Experiment 3

Aim:-

Write a python code for dealing with missing values and encoding categorical data. Apply data transformation and normalization.

Objective:- To learn how to read/find missing values To learn how to replace/deal missing values To learn how to encode categorical data To learn how to perform transformation and normalization

Description

Working with Missing Data Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed. Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

isnull() notnull() dropna() fillna() replace() interpolate() Now let's look at the different methods that we can use to deal with the missing data. Deleting the columns with missing data Deleting the rows with missing data Filling the missing data with a value – Imputation Imputation with an additional column Filling with a Regression Model

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

In [2]:
```
df = pd.read_csv("DataPreprocessing.csv")
```

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

Out[3]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	RoofStyle	TotalBsmt
0	1461	20.0	80	11622	Inside	NAmes	1Fam	1Story	5	6	1961	Gable	882
1	1462	20.0	##	14267	Corner	NAmes	1Fam	1Story	6	6	1958	Hip	1329
2	1463	60.0	74	13830	Inside	Gilbert	1Fam	2Story	NaN	5	1997	Gable	928
3	1464	60.0	78	9978	Inside	Gilbert	Na	?	6	6	1998	#	926
4	1465	120.0	43	5005	Inside	StoneBr	TwnhsE	1Story	8	Nan	1992	Gable	1280

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Id              0
MSSubClass      3
LotFrontage     227
LotArea         0
LotConfig       0
Neighborhood    0
BldgType        0
HouseStyle      0
OverallQual     4
OverallCond     0
YearBuilt       0
RoofStyle       0
TotalBsmtSF     1
GrLivArea       0
BedroomAbvGr    0
TotRmsAbvGrd    0
GarageArea      1
dtype: int64
```

In [5]:
```
df.dtypes
```

Out[5]:
```
Id               int64
MSSubClass     float64
LotFrontage     object
LotArea          int64
LotConfig       object
Neighborhood    object
BldgType        object
HouseStyle      object
OverallQual     object
OverallCond     object
YearBuilt        int64
RoofStyle       object
TotalBsmtSF    float64
GrLivArea        int64
BedroomAbvGr    object
TotRmsAbvGrd     int64
GarageArea     float64
dtype: object
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Id            1459 non-null   int64  
 1   MSSubClass    1456 non-null   float64
 2   LotFrontage   1232 non-null   object 
 3   LotArea       1459 non-null   int64  
 4   LotConfig     1459 non-null   object 
 5   Neighborhood  1459 non-null   object 
 6   BldgType      1459 non-null   object 
 7   HouseStyle    1459 non-null   object 
 8   OverallQual   1455 non-null   object 
 9   OverallCond   1459 non-null   object 
 10  YearBuilt     1459 non-null   int64  
 11  RoofStyle     1459 non-null   object 
 12  TotalBsmtSF   1458 non-null   float64
 13  GrLivArea     1459 non-null   int64  
 14  BedroomAbvGr  1459 non-null   object 
 15  TotRmsAbvGrd  1459 non-null   int64  
 16  GarageArea    1458 non-null   float64
dtypes: float64(3), int64(5), object(9)
memory usage: 193.9+ KB
```

Concise:-

By above analysis using info(), dtypes(), isnull() we can see that data contains missing values and some entries are having special characters/wrong data instead of empty cells.

Now using uqique function at every column, we can check special characters/wrong data and we can replace these with NaN.

In [7]:
```
df.head()
```

Out[7]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	RoofStyle	TotalBsmt
0	1461	20.0	80	11622	Inside	NAmes	1Fam	1Story	5	6	1961	Gable	88
1	1462	20.0	##	14267	Corner	NAmes	1Fam	1Story	6	6	1958	Hip	132
2	1463	60.0	74	13830	Inside	Gilbert	1Fam	2Story	NaN	5	1997	Gable	92
3	1464	60.0	78	9978	Inside	Gilbert	Na	?	6	6	1998	#	92
4	1465	120.0	43	5005	Inside	StoneBr	TwnhsE	1Story	8	Nan	1992	Gable	128

In [8]:
```
np.unique(df['MSSubClass'])
```

Out[8]:
```
array([ 20.,  30.,  40.,  45.,  50.,  60.,  70.,  75.,  80.,  85.,  90.,
       120., 150., 160., 180., 190., 590., 620.,  nan])
```

In [9]:
```
np.unique(df['LotConfig'])
```

Out[9]:
```
array(['Corner', 'CulDSac', 'FR2', 'FR3', 'Inside'], dtype=object)
```

In [10]:
```
np.unique(df['HouseStyle'])
```

Out[10]:
```
array(['1.5Fin', '1.5Unf', '1Story', '2.5Unf', '2Story', ':-', '?',
       'SFoyer', 'SLvl', '__'], dtype=object)
```

Here we have special characters or wrong data like ':-', '?', '__' so identify all such wrong data and replace with NaN

In [11]:
```
df.replace([':-', '?', '__', '_', '#', "##", 'Na', 'Nan'], np.nan, inplace = True)
```

In [12]:
```
df.head()
```

Out[12]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	RoofStyle	TotalBsmt
0	1461	20.0	80	11622	Inside	NAmes	1Fam	1Story	5	6	1961	Gable	88
1	1462	20.0	NaN	14267	Corner	NAmes	1Fam	1Story	6	6	1958	Hip	132
2	1463	60.0	74	13830	Inside	Gilbert	1Fam	2Story	NaN	5	1997	Gable	92
3	1464	60.0	78	9978	Inside	Gilbert	NaN	NaN	6	6	1998	NaN	92
4	1465	120.0	43	5005	Inside	StoneBr	TwnhsE	1Story	8	NaN	1992	Gable	128

As we can see all special characters or wrong data have been replaced by NaN, now we can apply drop/mean/median/mode/regression according to the requirement.

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
Id               0
MSSubClass       3
LotFrontage    228
LotArea          0
LotConfig        0
Neighborhood     0
BldgType         1
HouseStyle       4
OverallQual      5
OverallCond      1
YearBuilt        0
RoofStyle        1
TotalBsmtSF      1
GrLivArea        0
BedroomAbvGr     2
TotRmsAbvGrd     0
GarageArea       1
dtype: int64
```

In [14]:
```
df['MSSubClass'].fillna(int(df['MSSubClass'].mode()), inplace=True)
```

In [15]:
```
df['LotFrontage'].fillna(int(df['LotFrontage'].mode()), inplace=True)
```

In [16]:
```
df['BldgType'].fillna('1Fam', inplace=True)
```

In [17]:
```
df['HouseStyle'].fillna('1Story', inplace=True)
```

In [18]:
```
df['OverallQual'].fillna(int(df['OverallQual'].mode()), inplace=True)
```

In [19]:
```
df['OverallCond'].fillna(int(df['OverallCond'].mode()), inplace=True)
```

In [20]:
```
df['RoofStyle'].fillna('Gable', inplace=True)
```

In [21]:
```
df['TotalBsmtSF'].fillna(int(df['TotalBsmtSF'].mean()), inplace=True)
```

In [22]:
```
df['BedroomAbvGr'].fillna(int(df['BedroomAbvGr'].mode()), inplace=True)
```

In [23]:
```
df['GarageArea'].fillna(int(df['GarageArea'].mean()), inplace=True)
```

In [24]:
```
df.isnull().sum()
```

Out[24]:
```
Id              0
MSSubClass      0
LotFrontage     0
LotArea         0
LotConfig       0
Neighborhood    0
BldgType        0
HouseStyle      0
OverallQual     0
OverallCond     0
YearBuilt       0
RoofStyle       0
TotalBsmtSF     0
GrLivArea       0
BedroomAbvGr    0
TotRmsAbvGrd    0
GarageArea      0
dtype: int64
```

Now there is no Null Values, now convert object dtype into int64 or float64

In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Id            1459 non-null   int64  
 1   MSSubClass    1459 non-null   float64
 2   LotFrontage   1459 non-null   object 
 3   LotArea       1459 non-null   int64  
 4   LotConfig     1459 non-null   object 
 5   Neighborhood  1459 non-null   object 
 6   BldgType      1459 non-null   object 
 7   HouseStyle    1459 non-null   object 
 8   OverallQual   1459 non-null   object 
 9   OverallCond   1459 non-null   object 
 10  YearBuilt     1459 non-null   int64  
 11  RoofStyle     1459 non-null   object 
 12  TotalBsmtSF   1459 non-null   float64
 13  GrLivArea     1459 non-null   int64  
 14  BedroomAbvGr  1459 non-null   object 
 15  TotRmsAbvGrd  1459 non-null   int64  
 16  GarageArea    1459 non-null   float64
dtypes: float64(3), int64(5), object(9)
memory usage: 193.9+ KB
```

In [26]:
```
df['LotFrontage']=df['LotFrontage'].astype('int64')
```

In [27]:
```
df['OverallQual']=df['OverallQual'].astype('int64')
```

In [28]:
```
df['OverallCond']=df['OverallCond'].astype('int64')
```

In [29]:
```
df['BedroomAbvGr']=df['BedroomAbvGr'].astype('int64')
```

In [30]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

In [31]:
```
df['LotConfig'] = label_encoder.fit_transform(df['LotConfig'])
```

In [32]:
```
df['Neighborhood'] = label_encoder.fit_transform(df['Neighborhood'])
```

In [33]:
```
df['BldgType'] = label_encoder.fit_transform(df['BldgType'])
```

In [34]:
```
df['HouseStyle'] = label_encoder.fit_transform(df['HouseStyle'])
```

In [35]:
```
df['RoofStyle'] = label_encoder.fit_transform(df['RoofStyle'])
```

In [36]:
```
df.head()
```

Out[36]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	RoofStyle	TotalBsmt
0	1461	20.0	80	11622	4	12	0	2	5	6	1961	1	88
1	1462	20.0	60	14267	0	12	0	2	6	6	1958	3	132
2	1463	60.0	74	13830	4	8	0	4	5	5	1997	1	92
3	1464	60.0	78	9978	4	8	0	2	6	6	1998	1	92
4	1465	120.0	43	5005	4	22	4	2	8	5	1992	1	128

In [37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Id            1459 non-null   int64  
 1   MSSubClass    1459 non-null   float64
 2   LotFrontage   1459 non-null   int64  
 3   LotArea       1459 non-null   int64  
 4   LotConfig     1459 non-null   int32  
 5   Neighborhood  1459 non-null   int32  
 6   BldgType      1459 non-null   int32  
 7   HouseStyle    1459 non-null   int32  
 8   OverallQual   1459 non-null   int64  
 9   OverallCond   1459 non-null   int64  
 10  YearBuilt     1459 non-null   int64  
 11  RoofStyle     1459 non-null   int32  
 12  TotalBsmtSF   1459 non-null   float64
 13  GrLivArea     1459 non-null   int64  
 14  BedroomAbvGr  1459 non-null   int64  
 15  TotRmsAbvGrd  1459 non-null   int64  
 16  GarageArea    1459 non-null   float64
dtypes: float64(3), int32(5), int64(9)
memory usage: 165.4 KB
```

In [38]:
```
df.to_excel('AfterPreprocssing1.xlsx')
```

Now all columns are numerical without any null value.

Now we will apply data transformation

In [39]:
```
df.head()
```

Out[39]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	RoofStyle	TotalBsmt
0	1461	20.0	80	11622	4	12	0	2	5	6	1961	1	88
1	1462	20.0	60	14267	0	12	0	2	6	6	1958	3	132
2	1463	60.0	74	13830	4	8	0	4	5	5	1997	1	92
3	1464	60.0	78	9978	4	8	0	2	6	6	1998	1	92
4	1465	120.0	43	5005	4	22	4	2	8	5	1992	1	128

We can construct a new attribute "House_Age" from the attribute "YearBuilt" We can round off LotArea to the nearest 10

In [40]:
```
from datetime import date
df.insert(11, "House_Age", date.today().year - df['YearBuilt'])
```

In [41]:
```
df['LotArea'] = df['LotArea'].apply(lambda x: 10*round(x/10))
```

In [42]:
```
df.head()
```

Out[42]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	House_Age	RoofStyl
0	1461	20.0	80	11620	4	12	0	2	5	6	1961	63	
1	1462	20.0	60	14270	0	12	0	2	6	6	1958	66	
2	1463	60.0	74	13830	4	8	0	4	5	5	1997	27	
3	1464	60.0	78	9980	4	8	0	2	6	6	1998	26	
4	1465	120.0	43	5000	4	22	4	2	8	5	1992	32	

Data Normalization Data normalization involves converting all data variables into a given range. Techniques that are used for normalization are: Min-Max Normalization Z-Score Normalization Decimal Scaling

Convert LotArea in range from 10 to 50 using Min-Max Normalization

In [43]:
```
Min = df['LotArea'].min()
Max = df['LotArea'].max()
```

In [44]:
```
df['LotArea'] = df['LotArea'].apply(lambda x: ((x-Min)*40/(Max-Min))+10)
```

In [45]:
```
df['LotArea'] = df['LotArea'].apply(np.floor)
```

Z-score normalization In this technique, values are normalized based on mean and standard deviation of the data.

In [46]:
```
Mean = np.round(df['LotFrontage'].mean())
```

In [47]:
```
Std = df['LotFrontage'].std()
```

In [48]:
```
df['LotFrontage'] = df['LotFrontage'].apply(lambda x: ((x-Mean)/Std))
```

Decimal Scaling Method For Normalization To normalize the data by this technique, we divide each value of the data by the maximum absolute value of data. We are applying decimal scaling on "TotalBsmtSF"

In [50]:
```
Maximum = df['TotalBsmtSF'].max()
digits = len(str(Maximum))
Abs = pow(10,digits)
```

In [51]:
```
df['TotalBsmtSF'] = df['TotalBsmtSF'].apply(lambda x: (x/Abs))
```

In [52]:
```
df.head()
```

Out[52]:

	Id	MSSubClass	LotFrontage	LotArea	LotConfig	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	House_Age	RoofStyl
0	1461	20.0	0.625210	17.0	4	12	0	2	5	6	1961	63	
1	1462	20.0	-0.336652	19.0	0	12	0	2	6	6	1958	66	
2	1463	60.0	0.336652	18.0	4	8	0	4	5	5	1997	27	
3	1464	60.0	0.529024	16.0	4	8	0	2	6	6	1998	26	
4	1465	120.0	-1.154234	12.0	4	22	4	2	8	5	1992	32	

Learning Outcomes

Got to know missing values, finding them, replacing them appropriate method. Got to know encoding methods and how to apply them. Learned how to apply data transformation and standarization.

Result/ Conclusion All important functions of related to missing values have been studied and implemented in python sucessfully. Encodeing method label encoder had been studied and implemented in python sucessfully. Data transformation have been studied and implemented in python sucessfully.