EXPERIMENT 3

Aim:

Experiment to preprocess dataset using different preprocessing techniques.

Theory:

What is data cleaning?

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

How to Treat Outliers?

There are several ways to treat outliers in a dataset, depending on the nature of the outliers and the problem being solved. Here are some of the most common ways of treating outlier values.

- **Trimming:** It excludes the outlier values from our analysis. By applying this technique, our data becomes thin when more outliers are present in the dataset. Its main advantage is its fastest nature.
- **Capping:** In this technique, we cap our outliers data and make the limit i.e, above a particular value or less than that value, all the values will be considered as outliers, and the number of outliers in the dataset gives that capping number. For example, if you're working on the income feature, you might find that people above a certain income level behave similarly to those with a lower income. In this case, you can cap the income value at a level that keeps that intact and accordingly treat the outliers. Treating outliers as a missing value: By assuming outliers as the missing observations, treat them accordingly, i.e., same as missing values imputation.
- **Discretization:** In this technique, by making the groups, we include the outliers in a particular group and force them to behave in the same manner as those of other points in that group. This technique is also known as Binning.

How to Detect Outliers?

- **For Normal Distributions**

Use empirical relations of Normal distribution.

The data points that fall below mean-3*(sigma) or above mean+3*(sigma) are outliers, where mean and sigma are the average value and standard deviation of a particular column.

- **For Skewed Distributions**

Use Interquartile Range (IQR) proximity rule. The data points that fall below Q1 – 1.5 IQR or above the third quartile Q3 + 1.5 IQR are outliers, where Q1 and Q3 are the 25th and 75th percentile of the dataset, respectively. IQR represents the interquartile range and is given by Q3 – Q1.

- **For Other Distributions**

Use a percentile-based approach. For Example, data points that are far from the 99% percentile and less than 1 percentile are considered an outlier.

Data transformation:-

Data transformation in data mining refers to the process of converting raw data into a format that is suitable for analysis and modeling. The goal of data transformation is to prepare the data for data mining so that it can be used to extract useful insights and knowledge. Data transformation typically involves several steps, including:

- **Data cleaning:** Removing or correcting errors, inconsistencies, and missing values in the data.
- **Data integration:** Combining data from multiple sources, such as databases and spreadsheets, into a single format.
- **Data normalization:** Scaling the data to a common range of values, such as between 0 and 1, to facilitate comparison and analysis.
- **Data reduction:** Reducing the dimensionality of the data by selecting a subset of relevant features or attributes.
- **Data discretization:** Converting continuous data into discrete categories or bins.

Data aggregation:

Combining data at different levels of granularity, such as by summing or averaging, to create new features or attributes. Data transformation is an important step in the data mining process as it helps to ensure that the data is in a format that is suitable for analysis and modeling, and that it is free of errors and inconsistencies. Data transformation can also help to improve the performance of data mining algorithms, by reducing the dimensionality of the data, and by scaling the data to a common range of values. The data are transformed in ways that are ideal for mining the data. The data transformation involves steps that are:

1. **Smoothing:** It is a process that is used to remove noise from the dataset using some algorithms It allows for highlighting important features present in the dataset. It helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form. The concept behind data smoothing is that it will be able to identify simple changes to help predict different trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn’t see otherwise.
2. **Aggregation:** Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used. Gathering accurate data of high quality and a large enough quantity is necessary to produce relevant results. The collection of data is useful for everything from decisions concerning financing or business strategy of the product, pricing, operations, and marketing strategies. For example, Sales data may be aggregated to compute monthly & annual total amounts.
3. **Discretization:** It is a process of transforming continuous data into a set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes. Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values. For example, (1-10, 11-20) (age:- young, middle age, senior).
4. **Attribute Construction:** Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.
5. **Generalization:** It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example Age initially in Numerical form (22, 25) is converted into categorical value (young, old). For example, Categorical attributes, such as house addresses, may be generalized to higher-level definitions, such as town or country.
6. **Normalization:** Data normalization involves converting all data variables into a given range. Techniques that are used for normalization are:

Min-Max Normalization:

- This transforms the original data linearly.
- Suppose that: min_A is the minima and max_A is the maxima of an attribute, P
- Where v is the value you want to plot in the new range.
- v' is the new value you get after normalizing the old value.

Z-Score Normalization:

- In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation
- A value, v, of attribute A is normalized to v' by computing

Decimal Scaling:

- It normalizes the values of an attribute by changing the position of their decimal points
- The number of points by which the decimal point is moved can be determined by the absolute maximum value of attribute A.
- A value, v, of attribute A is normalized to v' by computing
- where j is the smallest integer such that Max(|v'|) < 1.
- Suppose: Values of an attribute P varies from -99 to 99.
- The maximum absolute value of P is 99.
- For normalizing the values we divide the numbers by 100 (i.e., j = 2) or (number of integers in the largest number) so that values come out to be as 0.98, 0.97 and so on.

Numerosity reduction:

Numerosity reduction is a technique used in data mining to reduce the number of data points in a dataset while still preserving the most important information. This can be beneficial in situations where the dataset is too large to be processed efficiently, or where the dataset contains a large amount of irrelevant or redundant data points.

There are several different numerosity reduction techniques that can be used in data mining, including:

- **Data Sampling:** This technique involves selecting a subset of the data points to work with, rather than using the entire dataset. This can be useful for reducing the size of a dataset while still preserving the overall trends and patterns in the data.
- **Clustering:** This technique involves grouping similar data points together and then representing each group by a single representative data point.

- **Data Aggregation:** This technique involves combining multiple data points into a single data point by applying a summarization function.
- **Data Generalization:** This technique involves replacing a data point with a more general data point that still preserves the important information.
- **Data Compression:** This technique involves using techniques such as lossy or lossless compression to reduce the size of a dataset.

Screenshots of implementation:

1. Importing libraries.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2. Read the dataset.

In [2]:
```
df = pd.read_csv('insurance_policy.csv')
df.head()
```

Out[2]:

	ID	City_Code	Region_Code	Accomodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health Indicator	Holding_Policy_Du
0	1	C3	3213	Rented	Individual	36	36	No	X1	
1	2	C5	1117	Owned	Joint	75	22	No	X2	
2	3	C5	3732	Owned	Individual	32	32	No	NaN	
3	4	C24	4378	Owned	Joint	52	48	No	X1	
4	5	C8	2190	Rented	Individual	44	44	No	X2	

3. Describe the dataset and check the datatype.

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50882 entries, 0 to 50881
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   ID                       50882 non-null  int64  
 1   City_Code                50882 non-null  object 
 2   Region_Code              50882 non-null  int64  
 3   Accomodation_Type        50882 non-null  object 
 4   Reco_Insurance_Type      50882 non-null  object 
 5   Upper_Age                50882 non-null  int64  
 6   Lower_Age                50882 non-null  int64  
 7   Is_Spouse                50882 non-null  object 
 8   Health Indicator         39191 non-null  object 
 9   Holding_Policy_Duration  30631 non-null  object 
 10  Holding_Policy_Type      30631 non-null  float64
 11  Reco_Policy_Cat          50882 non-null  int64  
 12  Reco_Policy_Premium      50882 non-null  float64
 13  Response                 50882 non-null  int64  
dtypes: float64(2), int64(6), object(6)
memory usage: 5.4+ MB
```

```
In [4]: df.describe()
```

Out[4]:

	ID	Region_Code	Upper_Age	Lower_Age	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response
count	50882.000000	50882.000000	50882.000000	50882.000000	30631.000000	50882.000000	50882.000000	50882.000000
mean	25441.500000	1732.788707	44.856275	42.738866	2.439228	15.150859	14183.950069	0.239947
std	14688.512535	1424.081652	17.310271	17.319375	1.025923	6.343378	6590.074873	0.427055
min	1.000000	1.000000	18.000000	16.000000	1.000000	1.000000	2280.000000	0.000000
25%	12721.250000	523.000000	28.000000	27.000000	1.000000	12.000000	9248.000000	0.000000
50%	25441.500000	1391.000000	44.000000	40.000000	3.000000	17.000000	13178.000000	0.000000
75%	38161.750000	2667.000000	59.000000	57.000000	3.000000	20.000000	18096.000000	0.000000
max	50882.000000	6194.000000	75.000000	75.000000	4.000000	22.000000	43350.400000	1.000000

4. Drop the ID column as it won’t be required for prediction.

```
In [5]: df.drop(['ID'], axis=1, inplace=True)
        df.head()
```

Out[5]:

	City_Code	Region_Code	Accomodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health Indicator	Holding_Policy_Duratic
0	C3	3213	Rented	Individual	36	36	No	X1	14
1	C5	1117	Owned	Joint	75	22	No	X2	Na
2	C5	3732	Owned	Individual	32	32	No	NaN	
3	C24	4378	Owned	Joint	52	48	No	X1	14
4	C8	2190	Rented	Individual	44	44	No	X2	

5. Finding the null values.

```
In [12]: df.isnull().sum()

Out[12]: City_Code                    0
         Region_Code                  0
         Accomodation_Type            0
         Reco_Insurance_Type          0
         Upper_Age                    0
         Lower_Age                    0
         Is_Spouse                    0
         Health Indicator         11691
         Holding_Policy_Duration  20251
         Holding_Policy_Type      20251
         Reco_Policy_Cat              0
         Reco_Policy_Premium          0
         Response                     0
         dtype: int64

In [13]: cols = df.columns
```

6. All the unique values of each column are printed and the type of data of each attribute is determined.

In [14]:
```
for i in cols:
    print(i)
    print(df[i].unique(),"\n")
```

```
City_Code
['C3' 'C5' 'C24' 'C8' 'C9' 'C1' 'C15' 'C28' 'C27' 'C7' 'C20' 'C25' 'C4'
 'C2' 'C34' 'C10' 'C17' 'C18' 'C16' 'C29' 'C33' 'C26' 'C19' 'C6' 'C12'
 'C13' 'C11' 'C14' 'C22' 'C23' 'C21' 'C36' 'C32' 'C30' 'C35' 'C31']

Region_Code
[3213 1117 3732 ... 5326 6149 5450]

Accomodation_Type
['Rented' 'Owned']

Reco_Insurance_Type
['Individual' 'Joint']

Upper_Age
[36 75 32 52 44 28 59 21 66 20 27 34 43 55 23 18 22 25 24 40 26 56 35 63
 49 64 67 42 71 57 73 31 19 48 65 54 33 30 69 68 37 29 62 58 38 39 60 41
 45 51 46 70 61 74 53 72 50 47]

Lower_Age
[36 22 32 48 44 52 28 73 43 26 21 47 66 20 27 34 55 23 18 25 24 56 35 63
 64 67 75 42 71 68 31 19 65 54 33 74 30 69 29 62 58 39 60 57 41 40 45 37
 51 59 49 38 46 70 61 53 16 72 50 17]

Is_Spouse
['No' 'Yes']

Health Indicator
['X1' 'X2' nan 'X4' 'X3' 'X6' 'X5' 'X8' 'X7' 'X9']

Holding_Policy_Duration
['14+' nan '1' '3' '5' '9' '14' '7' '2' '11' '10' '8' '6' '4' '13' '12']

Holding_Policy_Type
[ 3. nan  1.  4.  2.]

Reco_Policy_Cat
[22 19  1 21 18  2 16 20  3 11 12 17 13  6  5 15 14  4  7  8 10  9]

Reco_Policy_Premium
[11628. 30510.  7450. ... 25726.  6156. 11374.]

Response
[0 1]
```

1. Categorical Data-

City_Code, Accomodation_Type, Reco_Insurance_Type, Is_Spouse, Health Indicator, Holding_Policy_Duration, Holding_Policy_Type, Reco_Policy_Cat, Response

2. Numerical Data-

Region_Code, Upper_Age, Lower_Age, Reco_Policy_Premium

7. Mode is calculated for the Holding Policy Type column to fill the null values.

```
In [16]: print(df['Holding_Policy_Type'].mode())
         # print(df['Holding_Policy_Type'].value_counts())

0    3.0
Name: Holding_Policy_Type, dtype: float64
```

8. Null values in the Holding Policy Type column are replaced by mode values.

```
In [17]: df['Holding_Policy_Type'].fillna(df['Holding_Policy_Type'].mode()[0], inplace=True)
```

9. Now we can see that no null values are left.

```
In [18]: df['Holding_Policy_Type'].isnull().sum()

Out[18]: 0
```

10. We drop all rows which have null values.

```
In [19]: df.isnull().sum()

Out[19]: City_Code                      0
         Region_Code                    0
         Accomodation_Type              0
         Reco_Insurance_Type            0
         Upper_Age                      0
         Lower_Age                      0
         Is_Spouse                      0
         Health Indicator           11691
         Holding_Policy_Duration    20251
         Holding_Policy_Type            0
         Reco_Policy_Cat                0
         Reco_Policy_Premium            0
         Response                       0
         dtype: int64
```

In [20]:

```
df.dropna(inplace=True)
```

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23548 entries, 0 to 50881
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   City_Code                23548 non-null  object 
 1   Region_Code              23548 non-null  int64  
 2   Accomodation_Type        23548 non-null  object 
 3   Reco_Insurance_Type      23548 non-null  object 
 4   Upper_Age                23548 non-null  int64  
 5   Lower_Age                23548 non-null  int64  
 6   Is_Spouse                23548 non-null  object 
 7   Health Indicator         23548 non-null  object 
 8   Holding_Policy_Duration  23548 non-null  object 
 9   Holding_Policy_Type      23548 non-null  float64
 10  Reco_Policy_Cat          23548 non-null  int64  
 11  Reco_Policy_Premium      23548 non-null  float64
 12  Response                 23548 non-null  int64  
dtypes: float64(2), int64(5), object(6)
memory usage: 2.5+ MB
```

In [22]:

```
df
```

Out[22]:

	City_Code	Region_Code	Accomodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health Indicator	Holding_Policy_Duration
0	C3	3213	Rented	Individual	36	36	No	X1	14+
3	C24	4378	Owned	Joint	52	48	No	X1	14+
4	C8	2190	Rented	Individual	44	44	No	X2	3
5	C9	1785	Rented	Individual	52	52	No	X2	5
7	C1	3175	Owned	Joint	75	73	Yes	X4	9
...	...	...	...	...	...	...	...	...	...
50875	C6	231	Rented	Individual	36	36	No	X3	2
50878	C5	4188	Rented	Individual	27	27	No	X3	7
50879	C1	442	Rented	Individual	63	63	No	X2	14+
50880	C1	4	Owned	Joint	71	49	No	X2	2
50881	C3	3866	Rented	Individual	24	24	No	X3	2

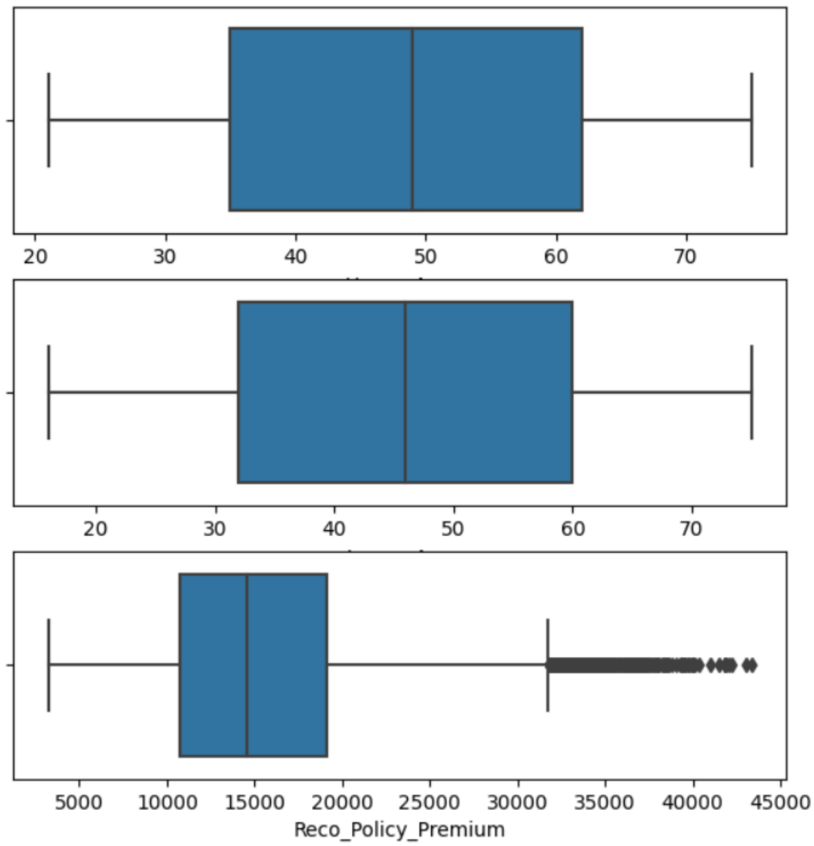
23548 rows × 13 columns

11. We plot boxplot to see the outliers and the distribution of the data.

```
In [13]: cols = ['Upper_Age', 'Lower_Age', 'Reco_Policy_Premium']

         fig, ax = plt.subplots(3,1, figsize = (7, 7))

         for i in range(len(cols)):
             sns.boxplot(df, x=cols[i], ax = ax[i])
         plt.show()
```



12. We use the get_dummies() function, present in pandas library, to convert categorical data to numerical data.

Converting Categorical values to Numerical Values

```
In [14]: df = pd.get_dummies(df, columns=['Accomodation_Type'])
         df.head()
```

Out[14]:

lealth icator	Holding_Policy_Duration	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response	Accomodation_Type_Owned	Accomodation_Type_Rented
X1	14+	3.0	22	11628.0	0	0	1
X1	14+	3.0	22	17780.0	0	1	0
X2	3	1.0	22	10404.0	0	0	1
X2	5	1.0	22	15264.0	1	0	1
X4	9	4.0	22	29344.0	1	1	0

13. We now normalize the data using min-max normalization so that the values are in the range of 0 and 1.

Normalization of Data

```
In [15]: df['Reco_Policy_Premium'] = ((df['Reco_Policy_Premium'] - df['Reco_Policy_Premium'].mean())/df['Reco_Policy_Premium'].std())
```

```
In [16]: df['Reco_Policy_Premium'].describe()
```

```
Out[16]: count    2.354800e+04
         mean     2.018656e-16
         std      1.000000e+00
         min     -1.900308e+00
         25%     -7.332856e-01
         50%     -1.292017e-01
         75%      5.814853e-01
         max      4.354734e+00
         Name: Reco_Policy_Premium, dtype: float64
```

Conclusion:

We successfully implemented data cleaning, data transformation and data reduction on the data set.