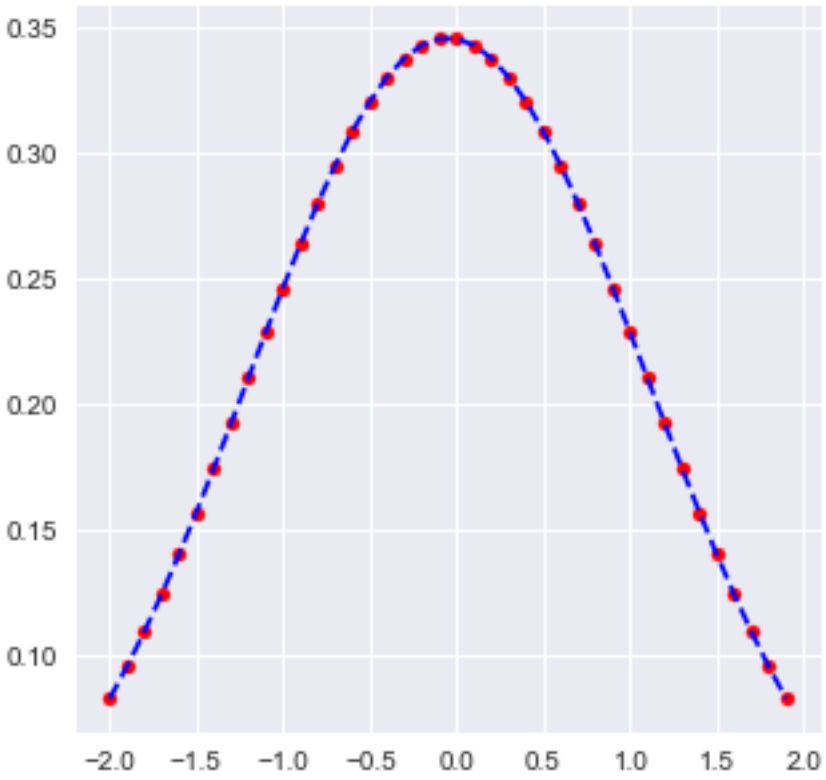
01_normality

April 1, 2022

[ ]:
```
#import libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[ ]:
```
#normal distribution  #How to draw
def pdf(x):
    mean = np.mean(x)
    std = np.std(x)
    y_out = 1/(std * np.sqrt(2 *np.pi)) *np.exp( -(x -mean)**2 / (2 * std **2))
    return y_out

# to generate an array of x
x = np.arange(-2, 2, 0.1)
y = pdf(x)

#plotting the normal curve / or gaussian ditribution

plt.style.use('seaborn')
plt.figure(figsize=(5,5))

plt.plot(x, y, color = 'blue', linestyle = 'dashed')

plt.scatter(x, y, marker = 'o', s=25, color='red')
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x215ab7732b0>
```



1 Normal Distribution and its tests

1. import dataset
2. subsetting a dataset
3. visual test for normal distribution 1.Histogram 2.qqnorm
4. statistics
    1.

[ ]:
```
# import  a dataset

kashti = sns.load_dataset('titanic')
kashti.head()
```

[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

	who	adult_male	deck	embark_town	alive	alone

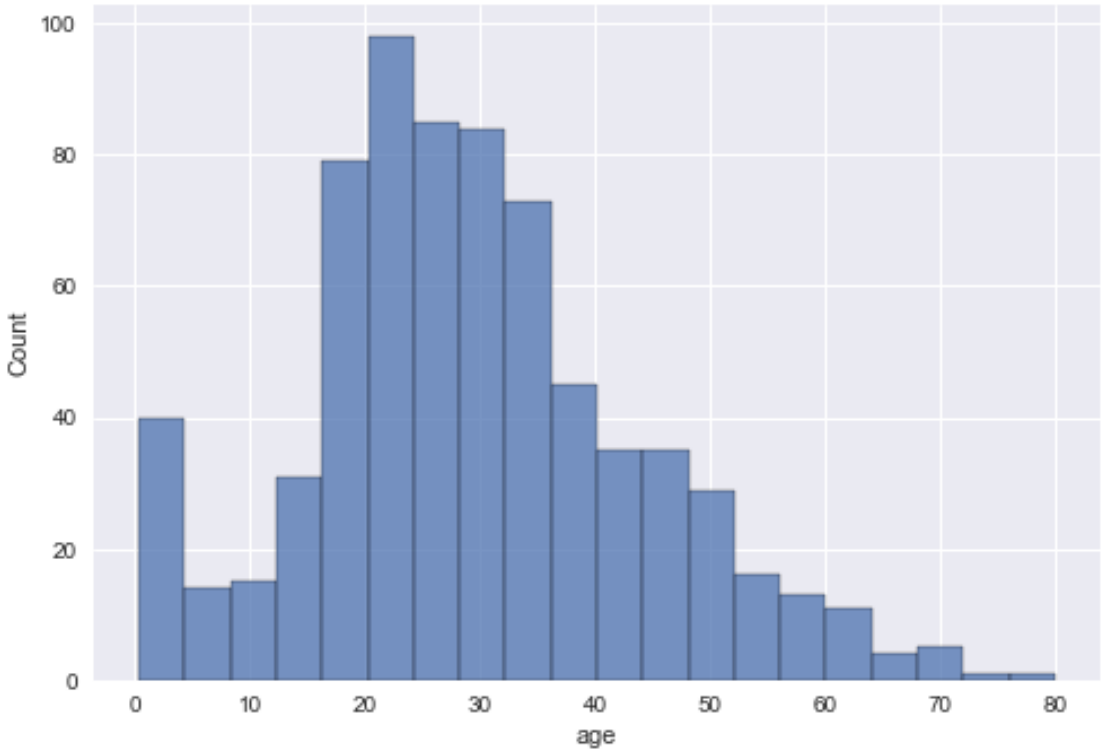
```
    0    man        True  NaN  Southampton    no  False
    1  woman       False    C    Cherbourg   yes  False
    2  woman       False  NaN  Southampton   yes   True
    3  woman       False    C  Southampton   yes  False
    4    man        True  NaN  Southampton    no   True
```

[ ]:
```
kashti = kashti[['sex', 'age', 'fare']]
kashti.head()
```

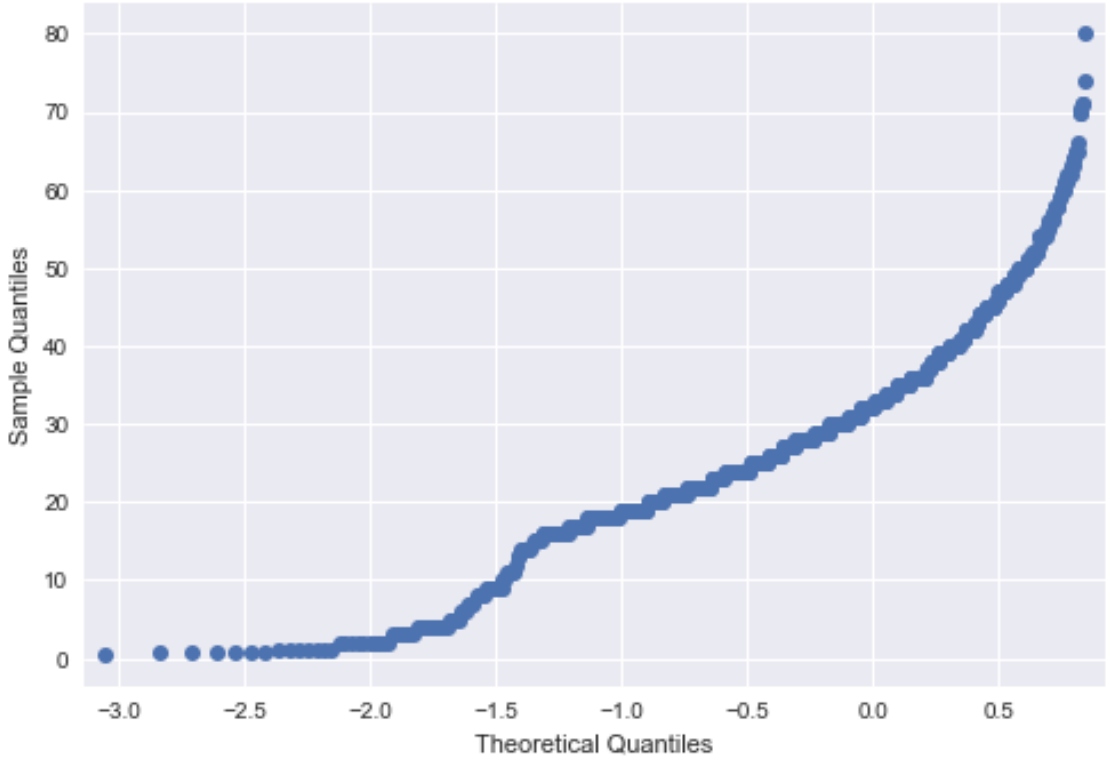
[ ]:
```
      sex   age     fare
0    male  22.0   7.2500
1  female  38.0  71.2833
2  female  26.0   7.9250
3  female  35.0  53.1000
4    male  35.0   8.0500
```

[ ]:
```
# Histogram test
sns.histplot(kashti['age'])
```

[ ]:
```
<AxesSubplot:xlabel='age', ylabel='Count'>
```



[ ]:
```
# qqplot
from statsmodels.graphics.gofplots import qqplot

#  q-q norm plot
qqplot(kashti['age'])
plt.show()
```



2 1. Normality Tests

There are many statistical tests that we can use to quantify whether asample of data looks as though it was drawn from a Guassian distribution . Each test makes different assumptions and considers different aspects of the data. We will look 3 commanly used tests in this section that you can apply to your own data samples..

```
1. Shapiro-Wilk Test
2. D'Agostino's K^2 Test
3. Anderson-Darling Test
p <= alpha: reject H0, not normal.
p >= alpha: fail to reject H0, normal.
```

3 1. Shapiro-WIlk Test

The Shapiro-Wilk test evaluates a data sample and quantifies how likely it is that the data was drawn from a Guassian distribution, named for Samuel Shapiro and Martin Wilk.

In, practice the Shapiro-Wilk test is believed to be a reliable test of nirmality, although there us some suggestion that the test may be suitable for smaller samples of data, e.g. thousands of observations or fewer.

The shapiro{} Scipy function will calculate the Shapiro-Wilk on a given dataset. The function returns bith the W-statistic calculated by the test and the p-value.

Assumptions
Observation in each sample are independent and identically distributed.

- H0: the sample has a Guasian distribution
- H1: the sample does not have a Guassian distribution Python code is here

[ ]:
```
# shapirowilk test
# import library

from scipy.stats import shapiro

stat, p = shapiro(kashti['age'])
print('stat=%.3f, p=%.3f' % (stat, p))

# make a conditional arguement for further for use
if p > 0.05:
    print('Probably Gaussian or Normal Distribution')
else:
    print('Probably not Gaussian nor normal Distribution')
```

```
stat=nan, p=1.000
Probably Gaussian or Normal Distribution
```

4 2. D'Agostino's K^2 Test

The D'Agostino's K^2 test calculates summary statisitcs frm the data, namely kurtosis, and skewness, to determine if the data distribution departs from the normal distribution, named for Ralph D'Agostino.

- **Skew** is a quantificstion of how much a distirbution is pushed left or right, a measure of asymmetry in the distribution.
- **Kutosis** quantifies how much of the distribution is in the tail. It is a simple and commanly used statistical test for p-vlaue.

Assumptions * Observation in each sampe are independent and identically distributed.

Interpretation

- H0: the sample has a Guassian distribution.

- H1: the sample does not have a Guassian distribution

[ ]:
```
# D' Agostino's K^2 test #ye kb lagna he #assignment
# import library

from scipy.stats import normaltest

stat, p = normaltest(kashti['age'])
print('stat=%.3f, p=%.3f' % (stat, p))

# make a conditional arguement for further for use
if p > 0.05:
    print('Probably Gaussian or Normal Distribution')
else:
    print('Probably not Gaussian nor normal Distribution')
```

```
stat=nan, p=nan
Probably not Gaussian nor normal Distribution
```

3. Anderson-Darling Test

A statistical test that can be used to evaluate whether a data sample comes from one of among many known data samples, named for Theodore Anderson and Donald Darling.

It can be used to check whether a data sample is normal. The test is a modified version of a more sophisticated nonparametric goodness-of-fit statistical test called the Kolmogorov-Smirnov test.

A feature of the Anderson-Darling test is that it returns a list of critical values rather than a single p-value. This can provide the basis for a more thorough interpretation of the result.

The anderson() SciPy function implements the Anderson-Darling test. It takes as parameters the data sample and the name of the distribution to test it against. By default, the test will check against the Gaussian distribution (dist='norm').

Assumptions

- Observations in each sample are independent and identically distributed. Interpretation
- H0: the sample has a Gaussian distribution.
- H1: the sample does not have a Gaussian distribution. Python code is here:

[ ]:
```
from scipy.stats import anderson

#select a column to check the normal distribtion
result = anderson(kashti["age"])
print('stat=%.3f' % (result.statistic))
for i in range(len(result.critical_values)):
    sl, cv = result.significance_level[i], result.critical_values[i]
    if result.statistic < cv:
        print('Probably Guassian/Normal Distribution at the %.1f level' % (sl))
    else:
```

```
        print('Probably not Guassian/Normal Distribution at the %.1f %% level'␣
↪% (sl))
```

```
stat=nan
Probably not Guassian/Normal Distribution at the 15.0 % level
Probably not Guassian/Normal Distribution at the 10.0 % level
Probably not Guassian/Normal Distribution at the 5.0 % level
Probably not Guassian/Normal Distribution at the 2.5 % level
Probably not Guassian/Normal Distribution at the 1.0 % level
```

5 kashti k ilawa ksi bhi data pr ye test laga kr send kren