EX-06-Feature-Transformation

AIM:

To Perform the various feature transformation techniques on a dataset and save the data to a file.

Explanation:

Feature Transformation is a mathematical transformation in which we apply a mathematical formula to a particular column(feature) and transform the values which are useful for our further analysis.

ALGORITHM:

Step 1: Read the given Data

Step 2: Clean the Data Set using Data Cleaning Process

Step 3: Apply Feature Transformation techniques to all the feature of the data set

Step 4: Save the data to the file.

```
 CODE:
Developed By : DHANUSH S
Register Number : 212221230020

 Titanic Dataset:
import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm

df=pd.read_csv("titanic_dataset.csv")
df.info()

df.isnull().sum()

df['Cabin']=df['Cabin'].fillna(df['Cabin'].mode()[0])
df['Age']=df['Age'].fillna(df['Age'].mean())
df['Embarked']=df['Embarked'].fillna(df['Embarked'].mode()[0])
df.isnull().sum()

df.skew()
df1=df.copy()
df1=df.info()
df1.skew()
df1["Sibsp_1"]=np.sqrt(df1.SibSp)
df1.SibSp.hist()
```

```
df1.skew()
df

del df['Name']
df

del df['Cabin']
del df['Ticket']
df.isnull().sum()

from sklearn.preprocessing import
OrdinalEncoder
embark=["C","S","Q"]
emb=OrdinalEncoder (categories =[embark])
df["Embarked"]=emb.fit_transform(df[["Embarked"]])
df

from category_encoders import BinaryEncoder
be1=BinaryEncoder()
df['Sex']=be1.fit_transform(df[["Sex"]])
df



#Function Transformation:
#Log Tranformation:
np.log(df["Age"])

#Reciprocal Transformation
np.reciprocal (df[["Fare"]])

#sqrt transformation
np.sqrt(df["Embarked"])

#power transformation
df["Age_boxcox"],parameters=stats.boxcox(df["Age"])
df



df["Pclass_boxcox"],parameters=stats.boxcox(df["Pclass"])
df

df["Fare_yeojohnson"],parameters = stats.yeojohnson(df["Fare"])
df

df["Parch_yeojohnson"],parameters = stats.yeojohnson(df["Parch"])
df

df.skew()

#Quantile transformation

from sklearn.preprocessing import QuantileTransformer
```

```
qt=QuantileTransformer(output_distribution ='normal',n_quantiles=891)

df["Age_1"]=qt.fit_transform(df[["Age"]])
sm.qqplot(df['Age'],line='45')

sm.qqplot(df['Age_1'],line='45')

df["Fare_1"]=qt.fit_transform(df[["Fare"]])
sm.qqplot(df["Fare"],line='45')
sm.qqplot(df['Fare_1'],line='45')

df["Parch_1"]=qt.fit_transform(df[["Parch"]])
sm.qqplot(df['Parch'],line='45')
sm.qqplot(df['Parch_1'],line='45')

df
```

Data to transform:

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm

df=pd.read_csv("Data_To_Transform.csv")
df

df.skew()

#Function Transformation
#Log Transformation
np.log(df["Highly Positive Skew"])
np.reciprocal(df["Moderate Positive Skew"])
np.sqrt(df["Highly Positive Skew"])

df["Highly positive Skew_boxcox"],parameters=stats.boxcox(df["Highly Positive Skew"])

df["Moderate Positive Skew_yeojohnson"],parameters=stats.boxcox(df["Moderate Positive
Skew"])
df

df["Moderate Negative Skew_yeojohnson"],parameters=stats.yeojohnson(df["Moderate Negative
Skew"])
df


df["Highly Negative Skew_yeojohnson"],parameters=stats.yeojohnson(df["Highly Negative
Skew"])
df
```

```
df.skew()
#Quantile Transformation
from sklearn.preprocessing import QuantileTransformer
qt=QuantileTransformer(output_distribution ='normal')

df["Moderate Negative Skew_1"]=qt.fit_transform(df[["Moderate Negative Skew"]])
sm.qqplot(df["Moderate Negative Skew"],line='45')

df["Highly Negative Skew_1"]=qt.fit_transform(df[["Highly Negative Skew"]])
sm.qqplot(df["Highly Negative Skew"],line='45')

df["Highly Positive Skew_1"]=qt.fit_transform(df[["Highly Positive Skew"]])
sm.qqplot(df["Highly Positive Skew"],line='45')

df
```

OUPUT:

In [15]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [4]:
```
df=pd.read_csv("titanic_dataset.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [6]:
```
df['Cabin']=df['Cabin'].fillna(df['Cabin'].mode()[0])
df['Age']=df['Age'].fillna(df['Age'].mean())
df['Embarked']=df['Embarked'].fillna(df['Embarked'].mode()[0])
df.isnull().sum()
```

Out[6]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

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

Out[7]:
```
PassengerId    0.000000
Survived       0.478523
Pclass        -0.630548
Age            0.434488
SibSp          3.695352
Parch          2.749117
Fare           4.787317
dtype: float64
```

In [8]:
```
df1=df.copy()
```

In [8]:
```
df1=df.copy()
```

In [9]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [10]:
```
df1.skew()
```

Out[10]:
```
PassengerId    0.000000
Survived       0.478523
Pclass        -0.630548
Age            0.434488
SibSp          3.695352
Parch          2.749117
Fare           4.787317
dtype: float64
```

In [11]:
```
df1["Sibsp_1"]=np.sqrt(df1.SibSp)
```


```
 10  Cabin        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

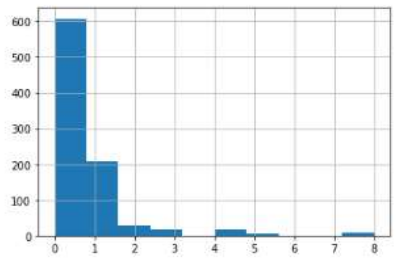
In [10]:
```
df1.skew()
```

Out[10]:
```
PassengerId    0.000000
Survived       0.478523
Pclass        -0.630548
Age            0.434488
SibSp          3.695352
Parch          2.749117
Fare           4.787317
dtype: float64
```

In [11]:
```
df1["Sibsp_1"]=np.sqrt(df1.SibSp)
```


In [12]:
```
df1.SibSp.hist()
```

Out[12]:
```
<AxesSubplot:>
```



In [31]:
```
df["Embarked"]=emb.fit_transform(df[["Embarked"]])
```

In [32]:
```
df
```

Out[32]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.000000	1	0	7.2500	1.0
1	2	1	1	female	38.000000	1	0	71.2833	0.0
2	3	1	3	female	26.000000	0	0	7.9250	1.0
3	4	1	1	female	35.000000	1	0	53.1000	1.0
4	5	0	3	male	35.000000	0	0	8.0500	1.0
...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.000000	0	0	13.0000	1.0
887	888	1	1	female	19.000000	0	0	30.0000	1.0
888	889	0	3	female	29.699118	1	2	23.4500	1.0
889	890	1	1	male	26.000000	0	0	30.0000	0.0
890	891	0	3	male	32.000000	0	0	7.7500	2.0

891 rows × 9 columns

In [35]:
```
from category_encoders import BinaryEncoder
```

In [36]:
```
be1=BinaryEncoder()
```

In [38]:
```
df['Sex']=be1.fit_transform(df[["Sex"]])
```

In [39]:
```
df
```

3	4	1	1	1	35.000000	1	0	53.1000	1.0
4	5	0	3	0	35.000000	0	0	8.0500	1.0
...	...	...	...	...	...	...	...	...	...
886	887	0	2	0	27.000000	0	0	13.0000	1.0
887	888	1	1	1	19.000000	0	0	30.0000	1.0
888	889	0	3	1	29.699118	1	2	23.4500	1.0
889	890	1	1	0	26.000000	0	0	30.0000	0.0
890	891	0	3	0	32.000000	0	0	7.7500	2.0

891 rows × 9 columns

In [42]:
```
#Function Transformation:
#Log Tranformation:
np.log(df["Age"])
```

Out[42]:
```
0      3.091042
1      3.637586
2      3.258097
3      3.555348
4      3.555348
         ...
886    3.295837
887    2.944439
888    3.391117
889    3.258097
890    3.465736
Name: Age, Length: 891, dtype: float64
```

In [43]:
```
#Reciprocal Transformation
np.reciprocal (df[["Fare"]])
```

Out[43]:

	Fare
0	0.137931

```
887    2.944439
888    3.391117
889    3.258097
890    3.465736
Name: Age, Length: 891, dtype: float64
```

In [43]:
```
#Reciprocal Transformation
np.reciprocal (df[["Fare"]])
```

Out[43]:

	Fare
0	0.137931
1	0.014029
2	0.126183
3	0.018832
4	0.124224
...	...
886	0.076923
887	0.033333
888	0.042644
889	0.033333
890	0.129032

891 rows × 1 columns

In [44]:
```
#sqrt transformation
np.sqrt(df["Embarked"])
```

Out[44]:
```
0      1.000000
1      0.000000
2      1.000000
3      1.000000
4      1.000000
         ...
886    1.000000
887    1.000000
888    1.000000
889    0.000000
890    1.414214
Name: Embarked, Length: 891, dtype: float64
```

In [45]:
```
#power transformation
df["Age_boxcox"],parameters=stats.boxcox(df["Age"])
```

In [53]:
```
df
```

Out[53]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age_boxcox	Pclass_boxcox	Fare_yeojohnson
0	1	0	3	0	22.000000	1	0	7.2500	1.0	14.251662	3.376116	1.906724
1	2	1	1	1	38.000000	1	0	71.2833	0.0	23.036649	0.000000	3.497640
2	3	1	3	1	26.000000	0	0	7.9250	1.0	16.531177	3.376116	1.970459
3	4	1	1	1	35.000000	1	0	53.1000	1.0	21.449592	0.000000	3.304258
4	5	0	3	0	35.000000	0	0	8.0500	1.0	21.449592	3.376116	1.981680
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	0	27.000000	0	0	13.0000	1.0	17.091018	1.359946	2.326029
887	888	1	1	1	19.000000	0	0	30.0000	1.0	12.493747	0.000000	2.916885
888	889	0	3	1	29.699118	1	2	23.4500	1.0	18.584256	3.376116	2.745246
889	890	1	1	0	26.000000	0	0	30.0000	0.0	16.531177	0.000000	2.916885
890	891	0	3	0	32.000000	0	0	7.7500	2.0	19.838238	3.376116	1.954457

891 rows × 12 columns

In [54]:
```
df["Pclass_boxcox"],parameters=stats.boxcox(df["Pclass"])
```

891 rows × 12 columns

In [58]:
```
df["Parch_yeojohnson"],parameters = stats.yeojohnson(df["Parch"])
```

In [59]:
```
df
```

Out[59]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age_boxcox	Pclass_boxcox	Fare_yeojohnson	Parch_yeojohnson
0	1	0	3	0	22.000000	1	0	7.2500	1.0	14.251662	3.376116	1.906724	-0.000000
1	2	1	1	1	38.000000	1	0	71.2833	0.0	23.036649	0.000000	3.497640	-0.000000
2	3	1	3	1	26.000000	0	0	7.9250	1.0	16.531177	3.376116	1.970459	-0.000000
3	4	1	1	1	35.000000	1	0	53.1000	1.0	21.449592	0.000000	3.304258	-0.000000
4	5	0	3	0	35.000000	0	0	8.0500	1.0	21.449592	3.376116	1.981680	-0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	0	27.000000	0	0	13.0000	1.0	17.091018	1.359946	2.326029	-0.000000
887	888	1	1	1	19.000000	0	0	30.0000	1.0	12.493747	0.000000	2.916885	-0.000000
888	889	0	3	1	29.699118	1	2	23.4500	1.0	18.584256	3.376116	2.745246	0.243296
889	890	1	1	0	26.000000	0	0	30.0000	0.0	16.531177	0.000000	2.916885	-0.000000
890	891	0	3	0	32.000000	0	0	7.7500	2.0	19.838238	3.376116	1.954457	-0.000000

891 rows × 13 columns

In [60]:
```
df.skew()
```

Out[60]:
```
PassengerId         0.000000
Survived            0.478523
Pclass             -0.630548
Sex                 0.618921
Age                 0.434488
SibSp               3.695352
Parch               2.749117
Fare                4.787317
Embarked           -0.147331
Age_boxcox          0.046849
Pclass_boxcox      -0.481963
Fare_yeojohnson    -0.040329
Parch_yeojohnson    1.228795
dtype: float64
```

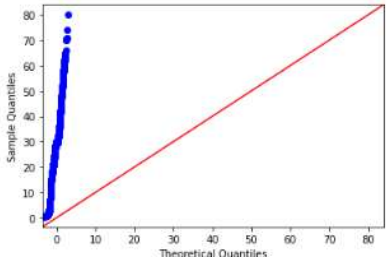
In [62]:
```
#Quantile transformation

from sklearn.preprocessing import QuantileTransformer
```

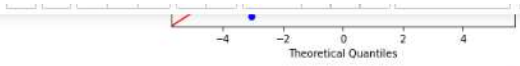
In [63]:
```
qt=QuantileTransformer(output_distribution ='normal',n_quantiles=891)
```

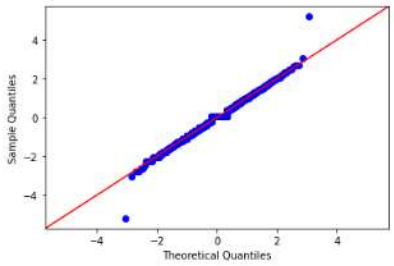
In [64]:
```
df["Age_1"]=qt.fit_transform(df[["Age"]])
```

In [66]:
```
sm.qqplot(df['Age'],line='45')
```

Out[66]:



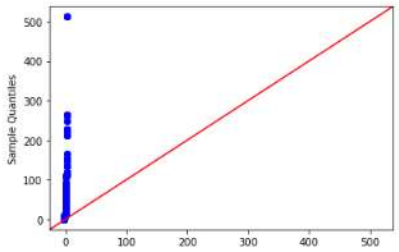


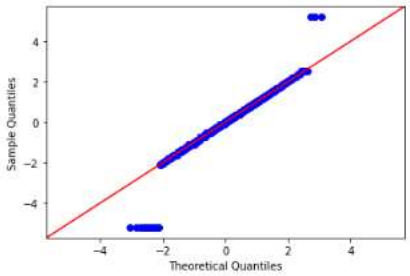


In [70]:
```
df["Fare_1"]=qt.fit_transform(df[["Fare"]])
```

In [71]:
```
sm.qqplot(df["Fare"],line='45')
```

Out[71]:

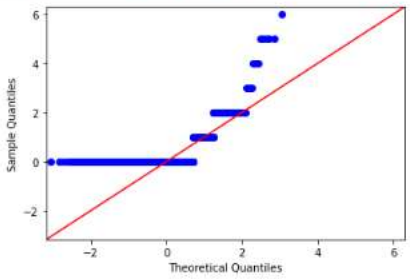


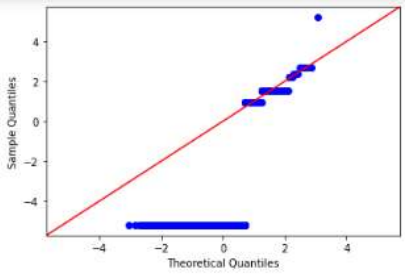


In [74]:
```
df["Parch_1"]=qt.fit_transform(df[["Parch"]])
```

In [75]:
```
sm.qqplot(df['Parch'],line='45')
```

Out[75]:





In [77]:
```
df
```

Out[77]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age_boxcox	Pclass_boxcox	Fare_yeojohnson	Parch_yeojohnson	A
0	1	0	3	0	22.000000	1	0	7.2500	1.0	14.251662	3.376116	1.906724	-0.000000	-0.6
1	2	1	1	1	38.000000	1	0	71.2833	0.0	23.036649	0.000000	3.497640	-0.000000	0.8
2	3	1	3	1	26.000000	0	0	7.9250	1.0	16.531177	3.376116	1.970459	-0.000000	-0.3
3	4	1	1	1	35.000000	1	0	53.1000	1.0	21.449592	0.000000	3.304258	-0.000000	0.6
4	5	0	3	0	35.000000	0	0	8.0500	1.0	21.449592	3.376116	1.981680	-0.000000	0.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	0	27.000000	0	0	13.0000	1.0	17.091018	1.359946	2.326029	-0.000000	-0.3
887	888	1	1	1	19.000000	0	0	30.0000	1.0	12.493747	0.000000	2.916885	-0.000000	-0.9
888	889	0	3	1	29.699118	1	2	23.4500	1.0	18.584256	3.376116	2.745246	0.243296	0.0
889	890	1	1	0	26.000000	0	0	30.0000	0.0	16.531177	0.000000	2.916885	-0.000000	-0.3
890	891	0	3	0	32.000000	0	0	7.7500	2.0	19.838238	3.376116	1.954457	-0.000000	0.4

891 rows × 16 columns

In [1]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [4]:
```
df=pd.read_csv("Data_To_Transform.csv")
```

In [5]:
```
df
```

Out[5]:

	Moderate Positive Skew	Highly Positive Skew	Moderate Negative Skew	Highly Negative Skew
0	0.899990	2.895074	11.180748	9.027485
1	1.113554	2.962385	10.842938	9.009762
2	1.156830	2.966378	10.817934	9.006134
3	1.264131	3.000324	10.764570	9.000125
4	1.323914	3.012109	10.753117	8.981296
...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882
9996	14.854474	16.396252	-3.147526	-3.772332
9997	15.262103	17.102991	-3.517256	-4.717950
9998	15.269983	17.628467	-4.689833	-5.670496
9999	16.204517	18.052331	-6.335679	-7.036091

10000 rows × 4 columns

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

Out[6]:
```
Moderate Positive Skew    0.656308
Highly Positive Skew      1.271249
```

```
Out[6]: Moderate Positive Skew    0.656308
        Highly Positive Skew      1.271249
        Moderate Negative Skew   -0.690244
        Highly Negative Skew     -1.201891
        dtype: float64
```

```
In [7]: #Function Transformation
        #Log Transformation
        np.log(df["Highly Positive Skew"])
```

```
Out[7]: 0       1.063011
        1       1.085995
        2       1.087342
        3       1.098720
        4       1.102640
                  ...
        9995    2.790522
        9996    2.797053
        9997    2.839253
        9998    2.869515
        9999    2.893275
        Name: Highly Positive Skew, Length: 10000, dtype: float64
```

```
In [8]: np.reciprocal(df["Moderate Positive Skew"])
```

```
Out[8]: 0       1.111123
        1       0.898026
        2       0.864431
        3       0.791057
        4       0.755336
                  ...
        9995    0.067801
        9996    0.067320
        9997    0.065522
        9998    0.065488
        9999    0.061711
        Name: Moderate Positive Skew, Length: 10000, dtype: float64
```

```
In [9]: np.sqrt(df["Highly Positive Skew"])
```

```
Out[9]: 0       1.701492
        1       1.721158
        2       1.722317
        3       1.732144
        4       1.735543
                  ...
        9995    4.036027
        9996    4.049229
        9997    4.135576
        9998    4.198627
        9999    4.248803
        Name: Highly Positive Skew, Length: 10000, dtype: float64
```

```
In [11]: np.square(df["Highly Negative Skew"])
```

```
Out[11]: 0       81.495480
         1       81.175811
         2       81.110452
         3       81.002257
         4       80.663680
                   ...
         9995    10.594259
         9996    14.230487
         9997    22.259048
         9998    32.154520
         9999    49.506580
         Name: Highly Negative Skew, Length: 10000, dtype: float64
```

```
In [12]: df["Highly positive Skew_boxcox"],parameters=stats.boxcox(df["Highly Positive Skew"])
```

```
In [13]: df
```

```
Out[13]:
```

	Moderate Positive Skew	Highly Positive Skew	Moderate Negative Skew	Highly Negative Skew	Highly positive Skew_boxcox
0	0.899990	2.895074	11.180748	9.027485	0.812909

9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	4.515148
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	4.585788
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	4.587141
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	4.744558

10000 rows × 6 columns

In [17]:
```
df["Moderate Negative Skew_yeojohnson"],parameters=stats.yeojohnson(df["Moderate Negative Skew"])
```

In [18]:
```
df
```

Out[18]:

	Moderate Positive Skew	Highly Positive Skew	Moderate Negative Skew	Highly Negative Skew	Highly positive Skew_boxcox	Moderate Positive Skew_yeojohnson	Moderate Negative Skew_yeojohnson
0	0.899990	2.895074	11.180748	9.027485	0.812909	-0.103432	29.137805
1	1.113554	2.962385	10.842938	9.009762	0.825921	0.109628	27.885272
2	1.156830	2.966378	10.817934	9.006134	0.826679	0.149502	27.793301
3	1.264131	3.000324	10.764570	9.000125	0.833058	0.244374	27.597360
4	1.323914	3.012109	10.753117	8.981296	0.835247	0.294988	27.555368
...	...	...	...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701	4.496676	-1.949345
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	4.515148	-2.028952
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	4.585788	-2.199693
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	4.587141	-2.697151
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	4.744558	-3.311402

10000 rows × 7 columns

In [20]:
```
df["Highly Negative Skew_yeojohnson"],parameters=stats.yeojohnson(df["Highly Negative Skew"])
```

Out[21]:

	Moderate Positive Skew	Highly Positive Skew	Moderate Negative Skew	Highly Negative Skew	Highly positive Skew_boxcox	Moderate Positive Skew_yeojohnson	Moderate Negative Skew_yeojohnson	Highly Negative Skew_yeojohnson
0	0.899990	2.895074	11.180748	9.027485	0.812909	-0.103432	29.137805	51.081487
1	1.113554	2.962385	10.842938	9.009762	0.825921	0.109628	27.885272	50.898041
2	1.156830	2.966378	10.817934	9.006134	0.826679	0.149502	27.793301	50.860530
3	1.264131	3.000324	10.764570	9.000125	0.833058	0.244374	27.597360	50.798432
4	1.323914	3.012109	10.753117	8.981296	0.835247	0.294988	27.555368	50.604084
...	...	...	...	...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701	4.496676	-1.949345	-1.433326
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	4.515148	-2.028952	-1.545673
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	4.585788	-2.199693	-1.722267
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	4.587141	-2.697151	-1.872430
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	4.744558	-3.311402	-2.053503

10000 rows × 8 columns

In [22]:
```
df.skew()
```

Out[22]:
```
Moderate Positive Skew              0.656308
Highly Positive Skew                1.271249
Moderate Negative Skew             -0.690244
Highly Negative Skew               -1.201891
Highly positive Skew_boxcox         0.023089
Moderate Positive Skew_yeojohnson   0.000155
Moderate Negative Skew_yeojohnson  -0.119651
Highly Negative Skew_yeojohnson    -0.274676
dtype: float64
```

In [23]:
```
#Quantile Transformation
from sklearn.preprocessing import QuantileTransformer
```

9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	4.687141	-2.697151	-1.872450
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	4.744558	-3.311402	-2.053503

10000 rows × 8 columns

In [22]:
```
df.skew()
```

Out[22]:
```
Moderate Positive Skew              0.656308
Highly Positive Skew                1.271249
Moderate Negative Skew             -0.690244
Highly Negative Skew               -1.201891
Highly positive Skew_boxcox         0.023089
Moderate Positive Skew_yeojohnson   0.000155
Moderate Negative Skew_yeojohnson  -0.119651
Highly Negative Skew_yeojohnson    -0.274676
dtype: float64
```

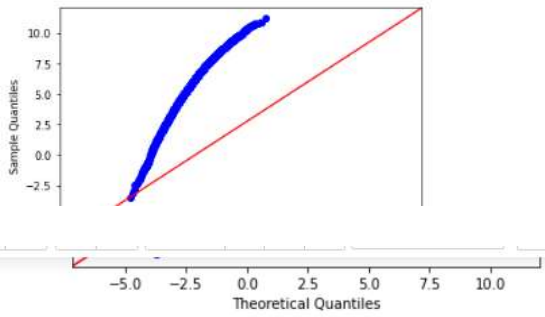
In [23]:
```
#Quantile Transformation
from sklearn.preprocessing import QuantileTransformer
```

In [25]:
```
qt=QuantileTransformer(output_distribution ='normal')
```

In [26]:
```
df["Moderate Negative Skew_1"]=qt.fit_transform(df[["Moderate Negative Skew"]])
```

In [27]:
```
sm.qqplot(df["Moderate Negative Skew"],line='45')
```

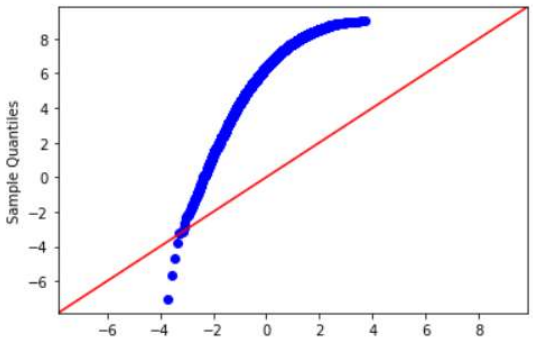
Out[27]:



In [28]:
```
df["Highly Negative Skew_1"]=qt.fit_transform(df[["Highly Negative Skew"]])
```

In [29]:
```
sm.qqplot(df["Highly Negative Skew"],line='45')
```

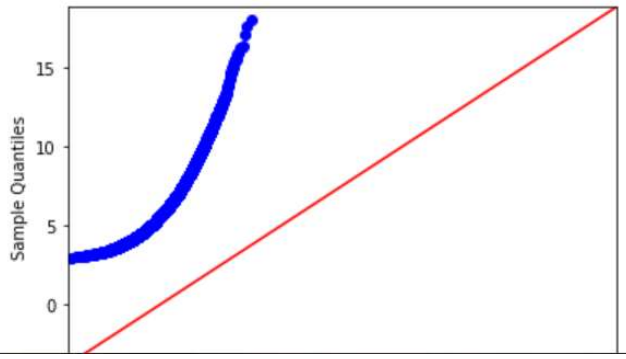
Out[29]:



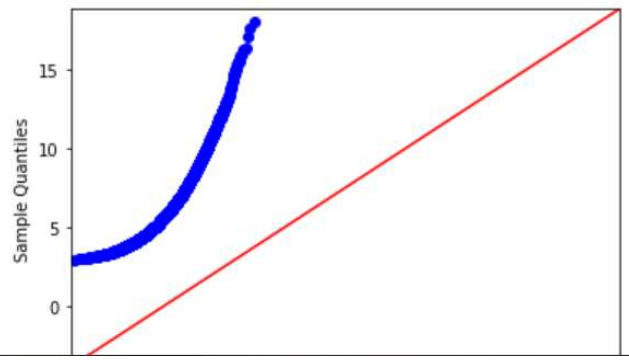
In [32]:
```
df["Highly Positive Skew_1"]=qt.fit_transform(df[["Highly Positive Skew"]])
```

In [33]:
```
sm.qqplot(df["Highly Positive Skew"],line='45')
```

Out[33]:



```
In [32]: df["Highly Positive Skew_1"]=qt.fit_transform(df[["Highly Positive Skew"]])

In [33]: sm.qqplot(df["Highly Positive Skew"],line='45')

Out[33]:
```



Result:

The various feature transformation techniques on a dataset and save the data to a file has been performed successfully.