ANOVA tests

1. one-way ANOVA test
2. two-way ANOVA test

one-way ANOVA test

- ANOVA : ANalysis Of VAriance
- used when we have multiple samples
- ANova uses F-Distribution(Fischer Distribution)
- H_o : all the pair of means are equal
- H_A / `H1` : there is atleast one pair of means where the difference is not equal to zero
- claim/response should follow normal distribution
- variances across all the samples are equal
- Predictor is attribute/categorical & claim/Response is continuous

import statsmodels

```
import numpy as np
import pandas as pd
# import numpy & pandas

import statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols
import statsmodels.stats.multicomp
# import statsmodels

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# import sklearn
```

upload dataset

```
from google.colab import files
uploaded=files.upload()
# MyData.xlsx
# to be used with google colab

# import os
# os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
# os.getcwd()
# to change current working directory to specified path
# to be used while running on local system
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving MyData.xlsx to MyData.xlsx

pd.read_excel('WorkBook_Name.xlsx', sheet_name='SheetName')

```
df = pd.read_excel('MyData.xlsx')
# reading excelfile with specifying the sheet name to load dataset
df.head()
```

	Sample	Value
0	A	5
1	A	4
2	A	6
3	A	7
4	A	5

```
import  statsmodels.api as sm
from statsmodels.formula.api import ols
import statsmodels.stats.multicomp
```

ols('Response ~ P1', data=dataSet).fit()

- 'sample' is called predictor
- 'value' is called as response
- `sample` `~` `value` means Sample column depends on value column, `~` is used to create relation between response & predctor

```
mod1 = ols('Value ~ Sample', data=df).fit()
# ols('colWithResponse ~ ColumnWithPredictor', data=dataFrame).fit()
# creating model using ols('Rsponse ~ P1', data=DataSet).fit()
```

sm.stats.anova_lm(model)

```
tbl = sm.stats.anova_lm(mod1)
# sm.stats.anova_lm(model)
# creates contingency table
tbl
```

	df	sum_sq	mean_sq	F	PR(>F)
Sample	2.0	130.278571	65.139286	37.416822	0.000012
Residual	11.0	19.150000	1.740909	NaN	NaN

- `df` : degree of freedom
- degree of freedom for residual = sample size 14 - sample 2 - 1 = 11
- `sum_sq` : sum of squares
- `mean_sq` : mean of squares
- `F` : F-Distribution
- `PR(>F)` : P-Value from F-distribution

import pairwise_tukeyhsd

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
# import pairwise_tukeyhsd
```

```
print(pairwise_tukeyhsd(df.Value, df.Sample))
# pairwise_tukeyhsd(Response_df, Predictor_df)
# create pairwise_tukeyhsd table
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
     A      B      0.2 0.9689 -2.0538 2.4538  False
     A      C     6.85    0.0  4.4595 9.2405   True
     B      C     6.65    0.0  4.2595 9.0405   True
---------------------------------------------------
```

two-way ANOVA test

- ANOVA : ANalysis Of VAriance
- used when we have multiple samples
- can give interaction between two factors/predictors
-

```
import numpy as np
import pandas as pd

```

```
from google.colab import files
uploaded=files.upload()
# CDAC_DataBook.xlsx
# to be used with google colab

# import os
# os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
# os.getcwd()
# to change current working directory to specified path
# to be used while running on local system
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving CDAC_DataBook.xlsx to CDAC_DataBook.xlsx

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='salaries')
df.head()
```

	rank	discipline	yrs_phd	yrs_service	gender	salary
0	Prof	B	19	18	Male	139750
1	Prof	B	20	16	Male	173200
2	AsstProf	B	4	3	Male	79750
3	Prof	B	45	39	Male	115000
4	Prof	B	40	41	Male	141500

```
import statsmodels
from statsmodels import stats
import statsmodels.api as sm
```

```
from statsmodels.formula.api import ols
import statsmodels.stats.multicomp
```

```
mydf = df[['rank', 'gender', 'salary']]
mydf.head()
```

	rank	gender	salary
0	Prof	Male	139750
1	Prof	Male	173200
2	AsstProf	Male	79750
3	Prof	Male	115000
4	Prof	Male	141500

```
# rank and gender are the predictors, salary is the response or output
mod1 = ols('salary~rank+gender', data = mydf).fit()
```

```
tbl = sm.stats.anova_lm(mod1)
```

```
print(tbl)
```

```
             df        sum_sq       mean_sq           F         PR(>F)
rank        2.0  1.432318e+11  7.161588e+10  128.382480  1.234599e-43
gender      1.0  8.408166e+08  8.408166e+08    1.507293  2.202874e-01
Residual  393.0  2.192281e+11  5.578322e+08         NaN           NaN
```

note

- average of response is same for all the pairs of predictor
- H_o : average salary is the same for all the pairs of genders
- H_o : average salary is same for all the pairs of ranks
- for rank: since P-Value(1.234599e-43) is less than 0.05, so we reject the H 0. Salary depends on rank
- for gender: since `P-Value(2.202874e-01)` > `0.05`, so we DO NOT reject the H_o. so Salary does not depend on gender

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

```
print(pairwise_tukeyhsd(mydf.salary, mydf.gender))
# this is without rank, so gender becomes a significant factor
```

```
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
==========================================================
group1 group2  meandiff  p-adj    lower     upper    reject
----------------------------------------------------------
Female   Male 14088.0087 0.0057 4131.1074 24044.9101   True
----------------------------------------------------------
```

Interaction between two factors: multiplying rank & gender

```
# rank and gender are the predictors, salary is the response or output
mod1 = ols('salary~rank*gender', data = mydf).fit()
```

```
tbl = sm.stats.anova_lm(mod1)
```

```
print(tbl)
```

```
                df        sum_sq       mean_sq           F        PR(>F)
rank           2.0  1.432318e+11  7.161588e+10  127.754543  2.010089e-43
gender         1.0  8.408166e+08  8.408166e+08    1.499921  2.214209e-01
rank:gender    2.0  4.360306e+07  2.180153e+07    0.038891  9.618588e-01
Residual     391.0  2.191845e+11  5.605741e+08         NaN           NaN
```

- if response is speed, and predicate are mode: Bike, car and weather: dry, rainy
- main effect plot, it is for one type of weather, but both modes: bike, car
- in interaction plot, response is on y-axis, and predictors are on the x-axis
- if the two lines are not parallel, then the two factor have an interaction
- H 0 : the rank & gender do not have an interaction
- H a : the tank & gender have interaction
- change in weather have different impact on speeds of car & bike
- change in traffic status also have a different impact on speeds of car & bike

Variance Tests

1. one-Sample Variance Test
2. two-Sample Variance Test

one-Sample Variance Test

- when we wish to compare the sample standard deviation of a sample against a claimed value to establish whether the difference is statistically significant or not
- used to comapre the ratios of variances of two samples and establish whether they are equal or not

```
my_std = 45
n=100
```

```
import scipy
from scipy import stats
from scipy.stats import chi2
# chi2 is not symmertric,
# because it is chi-squared distribution
```

```
crit1 = scipy.stats.chi2.ppf(0.025, 99)
crit1
```

```
73.36108019128368
```

```
crit2 = scipy.stats.chi2.ppf(0.975, 99)
crit2
```

```
128.4219886438403
```

```
low_est = ((n-1)/crit2)**0.5*my_std
low_est
```

```
39.51030804317418
```

```
upper_est = ((n-1)/crit1)**0.5*my_std
upper_est
```

```
52.27538648825794
```

two-Sample Variance Test

- compares the ratio of variances of two samples
- does not compare the difference of variance sof two samples
-

```
s1 = [5, 9, 8, 3, 4, 7, 6, 8, 9]
s2 = [2, 9, 8, 3, 5, 7, 6, 10, 12, 4]
# creating two samples
```

```
np.var(s1, ddof=1)
# actual ariance of s1
```

```
4.777777777777778
```

```
np.var(s2, ddof=1)
# actual ariance of s1
```

```
10.266666666666667
```

```
np.var(s1, ddof=1)/np.var(s2, ddof=1)
# finding ration of variances of two samples
```

```
0.4653679653679653
```

`Hₒ` : both variances are equal OR the ratio of the variances = 1 `HA` : both variances are equal OR the ratio of the variances is not 1

```
from scipy.stats import f
```

```
p_value = (scipy.stats.f.cdf(np.var(s1, ddof=1)/np.var(s2, ddof=1), len(s1)-1, len(s2)-1))*2
p_value
```

```
0.29495626539849623
```

Tools for discrete Data

- will be based on proportion and count
- ratio might be continuous, but we look at the discrete numbers for which we calculate ratio
- to draw conclusions, not predict

1. Proportion Tests
    - one-sample proportion Test
    - two-sample proportion test
2. chi square distribution tests

- goodness of fit test
- degree of association test

Proportion Tests

1. one-Sample Proportion Test
2. two-Sample Proportion Test

one-Sample Proportion Test

- the proportion calculated from a sample has to be compared against a claimed value, to establish the difference is staistically significant or not

1. sample is discrete
2. claim is proportion , which is continuous

- Q1.
- consultant claims that out of all the people who get jobs on DataScience, atleast 70% of them are engineers
- `Hₒ` : prop(engg) >= 0.7
- `HA` : prop(engg) < 0.7
- out of a sample of 270 professionals, 170 were engineers Does the data supports the claim ?

```
170/270
```

```
0.6296296296296297
```

- while dealing with proportions, `proportional S.D. = (p * (1-p))**0.5`

```
s = (0.6296*(1-0.6296))**0.5
# proportional S.D.
s
```

```
0.48291183460337767
```

```
low_est = 0.6296 - 1.96*0.4829/270**0.5
low_est
```

```
0.5719988180980312
```

```
upper_est = 0.6296 + 1.96*0.4829/270**0.5
upper_est
```

```
0.6872011819019689
```

```
import statsmodels
from statsmodels import stats
from statsmodels.stats import proportion
from statsmodels.stats.proportion import proportions_ztest
```

```
statsmodels.stats.proportion.proportions_ztest(170, 270, 0.7, alternative='smaller')
# here P-Value is less than 0.05, so we reject the H 0.
# sample proportion of 0.63(0.6296296296296297) cannot be considered >= 0.7
```

```
(-2.3944789436878944, 0.008321999540908103)
```

two-Sample Proportion Test

- difference of the proportions calculated from two samples is compared against the claimed value, to establish if the difference is statistically significant or not

1. sample is discrete, but proportion is continuous
2. claim is difference, which is continuous

- Q1.
- my claim is that BLR has atleast 12% more engineers than Kharghar
- H 0 : prop(BLR) - prop(KHA) >= 0.12
- H a : prop(BLR) - prop(KHA) < 0.12
- In BLR, in a sample of 150 students, 125 were engineers
- In KHA, in a sample of 100 students, 75 were engineers

```
125/150 - 75/100
# calculating actual difference of proportions
```

```
0.08333333333333337
```

- we need to check the difference between `actual difference 0.083` and `claimed difference 0.12` is statistically significant or not

```
ss = np.array([150, 100])
# creating sample1
```

```
en = np.array([125, 75])
# creating sample1
```

```
statsmodels.stats.proportion.proportions_ztest(en, ss, 0.12, alternative='smaller')
# statsmodels.stats.proportion.proportions_ztest(sample1, sample2, claimed value, alternative='sign of HA')
# returns (TestStatistics, P-Value)

```

```
(-0.7100469468046924, 0.23883751199295206)
```

- with `claimed difference of proportion = 0.12`, `P-value = 0.23883751199295206` > `0.05` , so we don't reject the claim

```

```