By grouping clients according to similar characteristics, this segmentation makes it possible to develop customised marketing and retention plans. The discovered clusters offer insightful information about consumer behaviour that directs data-driven business choices for focused interaction.

## Elbow Method to Determine Optimal k

The Elbow Method is a commonly used technique for selecting the optimal number of clusters (k) in K-Means clustering. This graph plots **Sum of Squared Errors (SSE)**

against the number of clusters (k) to visually inspect where the optimal number of clusters lies.



#### Detailed Breakdown:

* + **Sum of Squared Errors (SSE)**, also known as Inertia, measures the

compactness of the clusters. It sums the squared distances between each point and the centroid of the cluster it belongs to. The objective of K-Means is to

minimize this sum, thereby creating tight, well-separated clusters.

* + **The X-axis** represents the number of clusters (k). In your case, you explored values of k ranging from 1 to 10.
  + **The Y-axis** represents the SSE for each value of k.

#### Analysis:

* + **k = 1** shows the highest SSE because all data points are forced into a single cluster, resulting in high variance within the group.
  + As k increases, the SSE decreases because the data is divided into more clusters, reducing the distance between points and their centroids.
  + At **k = 4**, we observe the **elbow point**. This is the point where the SSE curve

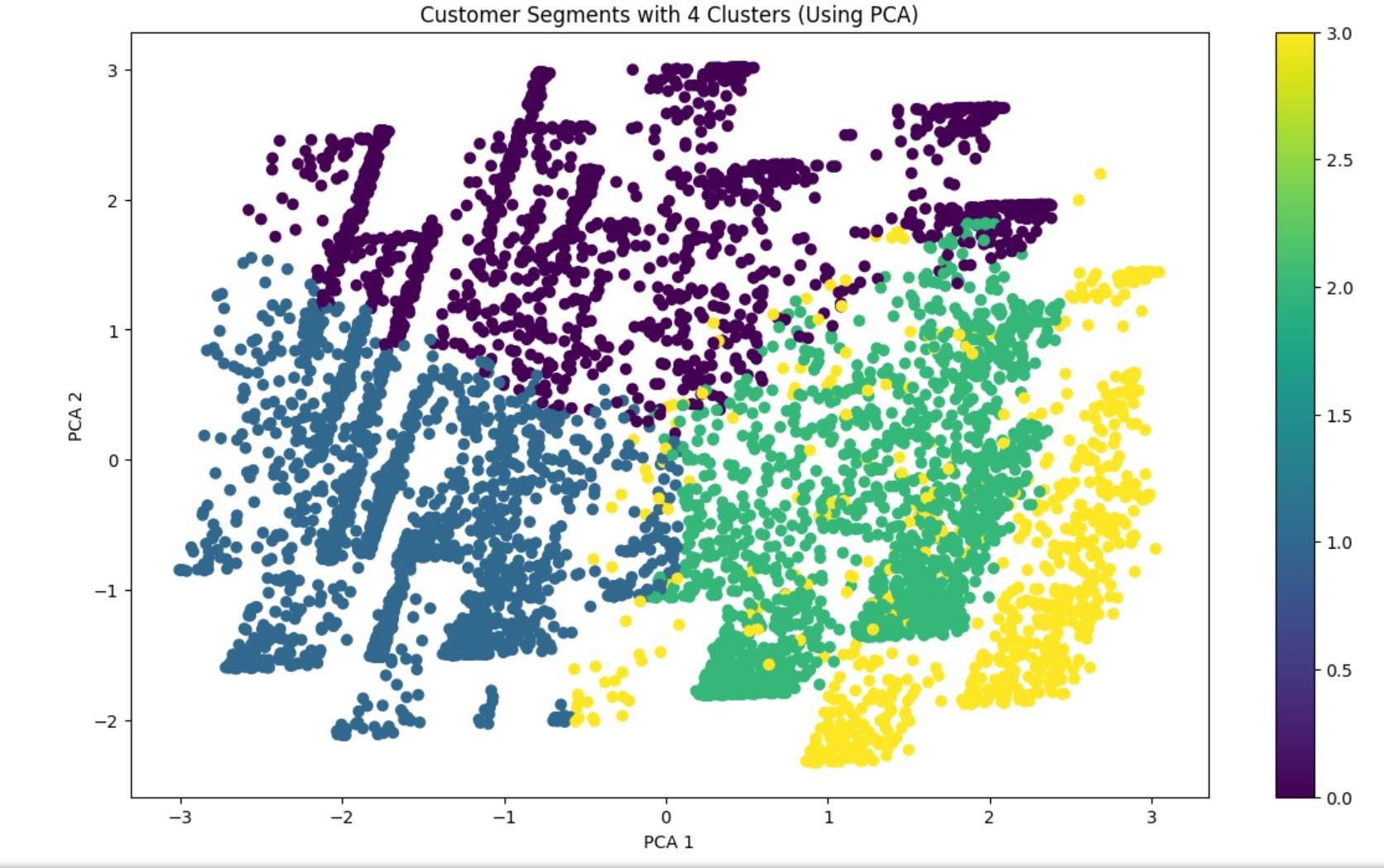
bends or flattens, indicating diminishing returns in the reduction of SSE with the addition of more clusters. Beyond this point, adding more clusters does not significantly reduce the SSE, suggesting that the data is sufficiently segmented into 4 clusters.

#### Why k = 4 is Optimal:

* + The elbow occurs at k = 4, meaning that this number of clusters provides a balance between reducing variance within clusters and avoiding overfitting the model by using too many clusters.
  + If you choose a number of clusters beyond the elbow point (e.g., k = 5 or higher), the SSE continues to decrease but at a slower rate, and the clustering solution may become unnecessarily complex without providing better segmentation.

## Customer Segments with 4 Clusters (Using PCA)

This second graph is a **visualization** of the K-Means clustering results. It uses **Principal Component Analysis (PCA)**, a dimensionality reduction technique, to project the high- dimensional data (with possibly many features) into two dimensions for easier interpretation and visualization.



#### Detailed Breakdown:

* + **Principal Component Analysis (PCA)** reduces the dimensionality of the dataset by identifying the principal components, which are directions (or axes) of maximum variance in the data. This allows for visualizing data in a two-

dimensional plane, even though the original dataset may have many features.

* + **Each point** on the scatter plot represents a customer from the dataset.
  + **The colors** represent the 4 clusters identified by K-Means, with each color corresponding to a different cluster label. For example, yellow points might represent one customer group, purple another, and so on.
  + **The X-axis and Y-axis** represent the first two principal components (PCA 1 and PCA 2), which capture the most variance in the data. These components are combinations of the original features but make it possible to visualize the entire dataset in two dimensions.

#### Cluster Separation:

* + The data points are grouped into 4 clusters as per the elbow method recommendation.
  + **Color-Coded Segments**: Each cluster is color-coded (purple, blue, green, and yellow). The distinct separation between some of these clusters (e.g., between purple and green) indicates that the K-Means algorithm was able to differentiate customer segments effectively.
  + **Cluster Spread**: Some clusters are more tightly grouped (like purple), while others (like yellow) are more spread out. This suggests that some customer segments have more homogeneous characteristics, while others are more diverse.
  + **Overlap**: There is some overlap between the clusters, which is expected

because PCA reduces the dimensions of the data, meaning some information loss can occur. However, the K-Means algorithm itself is working in a higher- dimensional space, where the separation between clusters may be clearer.

#### Insights:

* + **Customer Segments**: The 4 clusters likely represent distinct customer

segments based on the features you used in your model (e.g., demographic data, account information, service usage patterns).

* + - For instance, one cluster might represent long-term customers who are less likely to churn, while another might represent newer customers who frequently change service providers.
    - By identifying which features are most significant in determining these clusters, you can understand what differentiates one customer segment from another.
  + **Business Application**: Once you understand the characteristics of each cluster, you can tailor marketing strategies, service offers, and customer retention efforts to each segment. For example:
    - Customers in the yellow cluster (which may represent high churn risk) could be targeted with retention campaigns.
    - Customers in the purple cluster (perhaps representing loyal customers) could be offered loyalty rewards.

## Connecting Both Graphs:

* + **Elbow Method and Optimal Clustering**: The elbow graph tells us that 4 clusters are the best fit for this dataset, balancing simplicity and accuracy.
  + **PCA Visualization**: The PCA plot visually confirms that the clustering solution separates customers into 4 distinct groups, with varying degrees of spread and overlap. This suggests that K-Means clustering has successfully segmented the customers based on meaningful patterns in the data.

## Potential Next Steps:

1. **Feature Importance Analysis**: Analyze which features (like Monthly Charges, Tenure, etc.) are driving the separation of clusters. This will help you interpret the characteristics of each customer segment more precisely.
2. **Cluster Profiling**: Assign profiles or descriptions to each cluster based on key distinguishing features. For example:
   * Cluster 1: "High-Spending Loyal Customers"
   * Cluster 2: "Price-Sensitive Short-Term Customers"
3. **Business Strategy**: Use the clusters to inform business decisions such as marketing, product design, and customer retention strategies.