ORDINAL ENCODER

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

df = pd.read_csv('Encoding Data.csv')

pm = ['Hot','Warm','Cold']

oe = OrdinalEncoder(categories = [pm])
oe.fit(df[['ord_2']])

df['ord_2'] = oe.transform(df[['ord_2']])

df['ord_2']
```

```
0    0.0
1    1.0
2    2.0
3    1.0
4    2.0
5    0.0
6    2.0
7    2.0
8    1.0
9    0.0
Name: ord_2, dtype: float64
```

LABEL ENCODER

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

df = pd.read_csv('Encoding Data.csv')

le = LabelEncoder()
df['nom_0'] = le.fit_transform(df['nom_0'])

df['nom_0']
```

```
0    2
1    0
2    0
3    1
4    2
5    1
6    2
7    2
8    0
```

```
9    2
Name: nom_0, dtype: int64
```

ONE HOT ENCODER

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
ohe = OneHotEncoder(sparse=False)
df2=df.copy()
enc=pd.DataFrame(ohe.fit_transform(df2[['nom_0']]))
df2=pd.concat([df2,enc],axis=1)
pd.get_dummies(df2['nom_0'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

```
{"summary":"{\n  \"name\": \"pd\",\n  \"rows\": 10,\n  \"fields\": [\n
{\n      \"column\": 0,\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 1,\n      \"properties\": {\n
\"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 2,\n      \"properties\": {\n
\"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

df2

```
{"summary":"{\n  \"name\": \"df2\",\n  \"rows\": 10,\n  \"fields\": [\
n    {\n      \"column\": \"id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 0,\n
\"max\": 9,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          8,\n          1,\n          5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bin_1\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"T\",\n          \"F\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bin_2\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Y\",\n          \"N\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"nom_0\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 2,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          2,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ord_2\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Hot\",\n          \"Warm\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 0,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.4830458915396479,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          1.0,\n          0.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": 1,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.42163702135578396,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          1.0,\n          0.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": 2,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.5270462766947299,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          0.0,\n          1.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df2"}
```


```
pip install --upgrade category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl.metadata
(8.0 kB)
Requirement already satisfied: numpy>=1.14.0 in
/usr/local/lib/python3.10/dist-packages (from category_encoders)
(1.26.4)
Requirement already satisfied: scikit-learn>=0.20.0 in
/usr/local/lib/python3.10/dist-packages (from category_encoders)
(1.3.2)
Requirement already satisfied: scipy>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from category_encoders)
(1.13.1)
Requirement already satisfied: statsmodels>=0.9.0 in
/usr/local/lib/python3.10/dist-packages (from category_encoders)
(0.14.3)
Requirement already satisfied: pandas>=1.0.5 in
/usr/local/lib/python3.10/dist-packages (from category_encoders)
(2.1.4)
Requirement already satisfied: patsy>=0.5.1 in
/usr/local/lib/python3.10/dist-packages (from category_encoders)
(0.5.6)
Requirement already satisfied: python-dateutil>=2.8.2 in
```

```
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5-
>category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5-
>category_encoders) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5-
>category_encoders) (2024.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0-
>category_encoders) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0-
>category_encoders) (3.5.0)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0-
>category_encoders) (24.1)
Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 1.8 MB/s eta
0:00:00
```

BINARY ENCODER

```
from category_encoders import BinaryEncoder

df= pd.read_csv('data.csv')
be=BinaryEncoder()
nd=be.fit_transform(df['Ord_2'])
df=pd.concat([df,nd],axis=1)
dfb1=df.copy()
dfb1
```


```
{"summary":"{\n  \"name\": \"dfb1\",\n  \"rows\": 10,\n  \"fields\":
[\n    {\n      \"column\": \"id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 0,\n
\"max\": 9,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          8,\n          1,\n          5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bin_1\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"M\",\n          \"F\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bin_2\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Y\",\n          \"N\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n
```

```
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"Bangalore\",\n          \"Chennai\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ord_1\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"Warm\",\n          \"Cold\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ord_2\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"Masters\",\n          \"PhD\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Target\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ord_2_0\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ord_2_1\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ord_2_2\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"dfb1"}
```

TARGET ENCODER

```
from category_encoders import TargetEncoder
te=TargetEncoder()
nd=te.fit_transform(df['City'],df['Target'])
df=pd.concat([df,nd],axis=1)
dfte1=df.copy()
dfte1
```

```
{"summary":"{\n  \"name\": \"dfte1\",\n  \"rows\": 10,\n  \"fields\":
[\n    {\n      \"column\": \"id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 0,\n
\"max\": 9,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          8,\n          1,\n          5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bin_1\",\n      \"properties\": {\
```

```
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"M\",\n          \"F\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bin_2\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Y\",\n          \"N\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"Bangalore\",\n          \"Chennai\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ord_1\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"Warm\",\n          \"Cold\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ord_2\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"Masters\",\n          \"PhD\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Target\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ord_2_0\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ord_2_1\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ord_2_2\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.05269245298737523,\n
\"min\": 0.4452723428580931,\n        \"max\": 0.5650542371814989,\n
\"num_unique_values\": 3,\n        \"samples\": [\n
0.4452723428580931,\n          0.5650542371814989\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"dfte1"}
```


FEATURE TRANSFORMATION

```
import pandas as pd
import numpy as np
from scipy import stats
df= pd.read_csv('Data_to_Transform.csv')
df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10000,\n  \"fields\":
[\n    {\n      \"column\": \"Moderate Positive Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.047236872731055,\n        \"min\": 0.899990324,\n        \"max\":
16.204517,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          6.430480755,\n          5.600117786,\n
4.100672877\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8826852876441158,\n        \"min\":
2.895073786,\n        \"max\": 18.05233051,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
6.20205811,\n          5.496119142,\n          4.31233591\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Moderate Negative Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.0450599257681863,\n        \"min\": -6.335678967,\n        \"max\":
11.18074757,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          5.548719657,\n          6.389984146,\n
7.888212858\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Negative Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8605559572675425,\n        \"min\": -
7.036091246,\n        \"max\": 9.027484718,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
5.766479977,\n          6.498381507,\n          7.671506117\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
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
```

```
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
```

```
np.reciprocal(df['Highly Positive Skew'])
```

```
0       0.345414
1       0.337566
2       0.337112
3       0.333297
4       0.331993
          ...
9995    0.061389
9996    0.060990
9997    0.058469
9998    0.056726
9999    0.055395
Name: Highly Positive Skew, Length: 10000, dtype: float64
```

```
np.sqrt(df['Highly Positive Skew'])
```

```
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
```

```
np.square(df['Highly Positive Skew'])
```

```
0         8.381452
1         8.775724
2         8.799396
3         9.001942
```

```
4         9.072800
           ...
9995    265.348230
9996    268.837091
9997    292.512290
9998    310.762852
9999    325.886637
Name: Highly Positive Skew, Length: 10000, dtype: float64
```

```
df['Highly Positive Skew_boxcox'], parameters =
stats.boxcox(df['Highly Positive Skew'])
df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10000,\n  \"fields\":
[\n    {\n      \"column\": \"Moderate Positive Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.047236872731055,\n        \"min\": 0.899990324,\n        \"max\":
16.204517,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          6.430480755,\n          5.600117786,\n
4.100672877\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8826852876441158,\n        \"min\":
2.895073786,\n        \"max\": 18.05233051,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
6.20205811,\n          5.496119142,\n          4.31233591\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Moderate Negative Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.0450599257681863,\n        \"min\": -6.335678967,\n        \"max\":
11.18074757,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          5.548719657,\n          6.389984146,\n
7.888212858\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Negative Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8605559572675425,\n        \"min\": -
7.036091246,\n        \"max\": 9.027484718,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
5.766479977,\n          6.498381507,\n          7.671506117\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew_boxcox\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.11375225792955061,\n
\"min\": 0.8129086905344282,\n        \"max\": 1.4805251942556963,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
1.1700212996909898,\n          1.1225352234509938,\n
1.0175357770331757\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

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


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10000,\n  \"fields\":
[\n    {\n      \"column\": \"Moderate Positive Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.047236872731055,\n        \"min\": 0.899990324,\n        \"max\":
16.204517,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          6.430480755,\n          5.600117786,\n
4.100672877\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8826852876441158,\n        \"min\":
2.895073786,\n        \"max\": 18.05233051,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
6.20205811,\n          5.496119142,\n          4.31233591\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Moderate Negative Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.0450599257681863,\n        \"min\": -6.335678967,\n        \"max\":
11.18074757,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          5.548719657,\n          6.389984146,\n
7.888212858\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Negative Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8605559572675425,\n        \"min\": -
7.036091246,\n        \"max\": 9.027484718,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
5.766479977,\n          6.498381507,\n          7.671506117\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew_boxcox\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.11375225792955061,\n
\"min\": 0.8129086905344282,\n        \"max\": 1.4805251942556963,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
1.1700212996909898,\n          1.1225352234509938,\n
1.0175357770331757\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Moderate Negative Skew_yeojohnson\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 5.354700436048172,\n
\"min\": -3.311401378269837,\n        \"max\": 29.137807350266772,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
10.893374802975943,\n          13.234147636390649,\n
17.757279118367567\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
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 Negative Skew_yeojohnson   -0.119651
dtype: float64
```

```
df['Highly Negative Skew'].parameters=stats.yeojohnson(df['Highly
Negative Skew'])
df.skew()
```

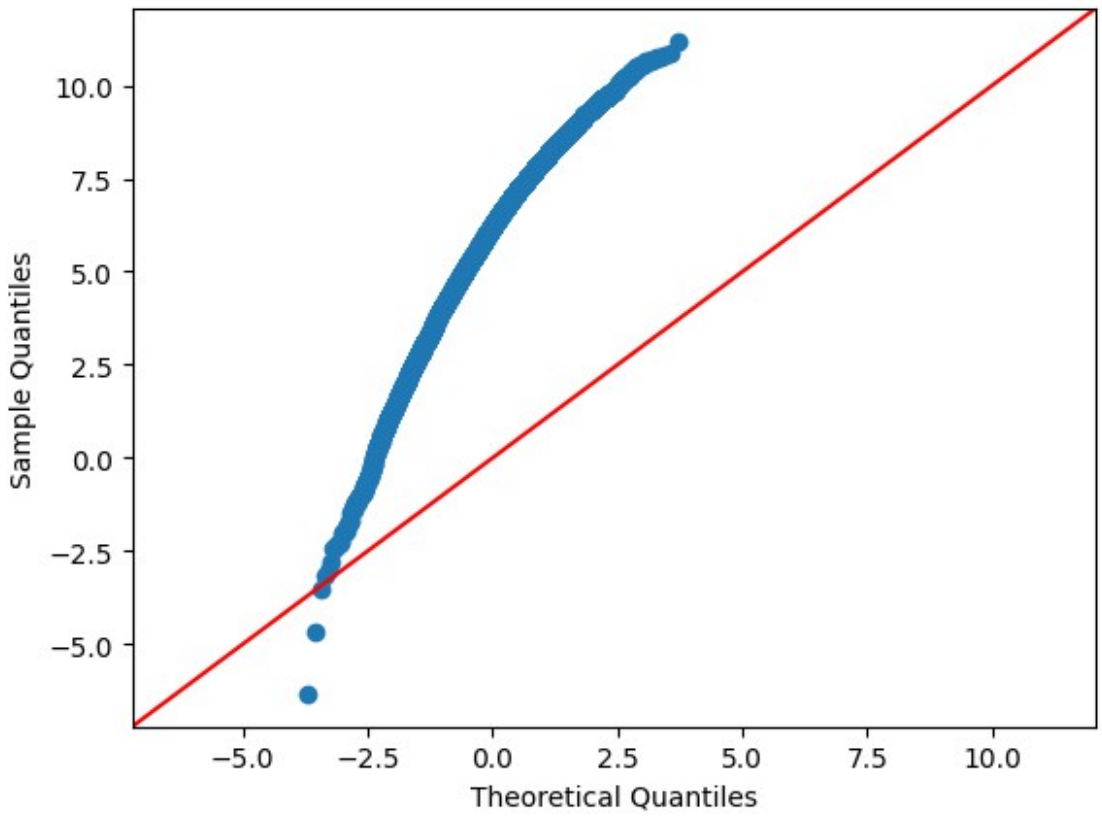
```
Moderate Positive Skew               0.656308
Highly Positive Skew                 1.271249
Moderate Negative Skew              -0.690244
Highly Negative Skew                -1.201891
Highly Positive Skew_boxcox          0.023089
Moderate Negative Skew_yeojohnson   -0.119651
dtype: float64
```

```
from sklearn.preprocessing import QuantileTransformer
qt=QuantileTransformer(output_distribution='normal')
df['Moderate Negative Skew_1']=qt.fit_transform(df[['Highly Negative
Skew']])
df
```

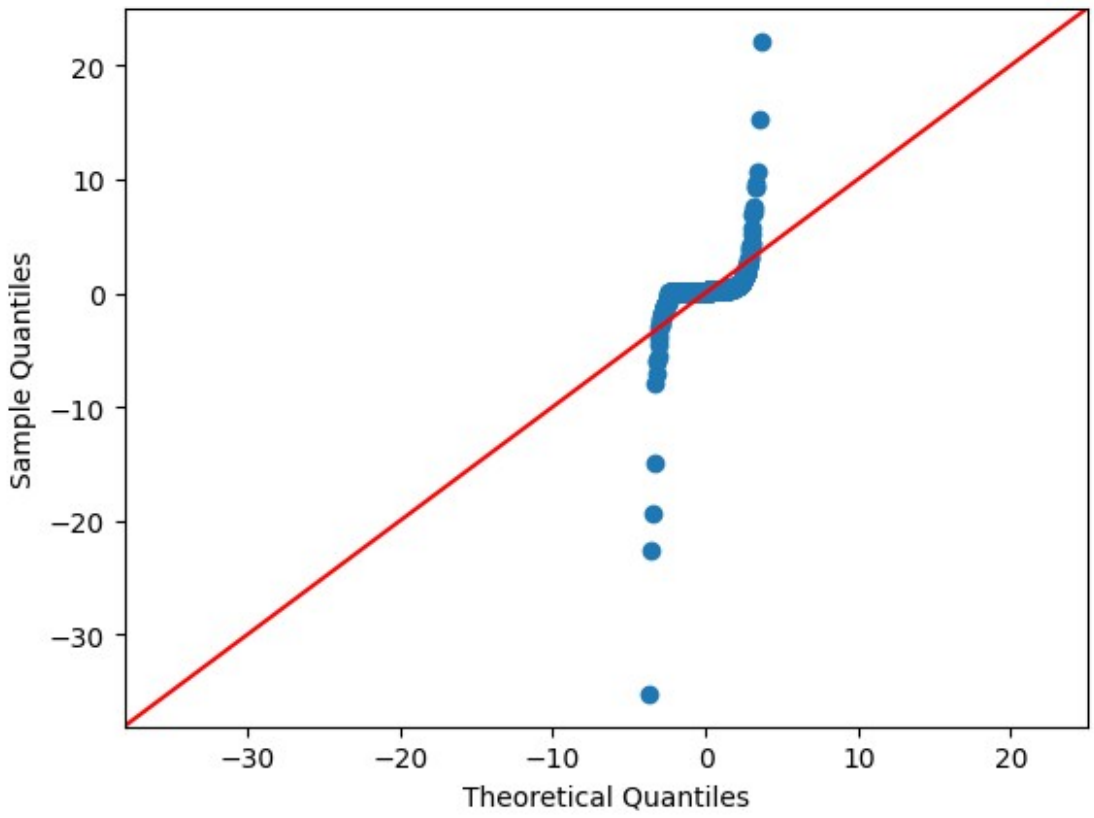

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10000,\n  \"fields\":
[\n    {\n      \"column\": \"Moderate Positive Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.047236872731055,\n        \"min\": 0.899990324,\n        \"max\":
16.204517,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          6.430480755,\n          5.600117786,\n
4.100672877\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8826852876441158,\n        \"min\":
2.895073786,\n        \"max\": 18.05233051,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
6.20205811,\n          5.496119142,\n          4.31233591\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Moderate Negative Skew\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.0450599257681863,\n        \"min\": -6.335678967,\n        \"max\":
11.18074757,\n        \"num_unique_values\": 10000,\n
\"samples\": [\n          5.548719657,\n          6.389984146,\n
7.888212858\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Negative Skew\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.8605559572675425,\n        \"min\": -
7.036091246,\n        \"max\": 9.027484718,\n
```

```
\"num_unique_values\": 10000,\n        \"samples\": [\n
5.766479977,\n          6.498381507,\n          7.671506117\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Highly Positive Skew_boxcox\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.11375225792955061,\n
\"min\": 0.8129086905344282,\n        \"max\": 1.4805251942556963,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
1.1700212996909898,\n          1.1225352234509938,\n
1.0175357770331757\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Moderate Negative Skew_yeojohnson\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 5.354700436048172,\n
\"min\": -3.311401378269837,\n        \"max\": 29.137807350266772,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n
10.893374802975943,\n          13.234147636390649,\n
17.757279118367567\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Moderate Negative Skew_1\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.000977971440535,\n
\"min\": -5.199337582605575,\n        \"max\": 5.19933758270342,\n
\"num_unique_values\": 10000,\n        \"samples\": [\n          -
0.3188745821384206,\n          0.07928214460099213,\n
0.9419213442771701\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

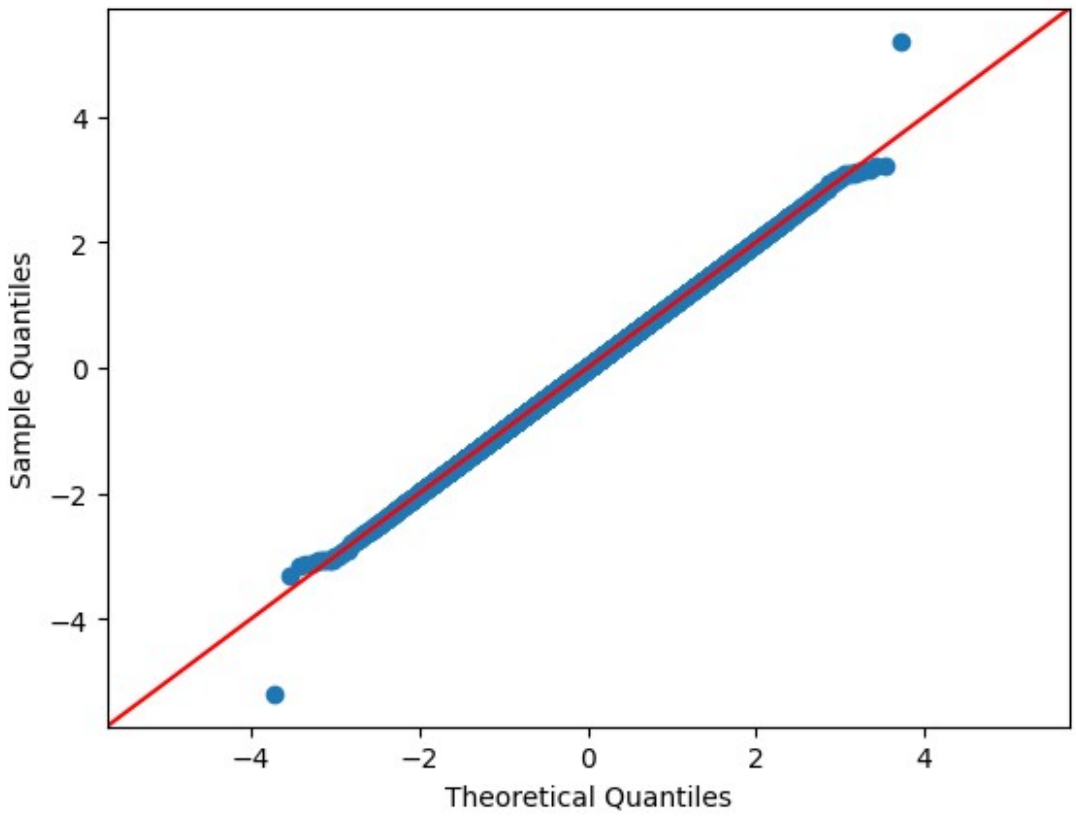

```
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
sm.qqplot(df['Moderate Negative Skew'],line='45')
plt.show()
```



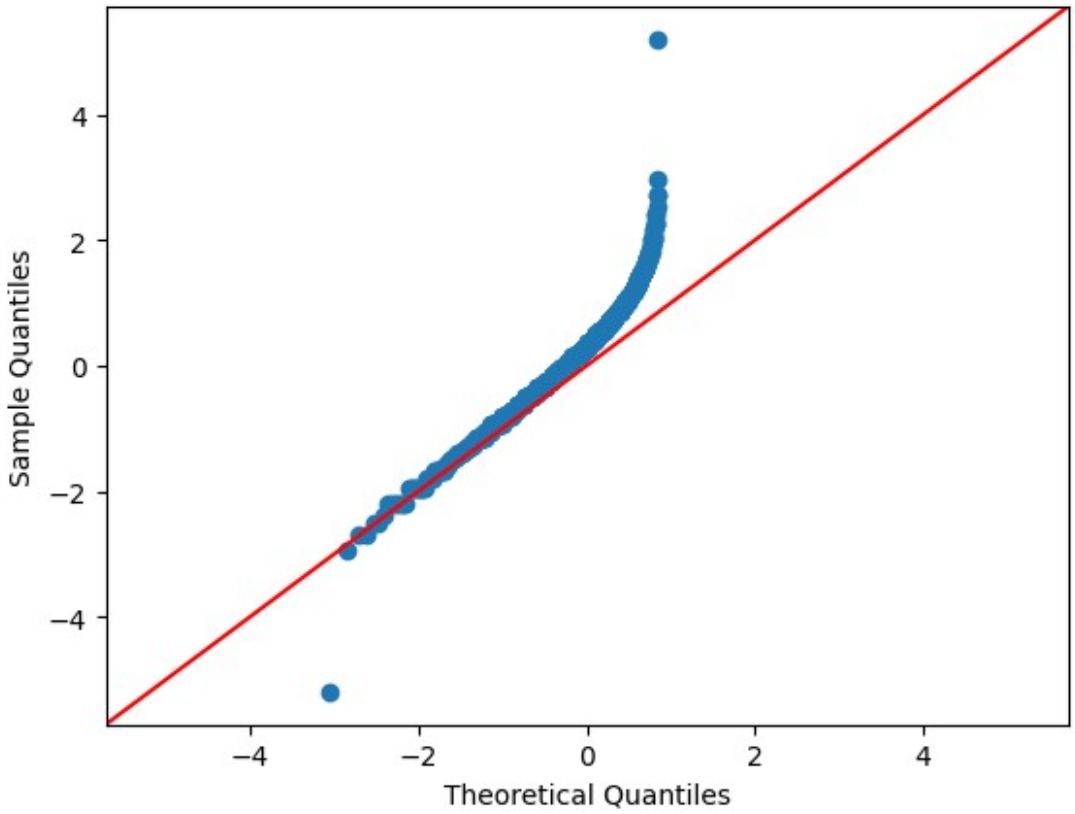

```
sm.qqplot(np.reciprocal(df['Moderate Negative Skew']),line='45')
plt.show()
```




```
from sklearn.preprocessing import QuantileTransformer
qt=QuantileTransformer(output_distribution='normal',n_quantiles=891)
df['Moderate Negative Skew_2']=qt.fit_transform(df[['Moderate Negative
Skew']])
sm.qqplot(df['Moderate Negative Skew_2'],line='45')
plt.show()
```




```
dt= pd.read_csv('titanic_dataset.csv')
from sklearn.preprocessing import QuantileTransformer
qt=QuantileTransformer(output_distribution='normal',n_quantiles=891)
dt['Age_1']=qt.fit_transform(dt[['Age']])
sm.qqplot(dt['Age_1'],line='45')
plt.show()
```


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