chi square distribution

1. goodness of fit - one variable
2. degree of association - two variable

goodness of fit test - one variable

- one variable, different categories
- based on "Pattern" of frequencies of data
- e.g.: student qualification- B.Tech., B.Sc. B.Com., etc.

```
import numpy as np
import pandas as pd
import scipy
from scipy import stats
from scipy.stats import chisquare
```

```
pattern = np.array([0.6, 0.25, 0.1, 0.05])
# creating an array to define the pattern of observations
```

```
sum(pattern)
# sum need to be necessarily as 1.0 as all of the
# elements in array are probabilities
```

```
1.0
```

```
obs_counts = np.array([250, 180, 120, 40])
# array of actual counts
# need to check if actual counts follow the pattern
obs_counts
```

```
array([250, 180, 120,  40])
```

- H 0 : counts from the sample follow the pattern

```
sum(obs_counts)
# sum of actual observations
```

```
590
```

```
exp_counts = 590 * pattern
# using ndarray broadcasting to calculate the expected counts as per pattern
exp_counts
```

```
array([354. , 147.5,  59. ,  29.5])
```

```
scipy.stats.chisquare(obs_counts, exp_counts)
# P-value is less than 0.05 , so 'H 0' is rejected and action is required
# actual/observation counts does not follow the expected counts

# Power_divergenceResult(statistic=104.51977401129943, pvalue=1.657508756495914e-22)
# here P-value is nearing zero
```

```
Power_divergenceResult(statistic=104.51977401129943, pvalue=1.657508756495914e-22)
```

Note:

- we can't rely on visual representation or practical significance and say that Null Hypothesis (H 0) is rejected, we have to check whether the difference is due to chance variation (check for statistical significance) by using stastical tools

ANOVA Test vs. Goodness of Fit test

- ANOVA test is for continuous data
- Goodness of fit test is for discrete data

```
d = obs_counts - exp_counts
# manually finding the deviation of obs_counts from exp_counts
d
```

```
array([-104. ,   32.5,   61. ,   10.5])
```

```
c = d**2 / exp_counts
# manually finding variation of obs_counts
c
```

```
array([30.55367232,  7.16101695, 63.06779661,  3.73728814])
```

```
sum(c)
# calculates test Statistic
```

```
# staistically difference between observed 120 and expected 59 is significant
```

```
104.51977401129943
```

TestStatistic vs. P-Value:

- as `TestStatistic` increases, `P-Value` will decrease
- Lower `TestStatistic` value means higher `P-value` means higher chance of Null Hypothesis being not rejected
- `TestStatistic = sum((ObservedCount - ExpectedCount)**2 / ExpectedCount)`

Degree of Association Test - two variable

- two variables, different categories
- based on strength & relation between two variables
- sample is discrete
- claimed value is categorical
- correlation coefficient is valid for continuous data only, not for categorical data
- e.g.:

```
a1 = np.array([[45, 27, 21], [31, 28, 27]])
# two variables are: count of students, count of students actually getting job
a1
```

```
array([[45, 27, 21],
       [31, 28, 27]])
```

- H_o : whether you get the job or not does not depend on the qualification

```
from scipy.stats import chi2_contingency
```

```
scipy.stats.chi2_contingency(a1)
#  testStatistic = 3.0780934692405153
# P-Value = 0.2145855609456467
# Degree of freedom = 2
# expected frequency array = [[39.48603352, 28.57541899, 24.93854749],
#                             [36.51396648, 26.42458101, 23.06145251]]

# Here , P-Value id 0.214 > 0.05, so we do not reject H 0
# dof = dof(row) x dof(column) = (2-1)x(3-1) = 1x2 = 2
```

```
Chi2ContingencyResult(statistic=3.0780934692405153, pvalue=0.2145855609456467, dof=2, expected_freq=array([[39.48603352, 28.57541899, 24.93854749],
       [36.51396648, 26.42458101, 23.06145251]]))
```

Poisson Rate tests

1. one-Sample Poisson Rate
2. two-Sample Poisson Rate

one-Sample Poisson Rate

- claim against the ratio of occurrences in one sample compare against claimed value
- to compare the counts/occurences obtained from a sample against the claimed value
- Q1.
- claimed that fewer than 3 accidents take place on the local every day
- H_o : n <= 3
- H_A : n > 3

two-Sample Poisson Rate

- ratio of the rates of occurrences in two different samples is compared against a claimed value

- Q1.
- in sample1 of 30 days, I saw 100 accidents were reported in Blore
- in sample2 of 20 days, I saw 75 accidents were reported in Mumbai

```
(100/30) / (75/20)
# ratio of rates of occurrences
```

- to compare the counts obtained from a sample against the claimed value
- claimed that fewer than 3 accidents take place on the local every day
- H_o : rate(Blr) / rate(Mum) = 1
- H_A : rate(Blr) / rate(Mum) != 1 --> two-tail test

inferential statistics DONE

→ Predictive Statistics

need for predictive statistics

1. Prediction & forecasting of numerical outcomes
2. Analysing & Quantifying the relationship between variables
3. identifying patterns

1. historical data is needed for prediction
2. when you need to make predictions
3. certain conditions in which data should be collected
4. supervised learning

- Decision tree, KNN, randomForest,

1. Supervised Learning
    1. Regression Models
        1. Linear Regression Model
            1. Simple Linear Regression Model
            2. Multiple Linear Regression Model
            3. Categorical Regression Model
        2. Non-Linear Regression Model

            3. Logistic Regression Model
                1. Binary Logistic Regression Model
                2. Nominal Logistic Regression Model
                3. Ordinal Logistic Regression Model
            4. Counts Regression Model
                1. Poisson Regression Model
                2. Negative Binomial Regression Model
        2. Classification Models
            1. Decision Tree Classification Model
            2. Random Forest Classification Model
            3. Support Vector Machine classification Model
            4. KNN (K-Nearest Neighbor) classification Model
            5. Gaussian Naive Bayes Classification Model
    2. Unsupervised Learning
        1. Clustering
            1. K-Mean Clustering
            2. Hierarchical Clustering
        2. Association

Supervised Learning

1. Regression Models
    - multiple input columns, one output column
2. Classification Models

Regression Models

Linear Regression Model

1. Simple Linear Regression Model
2. Multiple Linear Regression Model
3. Categorical Regression Model

Correlation Coefficient (r)

- correlation coefficient give information about the relation of two continuous variables
- range for correlation coefficient is `-1.0 to +1.0`

1. sign of correlation coefficient
   - tells the type of relation
     - +ve sign = +ve relation : if 1st value of x is inceasing, the value of 2nd variable y also increases

   e.g. age of car increases, cost of service also increases

     - -ve sign = +ve relation : if 1st value of x is increasing, the value of 2nd variable y also decreases

   e.g.: age of car increases, mileage/saleValue decreases

2. Magnitude of correlation coefficient:
   - tells the strength of relationship
   - also known as correlration factor
   - closer the magnitude to 1.0 stronger the relationship

Simple Linear Regression Model

- response/output - continuous data
- single predictor/input - continuous data
- response & predictor have linear relation
- basically calculates value of y & c for `y = mx + c`

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()
# 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
```

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

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

	eruptions	waiting
0	3.600	79
1	1.800	54
2	3.333	74
3	2.283	62
4	4.533	85

how to split dataset

- for training the model: 70-75%

- for testing the model: 25-30%
- dataSet splitting
  - X : predictor (P1, P2, ...), multiple predictor columns can exist
    - 70-75% : training
    - 25-30% : testing
  - Y : response (R), only one response column can exist
    - 70-75% : training
    - 25-30% : testing
- then compare the predicted value with actual value from testing dataset

train_test_split(predictor_cols, response_col, test_size=0.25)

```
x_train, x_test, y_train, y_test = train_test_split(df['waiting'], df['eruptions'], test_size=0.25)
# splitting dataset 4-ways
# rows are randomly selected for testing
```

```
x_train.head()
# indexing for training data will be same for predictor and response
```

```
80     75
118    59
63     82
103    83
134    46
Name: waiting, dtype: int64
```

```
y_train.head()
# indexing for training data will be same for predictor and response
```

```
80     4.133
118    1.817
63     4.800
103    4.500
134    1.833
Name: eruptions, dtype: float64
```

sm.add_constant(x_train, prepend=False)

```
x_train = sm.add_constant(x_train, prepend=False)
# adding constant to match equation
# if we do not add constant, model will follow y = mx
# if we add constant, model will follow y = mx + c
```

sm.OLS(y_train, x_train).fit()

```
mod1 = sm.OLS(y_train, x_train).fit()
# creating model using sm.OLS().fit()
```

```
print(mod1.summary())
# printing model summary / OLS Regression Results
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:              eruptions   R-squared:                       0.810
Model:                            OLS   Adj. R-squared:                  0.809
Method:                 Least Squares   F-statistic:                     860.8
Date:                Tue, 20 Jun 2023   Prob (F-statistic):           9.20e-75
Time:                        06:32:12   Log-Likelihood:                -144.16
No. Observations:                 204   AIC:                             292.3
Df Residuals:                     202   BIC:                             299.0
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
waiting        0.0742      0.003     29.339      0.000       0.069       0.079
const         -1.8057      0.183     -9.848      0.000      -2.167      -1.444
==============================================================================
Omnibus:                        2.692   Durbin-Watson:                   1.907
Prob(Omnibus):                  0.260   Jarque-Bera (JB):                2.107
Skew:                          -0.100   Prob(JB):                        0.349
Kurtosis:                       2.544   Cond. No.                         385.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

`y = mx + c`

- 1st Null Hypothesis, H_o : for m

  2nd Null Hypothesis, H_o : for c

- H_o : m = 0 OR x has no effect on y

- `Hₒ` : c = 0
- `Number of possible Hypothesis = Number of variables + 1`
    - for `y = mx + c`, number of variables = 1
    - so, number of possible hypothesis = 2

- as per model summary
    - in `y = mx + c`,
    - the value of m is 0.0742 from column 'coef'
    - the value of c is -1.8057 from column 'coef'
    - P-Value is 0.00 from column 'P>|t|'
    - the influence of x on y is statistically significant, it is not a mattter of chance variation
    - influence of waiting on eruption is statistically significant and cannot be ignored, it is not a matter of chance variation
    - R-Squared:
        - is the square of correlation coefficient incase of one predictor
        - value of R-Squared closer to 1.0 is a better model
        - indicates if model can explain its variation
        - 0.810 means model can expains 81% variation / chance variation
        - means model cannot explain 19% variation / chance variation

```
x_test = sm.add_constant(x_test, prepend=False)
# adding constant to test sample
```

```
y_pred = mod1.predict(x_test)
# generating prediction for test data
```

```
y_pred[:5]
# printing first five generated predictions
```

```
154    3.459169
152    3.014254
160    1.531204
211    4.126541
241    1.679509
dtype: float64
```

```
y_test[:5]
# printing first five records from test data
# to compare with the generated predictions
```

```
154    3.567
152    2.400
160    2.200
211    4.700
241    2.350
Name: eruptions, dtype: float64
```

Multiple Linear Regression Model

- response/output - continuous data
- multiple predictors/input - continuous data
- basically calculates value of y & c for `y = m1x1 + m2x2 + c`

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()
# 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
```

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

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

	AirFlow	WaterTemp	AcidConc	StackLoss
0	80	27	89	42
1	80	27	88	37
2	75	25	90	37
3	62	24	87	28
4	62	22	87	18

train_test_split(predictor_cols, response_col, test_size=0.25)

```
x_train, x_test, y_train, y_test = train_test_split(df.drop('StackLoss', axis=1), df['StackLoss'], test_size=0.25)
# splitting dataset 4-ways
# rows are randomly selected for testing
```

- H 0 : airflow has no effect on StackLoss
- H 0 : WaterTemp has no effect on StackLoss
- H 0 : AcidConc has no effect on StackLoss

sm.add_constant(x_train, prepend=False)

```
x_train = sm.add_constant(x_train, prepend=False)
# adding constant to match equation
# if we do not add constant, model will follow y = mx
# if we add constant, model will follow y = m1x1 + m2x2 + m3x3 + ... + c
```

sm.OLS(y_train, x_train).fit()

```
mod1 = sm.OLS(y_train, x_train).fit()
# creating model
```

```
print(mod1.summary())
# printing model summary / OLS Regression Results
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:              StackLoss   R-squared:                       0.904
Model:                            OLS   Adj. R-squared:                  0.877
Method:                 Least Squares   F-statistic:                     34.35
Date:                Tue, 20 Jun 2023   Prob (F-statistic):           7.01e-06
Time:                        07:12:30   Log-Likelihood:                -36.204
No. Observations:                  15   AIC:                             80.41
Df Residuals:                      11   BIC:                             83.24
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
AirFlow        0.6928      0.147      4.711      0.001       0.369       1.016
WaterTemp      1.1766      0.404      2.911      0.014       0.287       2.066
AcidConc      -0.1515      0.162     -0.933      0.371      -0.509       0.206
const        -36.7101     12.229     -3.002      0.012     -63.627      -9.793
==============================================================================
Omnibus:                        2.711   Durbin-Watson:                   2.904
Prob(Omnibus):                  0.258   Jarque-Bera (JB):                0.736
Skew:                          -0.140   Prob(JB):                        0.692
Kurtosis:                       4.048   Cond. No.                     1.62e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.62e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:1736: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=15
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

- here equation for linear regression is, `y = y1m1 + y2m2 + y3m3 + c`
- m1 = coeff of AirFlow, P-Value=0.001 < 0.05, so H 0 is rejected, so AirFlow has an impact on StackLoss
- m2 = coeff of WaterTemp, P-Value=0.014 < 0.05, so H 0 is rejected, so WaterTemp has an impact on StackLoss
- m3 = coeff of AcidConc, P-Value=0.371 < 0.05, so H 0 is NOT rejected, so AcidConc has an impact on StackLoss

Categorical Regression Model

- Multiple Regression Model with dummy columns
- response/output - continuous data, but at least one predictor is categorical
- multiple predictors/input - continuous data
- basically calculates value of y & c for `y = m1x1 + m2x2 + c`

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()
# 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
```

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

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='salaries')
# reading excelfile with specifying the sheet name to load dataset
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

```
df = df[['rank', 'yrs_service', 'salary']]
# selecting specific columns from dataset
df.head()
```

	rank	yrs_service	salary
0	Prof	18	139750
1	Prof	16	173200
2	AsstProf	3	79750
3	Prof	39	115000
4	Prof	41	141500

Note for dummies columns

- for continuous or discrete data, reference value is zero
- for categorical data, out of `n` possible values, one will be reference, and other `n-1` possible values will be target values
- while creaating dummies, alphabetically first category will be taken as reference value and dummy columns will be created for other target values

pd.get_dummies(categorical_col, drop_first=True)

- create dummies for categorical columns
- first value in alphabetical order is automatically taken as reference value

```
rank_dummy = pd.get_dummies(df['rank'], drop_first=True)
# creating dummy column
rank_dummy.head()
```

	AsstProf	Prof
0	0	1
1	0	1
2	1	0
3	0	1

```
df = df.drop('rank', axis=1)
# dropping actual categorical column
df.head()
# checking columns in DataFrame after dropping categorical column
```

	yrs_service	salary
0	18	139750
1	16	173200
2	3	79750
3	39	115000
4	41	141500

```
df = pd.concat([df, rank_dummy], axis=1)
# concatenating dummy column in place of actual categorical column
df.head()
# checking columns in DataFrame after concatenating categorical column
```

	yrs_service	salary	AsstProf	Prof
0	18	139750	0	1
1	16	173200	0	1
2	3	79750	1	0
3	39	115000	0	1
4	41	141500	0	1

train_test_split(predictor_cols, response_col, test_size=0.25)

```
x_train, x_test, y_train, t_test = train_test_split(df.drop('salary', axis=1), df['salary'], test_size=0.25)
# splitting dataset 4-ways
# rows are randomly selected for testing
```

sm.add_constant(x_train, prepend=False)

```
x_train = sm.add_constant(x_train, prepend=False)
# adding constant to match equation
# if we do not add constant, model will follow y = mx
# if we add constant, model will follow y = m1x1 + m2x2 + m3x3 + ... + c
x_train.head()
# checking columns in Predictor-training DataFrame after adding constant
```

sm.OLS(y_train, x_train).fit()

```
mod1 = sm.OLS(y_train, x_train).fit()
# creating model
```

```
print(mod1.summary())
# printing model summary / OLS Regression Results
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 salary   R-squared:                       0.404
Model:                            OLS   Adj. R-squared:                  0.398
Method:                 Least Squares   F-statistic:                     66.15
Date:                Tue, 20 Jun 2023   Prob (F-statistic):           1.08e-32
Time:                        11:16:39   Log-Likelihood:                -3419.4
No. Observations:                 297   AIC:                             6847.
Df Residuals:                     293   BIC:                             6862.
Df Model:                           3
Covariance Type:            nonrobust
===============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
yrs_service  -156.0714    138.097     -1.130      0.259    -427.860     115.717
AsstProf    -1.201e+04   5093.799     -2.357      0.019    -2.2e+04   -1982.358
Prof         3.748e+04   4158.028      9.015      0.000    2.93e+04    4.57e+04
const        9.399e+04   3976.732     23.635      0.000    8.62e+04    1.02e+05
==============================================================================
Omnibus:                       35.178   Durbin-Watson:                   2.022
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               54.656
Skew:                           0.730   Prob(JB):                     1.35e-12
Kurtosis:                       4.511   Cond. No.                         102.
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

interpreting predictors

- for rank, reference value is `AssocProf`, and all other categorical values are target values as per dummy column
- if rank changes from `AssocProf` --> `AsstProf`, keeping all other factors constant, salary is decreasing by 1.201e+04, decreasing due to -ve sign
- if rank changes from `AssocProf` --> `Prof`, keeping all other factors constant, salary is increasing by 3.748e+04, increasing due to +ve sign

interpreting const

- `const = 9.399e+04` , means if `x=0` in equation `y = mx + c`, then `y = c` called as y-intercept.
- This implies that the starting salary of employee is 9.399e+04 , when yrs_service = 0 and rank is also zero or fresher

Non-Linear Regression Model

```
1
```