Library

1. numpy
2. pandas
3. Matplotlib.pyplot - . This module is widely used for creating static, interactive, and animated visualizations in Python.
4. Seaborn - Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
5. scipy.stats.pearsonr, spearmanr
6. math
7. sklearn.model.train\_test\_split
8. sklearn.tree.DecisionTreeRegressor
9. sklearn.metrics.accuracy\_score

Correlation

Covariance

Outliers

For Positive - if x is increase & y is also increase then it is positive covariance & positive correlation

For Negative - If x is increase & y is decrease then it is negative covariance & negative correlation

For Zero - If x is increase & y is doesn't have any effect then there is no covariance & no correlation

Random Variable - Function that maps every outcome in the sample space to a real number

It can be both discrete & continuous

Covariance indicates the direction of the linear relationship between variables

Correlation measures both the strength & direction of the linear relationship between two variables

Correlation is a function of covariance

Covariance Calculation

| Day | x | y | X-x̄ | y-ȳ | (X-x̄)(y-ȳ) |
| --- | --- | --- | --- | --- | --- |
| 1 | 30 | 5 | -3 | -1 | 3 |
| 2 | 35 | 8 | +2 | +2 | 4 |
| 3 | 40 | 8 | +7 | +2 | 14 |
| 4 | 25 | 4 | -8 | -2 | 16 |
| 5 | 35 | 5 | +2 | -1 | -2 |
| Mean | x̄ = 33 | ȳ = 6 |  |  | Sum = 35 |

Mean = x1+x2 + x3 + x4 + ..+xn / n

n is no of samples

X̄ = 30 + 35 + 40 + 25 + 35 / 5 = 165/5 = 33

Covariance cov(x,y) = σₓᵧ = Σ(X-x̄)(y-ȳ) / n-1 = 35 /4 = 8.75

If there is need to know more than direction & similarity we can use correlation

Correlation value “ “ ->Strength

Sign indicates direction which is positive or negative

Different correlation :-

1. Pearson Correlation Coeffiecient “r“
2. Spearman’s Rank Correlation “⎷“

Note - Linear is a straight line on a graph ex - y = 2x + 5

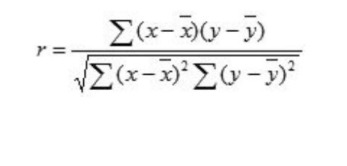
Non -Linear don't follow a straight line on a graph ex - y = x^2 + 3

Pearson Correlation is work on linear relationship

-1 strong negative correlation

+1 strong positive correlation

| Day | x | y | X-x̄ | y-ȳ | (X-x̄)(y-ȳ) | (X-x̄)2 | (y-ȳ)2 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 30 | 5 | -3 | -1 | 3 | 9 | 1 |
| 2 | 35 | 8 | +2 | +2 | 4 | 4 | 4 |
| 3 | 40 | 8 | +7 | +2 | 14 | 49 | 4 |
| 4 | 25 | 4 | -8 | -2 | 16 | 64 | 4 |
| 5 | 35 | 5 | +2 | -1 | -2 | 4 | 1 |
| Mean | x̄ = 33 | ȳ = 6 |  |  | Sum = 35 | Sum = 130 | Sum = 14 |

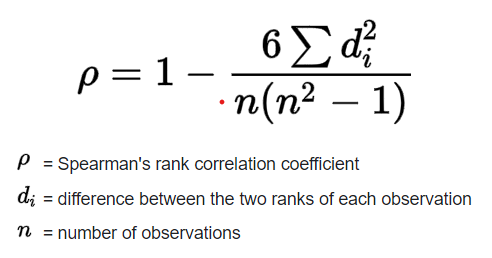


= 35 / √(130 x 14) = 35 / √1820 = 35 / 42.66 = 0.8204

It means giving direction positive & strength is 0.824 of relationship

Pearson correlation not sensitive to non-linear relationship of similarity & to slope of two variable To proceed that time of problem spearman’s rank correlation should be used

Spearman’s Correlation



| Student ID | Maths | Physics | Rank of Maths  = R(M) | Rank of Physics  = R(P) | d = R(m) - R(P) | d2 |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 35 | 30 | 7 | 5 | +2 | 4 |
| 2 | 23 | 33 | 5 | 7 | -2 | 4 |
| 3 | 47 | 45 | 9 | 8 | +1 | 1 |
| 4 | 17 | 23 | 4 | 4 | 0 | 0 |
| 5 | 10 | 8 | 3 | 2 | +1 | 1 |
| 6 | 43 | 49 | 8 | 9 | -1 | 1 |
| 7 | 9 | 12 | 2 | 3 | -1 | 1 |
| 8 | 6 | 4 | 1 | 1 | 0 | 0 |
| 9 | 28 | 31 | 6 | 6 | 0 | 0 |
|  |  |  |  |  |  | Σd2= 12 |

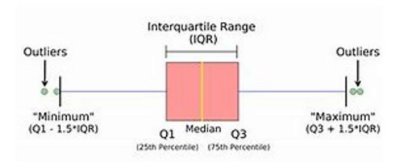
ρ = 1 - ((6 x 12) / 9(81 -1)) = 1 - (72 / 720) = 1 - 0.1 = 0.9

0.9 means 90% similarity obtained marks in maths & physics

| Day | x | y | R(x) | R(y) | d=R(x) - R(y) | d2 |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 30 | 5 | 2 | 2.5 | -0.5 | 0.25 |
| 2 | 35 | 8 | 3.5 | 4.5 | -1 | 1 |
| 3 | 40 | 8 | 5 | 4.5 | +0.5 | 0.25 |
| 4 | 25 | 4 | 1 | 1 | 0 | 0 |
| 5 | 35 | 5 | 3.5 | 2.5 | +1 | 1 |
|  |  |  |  |  |  | Σd2 = 2.5 |

ρ = 1 - ((6 x 2.5) / 5(25-1)) = 1 - 15/ 120 = 1 - 0.125 = 0.875

Outliers -



IQR 25%-75% Percentage which means 50% data

Outliers is a significant deviate of data

Total range of data(min value & max value)

Range = max - min

IQR = Q3 - Q1 = 50%

Note- Value of data appear bear min & bear max are called outliers

Imp - correlation between dependent & independent variable. Uncorrelated are dependent variable

| """Descriptive Analytics  1. Working with DataFrame  """  #Loading CSV file into python as DataFrame & print data  import pandas as pd  ipl\_auction\_df = pd.read\_csv( 'IPL IMB381IPL2013.csv' )  print(ipl\_auction\_df)  #Check the nature of variable  print(type(ipl\_auction\_df))  #Display First 5 entries with headers  print(ipl\_auction\_df.head(5))  #finding summary of DataFrame  #see data type is object or categorical  ipl\_auction\_df.info()  #To find meta data of frame  print(list(ipl\_auction\_df)) |
| --- |

| """Slicing and Indexing a dataframe"""  #1. By ROWS  #First five entries  ipl\_auction\_df[0:5]  #Last five entries  ipl\_auction\_df[-5:]  #Slicing By Colunm  ipl\_auction\_df['PLAYER NAME'][0:5]  #Slincing only 2 colunm 5 entries  ipl\_auction\_df[['PLAYER NAME', 'COUNTRY']][0:5]  #Sorting dataframe by column values  ipl\_auction\_df[['PLAYER NAME','SOLD PRICE']].sort\_values('SOLD PRICE')[0:5]  #Creating New Colunm  #Which player got the maximum premium on the base price?  ipl\_auction\_df['premium'] = ipl\_auction\_df['SOLD PRICE'] - ipl\_auction\_df['BASE PRICE']  #slicing 5 entries  ipl\_auction\_df[['PLAYER NAME', 'BASE PRICE', 'SOLD PRICE', 'premium']][0:5]  #sort values according to premium column in ascending order  print(ipl\_df[["PLAYER NAME","BASE PRICE","SOLD PRICE","PREMIUM"]].sort\_values("PREMIUM")[0:5])  #sort values according to premium column in descending order  ipl\_auction\_df[['PLAYER NAME',  'BASE PRICE',  'SOLD PRICE', 'premium']].sort\_values('premium',  ascending = False)[0:5] |
| --- |

| """Grouping and Aggregating  """  #grouping based on age & take mean of column sold price & reset index #so that it print according to new index  soldprice\_by\_age = ipl\_auction\_df.groupby('AGE')['SOLD PRICE'].mean().reset\_index()  #print new data  print(soldprice\_by\_age) |
| --- |

| import numpy as np  cov\_matrix=np.cov(ipl\_auction\_df['SOLD PRICE'],ipl\_auction\_df['BASE PRICE'])  cov\_matrix |
| --- |

| covariance = cov\_matrix[0, 1]  covariance |
| --- |

| cor\_matrix=np.corrcoef(ipl\_auction\_df['BASE PRICE'],ipl\_auction\_df['SOLD PRICE'])  cor\_matrix |
| --- |

| correlation = cor\_matrix[0,1]  correlation |
| --- |

| #Create a frequency distribution clock of discrete using bar plot  import matplotlib.pyplot as plt  import seaborn as sn  %matplotlib inline  sn.barplot(x='AGE',y='SOLD PRICE',data=Soldprice\_by\_age) |
| --- |

| #Create a frequency distribution clock of continous attribute using histogram  plt.hist(df["SOLD PRICE"],bins = 20) |
| --- |

| #To find relationship between two attributes that is sixer's vs sold price create a scatter plot  plt.scatter(x=df["SIXERS"],y=df["SOLD PRICE"])  plt.xlabel('SIXERS')  plt.ylabel('SOLD PRICE')  plt.title("Scatter plot between players sixers and sold price") |
| --- |

| #To find relationship between two attributes that is sixer's vs sold #price create a scatter plot with inclusion of direction of relationship  sn.regplot(x=df["SIXERS"],y=df["SOLD PRICE"])  #to create a scatter plot with a linear regression line. |
| --- |

| #Create an array of attribute called as influential attribute called as #influential attribute & prepare a pair plot  #Infliential Attributes=["SR-B","AVE","SIXERS","SOLD PRICE"]  influential\_features = ["SR-B","AVE","SIXERS","SOLD PRICE"]  sn.pairplot(df[influential\_features],height =2) |
| --- |

| #Find the correlation of influential feature array  df[influential\_features].corr() |
| --- |

| #Highlight correlation value with heat map function  sn.heatmap(df[influential\_features].corr(),annot = True) |
| --- |

When annot is set to True, the numerical values of the correlations will be displayed on the heatmap.

| #Prepare a box plot to calculate range of attribute sold price using #seaborn library  box = sn.boxplot(df["SOLD PRICE"]) |
| --- |

| #Prepare a box plot to calculate range of attribute sold price using #matplot library  box = plt.boxplot(df["SOLD PRICE"]) |
| --- |

| #Find minimum & maximum value of sold price attribute  [item.get\_ydata()[0] for item in box["caps"]] #This accesses the list of "caps" in the box plot. In a box plot, "caps" #are the horizontal lines at the ends of the whiskers, representing the #minimum and maximum values excluding outliers |
| --- |

| #Find IQR Value of sold Price  [item.get\_ydata()[0] for item in box["whiskers"]] |
| --- |

| # Find the median of sold price  [item.get\_ydata()[0] for item in box["medians"]] |
| --- |

Mean = sum of numbers / nos

Mode = Repeated number maximum times

Median = sort the list if odd middle number or if even add two middle number / 2

Median does not affect by outliers

Covariance & correlation

| cov\_matrix=np.cov(data['X'],data['Y'])  Or  cov\_matrix=data.cov()  cov\_matrix |
| --- |

| covariance = cov\_matrix[0, 1]  covariance |
| --- |

| cor\_matrix = np.corrcoef(data['X'], data['Y'])  Or  cov\_matrix=data.corr()  cor\_matrix |
| --- |

| correlation = cor\_matrix[0,1]  correlation |
| --- |

Random data

| import numpy as np  import pandas as pd  #Generate sample data  np.random.seed(0)#Configuration for random number  data=pd.DataFrame(np.random.randn(100,2), columns=['X','Y'])  #Create a two features data, here naming X and Y of size 100 samples  data.head(5) |
| --- |

seed(0) :- This means that the sequence of random numbers generated will be the same each time you run your program

| #Calculate covariance matrix  cov\_matrix=np.cov(data['X'],data['Y'])  cov\_matrix |
| --- |

| #Extract covariance  covariance = cov\_matrix[0, 1]  #b and c element represent covariance  covariance |
| --- |

| #Calculate correlation matix  cor\_matrix = np.corrcoef(data['X'], data['Y'])  cor\_matrix |
| --- |

| correlation = cor\_matrix[0,1]  correlation |
| --- |

| # Pearson correlation coefficient  pearson\_corr, \_ = pearsonr(data['X'], data['Y'])  print("Pearson correlation coefficient:", pearson\_corr)  # Spearman correlation coefficient  spearman\_corr, \_ = spearmanr(data['X'], data['Y'])  print("Spearman correlation coefficient:", spearman\_corr) |
| --- |

| #@title  import matplotlib.pyplot as plt  import seaborn as sns  def create\_box\_plot(data2):  plt.figure(figsize=(8,6))  sns.boxplot(data=data2)  plt.title("Box Plot")  plt.xlabel("Data")  plt.ylabel("Values")  plt.show()  #Example data  data1 = {  'Group A' :[-1, 12, 15, 18, 20 , 22], #add -a and see outlier  'Group B' :[11, 14, 17, 20, 23, 26],  'Group C' :[8, 10, 12, 14, 16, 18]  }  data2=pd.DataFrame(data1)  create\_box\_plot(data2) |
| --- |

Full Model

| #data creation for diabetics in which we have age bmi & outcomes where 1 for diabetics & 0 for non-diabetics  data = {  "Age" : [50, 30, 35, 40, 25, 45, 55, 60, 65, 70, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70],  "BMI" : [25, 30, 28, 27, 26, 29, 32, 33, 34, 36, 22, 24, 23, 21, 20, 31, 32, 34, 35, 37],  "Outcomes" : [1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1]  }  #converting data into dataframe & print data  import pandas as pd  df=pd.DataFrame(data)  print(df)  #scatter plot with seaborn  import matplotlib.pyplot as plt  import seaborn as sn  #The figsize parameter takes a tuple indicating the width and height in #inches.  plt.figure(figsize=(10,6))  sn.scatterplot(x="Age", y="BMI", hue="Outcomes", data=df, palette="coolwarm", s=100)  #The coolwarm palette ranges from cool (blue) to warm (red) colors.  #each point in the scatter plot will have a size  plt.show() |
| --- |

| # Separate features and labels  #This function is used to split datasets into training and testing #subsets.  from sklearn.model\_selection import train\_test\_split  # Defines a list of feature names that will be used as input variables #for the model.  features = ['Age','BMI']  # Defines the target variable (label) name that the model will predict.  label = 'Outcomes'  #X1 contains the feature values (columns Age and BMI).  #y1 contains the label values (column Outcomes).  X1, y1 = df[features].values, df[label].values  #test\_size=0.30 specifies that 30% of the data should be used for #testing, and the remaining 70% for training.  #random\_state=0 ensures that the split is reproducible by setting a fixed #seed for random number generation.  #if you run the split multiple times, you might get different results each time.  #By setting a fixed "random seed" (using random\_state=0), you tell the #computer to use the same starting point for the random number generator #every time you run the code  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X1, y1, test\_size=0.30, random\_state=0) |
| --- |

| # Train the model  #This class implements a decision tree for regression tasks.  from sklearn.tree import DecisionTreeRegressor  # train a decision tree model on the training set  model = DecisionTreeRegressor().fit(X\_train, y\_train)  print(model) |
| --- |

| #Validating the model and preparing metrics evaluation  #This function calculates the accuracy of predictions.  from sklearn.metrics import accuracy\_score  # Uses the trained model to make predictions on the test set (X\_test)  predictions = model.predict(X\_test)  print('Predicted labels: ', predictions)  print('Actual labels: ' ,y\_test)  #The accuracy\_score function compares y\_test (true labels) and predictions (predicted labels) to determine the accuracy.  print('Accuracy: ', accuracy\_score(y\_test, predictions)) |
| --- |

Handling Missing Data

| import pandas as pd  autos = pd.read\_csv("auto-mpg.data",  sep='\s+',  header=None)  #sep is a parameter that specifies the delimiter to use. The value '\s+' #is a regular expression for one or more whitespace characters (spaces, #tabs, etc.).  #header is a parameter that specifies which row to use as the header #(column names). The value None means that the file does not have a #header row and the columns should be numbered starting from 0.  autos.head(5) |
| --- |

| autos.columns = ['mgp','cylinders','displacement','horespower','weight','acceleration','year','origin','name']  autos.head(5) |
| --- |

| autos.info() |
| --- |

| autos["horespower"] = pd.to\_numeric(autos["horespower"], errors ='coerce')  #'coerce' means that any values that cannot be converted to a numeric #type will be set to NaN (Not a Number). This is useful for handling #missing or malformed data  autos.info() |
| --- |

| autos[autos.horespower.isnull()] |
| --- |

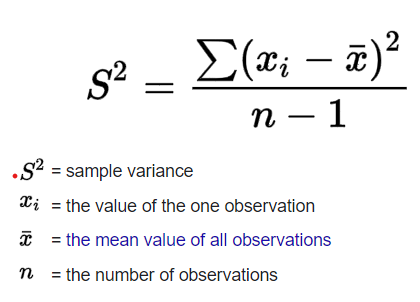
| autos = autos.dropna(subset = ["horespower"])  autos[autos.horespower.isnull()] |
| --- |

| print(autos[360:]) |
| --- |

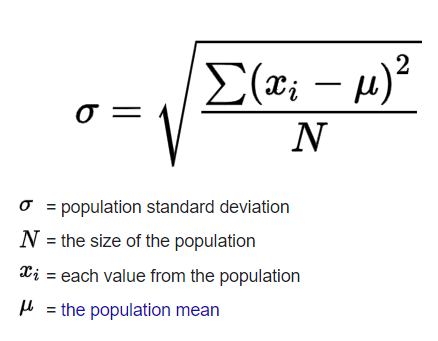
Measure of dispersion

Range = largest value - smallest value

Variance =



Standard deviation =



Probability Distribution and Data

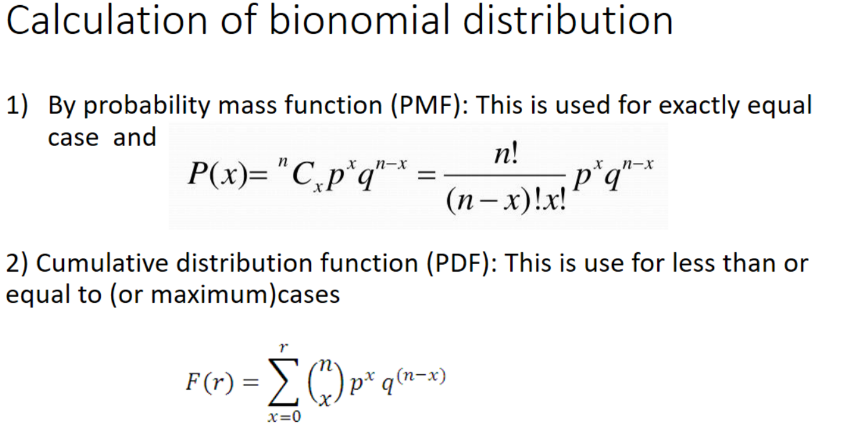
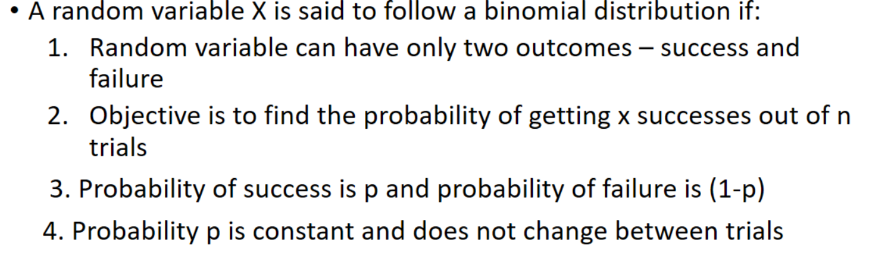
Discrete Distribution - Binomial, Poisson & Geometric distribution

Continuous Distribution - Uniformed, Exponential and Normal

Discrete Random Variable - sample X can assume finite or countable infinite

Continous Random Variable - sample X can assume from infinite set of values

They can be described by PMF(Probability Mass Function) & CDF(Cumulative Distribution Function)

Binomial Distribution

| #calculation using PMF  #Showing how color blindless effect 8% of men from a sample of 10 men  #a) Probabiltiy that all 10 mens are color blind  from scipy import stats  stats.binom.pmf(10,10,0.08) |
| --- |

| #b)Probabilty that all zero mens are color blind  stats.binom.pmf(0,10,0.08) |
| --- |

| #c)Probability that exactly 2 mens are color blind  stats.binom.pmf(2,10,0.08) |
| --- |

| #d) Probability that atleast 2 mens are color blind  1-stats.binom.pmf(0,10,0.08)-stats.binom.pmf(1,10,0.08) |
| --- |

| import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sn  pmf\_df = pd.DataFrame({'success': range(0,10),  'pmf' : list(stats.binom.pmf(range(0,10),  10,  0.08))})  pmf\_df |
| --- |

**Explanation of above code**

stats.binom.pmf is a function from the scipy.stats module that calculates the probability mass function (PMF) of a binomial distribution.

range(0, 10) provides the possible number of successes (k) in the binomial distribution.

10 is the number of trials (n) in the binomial distribution.

0.08 is the probability of success (p) on each trial.

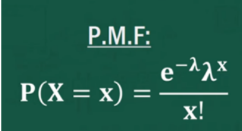
Converts the PMF values from an array to a list so it can be easily used as the values for the 'pmf' column.

The DataFrame has two columns: 'success' and 'pmf'.

| sn.barplot(x = pmf\_df.success,y=pmf\_df.pmf)  plt.ylabel('pmf')  plt.xlabel('Number of items Returned') |
| --- |

Df.column\_name

Poisson Distribution

It requires only one parameter (lambda=timeinterval)

We can use cdf as it maximum number finding probability

| #The number of calls arriving at a call center #follows a Poisson distribution at 10 calls per #hour  #a) Calculate the probability that the number of #calls will be maximum 5  from scipy import stats  stats.poisson.cdf(5,10) |
| --- |

| #b) Calculate the probability that the number of #calls over a 3-hour period will exceed 30  1 - stats.poisson.cdf(30,30) |
| --- |

| #A random poisson dsitribution plot  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sn  pmf\_df = pd.DataFrame({'success' :range(0,30),  "pmf":list(stats.poisson.pmf(range(0,30),  10))})  #A two column data with lamda=1  pmf\_df |
| --- |

| sn.barplot(x = pmf\_df.success,y=pmf\_df.pmf)  plt.ylabel('pmf')  plt.xlabel('Number of items Returned') |
| --- |

ETL (Extract Transform Load)

Normal Distribution known as gaussian distribution

| #Load Stock1 into dataframe. Display their 5 #entries  import pandas as pd  Import warnings  df = pd.read\_csv("/content/BEML.csv")  dfw = pd.read\_csv("/content/GLAXO.csv")  print(df.head(5))  print(dfw.head(5)) |
| --- |

| #Select a few attributes which is needed to #check performance of two attribute  df1 = df[["Date","Close"]]  df2 = dfw[["Date","Close"]]  print(df1.head(5))  print(df2.head(5)) |
| --- |

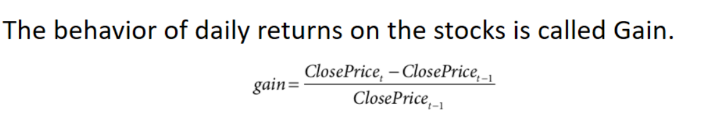
| #Preparing a time plot for a close attribute  import matplotlib.pyplot as plt  import seaborn as sn  %matplotlib inline  plt.plot(df1.Close);  plt.plot(df2.Close);  plt.xlabel("Time");  plt.ylabel("Close Price") |
| --- |

| # Arrange the data in the order of time  df1 = df1.set\_index(pd.DatetimeIndex(df1["Date"]))  print(df1.head(5))  df2 = df2.set\_index(pd.DatetimeIndex(df2["Date"]))  df2.head(5) |
| --- |

set\_index() is a method in pandas DataFrame that sets the DataFrame's index using existing columns.

| plt.plot(df1.Close);  plt.plot(df2.Close);  plt.xlabel("Time");  plt.ylabel("Close Price") |
| --- |

| #The dataframe which is created df1 move into #newbeml csv file  destination\_file = "NewBEML.csv"  df1.to\_csv(destination\_file, index=False)  print("ETL process completed")  destination\_file1 = "NewGLAXO.csv"  df2.to\_csv(destination\_file1, index=False)  print("ETL1 process completed") |
| --- |



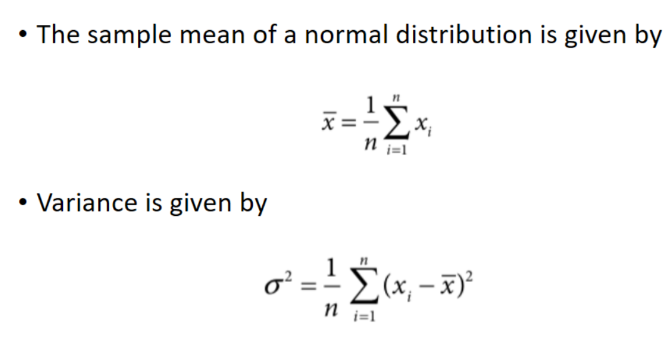
| df1['gain'] = df1.Close.pct\_change(periods =1)  df2['gain'] = df2.Close.pct\_change(periods =1)  print(df1.head(5))  df2.head(5) |
| --- |

.pct\_change() is a pandas DataFrame/Series method that calculates the percentage change between the current and a prior element.

periods=1 specifies the number of periods to shift for calculating the percentage change. Here, it calculates the percentage change between the current element and the element 1 position back.

| plt.figure(figsize = (8,6))  plt.plot(df1.index,df1.gain)  plt.plot(df2.index,df2.gain)  plt.xlabel("Time")  plt.ylabel("Gain") |
| --- |

| sn.distplot(df2.gain, label = "Glaxo")  sn.distplot(df1.gain, label = 'BEML')  plt.xlabel('gain')  plt.ylabel('Density')  plt.legend() |
| --- |



| print("Daily gain of Glaxo")  print("-------------------------")  print("mean: ",round(df2.gain.mean(),4))  print("Standard Deviation ",round(df2.gain.std(),4))  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  print("Daily gain of BEML")  print("-------------------------")  print("mean: ",round(df1.gain.mean(),4))  print("Standard Deviation ",round(df1.gain.std(),4)) |
| --- |

| print(df1.gain.describe())  df2.gain.describe() |
| --- |

.describe() is a pandas DataFrame/Series method that generates descriptive statistics of the column it is applied to.

It provides statistics such as count, mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile or Q2), 75th percentile (Q3), and maximum.

| import numpy as np  from scipy import stats  glaxo\_df\_ci = stats.norm.interval(0.95,  loc = df2.gain.mean(),  scale = df2.gain.std())  print("Gain at 95% confidence interval in GLAXO is : ",np.round(glaxo\_df\_ci,4))  beml\_df\_ci = stats.norm.interval(0.95,  loc = df1.gain.mean(),  scale = df1.gain.std())  print("Gain at 95% confidence interval in BEML is : ",np.round(beml\_df\_ci,4)) |
| --- |

stats.norm refers to the normal (Gaussian) distribution functions in the scipy.stats module.

.interval(0.95, loc=df2.gain.mean(), scale=df2.gain.std()) computes the 95% confidence interval for a normal distribution.

0.95 is the confidence level & alpha which is interval, meaning 95% confidence interval.

loc=df2.gain.mean() is the mean (average) of the 'gain' column in df2.

scale=df2.gain.std() is the standard deviation of the 'gain' column in df2.

Hypothesis Testing

It is an example of inferential testing

It is either to reject or retain a null hypothesis using data

Hypothesis testing consist of two complementary statements:-

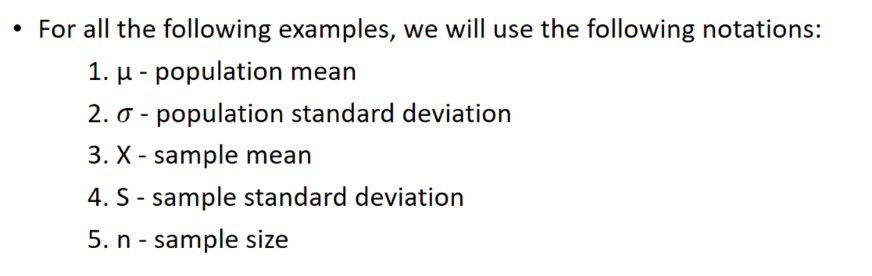
1. Null Hypothesis HO :- Existing Belief
2. Alternative Hypothesis HA :- What we want to establish with new evidence

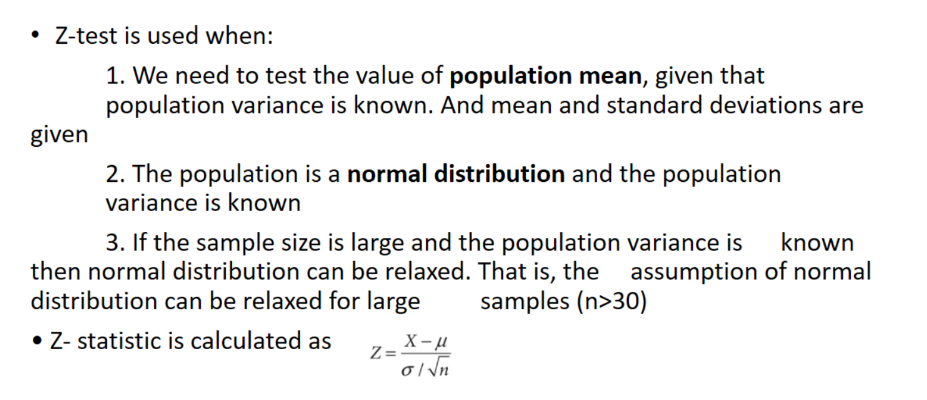
Types of Hypothesis Testing:-

1. Parametric Testing :- They make use of population parameters like mean & standard variation Ex- Ztest, Ttest & Anova
2. Non Parametric Testing :- They make use of data distribution to comment on the claim Ex- Chi-Square

Steps for Hypothesis Testing:-

1. Define Null or Alternative testing
2. Identify test statistic to be used for testing validity of null hypothesis
3. Decide the criteria for rejection or retention of null hypothesis. This is called Significant value
4. Calculate p-value
5. Take a decision to reject or not



Z Test

| #A passport office claims that the passport applications are processed #within 30days of submitting the application form and all  #40 passport applicants. The population standard deviation of the #processing time is 12.5 days. Conduct a hypothesis test at significance #level ∝ =0.05 to verify the claim made by the passport office.  import pandas as pd  passport\_df = pd.read\_csv("passport.csv")  passport\_df.head(5) |
| --- |

1. Null hypothesis = 30 days

ALternative hypothesis not equal to 30 days

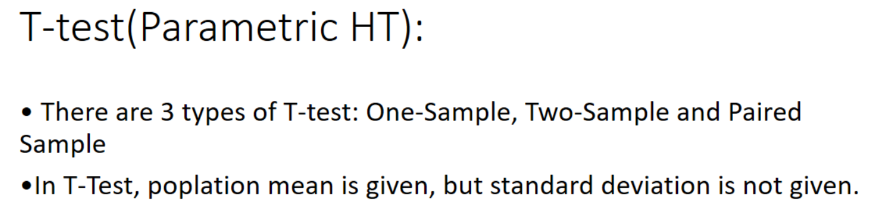
1. Z-Test

| import math  def z\_test(pop\_mean, pop\_std, sample):  z\_score = (sample.mean() - pop\_mean) / (pop\_std/math.sqrt(len(sample)))  return z\_score, stats.norm.cdf(z\_score) |
| --- |

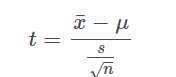
cdf stands for "Cumulative Distribution Function." The CDF of a distribution gives the probability that a random variable drawn from the distribution is less than or equal to a particular value

| z\_test(30,12.5,passport\_df.processing\_time)  #processing\_time is column sample it is containing |
| --- |

1. As p value is greater than ∝
2. Null hypothesis retained



T-Test 1sample



| #Aravind Productions (AP) is a newly formed movie production house based #out of Mumbai, India. AP was interested in understanding the production #cost required for producing Bollywood movie. The industry believes that #the production house will require INR 500 million on average. It is #assumed that the Bollywood movie production cost follows a normal #distribution. The production costs of 40 Bollywood movies in millions of #rupees are given in bollywoodmovies.csv file. Conduct and appropriate #hypothesis test at ∝ =0.05 to check whether the belief about average #production cost is correct.  bollywood\_movies\_df = pd.read\_csv('bollywoodmovies.csv')  bollywood\_movies\_df.head(5) |
| --- |

Population mean = 500

Sample (S) = 40

∝ =0.05

| from scipy import stats  stats.ttest\_1samp(bollywood\_movies\_df.production\_cost,500) |
| --- |

P-value < ∝

Null hypothesis is reject

T-Test 2 sample

| #A company claims that children who drink their health drink will grow #taller than the children who do not drink that health drink. Data in the #file healthdrink.xlsx shows average increase in height over one-year #period from two groups: one drinking the health drink and the other not #drinking the health drink. At ∝ =0.05, test whether the increase in #height for the children who drink the health drink is different than #those who do not drink health drink.  healthdrink\_yes\_df = pd.read\_excel("healthdrink.xlsx",'healthdrink\_yes')  healthdrink\_yes\_df.head(5) |
| --- |

| healthdrink\_no\_df = pd.read\_excel("healthdrink.xlsx",'healthdrink\_no')  healthdrink\_no\_df.head(5) |
| --- |

| stats.ttest\_ind(healthdrink\_yes\_df['height\_increase'],healthdrink\_no\_df['height\_increase']) |
| --- |

Paired sample T-Test

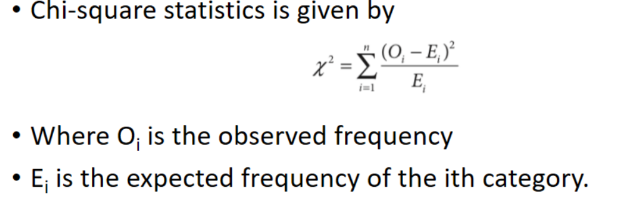
| #The file breakup.csv contains alcohol consumption before and after #breakup. Conduct a paired t-test to check whether the alcohol #consumption is same after the breakup at 95% confidence(∝ =0.05).  breakups\_df = pd.read\_csv('breakups.csv')  breakups\_df.head(5) |
| --- |

| import matplotlib.pyplot as plt  import seaborn as sn  sn.distplot(breakups\_df['Before\_Breakup'],label='Before\_Breakup')  sn.distplot(breakups\_df['After\_Breakup'],label='After\_Breakup')  plt.legend() |
| --- |

| stats.ttest\_rel(breakups\_df['Before\_Breakup'],breakups\_df['After\_Breakup']) |
| --- |

**Non Parametric Testing**

**Chi-Square**

****

| #Chi-Square Square Goodness for fit test  #Ques)Hanuman Airlines (HA) operated daily flights to several Indian #cities. One of the problems HA faces is the food preferences by the #passengers. Captain Cook, the operations manager of HA, believes that #35% of their passengers prefer vegetarian food, 40% prefer #non-vegetarian food, 20% low calorie food, and 5% request for diabetic #food. A sample of 500 passengers was chosen to analyze the food #preferences and the observed frequencies are as follows:  #1. Vegetarian: 190  #2. Non-vegetarian: 185  #3. Low-calorie: 90  #4. Diabetic: 35  #Conduct a chi-square test to check whether Captain Cook's belief is true #at ∝=0.05.  f\_obs = [190,185,90,35]  f\_exp = [500\*0.35,500\*0.4,500\*0.2,500\*0.05]  print(f\_exp) |
| --- |

| from scipy import stats  stats.chisquare(f\_obs,f\_exp) |
| --- |

**Anova (Analysis of Variance)**

| #Ques) Ms Rachael Khanna the brand manager of ENZO detergent powder at #the "one-stop" retail was interested in understanding whether the price #discounts have any impact on the sales quantity of ENZO. To test whether #the price discounts had any impact, price discounts of 0%, 10%, and 20% #were given on randomly selected days. The quantity of ENZO sold in a day #under different discount levels is shown in Table 3.1. Conduct a one way #ANOVA to check whether discount had any significant impact on the #average sales quantity at ∝ = 0.05  import pandas as pd  onestop\_df = pd.read\_csv("onestop.csv")  onestop\_df.head(5) |
| --- |

| import matplotlib.pyplot as plt  import seaborn as sn  sn.distplot(onestop\_df['discount\_0'],label ='No Discount' )  sn.distplot(onestop\_df['discount\_10'],label ='10% Discount' )  sn.distplot(onestop\_df['discount\_20'],label ='20% Discount' )  plt.legend() |
| --- |

| from scipy.stats import f\_oneway  f\_oneway(onestop\_df['discount\_0'],  onestop\_df['discount\_10'],  onestop\_df['discount\_20']) |
| --- |

| Symbol | Explanation |
| --- | --- |
| Σ | Simulation |
| σ | Standard Deviation |
| ⎷ | Radical Symbol |