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

OUPUT

titanic_dataset.csv:

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

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

df.drop("Name",axis=1,inplace=True)
df.drop("Cabin",axis=1,inplace=True)
df.drop("Ticket",axis=1,inplace=True)
df.isnull().sum()

df["Age"]=df["Age"].fillna(df["Age"].median())
df["Embarked"]=df["Embarked"].fillna(df["Embarked"].mode()[0])
df.info()

from sklearn.preprocessing import OrdinalEncoder

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

```
df

#FUNCTION TRANSFORMATION:
#Log Transformation
np.log(df["Fare"])
#ReciprocalTransformation
np.reciprocal(df["Age"])
#Squareroot 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["SibSp _yeojohnson"], parameters=stats.yeojohnson(df["SibSp"])
df
df["Parch _yeojohnson"], parameters=stats.yeojohnson(df["Parch"])
df

#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')
plt.show()
sm.qqplot(df['Age_1'],line='45')
plt.show()
df["Fare_1"]=qt.fit_transform(df[["Fare"]])
sm.qqplot(df["Fare"],line='45')
plt.show()
sm.qqplot(df['Fare_1'],line='45')
plt.show()
```

```
df.skew()
df
```

OUTPUT

Reading the data set

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

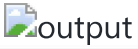
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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

Cleaning the dataset:

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Embarked         2
dtype: int64
```




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

```
df
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	1.0
1	2	1	1	female	38.0	1	0	71.2833	0.0
2	3	1	3	female	26.0	0	0	7.9250	1.0
3	4	1	1	female	35.0	1	0	53.1000	1.0
4	5	0	3	male	35.0	0	0	8.0500	1.0
...	...	...	...	...	...	...	...	...	...

886	887	0	2	male	27.0	0	0	13.0000	1.0
887	888	1	1	female	19.0	0	0	30.0000	1.0
888	889	0	3	female	28.0	1	2	23.4500	1.0
889	890	1	1	male	26.0	0	0	30.0000	0.0
890	891	0	3	male	32.0	0	0	7.7500	2.0

891 rows × 9 columns

FUNCTION TRANSFORMATION:

```
#ReciprocalTransformation
np.reciprocal(df["Age"])
```

```
0      0.045455
1      0.026316
2      0.038462
3      0.028571
4      0.028571
         ...
886    0.037037
887    0.052632
888    0.035714
889    0.038462
890    0.031250
Name: Age, Length: 891, dtype: float64
```

```
#Squareroot Transformation:
np.sqrt(df["Embarked"])
```

```
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
```

POWER TRANSFORMATION:

```
#Boxcox method:
df["Age _boxcox"], parameters=stats.boxcox(df["Age"])
df
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age _boxcox
0	1	0	3	male	22.0	1	0	7.2500	1.0	13.468119
1	2	1	1	female	38.0	1	0	71.2833	0.0	21.498728
2	3	1	3	female	26.0	0	0	7.9250	1.0	15.563417
3	4	1	1	female	35.0	1	0	53.1000	1.0	20.056110
4	5	0	3	male	35.0	0	0	8.0500	1.0	20.056110
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	16.076643
887	888	1	1	female	19.0	0	0	30.0000	1.0	11.845513
888	889	0	3	female	28.0	1	2	23.4500	1.0	16.586014
889	890	1	1	male	26.0	0	0	30.0000	0.0	15.563417
890	891	0	3	male	32.0	0	0	7.7500	2.0	18.588067

891 rows × 10 columns

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

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age _boxcox	Pclass _boxcox

0	1	0	3	male	22.0	1	0	7.2500	1.0	13.468119	3.376116
1	2	1	1	female	38.0	1	0	71.2833	0.0	21.498728	0.000000
2	3	1	3	female	26.0	0	0	7.9250	1.0	15.563417	3.376116
3	4	1	1	female	35.0	1	0	53.1000	1.0	20.056110	0.000000
4	5	0	3	male	35.0	0	0	8.0500	1.0	20.056110	3.376116
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	16.076643	1.359946
887	888	1	1	female	19.0	0	0	30.0000	1.0	11.845513	0.000000
888	889	0	3	female	28.0	1	2	23.4500	1.0	16.586014	3.376116
889	890	1	1	male	26.0	0	0	30.0000	0.0	15.563417	0.000000
890	891	0	3	male	32.0	0	0	7.7500	2.0	18.588067	3.376116

891 rows × 11 columns

```
#Yeojohnson method:
df["Fare _yeojohnson"], parameters=stats.yeojohnson(df["Fare"])
df
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age _boxcox	Pclass _boxcox	Fare _yeojohnson
0	1	0	3	male	22.0	1	0	7.2500	1.0	13.468119	3.376116	1.906724
1	2	1	1	female	38.0	1	0	71.2833	0.0	21.498728	0.000000	3.497640
2	3	1	3	female	26.0	0	0	7.9250	1.0	15.563417	3.376116	1.970459
3	4	1	1	female	35.0	1	0	53.1000	1.0	20.056110	0.000000	3.304258
4	5	0	3	male	35.0	0	0	8.0500	1.0	20.056110	3.376116	1.981680
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	16.076643	1.359946	2.326029
887	888	1	1	female	19.0	0	0	30.0000	1.0	11.845513	0.000000	2.916885
888	889	0	3	female	28.0	1	2	23.4500	1.0	16.586014	3.376116	2.745246
889	890	1	1	male	26.0	0	0	30.0000	0.0	15.563417	0.000000	2.916885
890	891	0	3	male	32.0	0	0	7.7500	2.0	18.588067	3.376116	1.954457

891 rows × 12 columns

```
df["SibSp _yeojohnson"], parameters=stats.yeojohnson(df["SibSp"])
df
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age _boxcox	Pclass _boxcox	Fare _yeojohnson	SibSp _yeojohnson
0	1	0	3	male	22.0	1	0	7.2500	1.0	13.468119	3.376116	1.906724	0.323389
1	2	1	1	female	38.0	1	0	71.2833	0.0	21.498728	0.000000	3.497640	0.323389
2	3	1	3	female	26.0	0	0	7.9250	1.0	15.563417	3.376116	1.970459	-0.000000
3	4	1	1	female	35.0	1	0	53.1000	1.0	20.056110	0.000000	3.304258	0.323389
4	5	0	3	male	35.0	0	0	8.0500	1.0	20.056110	3.376116	1.981680	-0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	16.076643	1.359946	2.326029	-0.000000
887	888	1	1	female	19.0	0	0	30.0000	1.0	11.845513	0.000000	2.916885	-0.000000
888	889	0	3	female	28.0	1	2	23.4500	1.0	16.586014	3.376116	2.745246	0.323389
889	890	1	1	male	26.0	0	0	30.0000	0.0	15.563417	0.000000	2.916885	-0.000000
890	891	0	3	male	32.0	0	0	7.7500	2.0	18.588067	3.376116	1.954457	-0.000000

891 rows × 13 columns

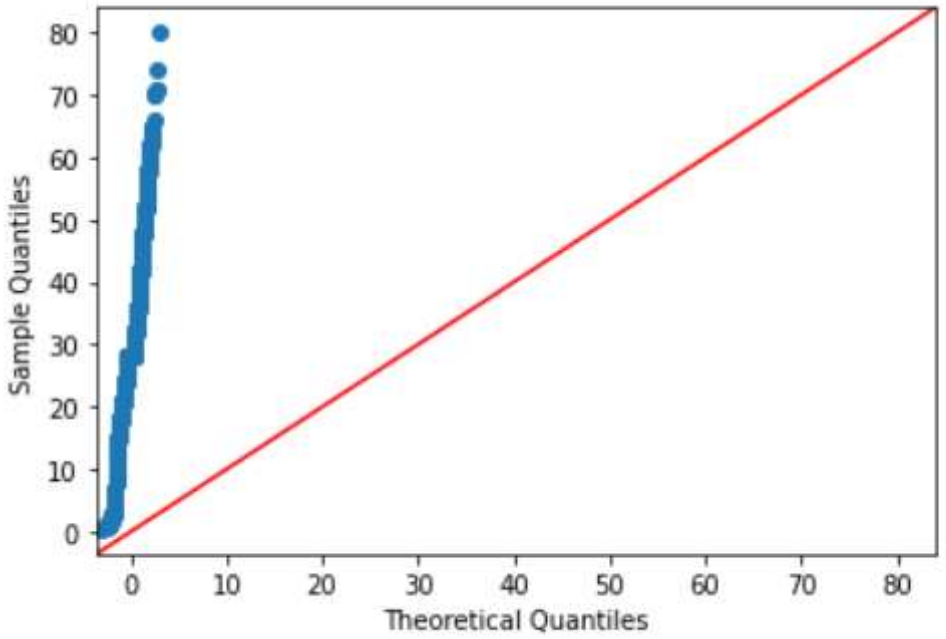
```
df["Parch _yeojohnson"], parameters=stats.yeojohnson(df["Parch"])
df
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age _boxcox	Pclass _boxcox	Fare _yeojohnson	SibSp _yeojohnson	Parch _yeojohnson
0	1	0	3	male	22.0	1	0	7.2500	1.0	13.468119	3.376116	1.906724	0.323389	-0.000000
1	2	1	1	female	38.0	1	0	71.2833	0.0	21.498728	0.000000	3.497640	0.323389	-0.000000
2	3	1	3	female	26.0	0	0	7.9250	1.0	15.563417	3.376116	1.970459	-0.000000	-0.000000
3	4	1	1	female	35.0	1	0	53.1000	1.0	20.056110	0.000000	3.304258	0.323389	-0.000000
4	5	0	3	male	35.0	0	0	8.0500	1.0	20.056110	3.376116	1.981680	-0.000000	-0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	16.076643	1.359946	2.326029	-0.000000	-0.000000
887	888	1	1	female	19.0	0	0	30.0000	1.0	11.845513	0.000000	2.916885	-0.000000	-0.000000
888	889	0	3	female	28.0	1	2	23.4500	1.0	16.586014	3.376116	2.745246	0.323389	0.243296
889	890	1	1	male	26.0	0	0	30.0000	0.0	15.563417	0.000000	2.916885	-0.000000	-0.000000
890	891	0	3	male	32.0	0	0	7.7500	2.0	18.588067	3.376116	1.954457	-0.000000	-0.000000

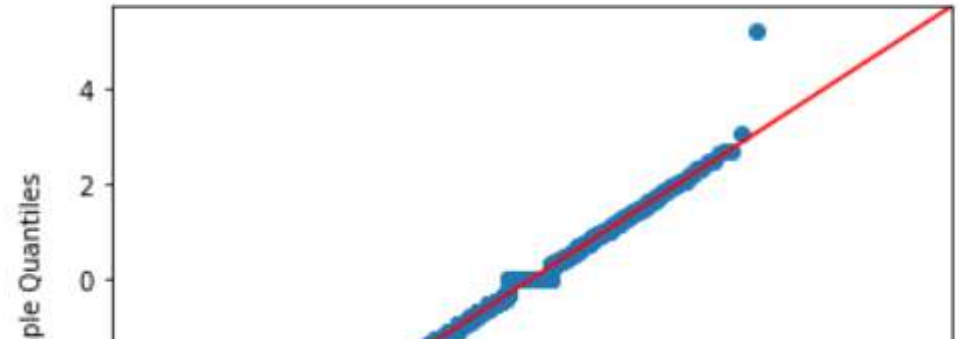
891 rows × 14 columns

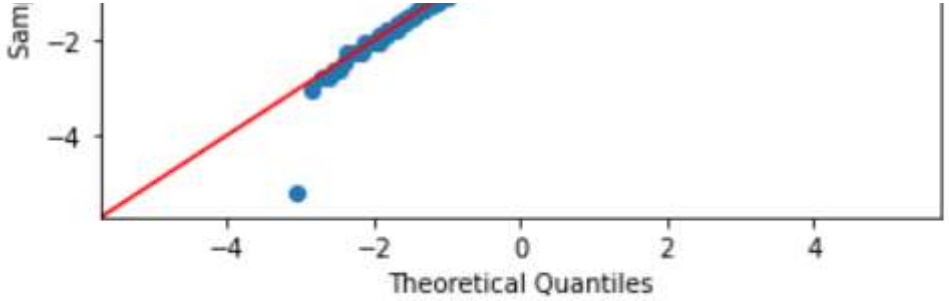
QUANTILE TRANSFORMATION

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

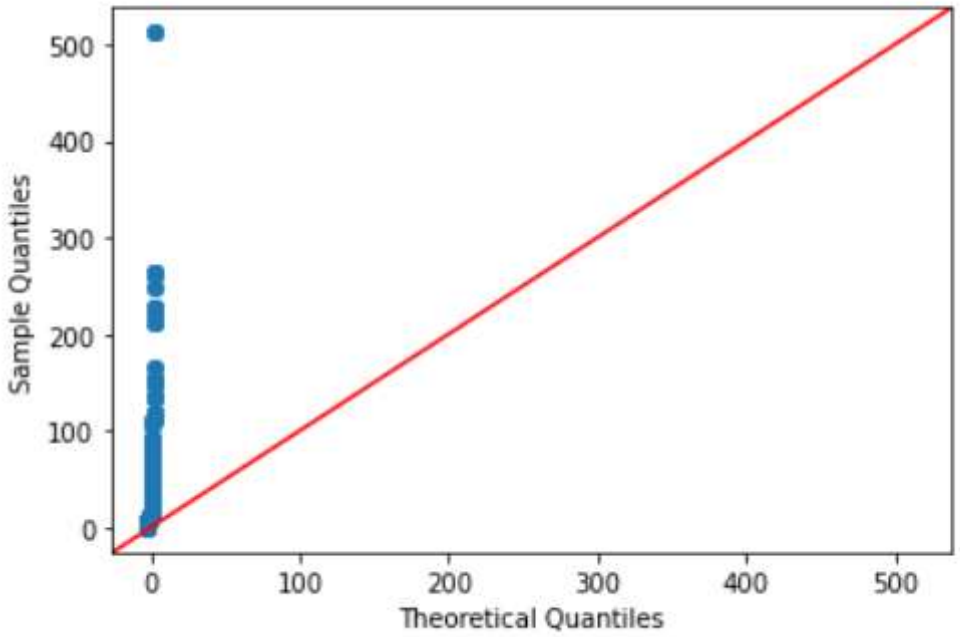



```
sm.qqplot(df['Age_1'],line='45')
plt.show()
```

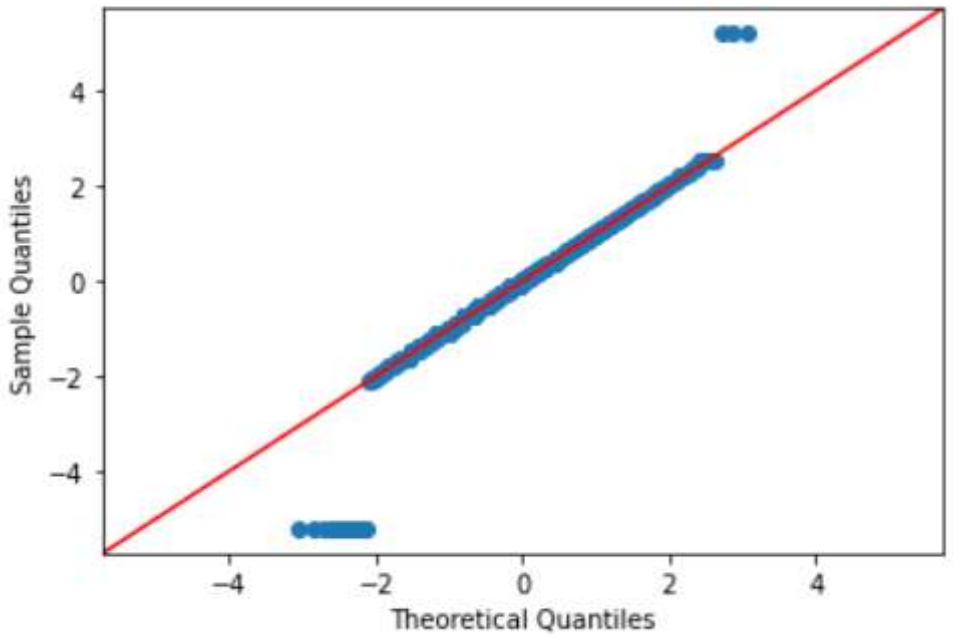



Sam
−2
−4
−4
−2
0
2
4
Theoretical Quantiles

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



```
sm.qqplot(df['Fare_1'],line='45')
plt.show()
```



Final Result:

```
PassengerId          0.000000
Survived             0.478523
Pclass              -0.630548
Age                  0.510245
SibSp                3.695352
Parch                2.749117
Fare                 4.787317
Embarked            -0.147331
Age _boxcox          0.060508
Pclass _boxcox      -0.481963
Fare _yeojohnson    -0.040329
SibSp _yeojohnson    0.808608
Parch _yeojohnson    1.228795
Age_1               -0.006827
Fare_1              -0.928213
dtype: float64
```

```
df
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Age _boxcox	Pclass _boxcox	Fare _yeojohnson	SibSp _yeojohnson	Parch _yeojohnson	Age_1	Fare_1
0	1	0	3	male	22.0	1	0	7.2500	1.0	13.468119	3.376116	1.906724	0.323389	-0.000000	-0.695859	-1.373288
1	2	1	1	female	38.0	1	0	71.2833	0.0	21.498728	0.000000	3.497640	0.323389	-0.000000	0.823696	1.202387
2	3	1	3	female	26.0	0	0	7.9250	1.0	15.563417	3.376116	1.970459	-0.000000	-0.000000	-0.391395	-0.644732
3	4	1	1	female	35.0	1	0	53.1000	1.0	20.056110	0.000000	3.304258	0.323389	-0.000000	0.662165	0.989391
4	5	0	3	male	35.0	0	0	8.0500	1.0	20.056110	3.376116	1.981680	-0.000000	-0.000000	0.662165	-0.537371
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	1.0	16.076643	1.359946	2.326029	-0.000000	-0.000000	-0.337215	-0.110063
887	888	1	1	female	19.0	0	0	30.0000	1.0	11.845513	0.000000	2.916885	-0.000000	-0.000000	-0.957723	0.624066
888	889	0	3	female	28.0	1	2	23.4500	1.0	16.586014	3.376116	2.745246	0.323389	0.243296	-0.021125	0.285474
889	890	1	1	male	26.0	0	0	30.0000	0.0	15.563417	0.000000	2.916885	-0.000000	-0.000000	-0.391395	0.624066
890	891	0	3	male	32.0	0	0	7.7500	2.0	18.588067	3.376116	1.954457	-0.000000	-0.000000	0.493940	-1.090982

891 rows × 16 columns

data_to_transform.csv:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import scipy.stats as stats
df=pd.read_csv("Data_To_Transform.csv")
df
df.skew()

#FUNCTION TRANSFORMATION:
#Log Transformation
np.log(df["Highly Positive Skew"])
#Reciprocal Transformation
np.reciprocal(df["Moderate Positive Skew"])
#Square Root Transformation
np.sqrt(df["Highly Positive Skew"])
#Square Transformation
np.square(df["Highly Negative Skew"])

#POWER TRANSFORMATION:
df["Highly Positive Skew_boxcox"], parameters=stats.boxcox(df["Highly Positive Skew"])
df
df["Moderate Positive Skew_yeojohnson"], parameters=stats.yeojohnson(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
```

```
#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')
plt.show()
sm.qqplot(df['Moderate Negative Skew_1'],line='45')
plt.show()
df["Highly Negative Skew_1"]=qt.fit_transform(df[["Highly Negative Skew"]])
sm.qqplot(df['Highly Negative Skew'],line='45')
plt.show()
sm.qqplot(df['Highly Negative Skew_1'],line='45')
plt.show()
df["Moderate Positive Skew_1"]=qt.fit_transform(df[["Moderate Positive Skew"]])
sm.qqplot(df['Moderate Positive Skew'],line='45')
plt.show()
sm.qqplot(df['Moderate Positive Skew_1'],line='45')
plt.show()
df["Highly Positive Skew_1"]=qt.fit_transform(df[["Highly Positive Skew"]])
sm.qqplot(df['Highly Positive Skew'],line='45')
plt.show()
sm.qqplot(df['Highly Positive Skew_1'],line='45')
plt.show()

df.skew()
df
```

Output:

Reading the data set:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import scipy.stats as stats
df=pd.read_csv("Data_To_Transform.csv")
df
```

	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

```
df.skew()
```

```
Moderate Positive Skew    0.656308
Highly Positive Skew      1.271249
Moderate Negative Skew   -0.690244
Highly Negative Skew     -1.201891
dtype: float64
```

FUNCTION TRANSFORMATION:

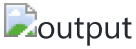
```
#FUNCTION TRANSFORMATION:
#Log Transformation
np.log(df["Highly Positive Skew"])
```

```
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
```

```
#Reciprocal Transformation
np.reciprocal(df["Moderate Positive Skew"])
```

```
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
```



```
#Square Transformation
np.square(df["Highly Negative Skew"])
```

```
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
```

POWER TRANSFORMATION:

```
#POWER TRANSFORMATION:
df["Highly Positive Skew_boxcox"], parameters=stats.boxcox(df["Highly Positive Skew"])
df
```

	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
1	1.113554	2.962385	10.842938	9.009762	0.825921
2	1.156830	2.966378	10.817934	9.006134	0.826679
3	1.264131	3.000324	10.764570	9.000125	0.833058
4	1.323914	3.012109	10.753117	8.981296	0.835247
...	...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525

10000 rows × 5 columns

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

	Moderate Positive Skew	Highly Positive Skew	Moderate Negative Skew	Highly Negative Skew	Highly Positive Skew_boxcox	Moderate Positive Skew_yeojohnson
0	0.899990	2.895074	11.180748	9.027485	0.812909	0.690865
1	1.113554	2.962385	10.842938	9.009762	0.825921	0.815560
2	1.156830	2.966378	10.817934	9.006134	0.826679	0.839629
3	1.264131	3.000324	10.764570	9.000125	0.833058	0.897735
4	1.323914	3.012109	10.753117	8.981296	0.835247	0.929191
...	...	...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701	3.828849
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	3.841318
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	3.888934
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	3.889845
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	3.995584

10000 rows × 6 columns

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

	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	0.690865	29.137805
1	1.113554	2.962385	10.842938	9.009762	0.825921	0.815560	27.885272
2	1.156830	2.966378	10.817934	9.006134	0.826679	0.839629	27.793301
3	1.264131	3.000324	10.764570	9.000125	0.833058	0.897735	27.597360
4	1.323914	3.012109	10.753117	8.981296	0.835247	0.929191	27.555368
...	...	...	...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701	3.828849	-1.949345
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	3.841318	-2.028952
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	3.888934	-2.199693
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	3.889845	-2.697151
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	3.995584	-3.311402

10000 rows × 7 columns

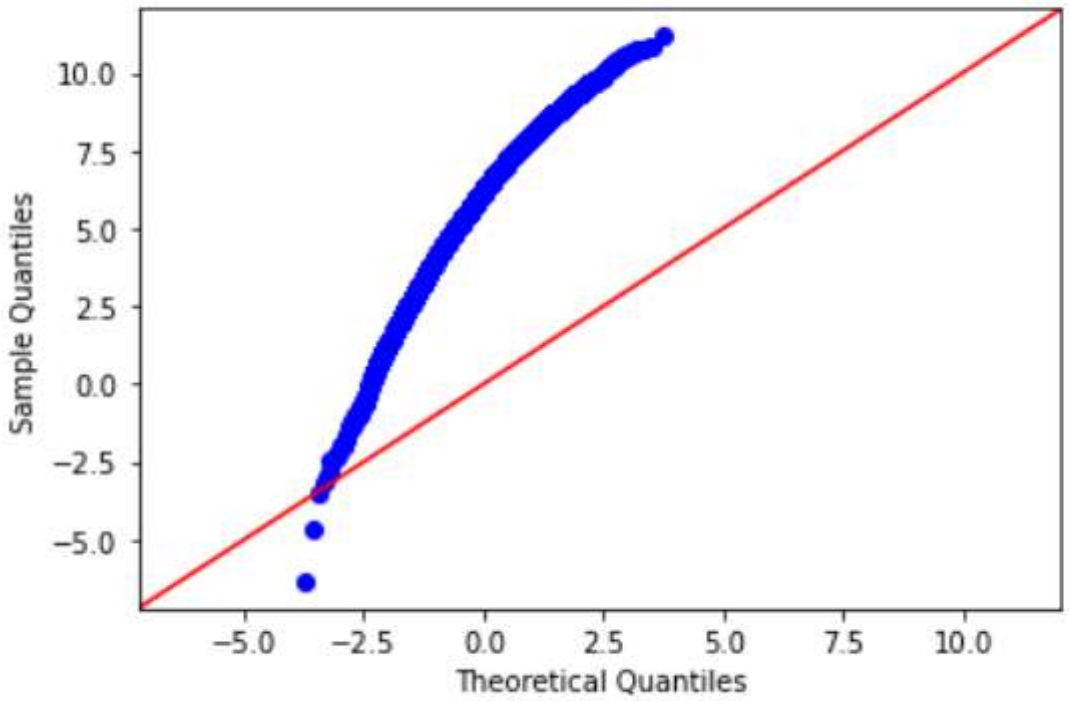
```
df["Highly Negative Skew_yeojohnson"], parameters=stats.yeojohnson(df["Highly Negative Skew"])
df
```

	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	0.690865	29.137805	51.081487
1	1.113554	2.962385	10.842938	9.009762	0.825921	0.815560	27.885272	50.898041
2	1.156830	2.966378	10.817934	9.006134	0.826679	0.839629	27.793301	50.860530
3	1.264131	3.000324	10.764570	9.000125	0.833058	0.897735	27.597360	50.798432
4	1.323914	3.012109	10.753117	8.981296	0.835247	0.929191	27.555368	50.604084
...	...	...	...	...	...	...	...	...
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701	3.828849	-1.949345	-1.433326
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	3.841318	-2.028952	-1.545673
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	3.888934	-2.199693	-1.722267
9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	3.889845	-2.697151	-1.872430
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	3.995584	-3.311402	-2.053503

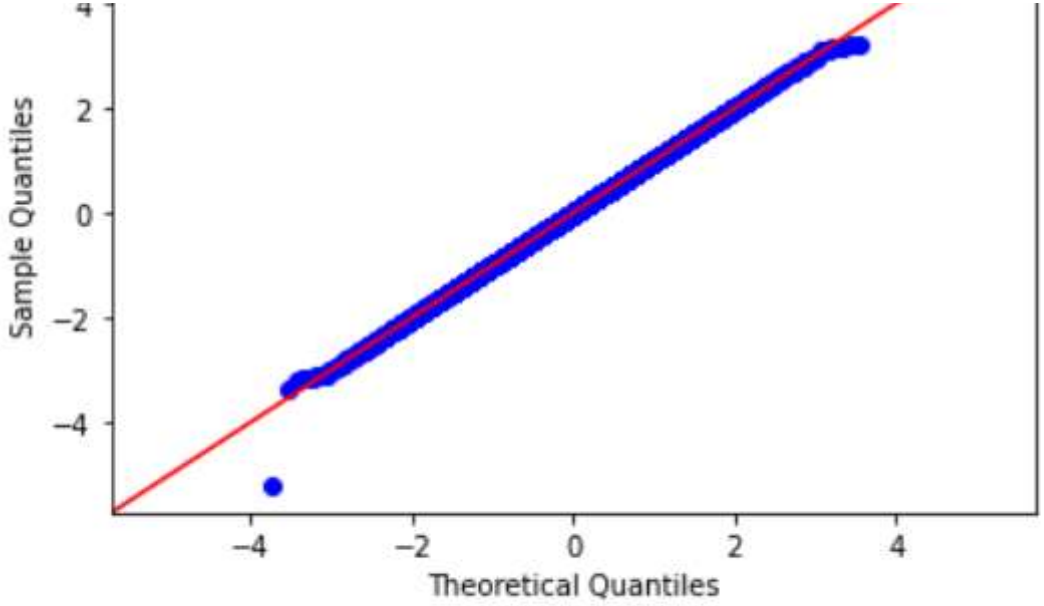
10000 rows × 8 columns

QUANTILE TRANSFORAMATION:

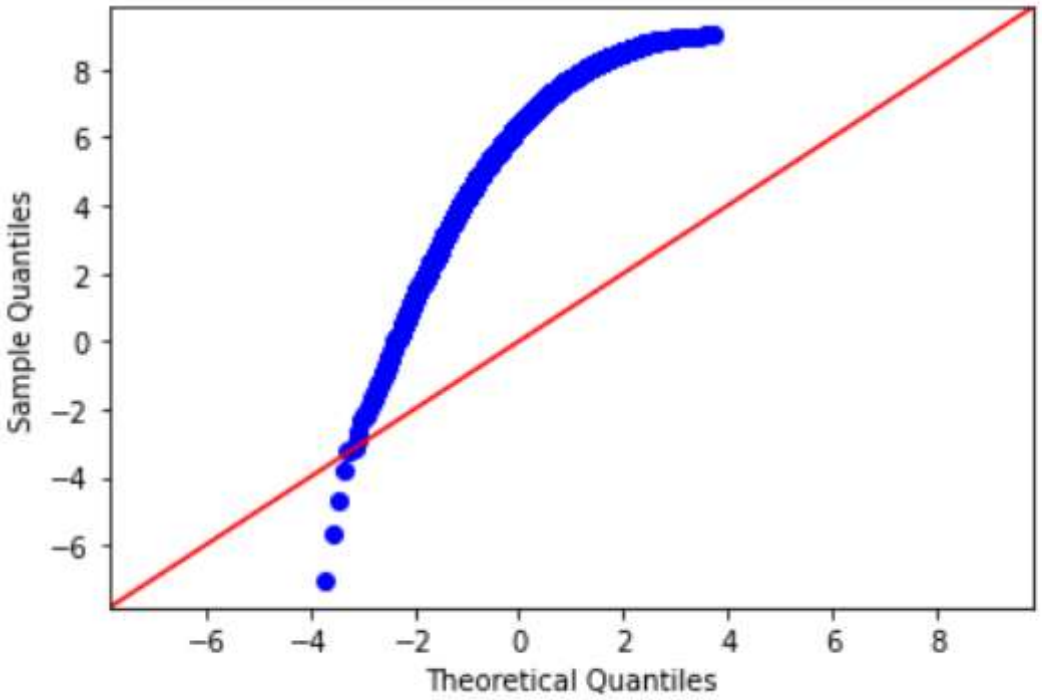
```
#QUANTILE TRANSFORMATION:
from sklearn.preprocessing import QuantileTransformer
qt=QuantileTransformer(output_distribution='normal')
df["Moderate Negative Skew_1"]=qt.fit_transform(df[["Moderate Negative Skew"]])
sm.qqplot(df['Moderate Negative Skew'],line='45')
plt.show()
```



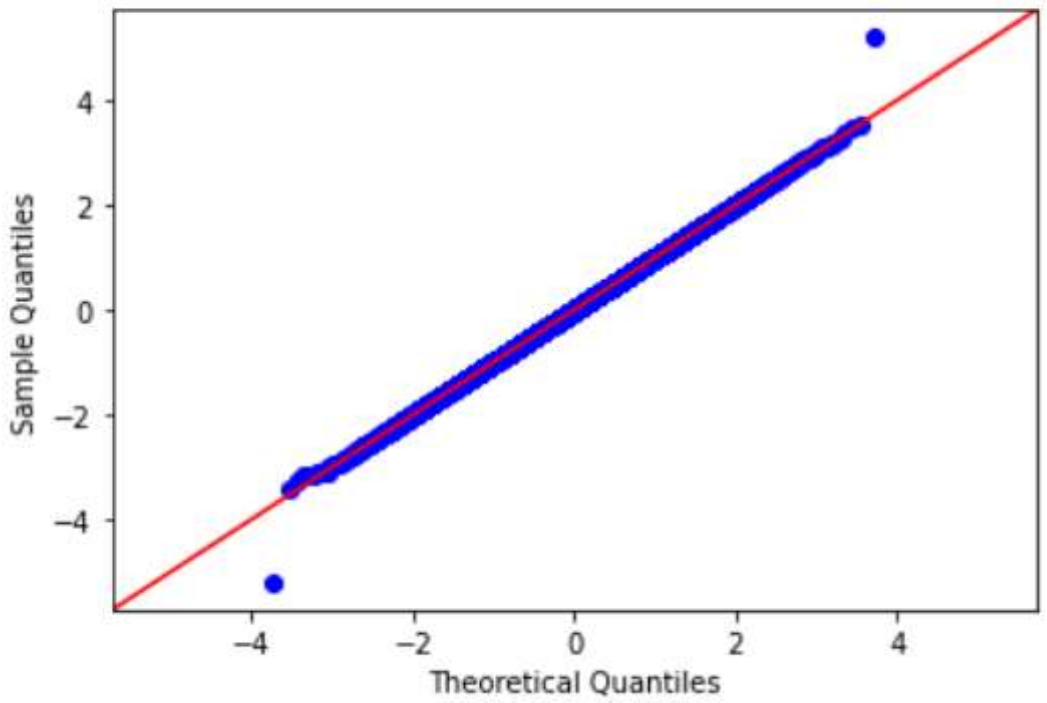

```
sm.qqplot(df['Moderate Negative Skew_1'],line='45')
plt.show()
```


Sample Quantiles
Theoretical Quantiles
4
2
0
−2
−4
−4
−2
0
2
4

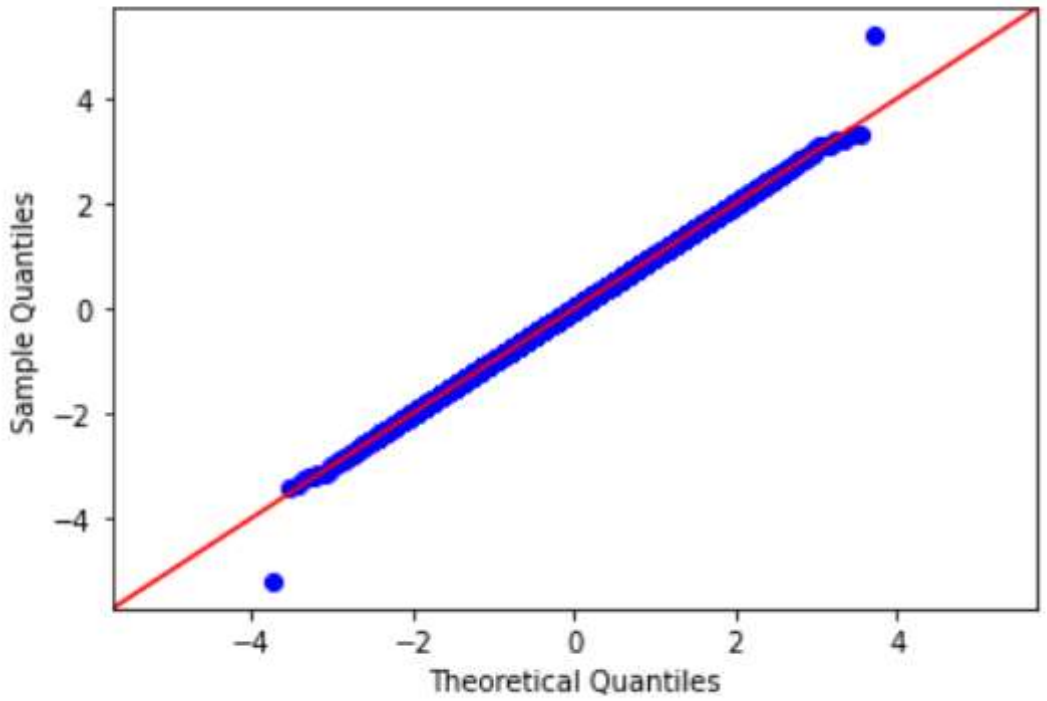
```
df["Highly Negative Skew_1"]=qt.fit_transform(df[["Highly Negative Skew"]])
sm.qqplot(df['Highly Negative Skew'],line='45')
plt.show()
```



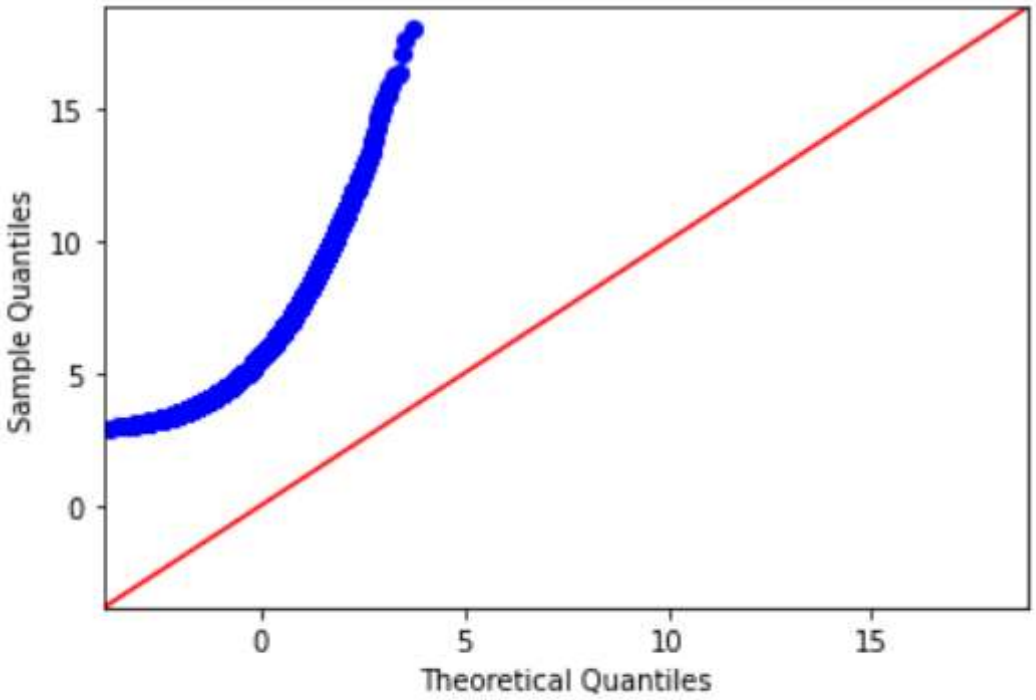
```
sm.qqplot(df['Highly Negative Skew_1'],line='45')
plt.show()
```



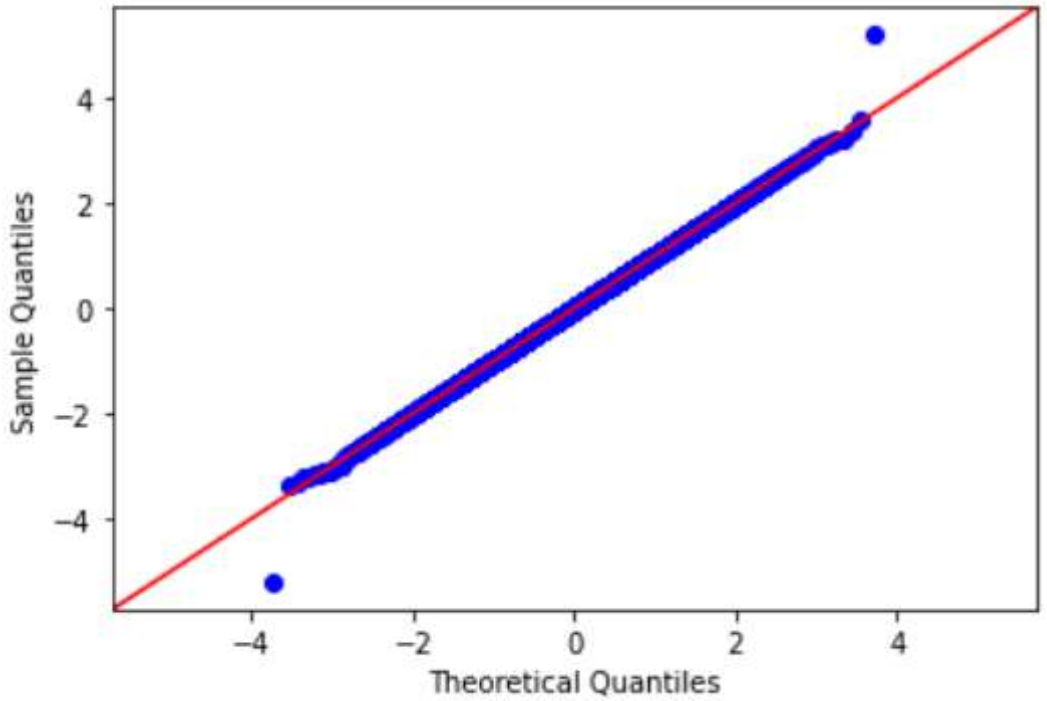
```
sm.qqplot(df['Moderate Positive Skew_1'],line='45')
plt.show()
```



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



```
sm.qqplot(df['Highly Positive Skew_1'],line='45')
plt.show()
```



Final Result:

```
df.skew()
```

```
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    -0.001168
Moderate Negative Skew_yeojohnson    -0.119651
Highly Negative Skew_yeojohnson      -0.274676
Moderate Negative Skew_1             -0.001436
Highly Negative Skew_1                0.003126
Moderate Positive Skew_1              0.000895
Highly Positive Skew_1               -0.000408
dtype: float64
```

```
df
```

	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	Moderate Negative Skew_1	Highly Negative Skew_1	Moderate Positive Skew_1	Hig Posi Ske
0	0.899990	2.895074	11.180748	9.027485	0.812909	0.690865	29.137805	51.081487	5.199338	5.199338	-5.199338	-5.199
1	1.113554	2.962385	10.842938	9.009762	0.825921	0.815560	27.885272	50.898041	3.227288	3.503580	-3.392734	-3.342
2	1.156830	2.966378	10.817934	9.006134	0.826679	0.839629	27.793301	50.860530	3.206801	3.453669	-3.341853	-3.326
3	1.264131	3.000324	10.764570	9.000125	0.833058	0.897735	27.597360	50.798432	3.167111	3.386210	-3.243698	-3.216
4	1.323914	3.012109	10.753117	8.981296	0.835247	0.929191	27.555368	50.604084	3.159208	3.239746	-3.200142	-3.186
...	...	...	...	...	...	...	...	...	...	...	...	
9995	14.749050	16.289513	-2.980821	-3.254882	1.457701	3.828849	-1.949345	-1.433326	-3.147619	-3.131880	3.203464	3.198
9996	14.854474	16.396252	-3.147526	-3.772332	1.459189	3.841318	-2.028952	-1.545673	-3.162489	-3.174835	3.225052	3.216
9997	15.262103	17.102991	-3.517256	-4.717950	1.468681	3.888934	-2.199693	-1.722267	-3.198205	-3.272809	3.326574	3.372

9998	15.269983	17.628467	-4.689833	-5.670496	1.475357	3.889845	-2.697151	-1.872430	-3.350199	-3.419532	3.328914	3.588
9999	16.204517	18.052331	-6.335679	-7.036091	1.480525	3.995584	-3.311402	-2.053503	-5.199338	-5.199338	5.199338	5.199

10000 rows × 12 columns

Result:

Hence, Feature transformation techniques is been performed on given dataset and saved into a file successfully.