HYPOTHESIS TESTING

- Hypothesis testing is a statistical method used to evaluate whether an observed data sample is consistent with a hypothesis or a null hypothesis. The null hypothesis is typically a statement that assumes there is no difference or relationship between two or more variables, while the alternative hypothesis is the opposite, suggesting there is a difference or relationship.
- To perform hypothesis testing, a statistical test is conducted using the data to calculate a test statistic, such as a t-score or p-value, which measures the probability of observing the data if the null hypothesis is true. If the p-value is less than a predetermined significance level, typically 0.05, then the null hypothesis is rejected, and the alternative hypothesis is supported.

- NOTE:
  - significance value(alpha) is either 0.05/0.01
  - pvalue > significance value --> accept the null hypothesis else reject

In [3]:

```
import numpy as np
import scipy.stats
from scipy.stats import t
```

Confidence Interval

-A confidence interval is a range of values that is likely to contain the true value of a population parameter with a certain degree of confidence.

The formula for calculating a confidence interval for a population mean is:

CI = X ± Z * (s / sqrt(n))

where:

-CI is the confidence interval

-X is the sample mean

-Z is the z-score for the desired level of confidence (e.g., 1.96 for a 95% confidence interval)

-s is the sample standard deviation

-n is the sample size

In [ ]:

```
#taking an example array:
arr = np.array([78 ,65 ,20 ,69 ,36 ,81 ,85 ,71 ,44 ,71 ,47 ,37 ,89 ,25 ,73 ,55 ,80 ,52 ,46,
```

In [ ]:

```
#using formula:
X = np.mean(arr)
Z = 1.96
s = np.std(arr)
n = len(arr)

CI1 = X - Z*(s/np.sqrt(n))
CI2 = X + Z*(s/np.sqrt(n))
print(CI1,CI2)
```

```
51.24255838509275 68.85744161490724
```

In [ ]:

```
#using scipy library:
sc = scipy.stats.sem(arr) #calculating standard error
# getting 95% confidence interval
t_val = t.interval(confidence=0.95,df=(n-1),loc=X,scale=sc)
t_val
```

Out[10]:

```
(50.400470937660145, 69.69952906233985)
```

In [ ]:

```

```

Shapiro Wilk Test

- to check if sample has normal distribution we use shapiro test , if pval>0.05 the data can be normaly distributed, else not.

In [ ]:

```
#using built in module:
from scipy.stats import shapiro
shapiro(arr)
```

Out[11]:

```
ShapiroResult(statistic=0.9377500414848328, pvalue=0.21734236180782318)
```

Here pvalue is more than 0.05, thus our array can be normally distributed.

In [ ]:

```

```

##Z test (when we have pop std available):

- A z-test is a statistical test that compares a sample mean to a known population mean when the population standard deviation is known. It is used to determine whether the difference between the sample mean and the population mean is statistically significant.
- The formula for calculating the z-score is:

z = (X - μ) / (σ / sqrt(n))

where:

X is the sample mean

μ is the population mean

σ is the population standard deviation

n is the sample size

-If the absolute value of the z-score is greater than the critical value for a given level of significance (e.g., 1.96 for a 95% confidence interval), then we can reject the null hypothesis and conclude that the sample mean is statistically significantly different from the population mean.

In [ ]:

```
#importing ztest:
from statsmodels.stats.weightstats import ztest
```

In [ ]:

```
#creating a population array
pop = np.random.uniform(0,50,size=(50))
pop
```

Out[24]:

```
array([32.74604246, 33.56518868, 43.50795015, 45.50614668, 38.8306301 ,
       22.20365074, 47.50829914, 23.27547057, 37.47812872, 41.60528632,
       44.85228347, 16.49288566, 40.3834803 , 30.52390039, 44.81210912,
       33.67518587, 12.79291627,  6.23261258, 49.39945951, 32.95685744,
        0.65689254, 49.10668004, 37.58585442,  7.73867263, 36.82612653,
       33.86925638,  9.45873698, 28.94881697, 15.21290017, 24.38346338,
       15.90480788, 44.48180982,  2.44119756, 26.67762388,  7.78469611,
       32.67353678, 26.50769285,  4.56499116,  3.38449021, 38.49501697,
       29.11635786, 37.25719946, 44.90976652, 32.15207807, 35.98638424,
       28.82627884,  7.77957377, 31.84726721, 14.87935178, 12.12519745])
```

In [ ]:

```
#randomly selecting sample from pop array:
Sample = np.random.choice(pop,size=30)
Sample
```

Out[25]:

```
array([43.50795015, 44.90976652, 35.98638424, 23.27547057,  7.77957377,
       24.38346338, 29.11635786, 22.20365074, 38.49501697, 31.84726721,
        7.77957377, 32.67353678,  6.23261258,  9.45873698,  7.77957377,
       30.52390039, 44.81210912, 49.10668004,  4.56499116, 44.48180982,
       29.11635786, 40.3834803 ,  7.73867263, 45.50614668, 15.90480788,
       28.94881697, 37.25719946, 43.50795015, 33.56518868,  6.23261258])
```

In [ ]:

```
#using the formula:
X = np.mean(Sample)
mu = np.mean(pop)
s = np.std(pop)
n = len(Sample)

#formula:
z = (X - mu) / (s / np.sqrt(n))
z
```

Out[28]:

```
-0.16701890401142058
```

In [ ]:

```
#getting z score and pvalue using scipy ztest
zsc,pval = ztest(Sample,value=mu)
print(zsc,pval)
```

```
-0.16015770562560863 0.872756845272247
```

In [ ]:

```

```

1 sample T test

- A one-sample t-test is used to determine whether a sample mean is significantly different from a hypothesized population mean when the population standard deviation is not known. The t-test is a statistical test that compares the mean of a sample to a known value or hypothesized population mean.
- Formula:

  t = (x̄ - μ) / (s / sqrt(n))

Where:

x̄ is the sample mean

μ is the hypothesized population mean

s is the sample standard deviation

n is the sample size

In [ ]:

```
#creating a population array
pop = np.random.uniform(0,50,size=(50))
pop
```

Out[31]:

```
array([ 5.18036289,  1.11277454,  0.96467437, 29.34713621, 47.74696878,
       25.89987369, 12.47829944, 35.56420211, 44.82387921, 21.59886285,
       46.51837308, 49.82584427, 46.37285163, 10.36310353, 40.54238469,
       48.42552787,  4.98772717, 42.75864617, 33.09138446, 45.65280508,
       38.5191423 , 35.94388045,  1.65996516,  8.5997978 , 30.38039978,
       20.33068123, 33.84763756,  8.52579212, 11.64226606, 31.36255499,
       28.49794178, 20.39327941, 39.56176362, 27.49154006, 45.18582971,
        2.23302955, 21.999271  , 20.63344138, 15.10922416, 23.97528398,
       47.07243369, 29.92872407, 22.85556021,  5.38556096, 27.73661023,
       39.99725439, 19.82553468, 30.18148773,  3.0918093 ,  4.91344648])
```

In [ ]:

```
#randomly selecting sample from pop array:
Sample = np.random.choice(pop,size=30)
Sample
```

Out[32]:

```
array([19.82553468,  3.0918093 , 31.36255499, 20.63344138, 39.56176362,
       39.56176362, 20.63344138, 44.82387921, 33.84763756, 48.42552787,
       27.73661023, 39.56176362, 44.82387921, 12.47829944, 39.99725439,
       25.89987369, 38.5191423 , 40.54238469, 27.49154006, 45.65280508,
       29.34713621, 27.73661023, 10.36310353, 45.18582971, 42.75864617,
       44.82387921, 29.34713621, 49.82584427,  2.23302955,  4.98772717])
```

In [ ]:

```
x = np.mean(Sample)
mu = np.mean(pop)
s = np.std(Sample)
n = len(Sample)
```

In [ ]:

```
#using formula:
t = (x - mu) / (s / np.sqrt(n))
t
```

Out[38]:

```
2.0796521700566624
```

In [ ]:

```
#importing ttest_1samp
from scipy.stats import ttest_1samp
```

In [ ]:

```
#getting t stats
ttest_1samp(Sample,mu)
```

Out[37]:

```
TtestResult(statistic=2.0446975432748005, pvalue=0.05005580140504867, df=29)
```

In [ ]:

```
# incase of one tailed pvalue*2 is the pvalue
```

In [ ]:

```

```

##Independent 2 sample T test

- An independent two-sample t-test is used to determine whether the means of two independent groups are significantly different from each other. The two groups should be independent of each other, meaning that the observations in one group are not related to the observations in the other group.

In [ ]:

```
#creating two arrays:
arr1 = np.array([5,3,3,5,5,6,6,7,3,8])
arr2 = np.array([6,7,9,3,6,5,4,2,6,8])
```

In [ ]:

```
#checking normality using shapiro
s1 = shapiro(arr1)
s2= shapiro(arr2)
print(s1)
print(s2)
```

```
ShapiroResult(statistic=0.910980761051178, pvalue=0.28779593110084534)
ShapiroResult(statistic=0.9741122126579285, pvalue=0.926135778427124)
```

In [ ]:

```
#checking if the variance is same using LEVENE test where if pval>0.05 the var is same else
from scipy.stats import levene
l = levene(arr1,arr2)
l
```

Out[41]:

```
LeveneResult(statistic=0.2842105263157895, pvalue=0.600475751924876)
```

In [ ]:

```
#performing the t test for 2 samples:
from scipy.stats import ttest_ind
ttest_ind(arr1,arr2,equal_var=True)
```

Out[42]:

```
Ttest_indResult(statistic=-0.5698028822981898, pvalue=0.5758546679640382)
```

- Since the p-value is greater than 0.05, we fail to reject the null hypothesis and conclude that there is insufficient evidence to suggest that the means of the two samples are different at the 95% confidence level.

2 sample dependent t test (Paired T test)

- A paired t-test is used to determine whether there is a significant difference between the means of two related groups.

In [ ]:

```
# Importing library
import scipy.stats as stats

# pre holds the mileage before
# applying the different engine oil
pre = [30, 31, 34, 40, 36, 35,
       34, 30, 28, 29]

# post holds the mileage after
# applying the different engine oil
post = [30, 31, 32, 38, 32, 31,
        32, 29, 28, 30]

# Performing the paired sample t-test
stats.ttest_rel(pre, post)
```

Out[43]:

```
TtestResult(statistic=2.584921310565987, pvalue=0.029457853822895275, df=9)
```

- Since the p-value is less than 0.05, we reject the null hypothesis and conclude that there is sufficient evidence to suggest that the means of the two samples are different at the 95% confidence level.

In [ ]:

F-test

- An F-test is a statistical hypothesis test that is used to compare the variances of two samples or the ratio of variances between two populations. The F-test is based on the F-distribution, which is a continuous probability distribution that arises in the analysis of variance (ANOVA) and regression.

In [ ]:

```
from scipy.stats import f_oneway

# create three samples
sample1 = np.array([1, 2, 3, 4, 5])
sample2 = np.array([2, 4, 6, 8, 10])
sample3 = np.array([3, 6, 9, 12, 15])

# perform F-test
f_statistic, p_value = f_oneway(sample1, sample2, sample3)

# print results
print("F-statistic:", f_statistic)
print("p-value:", p_value)
```

```
F-statistic: 3.857142857142857
p-value: 0.05086290933139865
```

- If the F-statistic is larger than the p-value, it means that the observed variability between groups is larger than would be expected by chance, and the null hypothesis of equal means across all groups is rejected at the chosen level of significance.

In [ ]:

```

```

ANOVA test:

- ANOVA stands for Analysis of Variance. It is a statistical method used to compare means of two or more groups to determine whether there is a significant difference between them.
  - The F-test is used in ANOVA to test the null hypothesis that the means of all groups are equal.

In [ ]:

```
import scipy.stats as stats

# Generate some sample data
group1 = [1, 2, 3, 4, 5]
group2 = [2, 4, 6, 8, 10]
group3 = [3, 6, 9, 12, 15]

# Perform one-way ANOVA
f_statistic, p_value = stats.f_oneway(group1, group2, group3)

# Print the results
print("F-statistic:", f_statistic)
print("p-value:", p_value)
```

```
F-statistic: 3.857142857142857
p-value: 0.05086290933139865
```

- If the F-statistic is larger than the p-value, it means that the observed variability between groups is larger than would be expected by chance, and the null hypothesis of equal means across all groups is rejected at the chosen level of significance.

In [ ]:

```

```

#Chi-Square test:

- The Chi-Square test is a statistical test used to determine if there is a significant association between two categorical variables. It is a non-parametric test, meaning that it does not make any assumptions about the distribution of the data.
- The formula for the Chi-Square test statistic is:

  χ^2 = Σ [(Oi - Ei)^2 / Ei]
  - where χ^2 is the test statistic, Oi is the observed frequency for category i, Ei is the expected frequency for category i (based on the assumption of independence), and the summation is taken over all categories

In [11]:

```
import pandas as pd
# Creating a dataframe with 2 categorical features:
data = pd.DataFrame({'Study': ['Yes', 'No', 'Yes','Yes', 'No', 'No','Yes', 'No', 'No'],
           'Result': ['Good', 'Bad', 'Bad','Good', 'Good', 'Bad','Good', 'Bad', 'Good']})

data
```

Out[11]:

	Study	Result
0	Yes	Good
1	No	Bad
2	Yes	Bad
3	Yes	Good
4	No	Good
5	No	Bad
6	Yes	Good
7	No	Bad
8	No	Good

In [13]:

```
# creating a crosstab table:
data_table = pd.crosstab(data['Study'],data['Result'])
data_table
```

Out[13]:

Result	Bad	Good
Study		
No	3	2
Yes	1	3

In [14]:

```
import scipy.stats as stats

observed = data_table.values

# Perform Chi-Square test
chi2_statistic, p_value, degrees_of_freedom, expected = stats.chi2_contingency(observed)

# Print the results
print("Chi-Square statistic:", chi2_statistic)
print("p-value:", p_value)
print("Degrees of freedom:", degrees_of_freedom)
print("Expected frequencies:", expected)
```

```
Chi-Square statistic: 0.1406249999999999
p-value: 0.7076604666545525
Degrees of freedom: 1
Expected frequencies: [[2.22222222 2.77777778]
 [1.77777778 2.22222222]]
```

- If the Chi-Square statistic is larger than the p-value, it means that the observed data has a larger deviation from the expected data than would be expected by chance, and the null hypothesis of independence between the two categorical variables is rejected at the chosen level of significance.

In [ ]:

```
```