SS_Task1

November 21, 2025

1 TASK 1 — DATA PREPARATION

1. Loading the dataset

1.1 Import Libraries & Load Dataset from GitHub

[1]:
```
import pandas as pd
import numpy as np

# GitHub RAW link
url = "https://raw.githubusercontent.com/ShivaHariny07/SaiKet_System_Internship/
 ↪main/Telco_Customer_Churn_Dataset%20%20(3).csv"

# Load dataset
data = pd.read_csv(url)

print(" Dataset Loaded Successfully!")
```

```
 Dataset Loaded Successfully!
```

1.2 Display First 5 Rows

[2]:
```
print(" First Five Rows of Dataset:\n")
data.head()
```

```
 First Five Rows of Dataset:
```

[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	\
0	7590-VHVEG	Female	0	Yes	No	1	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	
4	9237-HQITU	Female	0	No	No	2	Yes	

	MultipleLines	InternetService	OnlineSecurity	…	DeviceProtection	\
0	No phone service	DSL	No	…	No	
1	No	DSL	Yes	…	Yes	
2	No	DSL	Yes	…	No	
3	No phone service	DSL	Yes	…	Yes	

```
4                No     Fiber optic             No  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

1.3 Shape of the Dataset

[3]:
```
print(" Shape of Dataset:", data.shape)
```

```
 Shape of Dataset: (7043, 21)
```

2. Initial Data Exploration

2.1 Dataset Information

[4]:
```
print("\n Dataset Info:\n")
data.info()
```

```
 Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
```

```
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

2.2 Statistical Summary

[5]:
```
print("\n  Statistical Summary:\n")
data.describe(include='all')
```

```
 Statistical Summary:
```

[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
count	7043	7043	7043.000000	7043	7043	7043.000000	
unique	7043	2	NaN	2	2	NaN	
top	3186-AJIEK	Male	NaN	No	No	NaN	
freq	1	3555	NaN	3641	4933	NaN	
mean	NaN	NaN	0.162147	NaN	NaN	32.371149	
std	NaN	NaN	0.368612	NaN	NaN	24.559481	
min	NaN	NaN	0.000000	NaN	NaN	0.000000	
25%	NaN	NaN	0.000000	NaN	NaN	9.000000	
50%	NaN	NaN	0.000000	NaN	NaN	29.000000	
75%	NaN	NaN	0.000000	NaN	NaN	55.000000	
max	NaN	NaN	1.000000	NaN	NaN	72.000000	

	PhoneService	MultipleLines	InternetService	OnlineSecurity	…	\
count	7043	7043	7043	7043	…	
unique	2	3	3	3	…	
top	Yes	No	Fiber optic	No	…	
freq	6361	3390	3096	3498	…	
mean	NaN	NaN	NaN	NaN	…	
std	NaN	NaN	NaN	NaN	…	
min	NaN	NaN	NaN	NaN	…	
25%	NaN	NaN	NaN	NaN	…	
50%	NaN	NaN	NaN	NaN	…	
75%	NaN	NaN	NaN	NaN	…	
max	NaN	NaN	NaN	NaN	…	

```
       DeviceProtection TechSupport StreamingTV StreamingMovies  \
count              7043        7043        7043             7043
unique                3           3           3                3
top                  No          No          No               No
freq               3095        3473        2810             2785
mean                NaN         NaN         NaN              NaN
std                 NaN         NaN         NaN              NaN
min                 NaN         NaN         NaN              NaN
25%                 NaN         NaN         NaN              NaN
50%                 NaN         NaN         NaN              NaN
75%                 NaN         NaN         NaN              NaN
max                 NaN         NaN         NaN              NaN

              Contract PaperlessBilling     PaymentMethod MonthlyCharges  \
count             7043             7043              7043    7043.000000
unique               3                2                 4            NaN
top     Month-to-month              Yes  Electronic check            NaN
freq              3875             4171              2365            NaN
mean               NaN              NaN               NaN      64.761692
std                NaN              NaN               NaN      30.090047
min                NaN              NaN               NaN      18.250000
25%                NaN              NaN               NaN      35.500000
50%                NaN              NaN               NaN      70.350000
75%                NaN              NaN               NaN      89.850000
max                NaN              NaN               NaN     118.750000

       TotalCharges Churn
count          7043  7043
unique         6531     2
top                   No
freq             11  5174
mean            NaN   NaN
std             NaN   NaN
min             NaN   NaN
25%             NaN   NaN
50%             NaN   NaN
75%             NaN   NaN
max             NaN   NaN

[11 rows x 21 columns]
```

2.3 Data Types

[6]:
```
print("\n  Column Data Types:\n")
data.dtypes
```

```
 Column Data Types:
```

[6]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

2.4 Missing Value Count

[7]:
```
print("\n  Missing Values in Each Column:\n")
data.isnull().sum()
```

```
 Missing Values in Each Column:
```

[7]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
```

```
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

3. Handling Missing Values

3.1 Convert Blank Spaces → Null

[8]:
```
# Replace blank " " with actual NaN
data = data.replace(" ", np.nan)

print(" Missing values after cleaning spaces:\n")
data.isnull().sum()
```

```
 Missing values after cleaning spaces:
```

[8]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

3.2 Convert TotalCharges to Numeric

[9]:
```
data["TotalCharges"] = pd.to_numeric(data["TotalCharges"], errors='coerce')
```

3.3 Fill Missing TotalCharges with Median

[10]:
```
median_value = data["TotalCharges"].median()
data["TotalCharges"] = data["TotalCharges"].fillna(median_value)

print("  Missing values after filling TotalCharges:\n")
data.isnull().sum()
```

```
 Missing values after filling TotalCharges:
```

[10]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

4. Encoding Categorical Variables

4.1 Identify Categorical Columns

[11]:
```
from sklearn.preprocessing import LabelEncoder

categorical_cols = data.select_dtypes(include=['object']).columns
print("  Categorical Columns:\n", list(categorical_cols))
```

```
 Categorical Columns:
 ['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService',
'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling', 'PaymentMethod', 'Churn']
```

4.2 Separate Binary & Multi-category Columns

[12]:
```
# Binary columns (Yes/No)
binary_cols = [col for col in categorical_cols if data[col].nunique() == 2]

# Multi-category columns (>2 unique values)
multi_cat_cols = [col for col in categorical_cols if data[col].nunique() > 2]

print("\n Binary Columns:", binary_cols)
print(" Multi-Category Columns:", multi_cat_cols)
```

```
 Binary Columns: ['gender', 'Partner', 'Dependents', 'PhoneService',
'PaperlessBilling', 'Churn']
 Multi-Category Columns: ['customerID', 'MultipleLines', 'InternetService',
'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod']
```

4.3 Label Encoding (Binary Columns)

[13]:
```
le = LabelEncoder()

for col in binary_cols:
    data[col] = le.fit_transform(data[col])
```

4.4 One-Hot Encoding (Multi-category Columns)

[14]:
```
data = pd.get_dummies(data, columns=multi_cat_cols, drop_first=True)

print("\n Encoding Completed!")
print(" New Dataset Shape:", data.shape)
```

```
 Encoding Completed!
 New Dataset Shape: (7043, 7073)
```

5. Dataset Splitting (Train/Test)

5.1 Separate Target Variable

[15]:
```
y = data["Churn"]
X = data.drop("Churn", axis=1)
```

5.2 Train-Test Split

[16]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(" Training Set Shape:", X_train.shape)
print(" Testing Set Shape:", X_test.shape)
```

```
Training Set Shape: (5634, 7072)
Testing Set Shape: (1409, 7072)
```