In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

file_path=r"C:\Users\omkar\OneDrive\Documents\Data science\Naresh IT\Naresh IT\D
visa_df=pd.read_csv(file_path)
visa_df

cat_cols=visa_df.select_dtypes(include='object').columns
num_cols=visa_df.select_dtypes(exclude='object').columns
num_cols,cat_cols
```

Out[1]:
```
(Index(['no_of_employees', 'yr_of_estab', 'prevailing_wage'], dtype='object'),
 Index(['case_id', 'continent', 'education_of_employee', 'has_job_experience',
        'requires_job_training', 'region_of_employment', 'unit_of_wage',
        'full_time_position', 'case_status'],
       dtype='object'))
```

- Generally data has 3 types
  - Postive skew
  - Negtaive skew
  - No skew
- Skew ness happend becuase of Outliers
- eventhough we treat the outliers still we can see some skew
- And we know that all the math developed by make an assumption as **Data follows Normal distribution**
- so transformation methods used to convert data to Normal
- The important Transformations are
  - Log Transformation
  - Reciprocal Transformation
  - Sqrt Transformation
  - Exponential Transformation
  - Power Transformation

$Step - 1$

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

$Step - 2$: **read exponential data**

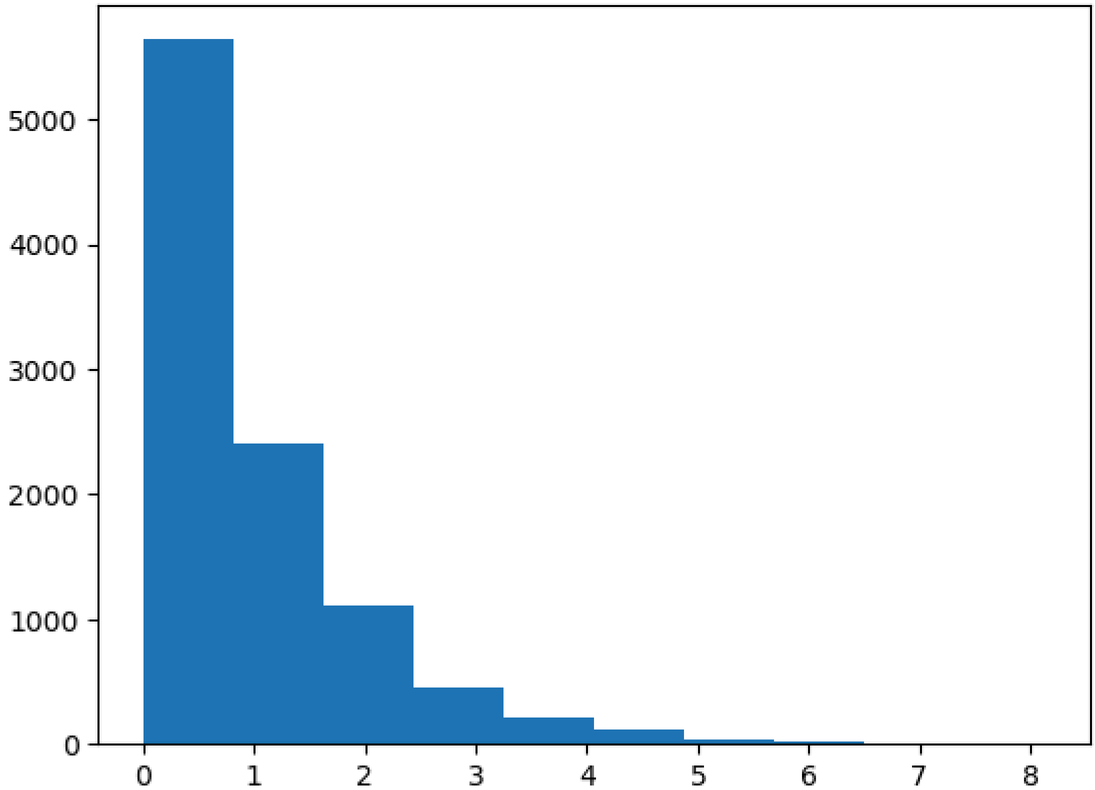
In [3]:
```
exp_data=np.random.exponential(size=10000)
exp_data
```

Out[3]:
```
array([0.46143947, 0.19155787, 0.61807367, ..., 0.08892538, 1.21819999,
       1.98742312])
```

$Step - 3$: **plot histogram**

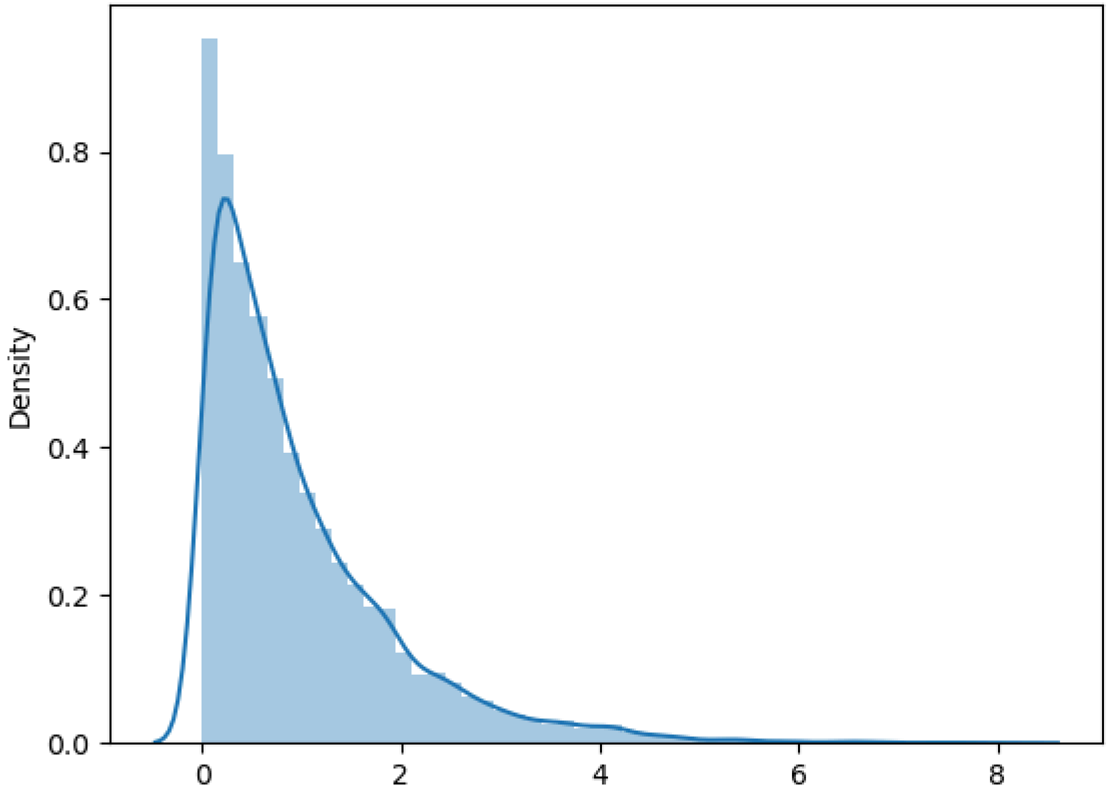
In [5]:
```
plt.hist(exp_data)
```

Out[5]:
```
(array([5.641e+03, 2.411e+03, 1.103e+03, 4.590e+02, 2.140e+02, 1.090e+02,
        3.500e+01, 1.700e+01, 7.000e+00, 4.000e+00]),
 array([7.58752117e-05, 8.13528500e-01, 1.62698112e+00, 2.44043375e+00,
        3.25388637e+00, 4.06733900e+00, 4.88079162e+00, 5.69424425e+00,
        6.50769687e+00, 7.32114950e+00, 8.13460212e+00]),
 <BarContainer object of 10 artists>)
```



In [7]:
```
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
sns.distplot(exp_data)
```

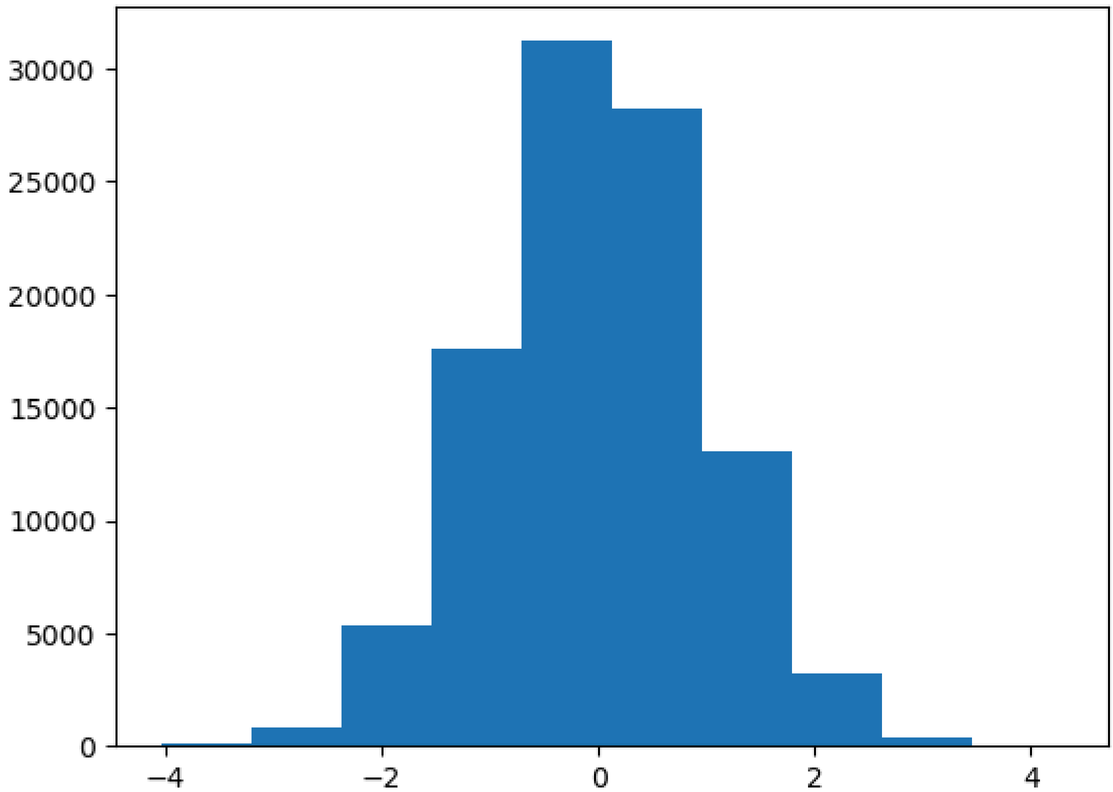
Out[7]:
```
<Axes: ylabel='Density'>
```



In [9]:
```
normal_data=np.random.normal(size=100000)
normal_data
```

Out[9]:
```
array([-0.17029366,  1.10486868, -0.43872987, ...,  0.50114078,
        0.07565132, -0.81979415])
```

In [11]:
```
plt.hist(normal_data)
plt.show()
```



Goal:

- Convert Exponential data to Normal data

Log Transformation

- Log transformation means performing logarithm operations on original data
- It is one of the approach to convert data to Normality
- Log means natural logarithm base=e

In [13]:
```
x=2
np.log(x)
```

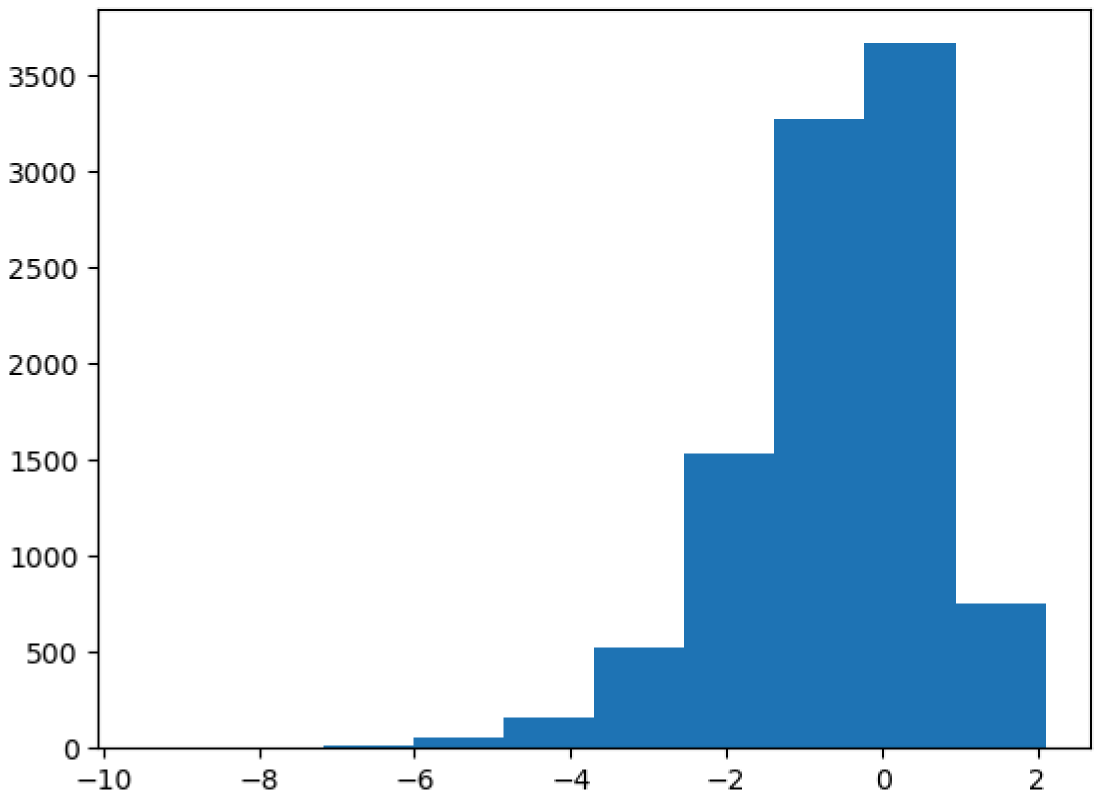
Out[13]:
```
0.6931471805599453
```

In [15]:
```
log_data=np.log(exp_data)
log_data
```

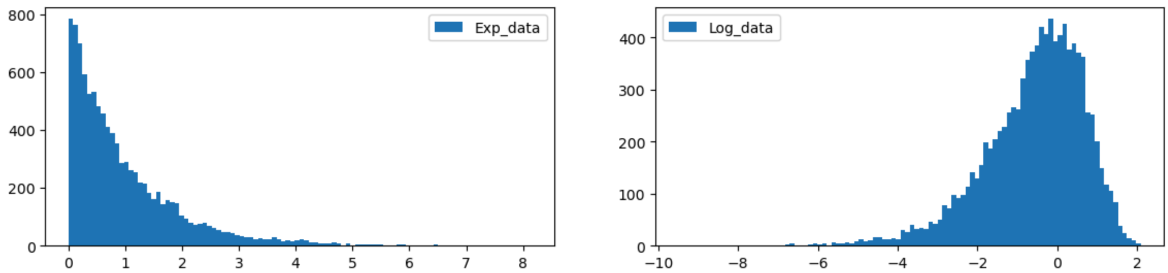
Out[15]:
```
array([-0.77340439, -1.65256531, -0.48114762, ..., -2.41995774,
        0.19737435,  0.68683889])
```

In [ ]:
```
# we are hoping log data might follows normality
# it might be possible or might not be possible
# we need to plot histogram again on log data
```

In [17]:
```
plt.hist(log_data)
plt.show()
```



In [19]:
```python
plt.figure(figsize=(14,3))
plt.subplot(1,2,1).hist(exp_data,bins=100,label='Exp_data')
plt.legend()
plt.subplot(1,2,2).hist(log_data,bins=100,label='Log_data')
plt.legend()
plt.show()
```

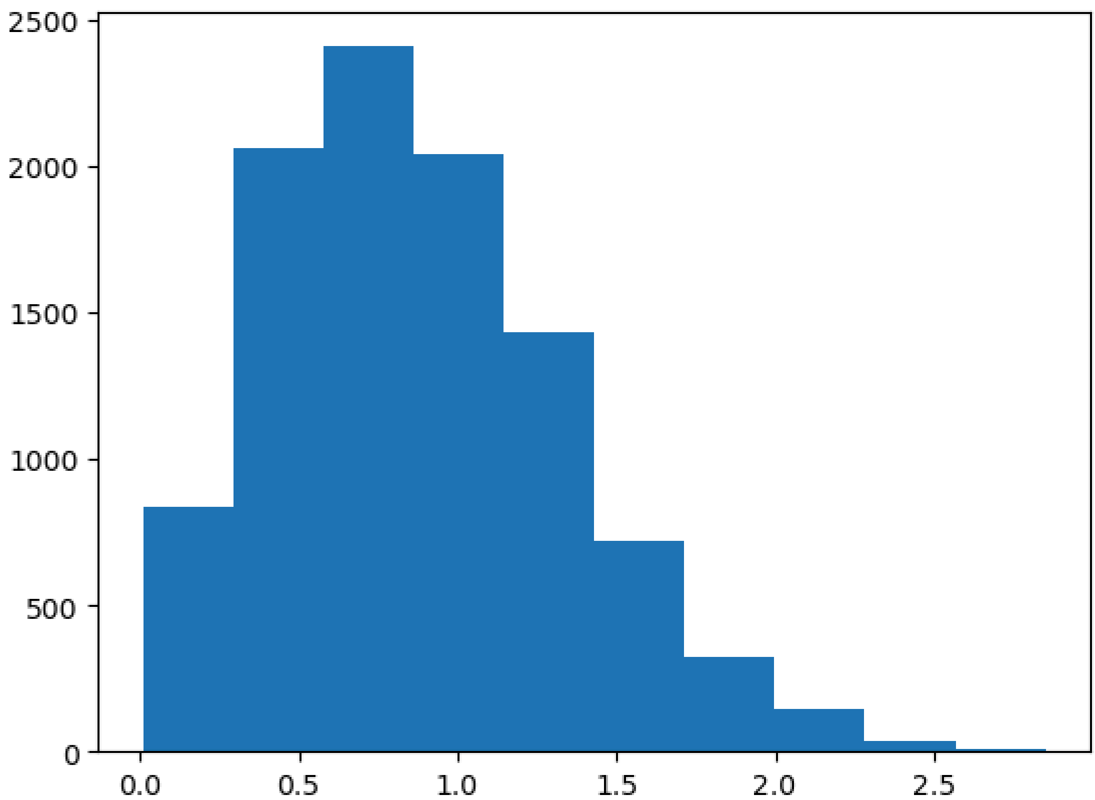


Square root Transformation: np.sqrt

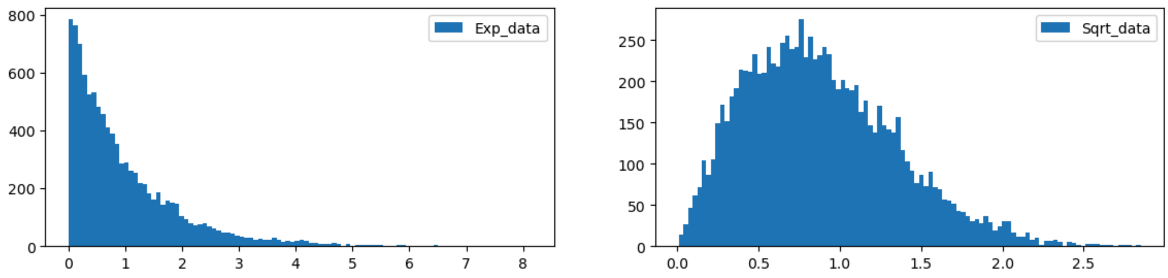
In [21]:
```python
sqrt_data=np.sqrt(exp_data)
sqrt_data
```

Out[21]:
```
array([0.67929336, 0.43767325, 0.78617662, ..., 0.29820358, 1.10372097,
       1.40975995])
```

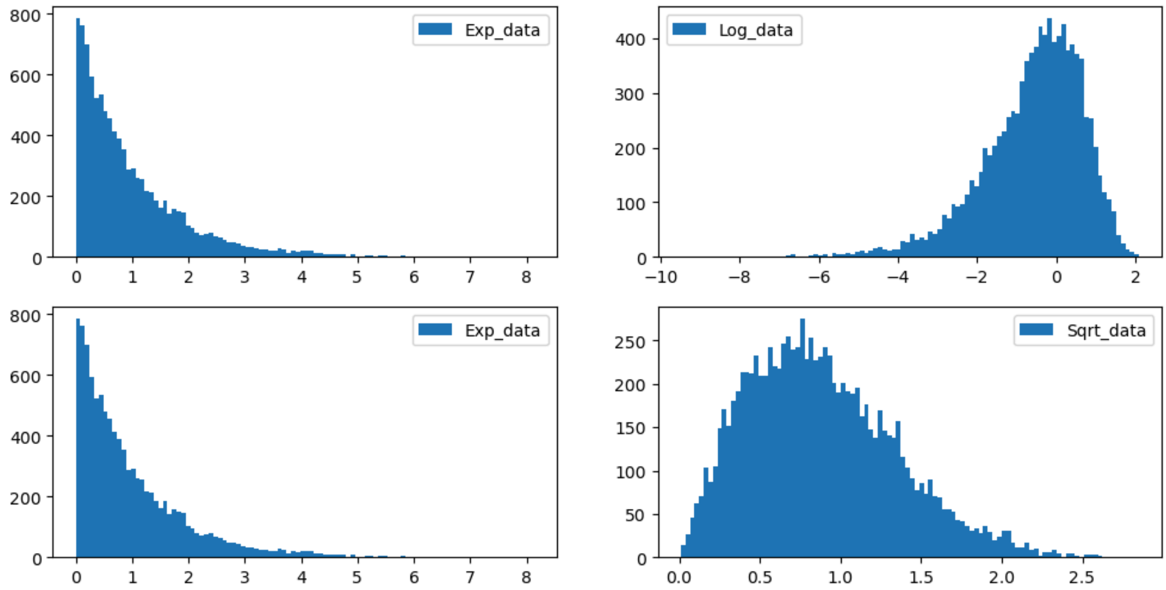
In [23]:
```python
plt.hist(sqrt_data)
plt.show()
```



In [25]:
```python
plt.figure(figsize=(14,3))
plt.subplot(1,2,1).hist(exp_data,bins=100,label='Exp_data')
plt.legend()
plt.subplot(1,2,2).hist(sqrt_data,bins=100,label='Sqrt_data')
plt.legend()
plt.show()
```



In [27]:
```
plt.figure(figsize=(12,6))
plt.subplot(2,2,1).hist(exp_data,bins=100,label='Exp_data')
plt.legend()
plt.subplot(2,2,2).hist(log_data,bins=100,label='Log_data')
plt.legend()
plt.subplot(2,2,3).hist(exp_data,bins=100,label='Exp_data')
plt.legend()
plt.subplot(2,2,4).hist(sqrt_data,bins=100,label='Sqrt_data')
plt.legend()
plt.show()
```



Power Transformation

- Power Transformation is used to Reduce the skewnes , so that distribution become symmetric
- Under these two types are there
  - Box-cox Transformation
  - It applies to only postive data
  - It has both Log transformation and square root transformation

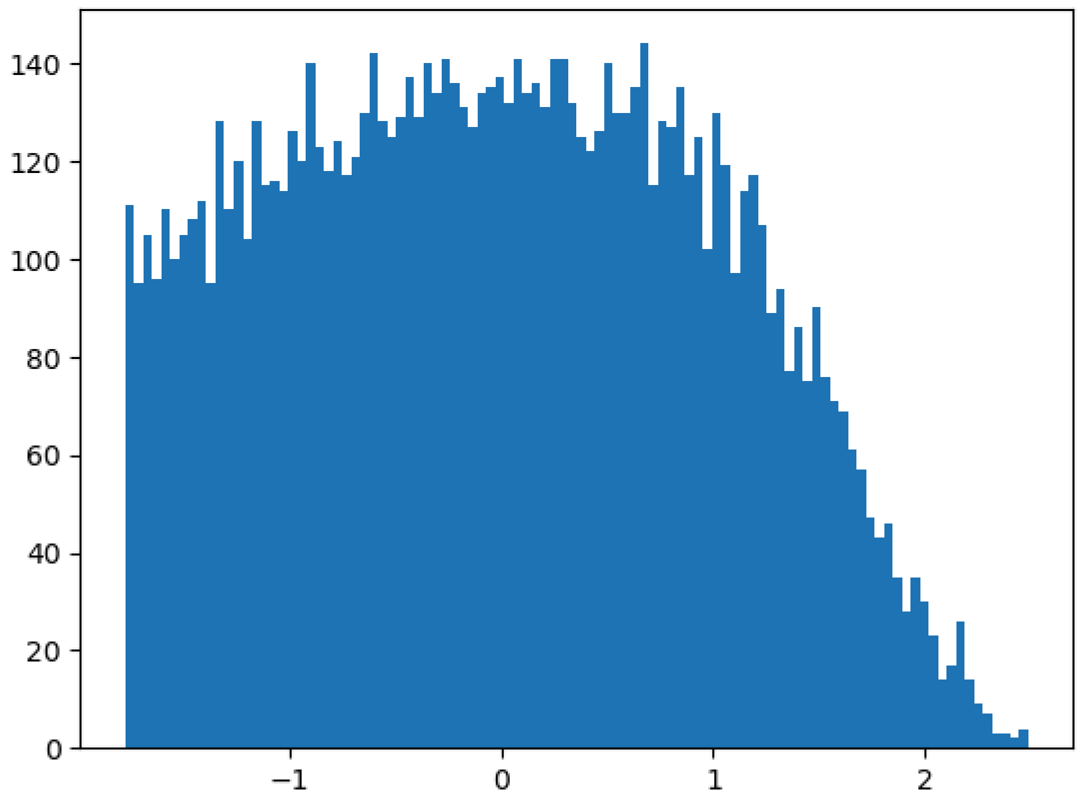
$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

- lambda =1 : No Transformation
- lambda = 0 : Log Transformation
- lambda =0.5 : Square root transformation

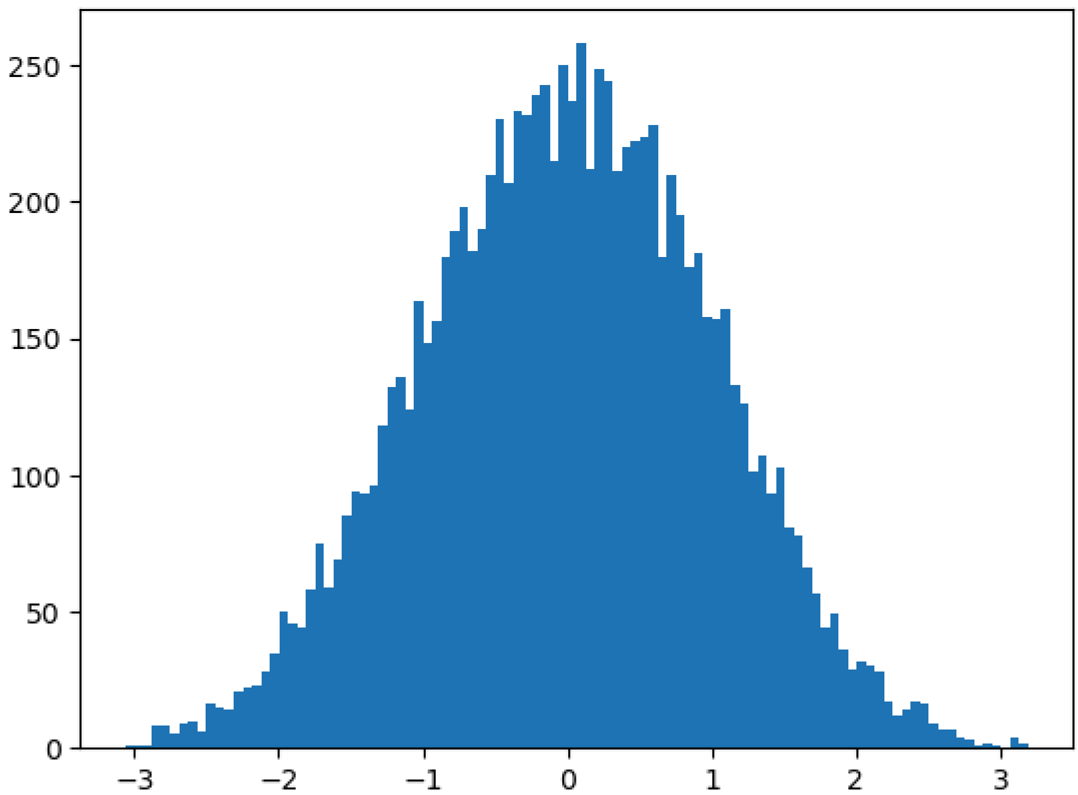
In [ ]:
```
**Your job is know about another method**

Yeo-Johnson ===== formaule tricky confused tomorrow i will ask you
```

- Power transformations are under sklearn package
- sklearn
  - preprocessing
    - PowerTransformation

In [29]:
```
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer(method='yeo-johnson')
exp_data=exp_data.reshape(-1,1)
trans_exp=pt.fit_transform(exp_data)
plt.hist(trans_exp,bins=100)
plt.show()
```

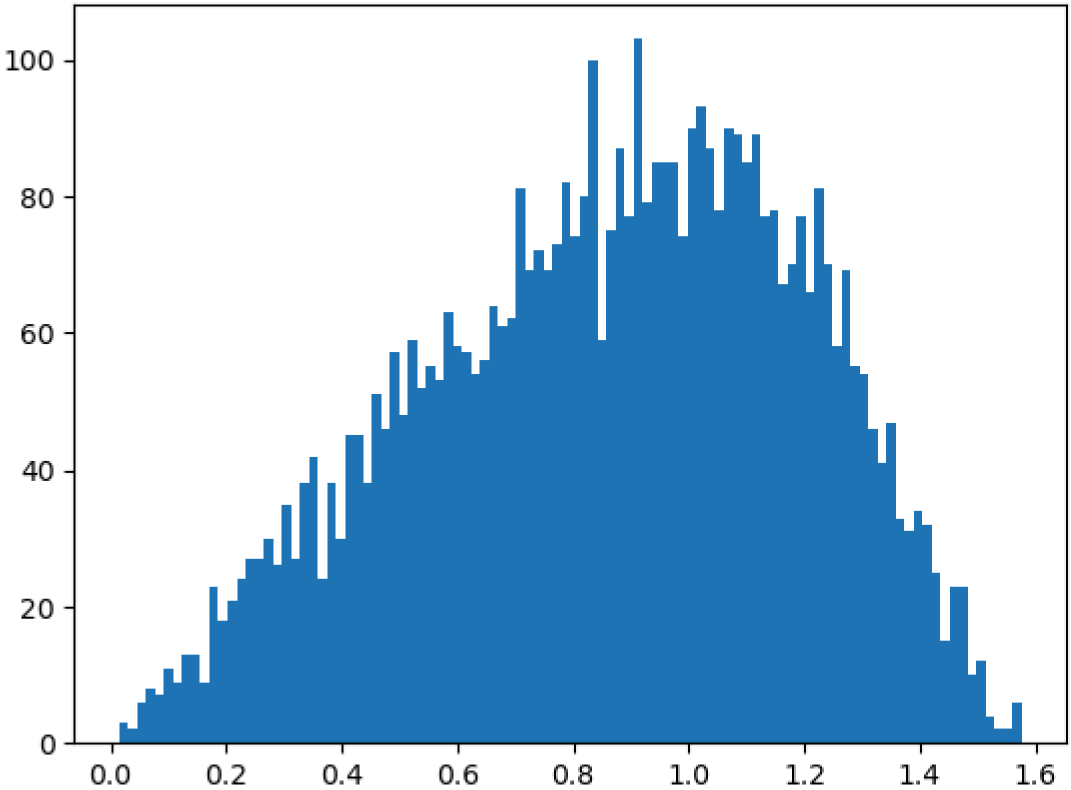


In [30]:
```
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer(method='box-cox')
exp_data=exp_data.reshape(-1,1)
trans_exp=pt.fit_transform(exp_data)
plt.hist(trans_exp,bins=100)
plt.show()
```



In [32]:
```
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer(method='yeo-johnson')
```

```
exp_data=exp_data.reshape(-1,1)
trans_exp=pt.fit_transform(exp_data)
trans_sqrt=np.sqrt(trans_exp)
plt.hist(trans_sqrt,bins=100)
plt.show()
```



In [ ]: