CL-I DMV 9

July 16, 2025

[1]:
```
''' NAME:Aher Swami Sandip
ROLL NO.01
COURSE: AI&DS
CLASS: BE
SUB:Computer Laboratory-I (DMV) '''
```

0.1 Data Cleaning and Preparation

[1]:
```
# Problem Statement: Analyzing Customer Churn in a Telecommunications Company
# Dataset: "Telecom_Customer_Churn.csv"
# Description: The dataset contains information about customers of a␣
 ↪telecommunications
# company and whether they have churned (i.e., discontinued their services).␣
 ↪The dataset
# includes various attributes of the customers, such as their demographics,␣
 ↪usage patterns, and
# account information. The goal is to perform data cleaning and preparation to␣
 ↪gain insights
# into the factors that contribute to customer churn.
# Tasks to Perform:
```

0.1.1 1. Import the "Telecom_Customer_Churn.csv" dataset.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("Telco-Customer-Churn.csv")
```

0.1.2 2. Explore the dataset to understand its structure and content.

[3]:
```
df.head()
```

[3]:
```
   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]
```

[5]:
```
df.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
```

[7]:
```python
df.describe()
```

[7]:
```
       SeniorCitizen       tenure  MonthlyCharges
count    7043.000000  7043.000000     7043.000000
mean        0.162147    32.371149       64.761692
std         0.368612    24.559481       30.090047
min         0.000000     0.000000       18.250000
25%         0.000000     9.000000       35.500000
50%         0.000000    29.000000       70.350000
75%         0.000000    55.000000       89.850000
max         1.000000    72.000000      118.750000
```

0.1.3 3. Handle missing values in the dataset, deciding on an appropriate strategy.

[9]:
```python
# Convert "TotalCharges" to numeric, handling any conversion errors.
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Drop rows with missing values in "TotalCharges."
df.dropna(subset=['TotalCharges'], inplace=True)
```

0.1.4 4. Remove any duplicate records from the dataset.

[11]:
```python
# Remove duplicates
df.drop_duplicates(inplace=True)
```

0.1.5 5. Check for inconsistent data, such as inconsistent formatting or spelling variations,and standardize it.

[13]:
```python
# Check for inconsistent data by looking at unique values in each column
for column in df.columns:
    print(f"Unique values in {column}:")
    print(df[column].unique())
    print("\n")
```

```
Unique values in customerID:
['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' … '4801-JZAZL' '8361-LTMKD'
 '3186-AJIEK']


Unique values in gender:
```

```
['Female' 'Male']


Unique values in SeniorCitizen:
[0 1]


Unique values in Partner:
['Yes' 'No']


Unique values in Dependents:
['No' 'Yes']


Unique values in tenure:
[ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]


Unique values in PhoneService:
['No' 'Yes']


Unique values in MultipleLines:
['No phone service' 'No' 'Yes']


Unique values in InternetService:
['DSL' 'Fiber optic' 'No']


Unique values in OnlineSecurity:
['No' 'Yes' 'No internet service']


Unique values in OnlineBackup:
['Yes' 'No' 'No internet service']


Unique values in DeviceProtection:
['No' 'Yes' 'No internet service']


Unique values in TechSupport:
['No' 'Yes' 'No internet service']
```

```
Unique values in StreamingTV:
['No' 'Yes' 'No internet service']


Unique values in StreamingMovies:
['No' 'Yes' 'No internet service']


Unique values in Contract:
['Month-to-month' 'One year' 'Two year']


Unique values in PaperlessBilling:
['Yes' 'No']


Unique values in PaymentMethod:
['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']


Unique values in MonthlyCharges:
[29.85 56.95 53.85 … 63.1  44.2  78.7 ]


Unique values in TotalCharges:
[  29.85 1889.5   108.15 …  346.45  306.6  6844.5 ]


Unique values in Churn:
['No' 'Yes']
```

[15]:
```
# Example: Standardize 'Yes'/'No' to lowercase
df = df.apply(lambda col: col.map(lambda s: s.lower() if type(s) == str else s))
```

[17]:
```
# Example: Standardize columns with similar values (e.g., 'Male'/'Female' to␣
↪'male'/'female')
if 'gender' in df.columns:
    df['gender'] = df['gender'].str.lower()
```

[19]:
```
# Display the first few rows after standardization
print(df.head())
```

```
   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       1889.50     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]
```

[21]:
```python
# Save the cleaned data
df.to_csv('Telco-Customer-Churn-cleaned.csv', index=False)
```

0.1.6 6. Convert columns to the correct data types as needed.

[23]:
```python
# Convert columns to appropriate data types
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')  #
 ↪Convert to numeric
df['Churn'] = df['Churn'].astype('category')  # Convert to category
df['tenure'] = df['tenure'].astype('int')  # Ensure tenure is integer
```

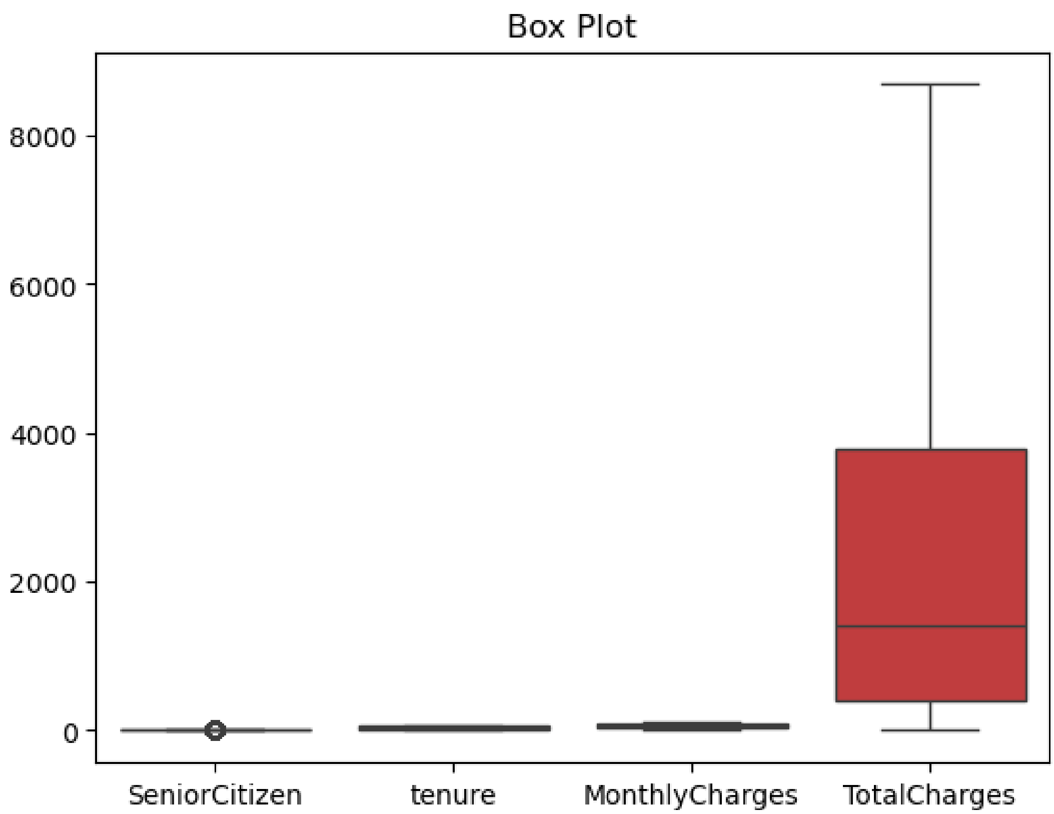
[25]:
```python
df.dtypes
```

[25]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int32
```

```
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         float64
Churn               category
dtype: object
```

0.1.7 7. Identify and handle outliers in the data.

[27]:
```
import seaborn as sns
# Box Plot
sns.boxplot(data=df)
plt.title("Box Plot")
plt.show()
```



0.1.8 8. Perform feature engineering, creating new features that may be relevant to predicting customer churn.

[29]:
```
# Create age group feature (replace thresholds as needed)
df['age_group'] = np.where(df['SeniorCitizen'] == 1, 'Senior', np.
 ↪where(df['tenure'] > 60, 'Middle-aged', 'Young'))

# Feature creation based on numerical features
# Tenure groups (replace thresholds as needed)
df['tenure_group'] = np.where(df['tenure'] <= 12, 'New Customer', 'Loyal␣
 ↪Customer')

# Average monthly spend
df['avg_monthly_spend'] = df['MonthlyCharges'] / df['tenure']
```

0.1.9 9. Normalize or scale the data if necessary.

[31]:
```
# Feature scaling or normalization (optional, consider data distribution)
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges',␣
 ↪'avg_monthly_spend'] # Adjust as needed
df[numerical_features] = scaler.fit_transform(df[numerical_features])
```

0.1.10 10. Split the dataset into training and testing sets for further analysis.

[33]:
```
from sklearn.model_selection import train_test_split

# Assuming your data is in a DataFrame named 'df' and your target variable is␣
 ↪'Churn' (replace if different)
X = df.drop('Churn', axis=1)  # Features (all columns except 'Churn')
y = df['Churn']  # Target variable

# Split data into training and testing sets (common split is 80/20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

print("Training data shapes:")
print(X_train.shape, y_train.shape)

print("Testing data shapes:")
print(X_test.shape, y_test.shape)
```

```
Training data shapes:
(5625, 23) (5625,)
Testing data shapes:
(1407, 23) (1407,)
```

[35]:
```
#cleaned data is in a DataFrame named 'df'
df.to_csv('cleaned_data.csv', index=False)  # Export without index
print("Cleaned data exported to cleaned_data.csv")
```

```
Cleaned data exported to cleaned_data.csv
```

[ ]:
```
```