Experiment No. 09

Aim : Data Cleaning and Preparation.

Problem Statement : Analyzing Customer Churn in a Telecommunications Company, and perform following tasks – 1. Import the dataset. 2. Explore the dataset. 3. Handle missing values. 4. Remove any duplicates. 5. Check for inconsistent data. 6. Convert columns to the correct data types. 7. Identify and handle outliers. 8. Perform feature engineering. 9. Normalize or scale the data. 10. Split the dataset into training and testing sets. and 11. Export the cleaned dataset.

Dataset: "Telecom_Customer_Churn.csv"

Software Requirements : Python and Jupyter notebook.

Hardware Requirements : 8 GB RAM, Intel I5 Processor, and Storage.

Objectives : i) Identify and handle missing values. ii) Convert categorical variables to numerical formats and scale numerical features. iii) Create a new feature that might be useful for predicting customer churn.

Theroy : Data cleaning and prepration are essential steps in any data analysis project. They ensure that the dataset is accurate, consistent, and ready for further analysis and modelling.

1. **Data Cleaning**

Data cleaning is an essential step in data analysis that involves preparing and improving raw data to make it suitable for analysis. Here’s a broad overview of the process:

1. **Remove Duplicates**: Ensure there are no redundant records in your dataset. Duplicates can skew results and analyses.
2. **Handle Missing Values**: Address gaps in data by either filling them in with a statistical measure (mean, median) or by using algorithms that can handle missing values, or by removing the incomplete records if appropriate.
3. **Correct Errors**: Identify and fix errors or inconsistencies in the data. This includes correcting typos, standardizing formats (e.g., date formats), and ensuring data consistency.
4. **Normalize Data**: Transform data into a common format or scale. For example, converting all text to lowercase, or standardizing units of measurement.

5. **Filter Outliers**: Detect and handle outliers—data points that significantly deviate from the norm. Depending on your analysis, you might choose to remove or adjust them.
6. **Validate Data**: Ensure that the data conforms to the expected format and constraints. This could include checking data types, ranges, or valid values.
7. **Standardize Data**: Ensure consistency in the dataset. For example, standardizing names, address formats, or categorical variables.
8. **Integrate Data**: Combine data from multiple sources and ensure they are compatible and consistent.
9. **Create Data Documentation**: Document the cleaning process, including any transformations or modifications made to the data. This ensures transparency and helps in future data management tasks.
10. **Automate Where Possible**: Use scripts or data cleaning tools to automate repetitive tasks, which can save time and reduce errors.

2. Data Transformation

Data transformation is a crucial process in data preparation that involves converting data from its original format or structure into a format that is more suitable for analysis or further processing. Here’s a detailed look at common data transformation techniques:

1. **Normalization**: Adjusting the scale of numerical data to ensure uniformity. For example, scaling values to a range between 0 and 1 or converting them to z-scores.
2. **Standardization**: Transforming data to have a mean of 0 and a standard deviation of This is particularly useful when comparing data that originally had different units or scales.
3. **Aggregation**: Combining multiple data points into a summary metric. For instance, calculating the average, sum, or count of values over a period or across categories.
4. **Encoding**: Converting categorical data into numerical format. Common methods include one-hot encoding (creating binary columns for each category) or label encoding (assigning a unique number to each category).

3. Exploratory Data Analysis [EDA]

Exploratory Data Analysis (EDA) is a crucial phase in data analysis that involves examining and understanding the data before applying formal statistical or machine learning models. The goal of EDA is to uncover patterns, spot anomalies, test hypotheses, and check assumptions through various techniques.

4. **Feature Engineering**

   Feature engineering is a crucial step in the data preprocessing phase of machine learning and data analysis. It involves creating, modifying, or selecting features (variables) from raw data to improve the performance and effectiveness of predictive models.

Implementation

Step No.1 - Import the Dataset.

```
import pandas as pd
import numpy as np
df = pd.read_csv(r"C:\Users\saira\Downloads\Telco-Customer-Churn.csv")
df.head()
```

```
customerID	gender	SeniorCitizen	Partner	Dependents	tenure
	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
	DeviceProtection	TechSupport	StreamingTV	StreamingMovies
	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	No
phone service	DSL	No	...	No	No	No	No	Month-
to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No
	DSL	Yes	...	Yes	No	No	No	One year	No
	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
	DSL	Yes	...	No	No	No	No	Month-to-month
	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No
phone service	DSL	Yes	...	Yes	Yes	No	No	One
year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	No
	Fiber optic	No	...	No	No	No	No	Month-
to-month	Yes	Electronic check	70.70	151.65	Yes
5 rows × 21 columns
```

```
df.tail()
```

```
customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
```

TechSupport StreamingTV StreamingMovies Contract PaperlessBilling PaymentMethod MonthlyCharges TotalCharges Churn

7038 6840-RESVB Male 0 Yes Yes 24 Yes Yes DSL Yes ... Yes Yes Yes Yes One year Yes Mailed check 84.80 1990.5 No

7039 2234-XADUHFemale0 Yes Yes 72 Yes Yes Fiber optic No ... Yes No Yes Yes One year Yes Credit card (automatic) 103.20 7362.9 No

7040 4801-JZAZL Female0 Yes Yes 11 No No phone service DSL Yes ... No No No No Month-to-month Yes Electronic check 29.60 346.45 No

7041 8361-LTMKD Male 1 Yes No 4 Yes Yes Fiber optic No ... No No No No Month-to-month Yes Mailed check 74.40 306.6 Yes

7042 3186-AJIEK Male 0 No No 66 Yes No Fiber optic Yes ... Yes Yes Yes Yes Two year

Step No.2 - Explore the Dataset.

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
df.columns
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
      'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
      'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
      'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
      'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
```

dtype='object')

df.describe()

	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

Step No.3 - Handle Missing Values.

df.isnull().sum()

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

Step No.4 - Remove Duplicate Records.

df.duplicated()

0 False

1 False

2 False

3 False

4 False

...

7038 False

7039 False

7040 False

7041 False

7042 False

```
Length: 7043, dtype: bool

df = df.drop_duplicates()

df.duplicated().sum()

0
```

Step No.5 - Check for Inconsistent Data.

```
def standardize_text(df, text_columns):
    for col in text_columns:
        df[col] = df[col].str.strip().str.lower()
    return df

text_columns = df.select_dtypes(include='object').columns

text_columns

Index(['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService',
       'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'Contract', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges',
       'Churn'],
      dtype='object')

df = standardize_text(df, text_columns)

df.head()

customerID gender SeniorCitizen Partner Dependents tenure PhoneService
    MultipleLines InternetService OnlineSecurity ... DeviceProtection
    TechSupport StreamingTV StreamingMovies Contract PaperlessBilling
    PaymentMethod MonthlyCharges TotalCharges Churn

0 7590-vhveg female 0 yes no 1 no no phone service dsl
    no ... no no no no month-to-month yes electronic
check 29.85 29.85 no
```

1 5575-gnvde male 0 no no 34 yes no dsl yes ...
yes no no no one year no mailed check 56.95 1889.5 no

2 3668-qpybk male 0 no no 2 yes no dsl yes ...
no no no no month-to-month yes mailed check 53.85
108.15 yes

3 7795-cfocw male 0 no no 45 no no phone service dsl
yes ... yes yes no no one year no bank transfer
(automatic) 42.30 1840.75 no

4 9237-hqitu female 0 no no 2 yes no fiber optic no
... no no no no month-to-month yes electronic check
70.70 151.65 yes

5 rows × 21 columns

Step No.6 - Convert Columns to Correct Data Types.

df.dtypes

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
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['SeniorCitizen'] = df['SeniorCitizen'].astype(bool)
df.dtypes
customerID         object
gender             object
SeniorCitizen      bool
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        float64
Churn               object
dtype: object
```

Step No.7 - Identify and Handle Outliers.

```
# Function to identify outliers using the IQR method
def identify_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]
# Identify outliers for each numerical column
numerical_columns = df.select_dtypes(include=[np.number]).columns
outliers = {col: identify_outliers_iqr(df, col) for col in numerical_columns}
```

```
# Display the number of outliers for each numerical column
outlier_counts = {col: len(outliers[col]) for col in outliers}
outlier_counts
{'tenure': 0, 'MonthlyCharges': 0, 'TotalCharges': 0}
# Function to remove outliers using the IQR method
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
# Remove outliers for each numerical column
for col in numerical_columns:
    df = remove_outliers_iqr(df, col)
# Display the dataframe shape after outlier removal
df.shape
(7032, 21)
```

Step No.8 - Perform Feature Engineering.

```
# Create a new feature for total services count
service_columns = [
    'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
    'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies'
]
```

```
df['TotalServices'] = df[service_columns].apply(lambda x: x.eq('yes').sum(), axis=1)

# Create a new feature for the ratio of MonthlyCharges to TotalCharges

df['ChargesRatio'] = df['MonthlyCharges'] / (df['TotalCharges'] + 1)  # Add 1 to avoid division by zero

# Create tenure groups

def tenure_group(tenure):

    if tenure <= 12:

        return '0-1 year'

    elif tenure <= 24:

        return '1-2 years'

    elif tenure <= 48:

        return '2-4 years'

    elif tenure <= 60:

        return '4-5 years'

    else:

        return '5+ years'

df['TenureGroup'] = df['tenure'].apply(tenure_group)

# Display the first few rows to verify the new features

df[['TotalServices', 'ChargesRatio', 'TenureGroup']].head()
```

	TotalServices	ChargesRatio	TenureGroup
0	1	0.967585	0-1 year
1	3	0.030124	2-4 years
2	3	0.493358	0-1 year
3	3	0.022967	2-4 years
4	1	0.463151	0-1 year

```
# Step No.9 - Normalize or Scale the Data.
from sklearn.preprocessing import MinMaxScaler
# List of numerical columns to scale
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
# Initialize the scaler
scaler = MinMaxScaler()
# Apply the scaler to the numerical columns
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
# Display the first few rows to verify the changes
df[numerical_columns].head()
```

	tenure	MonthlyCharges	TotalCharges
0	0.000000	0.115423	0.001275
1	0.464789	0.385075	0.215867
2	0.014085	0.354229	0.010310
3	0.619718	0.239303	0.210241
4	0.014085	0.521891	0.015330

```
from sklearn.preprocessing import StandardScaler
# List of numerical columns to scale
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
# Initialize the scaler
scaler = StandardScaler()
# Apply the scaler to the numerical columns
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
# Display the first few rows to verify the changes
```

df[numerical_columns].head()

	tenure	MonthlyCharges	TotalCharges
0	-1.280248	-1.161694	-0.994194
1	0.064303	-0.260878	-0.173740
2	-1.239504	-0.363923	-0.959649
3	0.512486	-0.747850	-0.195248
4	-1.239504	0.196178	-0.940457

Step No.10 - Split the Dataset into Training and Testing Sets.

from sklearn.model_selection import train_test_split

Separate features (X) and target variable (y)

X = df.drop('Churn', axis=1) # Features

y = df['Churn'] # Target variable

Split the dataset into training and testing sets

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

Display the shapes of the resulting datasets

(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

((5625, 23), (1407, 23), (5625,), (1407,))

Step no.11 - Export the Cleaned Dataset.

df.to_csv("Cleaned_Telecom_Customer_Churn.csv", index=False)

Conclusion : We can successfully Analysing customer churn in a Telecommunications company and also we can easily data cleaning, prepration, and visualize the data on a “Telecom_customer_churn.csv”.