EX-05-Feature-Generation

AIM

To read the given data and perform Feature Generation process and save the data to a file.

Explanation

Feature Generation (also known as feature construction, feature extraction or feature engineering) is the process of transforming features into new features that better relate to the target.

FEATURE ENCODING:

1. Ordinal Encoding An ordinal encoding involves mapping each unique label to an integer value. This type of encoding is really only appropriate if there is a known relationship between the categories. This relationship does exist for some of the variables in our dataset, and ideally, this should be harnessed when preparing the data.

2. Label Encoding Label encoding is a simple and straight forward approach. This converts each value in a categorical column into a numerical value. Each value in a categorical column is called Label.

3. Binary Encoding Binary encoding converts a category into binary digits. Each binary digit creates one feature column. If there are n unique categories, then binary encoding results in the only log(base 2)^n features.

4. One Hot Encoding We use this categorical data encoding technique when the features are nominal(do not have any order). In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category.

FEATURE SCALING:

1. Standard Scaler It is also called Z-score normalization. It calculates the z-score of each value and replaces the value with the calculated Z-score. The features are then rescaled with $\bar{x} = 0$ and $\sigma = 1$

2. MinMaxScaler It is also referred to as Normalization. The features are scaled between 0 and 1. Here, the mean value remains same as in Standardization, that is, 0.

3. Maximum absolute scaling Maximum absolute scaling scales the data to its maximum value; that is, it divides every observation by the maximum value of the variable.The result of the preceding transformation is a distribution in which the values vary approximately within the range of -1 to 1.

4. RobustScaler RobustScaler transforms the feature vector by subtracting the median and then dividing by the interquartile range (75% value — 25% value).

ALGORITHM

STEP 1

Read the given Data

STEP 2

Clean the Data Set using Data Cleaning Process

STEP 3

Apply Feature Generation techniques to all the feature of the data set

STEP 4

Save the data to the file

CODE:

1.FEATURE GENERATION FOR Data.csv

CODE FOR FEATURE ENCODING AND FEATURE SCALING:

```
import pandas as pd
df=pd.read_csv("data.csv")
df
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
temp=["Cold","Warm","Hot","Very Hot"]
enc=OrdinalEncoder(categories=[temp])
df["Ord_1"]=enc.fit_transform(df[["Ord_1"]])
df
education=["High School","Diploma","Bachelors","Masters","PhD"]
ed=OrdinalEncoder(categories=[education])
df["Ord_2"]=ed.fit_transform(df[["Ord_2"]])
df
pip install category_encoders
from category_encoders import BinaryEncoder
be=BinaryEncoder()
be.fit_transform(df["bin_1"])
df["bin_1"]=be.fit_transform(df[["bin_1"]])
df
```

```
be1=BinaryEncoder()
be1.fit_transform(df["bin_2"])
df["bin_2"]=be1.fit_transform(df[["bin_2"]])
df
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(sparse=False)
ohe.fit_transform(df[["City"]])
from category_encoders import TargetEncoder
te=TargetEncoder()
te.fit_transform(X=df['Ord_1'],y=df["Target"])
le=LabelEncoder()
df["City"]=le.fit_transform(df[["City"]])
df

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df1=pd.DataFrame(sc.fit_transform(df),columns=
['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df1

from sklearn.preprocessing import MaxAbsScaler
mas=MaxAbsScaler()
df2=pd.DataFrame(mas.fit_transform(df),columns=
['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df2

from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df3=pd.DataFrame(mms.fit_transform(df),columns=
['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df3

from sklearn.preprocessing import RobustScaler
rs=RobustScaler()
df4=pd.DataFrame(rs.fit_transform(df),columns=
['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df4
```

OUPUT for Feauture Encoding and scaling:

Given data:

In [1]:
```
import pandas as pd
df=pd.read_csv("data.csv")
```

In [2]:
```
df
```

Out[2]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0	F	N	Delhi	Hot	High School	0
1	1	F	Y	Bangalore	Warm	Masters	1
2	2	M	N	Mumbai	Very Hot	Diploma	1
3	3	M	Y	Chennai	Cold	Bachelors	0
4	4	M	Y	Delhi	Cold	Bachelors	1
5	5	F	N	Delhi	Very Hot	Masters	0
6	6	M	N	Chennai	Warm	PhD	1
7	7	F	N	Chennai	Hot	High School	1
8	8	M	N	Delhi	Very Hot	High School	0
9	9	F	Y	Delhi	Warm	PhD	0

Feature encoding using Ordinal Encoder:

!

In [3]:
```
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
```

In [4]:
```
temp=["Cold","Warm","Hot","Very Hot"]
```

In [5]:
```
enc=OrdinalEncoder(categories=[temp])
df["Ord_1"]=enc.fit_transform(df[["Ord_1"]])
df
```

Out[5]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0	F	N	Delhi	2.0	High School	0
1	1	F	Y	Bangalore	1.0	Masters	1
2	2	M	N	Mumbai	3.0	Diploma	1
3	3	M	Y	Chennai	0.0	Bachelors	0
4	4	M	Y	Delhi	0.0	Bachelors	1
5	5	F	N	Delhi	3.0	Masters	0
6	6	M	N	Chennai	1.0	PhD	1
7	7	F	N	Chennai	2.0	High School	1
8	8	M	N	Delhi	3.0	High School	0
9	9	F	Y	Delhi	1.0	PhD	0

In [7]:
```
education=["High School","Diploma","Bachelors","Masters","PhD"]
```

In [8]:
```
ed=OrdinalEncoder(categories=[education])
df["Ord_2"]=ed.fit_transform(df[["Ord_2"]])
df
```

Out[8]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0	F	N	Delhi	2.0	0.0	0
1	1	F	Y	Bangalore	1.0	3.0	1
2	2	M	N	Mumbai	3.0	1.0	1
3	3	M	Y	Chennai	0.0	2.0	0
4	4	M	Y	Delhi	0.0	2.0	1
5	5	F	N	Delhi	3.0	3.0	0
6	6	M	N	Chennai	1.0	4.0	1
7	7	F	N	Chennai	2.0	0.0	1
8	8	M	N	Delhi	3.0	0.0	0
9	9	F	Y	Delhi	1.0	4.0	0

Feature encoding using Binary Encoder:

In [10]:
```
from category_encoders import BinaryEncoder
be=BinaryEncoder()
```

In [11]:
```
be.fit_transform(df["bin_1"])
```

Out[11]:

	bin_1_0	bin_1_1
0	0	1
1	0	1
2	1	0
3	1	0
4	1	0
5	0	1
6	1	0
7	0	1
8	1	0
9	0	1

In [13]:
```
df["bin_1"]=be.fit_transform(df[["bin_1"]])
df
```

Out[13]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0	0	N	Delhi	2.0	0.0	0
1	1	0	Y	Bangalore	1.0	3.0	1
2	2	1	N	Mumbai	3.0	1.0	1
3	3	1	Y	Chennai	0.0	2.0	0
4	4	1	Y	Delhi	0.0	2.0	1
5	5	0	N	Delhi	3.0	3.0	0
6	6	1	N	Chennai	1.0	4.0	1
7	7	0	N	Chennai	2.0	0.0	1
8	8	1	N	Delhi	3.0	0.0	0
9	9	0	Y	Delhi	1.0	4.0	0

In [14]:
```
be1=BinaryEncoder()
```

In [16]:
```
be1.fit_transform(df["bin_2"])
```

Out[16]:

	bin_2_0	bin_2_1
0	0	1
1	1	0
2	0	1
3	1	0
4	1	0
5	0	1
6	0	1
7	0	1
8	0	1
9	1	0

In [17]:
```
df["bin_2"]=be1.fit_transform(df[["bin_2"]])
df
```

Out[17]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0	0	0	Delhi	2.0	0.0	0
1	1	0	1	Bangalore	1.0	3.0	1
2	2	1	0	Mumbai	3.0	1.0	1
3	3	1	1	Chennai	0.0	2.0	0
4	4	1	1	Delhi	0.0	2.0	1
5	5	0	0	Delhi	3.0	3.0	0
6	6	1	0	Chennai	1.0	4.0	1
7	7	0	0	Chennai	2.0	0.0	1
8	8	1	0	Delhi	3.0	0.0	0
9	9	0	1	Delhi	1.0	4.0	0

Feature encoding using One Hot Encoder:

In [18]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [19]:
```
ohe=OneHotEncoder(sparse=False)
```

In [20]:
```
ohe.fit_transform(df[["City"]])
```

Out[20]:
```
array([[0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [0., 0., 0., 1.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.]])
```

Feature encoding using Target Encoder:

In [21]:
```
from category_encoders import TargetEncoder
```

In [22]:
```
te=TargetEncoder()
```

In [23]:
```
te.fit_transform(X=df['Ord_1'],y=df["Target"])
```

Out[23]:

	Ord_1
0	2.0
1	1.0
2	3.0
3	0.0
4	0.0
5	3.0
6	1.0
7	2.0
8	3.0
9	1.0

Feature encoding using Label Encoder:

In [24]:
```
le=LabelEncoder()
```

In [26]:
```
df["City"]=le.fit_transform(df[["City"]])
df
```

Out[26]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0	0	0	2	2.0	0.0	0
1	1	0	1	0	1.0	3.0	1
2	2	1	0	3	3.0	1.0	1
3	3	1	1	1	0.0	2.0	0
4	4	1	1	2	0.0	2.0	1
5	5	0	0	2	3.0	3.0	0
6	6	1	0	1	1.0	4.0	1
7	7	0	0	1	2.0	0.0	1
8	8	1	0	2	3.0	0.0	0
9	9	0	1	2	1.0	4.0	0

Feature scaling using Standard Scaler:

In [27]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df1=pd.DataFrame(sc.fit_transform(df),columns=['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df1
```

Out[27]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	-1.566699	-1.0	-0.816497	0.50	0.359211	-1.255555	-1.0
1	-1.218544	-1.0	1.224745	-2.00	-0.538816	0.726900	1.0
2	-0.870388	1.0	-0.816497	1.75	1.257237	-0.594737	1.0
3	-0.522233	1.0	1.224745	-0.75	-1.436842	0.066082	-1.0
4	-0.174078	1.0	1.224745	0.50	-1.436842	0.066082	1.0
5	0.174078	-1.0	-0.816497	0.50	1.257237	0.726900	-1.0
6	0.522233	1.0	-0.816497	-0.75	-0.538816	1.387719	1.0
7	0.870388	-1.0	-0.816497	-0.75	0.359211	-1.255555	1.0
8	1.218544	1.0	-0.816497	0.50	1.257237	-1.255555	-1.0
9	1.566699	-1.0	1.224745	0.50	-0.538816	1.387719	-1.0

Feature scaling using MaxAbs Scaler:

In [29]:
```
from sklearn.preprocessing import MaxAbsScaler
mas=MaxAbsScaler()
df2=pd.DataFrame(mas.fit_transform(df),columns=['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df2
```

Out[29]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0.000000	0.0	0.0	0.666667	0.666667	0.00	0.0
1	0.111111	0.0	1.0	0.000000	0.333333	0.75	1.0
2	0.222222	1.0	0.0	1.000000	1.000000	0.25	1.0
3	0.333333	1.0	1.0	0.333333	0.000000	0.50	0.0
4	0.444444	1.0	1.0	0.666667	0.000000	0.50	1.0
5	0.555556	0.0	0.0	0.666667	1.000000	0.75	0.0
6	0.666667	1.0	0.0	0.333333	0.333333	1.00	1.0
7	0.777778	0.0	0.0	0.333333	0.666667	0.00	1.0
8	0.888889	1.0	0.0	0.666667	1.000000	0.00	0.0
9	1.000000	0.0	1.0	0.666667	0.333333	1.00	0.0

Feature scaling using MinMax Scaler:

In [43]:
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df3=pd.DataFrame(mms.fit_transform(df),columns=['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df3
```

Out[43]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	0.000000	0.0	0.0	0.666667	0.666667	0.00	0.0
1	0.111111	0.0	1.0	0.000000	0.333333	0.75	1.0
2	0.222222	1.0	0.0	1.000000	1.000000	0.25	1.0
3	0.333333	1.0	1.0	0.333333	0.000000	0.50	0.0
4	0.444444	1.0	1.0	0.666667	0.000000	0.50	1.0
5	0.555556	0.0	0.0	0.666667	1.000000	0.75	0.0
6	0.666667	1.0	0.0	0.333333	0.333333	1.00	1.0
7	0.777778	0.0	0.0	0.333333	0.666667	0.00	1.0
8	0.888889	1.0	0.0	0.666667	1.000000	0.00	0.0
9	1.000000	0.0	1.0	0.666667	0.333333	1.00	0.0

Feature scaling using Robust Scaler:

In [44]:
```
from sklearn.preprocessing import RobustScaler
rs=RobustScaler()
df4=pd.DataFrame(rs.fit_transform(df),columns=['id','bin_1','bin_2','City','Ord_1','Ord_2','Target'])
df4
```

Out[44]:

	id	bin_1	bin_2	City	Ord_1	Ord_2	Target
0	-1.000000	-0.5	0.0	0.0	0.285714	-0.727273	-0.5
1	-0.777778	-0.5	1.0	-2.0	-0.285714	0.363636	0.5
2	-0.555556	0.5	0.0	1.0	0.857143	-0.363636	0.5
3	-0.333333	0.5	1.0	-1.0	-0.857143	0.000000	-0.5
4	-0.111111	0.5	1.0	0.0	-0.857143	0.000000	0.5
5	0.111111	-0.5	0.0	0.0	0.857143	0.363636	-0.5
6	0.333333	0.5	0.0	-1.0	-0.285714	0.727273	0.5
7	0.555556	-0.5	0.0	-1.0	0.285714	-0.727273	0.5
8	0.777778	0.5	0.0	0.0	0.857143	-0.727273	-0.5
9	1.000000	-0.5	1.0	0.0	-0.285714	0.727273	-0.5

2.FEATURE GENERATION FOR Encoding.csv

CODE FOR FEATURE ENCODING AND FEATURE SCALING:

```
import pandas as pd
df=pd.read_csv("Encoding Data.csv")
df
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
temp=["Cold","Warm","Hot"]
enc=OrdinalEncoder(categories=[temp])
df["ord_2"]=enc.fit_transform(df[["ord_2"]])
df
from category_encoders import BinaryEncoder
be=BinaryEncoder()
df['bin_1']=be.fit_transform(df[['bin_1']])
df
be1=BinaryEncoder()
df['bin_2']=be1.fit_transform(df[['bin_2']])
df
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(sparse=False)
ohe.fit_transform(df[["nom_0"]])
le=LabelEncoder()
df["nom_0"]=le.fit_transform(df[["nom_0"]])
df

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df1=pd.DataFrame(sc.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df1

from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df2=pd.DataFrame(mms.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df2
```

```
from sklearn.preprocessing import MaxAbsScaler mas=MaxAbsScaler()
df3=pd.DataFrame(mas.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df3

from sklearn.preprocessing import RobustScaler
rs=RobustScaler()
df4=pd.DataFrame(rs.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df4
```

Output for FEATURE GENERATION FOR Encoding.csv:

Given DataFrame:

In [1]:
```
import pandas as pd
df=pd.read_csv("Encoding Data.csv")
```

In [2]:
```
df
```

Out[2]:

	id	bin_1	bin_2	nom_0	ord_2
0	0	F	N	Red	Hot
1	1	F	Y	Blue	Warm
2	2	F	N	Blue	Cold
3	3	F	N	Green	Warm
4	4	T	N	Red	Cold
5	5	T	N	Green	Hot
6	6	F	N	Red	Cold
7	7	T	N	Red	Cold
8	8	F	N	Blue	Warm
9	9	F	Y	Red	Hot

Feature encoding using Ordinal Encoder:

In [3]:
```
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
```

In [4]:
```
temp=["Cold","Warm","Hot"]
```

In [5]:
```
enc=OrdinalEncoder(categories=[temp])
```

In [6]:
```
df["ord_2"]=enc.fit_transform(df[["ord_2"]])
df
```

Out[6]:

	id	bin_1	bin_2	nom_0	ord_2
0	0	F	N	Red	2.0
1	1	F	Y	Blue	1.0
2	2	F	N	Blue	0.0
3	3	F	N	Green	1.0
4	4	T	N	Red	0.0
5	5	T	N	Green	2.0
6	6	F	N	Red	0.0
7	7	T	N	Red	0.0
8	8	F	N	Blue	1.0
9	9	F	Y	Red	2.0

Feature encoding using Binary Encoder:

In [7]:
```
from category_encoders import BinaryEncoder
```

In [9]:
```
be=BinaryEncoder()
df['bin_1']=be.fit_transform(df[['bin_1']])
df
```

Out[9]:

	id	bin_1	bin_2	nom_0	ord_2
0	0	0	N	Red	2.0
1	1	0	Y	Blue	1.0
2	2	0	N	Blue	0.0
3	3	0	N	Green	1.0
4	4	1	N	Red	0.0
5	5	1	N	Green	2.0
6	6	0	N	Red	0.0
7	7	1	N	Red	0.0
8	8	0	N	Blue	1.0
9	9	0	Y	Red	2.0

In [10]:
```
be1=BinaryEncoder()
df['bin_2']=be1.fit_transform(df[['bin_2']])
df
```

Out[10]:

	id	bin_1	bin_2	nom_0	ord_2
0	0	0	0	Red	2.0
1	1	0	1	Blue	1.0
2	2	0	0	Blue	0.0
3	3	0	0	Green	1.0
4	4	1	0	Red	0.0
5	5	1	0	Green	2.0
6	6	0	0	Red	0.0
7	7	1	0	Red	0.0
8	8	0	0	Blue	1.0
9	9	0	1	Red	2.0

Feature encoding using One Hot Encoder:

In [11]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [13]:
```
ohe=OneHotEncoder(sparse=False)
ohe.fit_transform(df[["nom_0"]])
```

Out[13]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.]])
```

Feature encoding using Label Encoder:

In [15]:
```
le=LabelEncoder()
```

In [16]:
```
df["nom_0"]=le.fit_transform(df[["nom_0"]])
df
```

Out[16]:

	id	bin_1	bin_2	nom_0	ord_2
0	0	0	0	2	2.0
1	1	0	1	0	1.0
2	2	0	0	0	0.0
3	3	0	0	1	1.0
4	4	1	0	2	0.0
5	5	1	0	1	2.0
6	6	0	0	2	0.0
7	7	1	0	2	0.0
8	8	0	0	0	1.0
9	9	0	1	2	2.0

Feature scaling using Standard Scaler:

In [17]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df1=pd.DataFrame(sc.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df1
```

Out[17]:

	id	bin_1	bin_2	nom_0	Ord_2
0	-1.566699	-0.654654	-0.5	0.917663	1.324244
1	-1.218544	-0.654654	2.0	-1.376494	0.120386
2	-0.870388	-0.654654	-0.5	-1.376494	-1.083473
3	-0.522233	-0.654654	-0.5	-0.229416	0.120386
4	-0.174078	1.527525	-0.5	0.917663	-1.083473
5	0.174078	1.527525	-0.5	-0.229416	1.324244
6	0.522233	-0.654654	-0.5	0.917663	-1.083473
7	0.870388	1.527525	-0.5	0.917663	-1.083473
8	1.218544	-0.654654	-0.5	-1.376494	0.120386
9	1.566699	-0.654654	2.0	0.917663	1.324244

Feature scaling using MinMax Scaler:

In [18]:
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df2=pd.DataFrame(mms.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df2
```

Out[18]:

	id	bin_1	bin_2	nom_0	Ord_2
0	0.000000	0.0	0.0	1.0	1.0
1	0.111111	0.0	1.0	0.0	0.5
2	0.222222	0.0	0.0	0.0	0.0
3	0.333333	0.0	0.0	0.5	0.5
4	0.444444	1.0	0.0	1.0	0.0
5	0.555556	1.0	0.0	0.5	1.0
6	0.666667	0.0	0.0	1.0	0.0
7	0.777778	1.0	0.0	1.0	0.0
8	0.888889	0.0	0.0	0.0	0.5
9	1.000000	0.0	1.0	1.0	1.0

Feature scaling using MaxAbs Scaler:

In [19]:
```
from sklearn.preprocessing import MaxAbsScaler
mas=MaxAbsScaler()
df3=pd.DataFrame(mas.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df3
```

Out[19]:

	id	bin_1	bin_2	nom_0	Ord_2
0	0.000000	0.0	0.0	1.0	1.0
1	0.111111	0.0	1.0	0.0	0.5
2	0.222222	0.0	0.0	0.0	0.0
3	0.333333	0.0	0.0	0.5	0.5
4	0.444444	1.0	0.0	1.0	0.0
5	0.555556	1.0	0.0	0.5	1.0
6	0.666667	0.0	0.0	1.0	0.0
7	0.777778	1.0	0.0	1.0	0.0
8	0.888889	0.0	0.0	0.0	0.5
9	1.000000	0.0	1.0	1.0	1.0

Feature scaling using Robust Scaler:

In [20]:
```
from sklearn.preprocessing import RobustScaler
rs=RobustScaler()
df4=pd.DataFrame(rs.fit_transform(df),columns=['id','bin_1','bin_2','nom_0','Ord_2'])
df4
```

Out[20]:

	id	bin_1	bin_2	nom_0	Ord_2
0	-1.000000	0.000000	0.0	0.285714	0.571429
1	-0.777778	0.000000	1.0	-0.857143	0.000000
2	-0.555556	0.000000	0.0	-0.857143	-0.571429
3	-0.333333	0.000000	0.0	-0.285714	0.000000
4	-0.111111	1.333333	0.0	0.285714	-0.571429
5	0.111111	1.333333	0.0	-0.285714	0.571429
6	0.333333	0.000000	0.0	0.285714	-0.571429
7	0.555556	1.333333	0.0	0.285714	-0.571429
8	0.777778	0.000000	0.0	-0.857143	0.000000
9	1.000000	0.000000	1.0	0.285714	0.571429

3.FEATURE GENERATION FOR titanic_dataset.csv

CODE FOR FEATURE ENCODING AND FEATURE SCALING:

```
import pandas as pd
df=pd.read_csv("titanic_dataset.csv")
df
df.isnull().sum()
df["Age"]=df["Age"].fillna(df["Age"].median())
df["Embarked"]=df["Embarked"].fillna(df["Embarked"].mode()[0])
df
df.drop("Cabin",axis=1,inplace=True)
df.drop("Ticket",axis=1,inplace=True)
df.drop("Name",axis=1,inplace=True)
df.isnull().sum()
df
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
embark=["C","S","Q"]
emb=OrdinalEncoder(categories=[embark])
```

```
df["Embarked"]=emb.fit_transform(df[["Embarked"]])
df
from category_encoders import BinaryEncoder
be=BinaryEncoder()
df["Sex"]=be.fit_transform(df[["Sex"]])
df

from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
df1=pd.DataFrame(ss.fit_transform(df),columns=
['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PClass'])
df1

from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df2=pd.DataFrame(mms.fit_transform(df),columns=
['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PClass'])
df2

from sklearn.preprocessing import MaxAbsScaler
mas=MaxAbsScaler()
df3=pd.DataFrame(mas.fit_transform(df),columns=
['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PClass'])
df3

from sklearn.preprocessing import RobustScaler
rs = RobustScaler()
df4=pd.DataFrame(rs.fit_transform(df),columns=
['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PClass'])
df4
```

Output for FEATURE GENERATION FOR titanic_dataset.csv:

Given DataFrame:

In [1]:
```
import pandas as pd
df=pd.read_csv("titanic_dataset.csv")
```

In [2]:
```
df
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Resolving null value data:

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [4]:
```
df["Age"]=df["Age"].fillna(df["Age"].median())
```

In [5]:
```
df["Embarked"]=df["Embarked"].fillna(df["Embarked"].mode()[0])
```

In [6]:
```
df
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Dropping unnecessary columns:

In [7]:
```
df.drop("Cabin",axis=1,inplace=True)
```

In [8]:
```
df.drop("Ticket",axis=1,inplace=True)
```

In [11]:
```
df.drop("Name",axis=1,inplace=True)
```

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

Out[12]:
```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

Feature encoding using Ordinal Encoder:

In [14]:
```
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
```

In [19]:
```
embark=["C","S","Q"]
emb=OrdinalEncoder(categories=[embark])
df["Embarked"]=emb.fit_transform(df[["Embarked"]])
```

In [20]:
```
df
```

Out[20]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	PClass
0	1	0	3	male	22.0	1	0	7.2500	1.0	2.0
1	2	1	1	female	38.0	1	0	71.2833	0.0	0.0
2	3	1	3	female	26.0	0	0	7.9250	1.0	2.0
3	4	1	1	female	35.0	1	0	53.1000	1.0	0.0
4	5	0	3	male	35.0	0	0	8.0500	1.0	2.0
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	1.0
887	888	1	1	female	19.0	0	0	30.0000	1.0	0.0
888	889	0	3	female	28.0	1	2	23.4500	1.0	2.0
889	890	1	1	male	26.0	0	0	30.0000	0.0	0.0
890	891	0	3	male	32.0	0	0	7.7500	2.0	2.0

891 rows × 10 columns

Feature encoding using Binary Encoder:

In [21]:
```
from category_encoders import BinaryEncoder
```

In [22]:
```
be=BinaryEncoder()
```

In [23]:
```
df["Sex"]=be.fit_transform(df[["Sex"]])
```

In [24]:
```
df
```

Out[24]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	PClass
0	1	0	3	0	22.0	1	0	7.2500	1.0	2.0
1	2	1	1	1	38.0	1	0	71.2833	0.0	0.0
2	3	1	3	1	26.0	0	0	7.9250	1.0	2.0
3	4	1	1	1	35.0	1	0	53.1000	1.0	0.0
4	5	0	3	0	35.0	0	0	8.0500	1.0	2.0
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	0	27.0	0	0	13.0000	1.0	1.0
887	888	1	1	1	19.0	0	0	30.0000	1.0	0.0
888	889	0	3	1	28.0	1	2	23.4500	1.0	2.0
889	890	1	1	0	26.0	0	0	30.0000	0.0	0.0
890	891	0	3	0	32.0	0	0	7.7500	2.0	2.0

891 rows × 10 columns

Feature scaling using Standard Scaler:

In [29]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
df1=pd.DataFrame(ss.fit_transform(df),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PCl
df1
```

Out[29]:

	Passenger	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	PClass
0	-1.730108	-0.789272	0.827377	-0.737695	-0.565736	0.432793	-0.473674	-0.502445	0.198572	0.827377
1	-1.726220	1.266990	-1.566107	1.355574	0.663861	0.432793	-0.473674	0.786845	-1.745685	-1.566107
2	-1.722332	1.266990	0.827377	1.355574	-0.258337	-0.474545	-0.473674	-0.488854	0.198572	0.827377
3	-1.718444	1.266990	-1.566107	1.355574	0.433312	0.432793	-0.473674	0.420730	0.198572	-1.566107
4	-1.714556	-0.789272	0.827377	-0.737695	0.433312	-0.474545	-0.473674	-0.486337	0.198572	0.827377
...	...	...	...	...	...	...	...	...	...	...
886	1.714556	-0.789272	-0.369365	-0.737695	-0.181487	-0.474545	-0.473674	-0.386671	0.198572	-0.369365
887	1.718444	1.266990	-1.566107	1.355574	-0.796286	-0.474545	-0.473674	-0.044381	0.198572	-1.566107
888	1.722332	-0.789272	0.827377	1.355574	-0.104637	0.432793	2.008933	-0.176263	0.198572	0.827377
889	1.726220	1.266990	-1.566107	-0.737695	-0.258337	-0.474545	-0.473674	-0.044381	-1.745685	-1.566107
890	1.730108	-0.789272	0.827377	-0.737695	0.202762	-0.474545	-0.473674	-0.492378	2.142828	0.827377

891 rows × 10 columns

Feature scaling using MinMax Scaler

In [30]:
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df2=pd.DataFrame(mms.fit_transform(df),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PC
df2
```

Out[30]:

	Passenger	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	PClass
0	0.000000	0.0	1.0	0.0	0.271174	0.125	0.000000	0.014151	0.5	1.0
1	0.001124	1.0	0.0	1.0	0.472229	0.125	0.000000	0.139136	0.0	0.0
2	0.002247	1.0	1.0	1.0	0.321438	0.000	0.000000	0.015469	0.5	1.0
3	0.003371	1.0	0.0	1.0	0.434531	0.125	0.000000	0.103644	0.5	0.0
4	0.004494	0.0	1.0	0.0	0.434531	0.000	0.000000	0.015713	0.5	1.0
...	...	...	...	...	...	...	...	...	...	...
886	0.995506	0.0	0.5	0.0	0.334004	0.000	0.000000	0.025374	0.5	0.5
887	0.996629	1.0	0.0	1.0	0.233476	0.000	0.000000	0.058556	0.5	0.0
888	0.997753	0.0	1.0	1.0	0.346569	0.125	0.333333	0.045771	0.5	1.0
889	0.998876	1.0	0.0	0.0	0.321438	0.000	0.000000	0.058556	0.0	0.0
890	1.000000	0.0	1.0	0.0	0.396833	0.000	0.000000	0.015127	1.0	1.0

891 rows × 10 columns

Feature scaling using MaxAbs Scaler:

In [32]:
```
from sklearn.preprocessing import MaxAbsScaler
mas=MaxAbsScaler()
df3=pd.DataFrame(mas.fit_transform(df),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PC
df3
```

Out[32]:

	Passenger	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	PClass
0	0.001122	0.0	1.000000	0.0	0.2750	0.125	0.000000	0.014151	0.5	1.0
1	0.002245	1.0	0.333333	1.0	0.4750	0.125	0.000000	0.139136	0.0	0.0
2	0.003367	1.0	1.000000	1.0	0.3250	0.000	0.000000	0.015469	0.5	1.0
3	0.004489	1.0	0.333333	1.0	0.4375	0.125	0.000000	0.103644	0.5	0.0
4	0.005612	0.0	1.000000	0.0	0.4375	0.000	0.000000	0.015713	0.5	1.0
...	...	...	...	...	...	...	...	...	...	...
886	0.995511	0.0	0.666667	0.0	0.3375	0.000	0.000000	0.025374	0.5	0.5
887	0.996633	1.0	0.333333	1.0	0.2375	0.000	0.000000	0.058556	0.5	0.0
888	0.997755	0.0	1.000000	1.0	0.3500	0.125	0.333333	0.045771	0.5	1.0
889	0.998878	1.0	0.333333	0.0	0.3250	0.000	0.000000	0.058556	0.0	0.0
890	1.000000	0.0	1.000000	0.0	0.4000	0.000	0.000000	0.015127	1.0	1.0

891 rows × 10 columns

Feature scaling using Robust Scaler

In [33]:
```
from sklearn.preprocessing import RobustScaler
rs = RobustScaler()
df4=pd.DataFrame(rs.fit_transform(df),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','PCl
df4
```

Out[33]:

	Passenger	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	PClass
0	-1.000000	0.0	0.0	0.0	-0.461538	1.0	0.0	-0.312011	0.0	0.0
1	-0.997753	1.0	-2.0	1.0	0.769231	1.0	0.0	2.461242	-1.0	-2.0
2	-0.995506	1.0	0.0	1.0	-0.153846	0.0	0.0	-0.282777	0.0	0.0
3	-0.993258	1.0	-2.0	1.0	0.538462	1.0	0.0	1.673732	0.0	-2.0
4	-0.991011	0.0	0.0	0.0	0.538462	0.0	0.0	-0.277363	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
886	0.991011	0.0	-1.0	0.0	-0.076923	0.0	0.0	-0.062981	0.0	-1.0
887	0.993258	1.0	-2.0	1.0	-0.692308	0.0	0.0	0.673281	0.0	-2.0
888	0.995506	0.0	0.0	1.0	0.000000	1.0	2.0	0.389604	0.0	0.0
889	0.997753	1.0	-2.0	0.0	-0.153846	0.0	0.0	0.673281	-1.0	-2.0
890	1.000000	0.0	0.0	0.0	0.307692	0.0	0.0	-0.290356	1.0	0.0

891 rows × 10 columns

RESULT:

Feature Encoding process and Feature Scaling process is applied to the given data frame sucessfully.