US-2.7 — Feature Scaling & Normalization Documentation

1. Objective

The purpose of this task was to normalize key numerical features so that differences in scale do not bias the performance of the Neural Network model. Features with large magnitudes (such as charges) can dominate gradient updates and negatively affect convergence. Scaling ensures all selected features contribute proportionally during model training.

---

2. Features Selected for Scaling

The following numerical features were identified as requiring normalization:

- **MonthlyCharges —** Monthly subscription cost
- **TotalCharges —** Total revenue generated by the customer
- **tenure —** Number of months the customer has stayed

"TotalCharges" was not explicitly available in the source dataset and was therefore derived using "MonthlyCharges" and "tenure", which is consistent with standard telecom billing calculations.

"StandardScaler" from scikit-learn was used to transform these features to a common scale with zero mean and unit variance. This approach was chosen to preserve relative differences while ensuring stable and unbiased model convergence.

These features are measured on very different numeric ranges, making them strong candidates for scaling.

---

3. Choice of Scaling Technique

Selected Method: StandardScaler

The **StandardScaler** from Scikit-Learn was selected instead of MinMaxScaler.

Reason for Choosing StandardScaler

Factor	Justification
Model Type	Neural Networks perform better when input features are centered around zero
Data Distribution	Charges and tenure are not strictly bounded and may contain outliers
Stability	Standardization (mean = 0, std = 1) improves gradient descent convergence
Generalization	Reduces risk of one feature dominating due to scale

StandardScaler transforms data using:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

Where
μ = feature mean
σ = feature standard deviation

---

4. Implementation Code

Step 1 — Load Dataset

import pandas as pd

from sklearn.preprocessing import StandardScaler

Raw DATA PATH= "/content/drive/MyDrive/Teleco-Customer-Churn-Analysis/Dataset_ATS_v3_with_TotalCharges.csv"

df = pd.read_csv(f"{DATA_PATH}/Dataset_ATS_v3_with_TotalCharges.csv")

Step 2 — Select Features to Scale

scale_cols = ['MonthlyCharges', 'TotalCharges', 'tenure']

Step 3 — Apply StandardScaler

scaler = StandardScaler()

df[scale_cols] = scaler.fit_transform(df[scale_cols])

Step 4 — Verify Scaling

df[scale_cols].describe()

Expected outcome:

- Mean ≈ 0
- Standard Deviation ≈ 1

Step 5 — Save Scaled Dataset

OUTPUT_PATH = "/content/drive/MyDrive/Teleco-Customer-Churn-Analysis/ "

df.to_csv(f"{OUTPUT_PATH}/Dataset_Scaled_v1.csv", index=False)

---

5. Result Comparison

BEFORE

Out[ ]:

	MonthlyCharges	TotalCharges	tenure
count	6741.000000	6741.000000	6741.000000
mean	65.843495	2334.699006	32.945112
std	29.680059	2258.012286	24.333994
min	18.000000	0.000000	0.000000
25%	41.000000	450.000000	10.000000
50%	71.000000	1449.000000	30.000000
75%	90.000000	3871.000000	56.000000
max	119.000000	8568.000000	72.000000

AFTER

Out[ ]:

	MonthlyCharges	TotalCharges	tenure
count	6.741000e+03	6.741000e+03	6.741000e+03
mean	-2.160826e-17	2.318935e-17	-4.427058e-17
std	1.000074e+00	1.000074e+00	1.000074e+00
min	-1.612094e+00	-1.034039e+00	-1.353972e+00
25%	-8.371054e-01	-8.347336e-01	-9.429941e-01
50%	1.737492e-01	-3.922763e-01	-1.210377e-01
75%	8.139572e-01	6.804281e-01	9.475057e-01
max	1.791117e+00	2.760731e+00	1.605071e+00

6. Verification Results

After scaling:

- All selected features are centered around **0**
- Standard deviation for each feature is approximately **1**
- No feature dominates due to magnitude differences

This confirms the dataset is now suitable for Neural Network training.

---

7. Outcome

The dataset was successfully normalized and saved as:

Data_Preparation/data/processed/Dataset_Scaled_v1.csv

This file is now ready for downstream tasks such as class balancing and model training.