**Data Preparation and Normalisation**

**Data Preparation and Scaling Techniques**

# 1. Overview of the Code

This code takes a dataset as input and performs these data preprocessing steps to the given dataset. This includes loading data, missing value treatment, categorical variable encoding, data splitting, feature scaling, and visualization of data distributions before and after scaling.

# 2. Libraries Used

* **pandas**: For data manipulation and analysis.
* **numpy**: Provides support for numerical operations.
* **sklearn**: A machine learning library used for model selection, preprocessing, and scaling.
  + **train\_test\_split**: Splits the data into training and testing sets.
  + **StandardScaler**: Used for feature scaling (standardization).
  + **LabelEncoder**: Converts categorical data to numeric values.
* **seaborn**: For statistical data visualization.
* **matplotlib**: For creating plots and visualizing data.
* **style**: A module from matplotlib used to set the style of plots.

# 3. Step-by-Step Explanation of the Code

## 3.1 Load the Dataset

***file\_path = "/Users/bikeshkhadka/Downloads/Dataset\_ATS\_v2.csv"***

***df = pd. read\_CSV (file\_path)***

Load the dataset from a CSV file with pandas Customer Info which presumably contains customer related attributes, Gender, Dependents, Phone service, Internet service, Contract type and target variable (Churn)

## 3.2 Display Basic Information About the Dataset

## *print(df.info())*

***print(df.head())***

* **info()**: Displays the structure of the dataset, including column names, data types, and the number of non-null entries.
* **describe()**: Provides summary statistics for numerical columns (e.g., mean, standard deviation, min, and max values).

## 3.3 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.

## 3.4 Encode Categorical Variables

***from sklearn.preprocessing import LabelEncoder***

***binary\_cols = ['gender', 'Dependents', 'PhoneService', 'Churn']***

***le = LabelEncoder ()***

***for col in binary\_cols:***

***df[coll = le.fit\_transform(df[coll)***

***df = pd.get\_dummies(df, columns=['MultipleLines', 'InternetService', 'Contract'], drop\_first=True)***

***print("\nData After Encoding:")***

**Label Encoding**:

* It is used to convert categorical variables into numerical values. For instance, "Male" and "Female" in the gender column could be encoded as 0 and 1, respectively.
* This is needed because numeric data usually performs better with machine learning model than does categorical data.

## 3.5 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 holds all the features (independent variables), and y holds the target variable (Churn) which we want to predict.
* **train\_test\_split()**: Divides the dataset into training (80%) and testing (20%) sets. This guarantees that no overfitting occurs, as the model will be trained on a part of the data and tested on an entirely different one.

# 4. Emphasis on Scaling Techniques

**4.1 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 to ensure that the data has mean of 0 and 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. It helps to ensure that features with large values don’t have more influence than features with smaller values.

**4.1.1 How StandardScaler Works**

* **StandardScaler** transforms the data by:

# 𝑍 =

where:

o **X** is the feature,

o **μ** is the mean of the feature values in the training data,

o **σ** is the standard deviation.

The scaler first  fits on X\_train which are the mean and standard deviation, and then applies the transformation (fit\_transform) The same mean and standard deviation values from X\_train are used to scale X\_test (transform), resulting in consistent scaling between the training and testing sets.

## 4.2 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 part is visualizing the distribution of a specific feature (MonthlyCharges) as a before and after the Scale.
* **Before Scaling**: Distribution of MonthlyCharges as it appears in the original dataset. The values can be all over the place.
* **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.

## 5. Summary

This code snippet walks through some very important tasks to prepare a dataset for machine learning with an emphasis on scaling techniques. The preprocessing includes these steps:

* **Data Loading**: Load the data using pandas to a DataFrame.
* **Data Cleaning**:  It performs forward-fill to fill the missing values.
* **Label Encoding**: The categorical variables are converted into numeric form by applying LabelEncoder.
* **Train-Test Split**: This allows to split the dataset into train and test sets.
* **Scaling**: StandardScaler is used to get all of features with mean=0 and standard deviation=1

The **scaling** step is essential for many machine learning algorithms as it helps bring the features on the same scale and with that, improve the performance and stability of the models. By visualizing data before and after scaling, you can clearly see the effect of normalization on feature distributions.