python for data science.

- Pandas
- Matplot lib
- Numpy
- Seaborn.

- dataframe comprises of multiple columns.
- series comprises of single columns.

Preferences > IPython console > Graphis > Backend = automatic

A series is used to model one dimensional data, similar to list in python. The series object also have a few more bits of data including an index and a name.

- To Import excel file.
  Import pandas as pd
  f1 = pd.read_excel(" directory").

- To Import csv file
  pd.read_csv (" directory)

⊙ difference bet" list, series and array.

⊙ Series.

```
Import pandas as pd
song = pd.Series([12, 11, 10, 9], name = "count")
song.Index
```

o/p:

Index	count
0	12
1	11
2	10
3	9

```
george = pd.Series([10,7,1,22],
          Index=[1968,1982,1945,1933],
          name = 'george songs')
george        # display series

george[1968] # To read or select item.

george[1968]=63  # To update value.

del george[1]  # To delete item.
george.mean()  # display mean.
```

⊙ arrays.

```
Import numpy as np.
num_ser = np.array([145,141,39,9])
num_ser[1]  # accessing array
num_ser.mean() # display mean.
```

o/p: george (series)

Index	george song
1968	68
1952	7
1945	1
1933	22

o/p: numser (array)

	0
0	145
1	141
2	38
3	9

* Convert types.
- string : .astype(str)
- numeric : pd.to_numeric
- Integer : .astype(int)
- datetime : pd.to_datetime

Songs_66 = pd.Series([3, None, 11, 9],
Index = ["george", "ringo", "John", "Paul"],
name = "counts")

Songs_66.dtypes. # display datatypes. float'64'

Pd.to_numeric(Songs_66.apply(str))
# 1st we want to convert in str and then in numeric It will display error.

to ignore the error use errors = 'coerce'

pd.to_numeric(songs_66.astype(str), errors = 'coerce').

Songs_66.dtypes

To deal with none use .fillna.

songs_66.fillna(-1)

# nan values will be droped using .dropna (not good practice)
Songs_66.dropna()

o/p : float64
' error
• (displays series) dtype : float64
• dtype ('float64')
• george 3.6
ringo -1.0
John 11.0
Paul 9.6
Name : counts, dtype : float64
• (drops ringo & -1.0).

* Combining 2 series.

Songs = songs_66.append(songs_69)

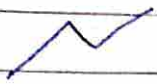
o/p : After prints series 1 it will print 2nd series vertically.

Page No.
Date

* Plot

```
Import matplotlib.pyplot as plt

fig = plt.figure # open canvas
songs_69.plot() OR plot (kind = "bar", color = 'r')
plt.legend()
```



* Pandas Data Frame

- It is two dimensional datastructure.
- It is Immutable, heterogeneous tabular
- data structure with labeled axes rows, and columns.

To ~~upra~~ upgrade pandas
conda install --upgrade pandas.
conda install -C anaconda pandas.

Conda Install pandas == 2.0.3
update

# To check pandas version (on thespyder).
import pandas as pd
pd.__version__

* Creating dataframe using list of list.

```
Import pandas as pd

technologies = [["Sparks", 2000, "30days"],
                ["pandas", 2000, "40days"]]
pf=pd.DataFrame (technologies)
```

o/p

Index	0	1	2
0	sparks	2000	30days
1	pandas	2008	40days.

Since we have not given labels to colums and ~~rows~~ indexes, Dataframe by default assigns Incremental sequence numbers as labels to both rows and columns, these are called Index.

```
Colume_name = ["course", "fee", "duration"]
row_label = ["a", "b"]
df = pd.Dataframe (technologies, columns = colum-
                   name, Index = row_label).
print df
```

o/p:

Index	courses	fee	duration
a	Sparks	20000	30 days
b	pandas	2000	40 "

* Creating dataframe using dictionary.

```
technologies = {
    "courses": ["spark's", "pysparks", "hadoop", "python"],
    "fee" : [2000, 2500, 3000, 4200],
    "duration : ["30days", "40 days", "25days", "50days"]
    }
pd = df.DataFrame(technologies)
print (df.dtypes)
```

o/p : (Dataframe)

courses	object
fee	Int64
duration	object

* Convert object into string.

```
df2 = df.convert_dtypes()
print (df.dtypes
```

o/p :

course	string
fee	Int64
duration	string

* Convert all column to object.

```
df = df.astype(str)
print (df.dtypes)
```

o/p:

Course	object
fee	"
duration	"

* Change specific column

```
df = df.astype({"fee": int, "discount": float})
print(df.dtypes)
```

o/p:

* convert all datatypes for all column in list.

```
pf = pd.DataFrame(technologies)
cols = ["fee", "discount"]
df[cols] = df[cols].astype("float")
df.dtypes.
```

o/p: ~~living~~ leaving object all will be float.

* Ignore error

when we will try to convert object into int.

datatype will not change but error will be ignored.

```
df = df.astype({"courses": int}, errors = "ignore")
df.dtypes.
```

* generate error

```
df = df.astype({"course": int}, errors = "raise")
df.dtypes.
```

o/p: error.

* Convert dataframe to csv.

```
Import pandas a pd.

technologies = {
    "courses" :["sparks", "pysparks", "hadoop",
                "python"],
    "fee" :[2000, 2500, 4000, 2500],
    "duration:["30days","40days", "35days","20days"]
    }
df = pd.DataFrame(technologies)
df      # displays table in kernal.

df.to_csv('data_file.csv')
# Convert dataframe to csv.
```

Exploratory Data Analysis (EDA).

```
df.shape
# (8,4)  (rows, columns)

df.size
# 32  (rows x columns)

df.columns / df.columns.values
# Index(['courses', 'fee', 'duration'], dtype='object')
array.

df.index
# Index(['r0', 'r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7'], dtype =
        'object').
```

* accessing one column

df['fee']

* accessing two column or more.

df[['fee', 'discount']]

* accessing specific rows, column.

df.iloc[rows, column]

df = df.iloc[1:3, 0:2]

df = df.iloc[:, 0:2] all row & 0,1 column

* accessing one column contents

df['fee'] ~~##~~

* accessing Individual cell.

df['duration'][3]

* Substracting specific number from column

df['fee'] = df['fee'] - 500

df['fee']

Page No.
Date

* shows 5 no. summary Describe()

```
df.describe()
```

* changing column names.

```
df.columns = ["A", "B", "C", "D"]
df
```

```
rows = axis=0
column = axis=1
```

*
```
df2 = df.rename({'A':'C1', 'B':'C2'}, axis=1)
df2 = df.rename({'C':'C3', 'D':'C4'}, axis='columns')
df2 = df.rename(columns={'A':'C1', 'B':'C2'})
```

** To convert series into list.

```
Import pandas as pd.
ds = pd.Series([1, 2, 3])
print(ds)   # series

print(ds.tolist())   # converted into list
print(type(ds.tolist()))   # <class 'list'>
```

* Tuples, list & dict can be converted into series.
* Numpy array can be converted into series.

```
import numpy as np
ary = np.array([1, 4, 9, 8])
ds1 = pd.Series(ary)
print(ds1)
```

Page No.
Date

* Write pandas program to convert one column as series.

```
d = {"col1" : [1, 2, 3],
     "col2" : [4, 5, 6],
     "col3" : [7, 8, 9]
    }
df = pd.DataFrame(data=d)
print(df) # display dataframe.

ds1 = df.iloc[:, 0] # all row 0th column.
print(ds1) # displays series.
```

⊙ stack()

It is used to stack the prescribed level(s) from column to index.

It converts multilevel index into single.

```
eg. a = [[1, 2], [3, 4]]
    .stack() = [1, 2, 3, 4].
```

```
eg:
s = pd.Series([
    [1, 2],
    [3, 4, 5],
    [6, 7]
    ])
print("original Series", s)
print("single Series")
s = s.apply(pd.Series).stack().reset_index(
                                drop = True)
```

o/p:

0	[1, 2]
1	[3, 4, 5]
2	[6, 7]

reset_index will reset index upto expected value,
drop = True means drop the position index

0	1
1	2
2	3
3	4
4	5
5	6
6	7

```
# S = S.apply(pd.Series)
S = S.stack()
S = reset_index(drop=True)
```

* Add existing series to the series.

```
S1 = pd.Series([2,3,4])
print(S1)
news1 = pd.concat([S1, pd.Series(["Hi", 5])], ignore_
                                    index = True)
print(news1)
```

o/p:

0	2
1	3
2	4

0	2
1	3
2	4
3	Hi
4	5

* Sort the series

```
S1 = Pd.Series(["1", "2", "Python", "0"])
SS1 = Pd.Series(S1).sort_values()
print(SS1)
```

```
o/p: 0  0
     1  1
     2  2
     3  Python.
```

* Reindexing Series

```
S1 = Pd.Series(data=[1, 2, 3],
               index=["A", "B", "C"])

S1 = S1.reindex(index=["B", "C", "A"])
print(S1)
```

```
o/p: B  2
     C  3
     A  1.
```

* dropping rows using row labels.

```
df = df.drop['r1"].

# r1 row will be deleted.
```

```
df = df.drop(df.index[1]) # 1st row deleted.
     (df.Index[[2,5]]) # 2nd, 5th row deleted
     (df.Index[4:]) # from 4th deleted.
          [2:5] # 2-4 deleted.
     df.drop(0) # 0th Index deleted
     df.drop([1,4]) # 1, 4 deleted
     df.drop(range(2,5)) # 2-4
```

Droping Column.

~~Mant~~ Mandatory to write axis=1

```
df = df.drop(["courses"], axis=1)
     (labels=["courses"], axis=1)
     (columns=["fee"], axis=1)
```

When we are making changes in same dataframe it is mandatory to use Inplace = True

eg: df.drop(df.column[1], axis=1, Inplace = True)

- To delete multiple columns

```
df = df.drop(df.columns[[1,2]], axis = 1)
   # It will delete columns Index 1,2.

litcol = ["courses", "fee"]
df = df.drop(litcol, axis=1)
```

df.iloc[[',3],[1,2]] # row, column

* Iloc (iloc)

df.iloc [start row : end row, start column : end column]

df = df.iloc[1:3, 0:2] # 1 & 2 row, 0,1 column

#

df = df.iloc [0:2, :] # 0,1 row, all column

df = df.iloc [2] # 2nd Index row (series)

df = df.iloc [[2,3,6]] # It will select 2nd, 3rd, 6th Index

df = df.iloc [1:5] # It will select 1st to 4th Index

df = df.iloc [:2] # 0th, 1th row

df = df.iloc[-2:] # -1th & -2th row

df = df.iloc[:-3] # all rows till -2th Index.

df = df.iloc[::3] # all rows in step of 3.

* loc

df = df.loc['r1'] # display 'r1' row Series.

df = df.loc["r1", "r2"] # display "r1", "r2" row

df = df.loc['r1':'r7':2] # display r1 to r7 in step of 2

df = df.loc ['r1':'r4'] # display from r1 to r4 including r4

df = df.loc["r0":"r5", ["course", "fee"]]

df = df.loc ["r0", 'fee':'discount'] # Series

df = df.loc[:, "duration":] # all rows & all columns from duration.

* To Add columns.

tutors = ['ram', 'sham', 'ghansham', 'radhesham']
df2 = df.assign(Tutor_assigned = tutors)
column name values.

* Add multiple columns

mncompany = ['tata', 'Infosys', 'amazon', 'google', 'TCS']
df2 = df.assign(Tutors = tutors, Company = mncompany).

* Derive new columns from existing columns

df2 = df.assign(per = lambda x: x.fee * x.disc/100)

* Add column to existing columns.

df['company'] = mnc company.

* Add new column to specific position

df.insert(0, 'tutor', tutors).

* To Add row

df.loc["f"] = ["avanti", 20, 3, "no"].
or
index

* To find no. of column & rows in DF

rows_count = len(df.index) or df.shape[0]
= len(df.axes[0])
column_count = len(df.axes[1]) or df.shape[1]

Assignment

* no. of attempts must be less than 2.

```
print (df[df['attempt']>2])

df2 = df.loc[df.attempt>2]
```

* To find null value.

```
print(df[df["score"].isnull()])
print(df.loc[df.score.isnull()])
```

* Select Score bet^n 15 to 20

```
df2 = df[df['score'].between(15,20)]
df2 = df.loc[df.score.between(15,20)]
```

* no. of attempts is <2 & score >15

```
print(df[(df['attempt']<2) & (df['score']>15)])
print(df.loc[(df.attempt<2) & (df.score>15)])
```

* To change specific value in Dataframe.

```
df.loc["d","score"] = 11.5
```

* Sum(), mean()

```
print(df["attempts"].sum())
print(df["attempts"].mean())
```

* Sort in ascending, descending

```
df = df.sort_values(by=["column name"], ascending=[False])
# sorted in descending order
df = df.sort_values[by=["column"], ascending=[True])
```

* Replace values in DF map()

```
df["column name"] = df["column name"].map({"Hi": "Hello",
                                           "By": "Bye"})
```

```
df["column name"] = df["column name"].replace("Hi", "Hello")
```

* Iterate over rows

```
for index, row in df.iterrows():
    print(row["name"], row["score"])
```

* While renaming the values in column, rename all the values else it will be replace by nan.

apply()

```
data = {"A": [1,2],
        "B": [3,4]
       }
df = pd.DataFrame(data)
df

def add_3(x):
    return x+3

df2 = df.apply(add_3)  # all column
or
df2 = ((df.A).apply(add_3)  # single column

df["B"] = df["B"].apply(add_3) # Single column

#To apply on multiple column list of column is mandatory to pass
df[["A","B"]] = df[["A","B"]].apply(add_3)
df                  # multiple column

# Using lamda function on all column

df2 = df.apply(lamda x : x+10)
df2

# Using lambda fun on multiple columns

df["A","B"] = df["A","B"].apply(lambda x : x+10)
df
```

* Transform() Instead of apply.

```
df = df.transform(add_5)
df
```

* Map() same as apply & Transform.

* Using numpy function

```
import numpy as np
df["A"] = df["A"].apply(np.square)
print(df
```

* groupby()

Single column.

```
df2 = df.groupby(["courses"]).sum()
print(df2)        # Not have Index

df2 = df.groupby(["courses"]).sum().reset_index()

# on multiple columns
df2 = df.groupby(["courses", "duration"]).sum()
                        # with Index.
```

* Shuffle

shuffle DF rows.

```
df1 = df.sample(frac=1)  # 100% Shuffle.
df1 = df.sample(frac=0.5) # 50% shuffle.
```

#

drop = True

If we set drop = True, reset index will delete the index instead of inserting it back in column.

```
df1 = df.sample(frac=1).reset_index(drop=True)
```

* Joining DF

```
tech2 = {
    "courses": ["cpp", "Java", "Python"]
    "fee": [100, 200, 300]
    "duration": ["30 days", "10 days", "20days"]
    }

index1 = ["r1", "r2", "r3"]

df1 = pd.DataFrame(tech2, index1)

tech3 = {
    "courses": ["Cpp", "Java", "C"]
    "discount": [300, 400, 500]
    }
index1 = ["r1", "r5", "r3"]
df2 = pd.DataFrame(tech3, index)

# Left Join
df3 = df1.join(df2, lsuffix="left", rsuffix="right")
print(df3)
```

r1	Cpp	100	30days	300
r2	Java	200	10days	nan
r3	python	300	20days	nan

Inner Join

```
df3 = df1.join(df2, lsuffix="left", rsuffix="right",
                how="inner")
```

o/p:

r1	Cpp	100	30days	300

Right Join.

```
how = "right"
```

o/p:

r1	Cpp	300	100	30days,
r2	Java	nan	nan	nan
r3	C	nan	nan	nan

Joining according to columns

~~If~~ Index don't matter.

```
df3 = df1.set_Index("courses").join(df2.set_Index
      ("courses"), how = "inner")
```

o/p.

Cpp	100	30days	300
Java	200	10days	400

* Merge() = Inner Join

df3 = pd.merge (df2, df3)
df3 = df1.merge(df2)

* Concat()
Vertically joins all dataframe.

data = (df1, df2)
df3 = pd.concat (data)
df3.

Numpy

* Create an array.

```
Import numpy as np
arr = np.array([10,20,30])  # 1 dimension
print(arr)
```

* Create multidimensional array.

```
arr = np.array[[10,20,30],[40,50,60]] # 2dimension.
```

* ndmin - to specify how many dimensions you need.

```
arr = np.array([10,20,30], ndmin=3)
o/p: [[[10 20 30]]]
```

* To change datatype.

```
arr = np.array([10,20,30], dtype=complex)
o/p: [10.+0.j 20.+0.j 30.+0.j]
```

* To get dimension of array.

```
arr = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(arr.ndim)
      o/p: 2
```

* to get array type, size, shape.

```
arr.dtype    # int32        ([10,20,30],
arr.size     # 6             [4,5,6])
arr.shape    # (2,3) (row, column)
```

* create sequence using range fu"

```
arr = np.arange(0,20,3)
print(arr)
```

* accessing array using Index.

```
arr = np.arange(11)
print(arr)
print(arr[2])
# 2
print(arr[-2])
# 9.
```

* accessing multidimensional array.

```
arr = np.array([[10,20,30],[40,50,60]])
print(arr[1,2]) #60
print(arr[0,2]) #30
print(arr[0,-1] #30

arr = ([1,2,3,4,5,6,7,8,9,0])
print(arr[1:8:2] # start:stop:step.
  # [2,4,6,8]
print [-2:3:-1] # -2 to 3 in step 1
  # [9,8,7,6,5]

print(arr[-2:11])
      # [9,0]
```

Page No.
Date

```
arr = ([[1,2,3],[4,5,6],[7,8,9]])

arr[1,2] = 6
arr[1,:] # one row all column
arr[:,2] # 2nd column represented as array.

arr[:3,1::2] # all rows till 3 & alternate
               columns.
```

* Integer array indexing, allows selection of arbitary items

```
arr = np.arange(35).reshape(5,7)  (row, column)
```

* boolen array Indexing.

```
rows = np.array([False, True, True]) # 0=false
                                       1,2 = True
wanted_rows = arr[rows, :] # 1st, 2nd row &
                             all column
```

* Convert array into list

```
array = np.array([1,2,3])
lst = array.tolist()
```

Multidimensional array → nested list.
1st, 2nd, 3rd row → 1st, 2nd, 3rd index of list.

* Convert list into array.

```
lst = [1,2,3]
array = np.array(lst) or np.asarray(lst)
print(array)
```

* resize array.

```
array = np.array([[1,2,3],[4,5,6]])
print(array.reshape(3,2))
```

```
array.shape = (3,2)
print(array.shape)
```

* find if any number is zero

① all()

```
Import numpy as np
x = np.array([1,2,3,4])
print(np.all(x))   # True

y = np.array([0,2,3])
print(np.all(y))   # false.
```

If array contains zero then o/p is false.

② any()

```
x = np.array([1,0,0])
print(np.any(x))
```

o/p: True

```
y = np.array([0,0,0])
print(np.any(y))
  o/p : false
```

True if any non zero number in array.

⑧ isfinite()

check element by element if is any not indefinite or nan value.

```
x = np.array([1,0,np.nan,np.inf])
print(np.isfinite(x))
  # o/p : [True True false False]
```

⑨ isnan()

```
print(np.isnan(x))
    op: [False, False, True, False]
```

⑩ Comparision element by element.

```
x = np.array([2,3])
y = np.array([1,4])
print (np.greater(x,y))
       (np.greater_equal(x,y))
       (np.less(x,y))
       (np.less_equal(x,y))
```

```
o/p : [True False]
      [True False]
      [False True]
      [False True.
```

* create 3x3 Identity matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

```
array = np.Identity(3)
```

* generate random number bet^n 0 to 1.

syntax: center of location, scale, size

```
rand_num = np.random.normal(0,1,1)
print(rand_num).
```

* create 3x4 ~~matrix~~ array & Iterate over it.

```
arr = np.arange(10,22).reshape(3,4)
print(arr)  # display array of 3,4
for x in np.nditer(arr):
    print(x, end=" ")
    print()
```

$$\begin{bmatrix} 10 & 11 & 12 & 13 \\ 14 & 15 & 16 & 17 \\ 18 & 19 & 20 & 21 \end{bmatrix}$$

* Vector creation of length 5 bet^n 10 to 50

```
v = np.linspace(10,49,5)
```

start, stop, length.

* reverse an array.

```
x = np.arange(0,10)  # 0 .. 9
x = x[::-1]
print(x)  # 9 .... 0
```

* dot product of 2 matrix

```
x = [[1,2],[3,2]]
y = [[3,2],[3,5]]
result = np.dot(x,y)
print(result)
```

o/p: 9 12
15 16

*

cross product

```
result1= np.cross(x,y)
result2= np.cross(y,x)
```

#result1 result2
[-4 9] [4 -9]

* To find determinant of given square array.

```
from numpy import linalg as la
a = np.array([[2,2],[3,1]])
print(np.linalg.det(a))
# o/p: -5.0000...1
```

* find eigen values & eigen vectors.

```
m= np.mat("3 -2;1 0")
print(m)
w,v= np.linalg.eig(m)
print("values, vectors:", w,v)
```

#o/p: [[3 -2]
[1 0]]
values =[2. 1.]
vector =[[0... 0...
0... 0...]]

* Inverse of matrix

```
result = np.linalg.inv(a)
print(result)
```

* Data science processes

1) Setting the research goal
2) Retriving data
3) Data preparation
4) Data exploration
5) Data modeling or model building
6) ~~preparat~~ presentation & automation

Matplotlib.

* Import Matplotlib.pyplot as plt
plt.plot([1,3,2,4])
plt.show()

* Multiple plot
x = range(6,5)
plt = plot(x, [xi*1.5 for xi in x])
plt = plot(x, [xi*3 for xi in x])
plt.show()

* Grid in graph.

Import matplotlib.pyplot as plt
Import numpy as np.
x = np.arange(1,5)

plt.plot(x, x*1.5, x, x*1.3, x, x/3.0)
plt.grid(True)
plt.show

* handling axes.

x = np.arange(1,5)
plt.plot(x, x*3, x, x*1.5, x, x/3.0)
plt.axis()
plt.axis([0,5,-1,13])
plt.grid(True)
plt.show()

* adding labels.

```
plt.plot([1,3,2,4])
plt.xlabel("This is x axis")
plt.ylabel("This is y axis")
plt.show()
```

* adding title

```
plt.plot([1,3,2])
plt.title("simple plot")
plt.show()
```

* adding legend

```
x = np.arange(1,5)
plt.plot(x, x*3, label="Normal")
plt.plot(x, x*1.5, label="Fast")
plt.plot(x, x/3.0, label="slow")
plt.legend()
plt.show()
```

* Color abbrevation

b	blue
c	cyan
g	green
k	black
m	magenta
r	red
w	white
y	yellow.

```
Import matplotlib.pyplot as plt
import numpy as np
y = np.arange(1,3)
plt.plot(y, "y")
plt.plot(y+1, "m")
plt.plot(y+2, "c")
plt.show()
```

* Specifying style in multiline plot.

```
y = np.arange(1,3)
plt.plot(y, "--", y+1, "-.", y+2, ":")
plt.show()
```

* Markers.

```
y = np.arange(1,3,0.2)
plt.plot(y, "x", y+0.5, "o", y+1, "D", y+1.5, "^", y+2,
plt.show()
```

* histogram charts.

```
Import matplotlib.pyplot as plt
import numpy as np
y = np.random.randn(1000)
plt.hist(y)
plt.show()
```

* Bar graph.

sytax: bar([x cordin], [y cordi])

```
plt.bar([1,2,3],[3,2,5])
plt.show()
```

* Scatter plots.
  displays values of 2 sets of ~~plot~~ data at x,y axis

```
Import matplotlib.pyplot as plt
Import numpy as np
x=np.random.randn(1000)
y= np.random.randn(1000)
plt.scatter(x,y)
plt.show()
```

* change size & color

```
size = 50 * np.random.randn(1000)
color = np.random.randn(1000)
plt.scatter(x, y, s=size, c=colour)
plt.show()
```

* adding text

```
x = np.linspace(-4, 4, 1024)
y = .25 * (x+4.) * (x+1.) * (x-2.)
plt.text(-0.5, -0.25, "brackmard minimum")
plt.plot(x, y, c='k')
```

Statistics

1) discriptive Statistics
2) Inferential Statistics

- discriptive is with processing of data without attempting to draw any Inferences from it.
- Inferential statistics uses mathematical tools to make ~~or~~ podcast and projection by analy-izing the ~~giving~~ given data.

* Population
The population offent consist of large group of specifically defined element.
ex: population of toon means that all the people living within that boundries of toon.
A small group of elements from the ~~grup~~ population is called sample.

* Data:
It is collection of numbers gather to give some Information.
Types of Data (depending upon source):
- primary data
- secondary data.

1) Primary data.
The data collected by experimentation.

2) Secondary data.
data collected from someone's experimental

* Variable
The ~~density~~ quantity/entity which can vary from one individual to another is called variable.

* Continuous Variable/data.
A variable which can assume each an every value within given range is called continuous variable/data.
e.g monitoring of temperature.

* discrete variable or data
A variable which can assume only some specific values within given range is called discrete variable or discrete data.
e.g students marks, population of village.

* Raw data.
The Data obtain from original source but not arranged properly is called raw data.

* Preprocess data./Tabular Data.
The process of placing classified data in tabular form is known as tabular data of processed data.

* Structured/Unstructural data.
Structured data is tabular data, represented in form of numerical values.

* Unstructured dl data is in original form and not numerical form.

* Range.
The difference between highest & lowest value of observation.

* Frequency of observation
The frequency of particular data value is number of times data value occurs.

* Frequency distribution
It is tabulation of values that one or more variables take in a sample.

DATA

Quantitative

Continuous
3.5, 4.06

discrete
→ 8

Qualitative

nominal
like, beautiful etc.

Ordi
order
:- good
bad
slow, fa

* Mean.

Name	monthly Income
Rob	5000
Rafiq	6000
Nina	4000
Sofia	7500
mohan	8000
Tao	7000

average = 6250

Name	$
Rob	5000
Rafiq	6000
Nina	4000
Sofia	7500
mohan	8000
Tao	7000
elonmusk	10 million

average : 1.43 million.

If ~~outrange~~ outlayer value gets added & mean is wrong.

* Median

x	y	z	a	b	c	d	e
4000	6000	6000	7000	8000	9000	10,000	11000

↓

7500

* Handing missing values

We can replace missing value with mean or median.

* Replacing missing values using mean values.

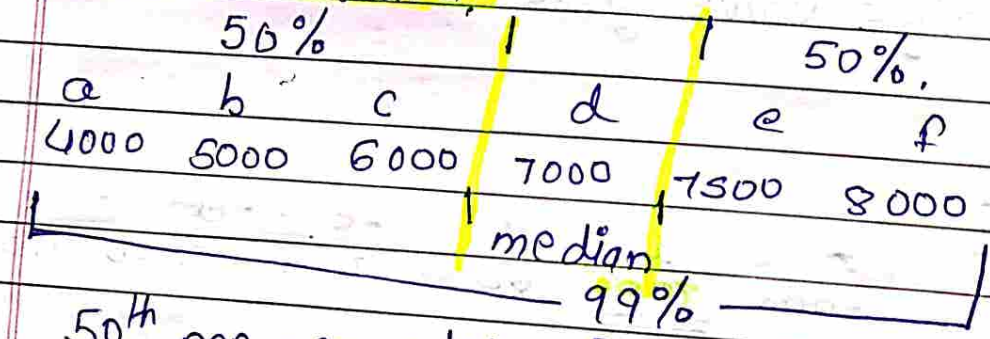
name	$	credit Score	Approv lone?
x	5000	650	No
y	6000	400	No
z	4000	780	Yes
a	1.6 million	810	Yes
b	8000	410	No
c	7000	850	Yes
d	10 million.	880	Yes.

* Replacing missing value with median.
6500 ≈ 1.6 million

* median used in descriptive analysis & Data cleaning (filling NA values)

* Outlayer is an extreme value of data in given dataset.

* Percentile value.



50th percentile for this dataset is 7000
0th percentile for this dataset is 4000.

* Mode
most frequently appearing value.

Name	Restaurant vote.
a	mexican
b	mexican
c	Italian
d	thai
e	mexican
f	India

Mode = Mexican.

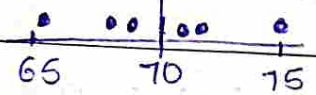
* Dealing with outlayer. (Mean/Median absolute Deviation) (MAD)

perfect example:
History Test.

name	score	Abs
a	75	5
b	72	2
c	68	2
d	65	5
e	67	3
f	73	2

(average = 70)

mean = 3.36.



here data (score) is near to mean value.

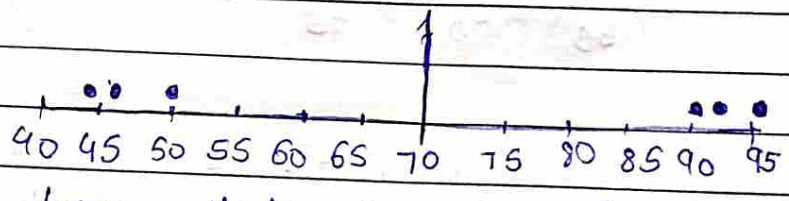
After calculating Absolute value: (score-avg)

The mean is 3.36 & deviation is less.

Not perfect:
Math Test

name	score	Abs (score-avg)
a	91	23
b	96	26
c	43	27
d	47	23
e	53	19
f	90	20

average = 70.
mean = 23



here all the scores are far from mean value :. model will be poor.

After Calculating Absolute value mean is 23 & deviation is less.

The Above was mean Absolute deviation (MAD)/MAR.

If there is still outlayer after Applying mean Absdute deviation then go for median absolute deviation

$\sigma = \sqrt{\frac{\sum (x_i - u)^2}{N}}$ standard deviation. $= \sqrt{\text{Var.}}$

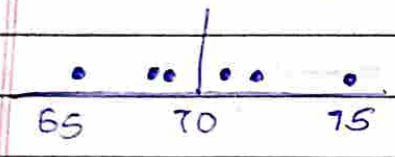
Page No.
Date

let us consider another table,

History Test.

Name	Score	Abs	(score-avg)^2
a	76	5	25
b	72	2	4
c	68	2	4
d	65	5	25
e	67	3	9
f	73	3	9
	MAD = 3.33		avg

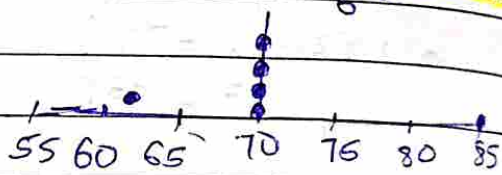
avg = 12.66 $\sqrt{\text{avg}} = 3.55$



here, after MAD the deviation spread is lower, which is good for model.

Name	Score	Abs	(Score-A
a	83	13	169
b	70	0	0
c	70	0	0
d	63	7	49
e	70	0	0
f	70	0	0
	MAD = 3.33		avg

avg = 36.33 $\sqrt{\text{avg}} = 6.01$



here, even after mean val the deviation spread is higher, which is not good allthough MAD is same for both table but it's standa deviation is more, It mean Is't spread is more.

so, If standard deviation is more than we can say spread is also more

* L1 norm & L2 norms

L1 norm that is calculated as the sum of absolute values of vector. L2 norm that is calculated as square root of sum of squared vector values.

mean absolute = 0
standard deviation = 0

Page No.
Date

* Transforming data on spread and centre.

What is the mean weight and what is standard deviation

$\bar{x} = 135.6 \quad \sigma = 31.75$

a	b	c	d	e
105	156	145	172	100

1) Transform data with addition.

a	105	+	5	= 110	$\bar{x}_{new} = \frac{a+b+c+d+e}{5}$
b	156	+	5	= 161	
c	145	+	5	= 150	$\bar{x}_{new} = 140.6$
d	172	+	5	= 177	$\sigma_{new} = 31.75$
e	100	+	5	= 106.	

When we are going to add same value in all data point then there is no transformation in given data set.

measure of centre Affected by:	measure of spread Affected by:
⊕ ⊖ ⊗ ⊕	⊗ ÷
(mean, median, mode)	(standard deviation, range)

Pie, Bar = ratio/categorical data

stamp plot, histogram = continuous (numeric) (quantitative)
time plot.

2) a 105 $\bar{x} = 135.6$ (X) 2.5 + 750
b 156
c 145 $\bar{x}_{new} = (135.6)(2.5) + 750 = 1089$
d 172
e 100 $s = 31.75$ (X) 2.5
$s_{new} = (31.75)(2.5) = 79.38$.

~~2.5ml of water & for every pound They~~

measure of centre

Centre new = (centre old) (X) + B

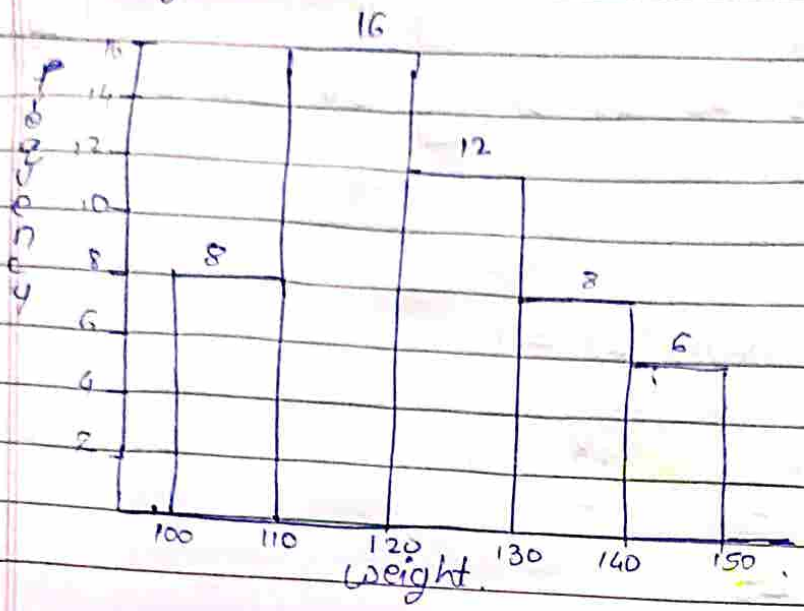
Spread new = (Spread old) (X)

* Impact of Outlayer on mean, median, Mode & Rang

a	b	c	d	e	f	g.
26.0	15.0	20.5	31.0	-350.0	31.0	30.5
				outlayer.		

measure		with outlayer	without outlayer
Affected	mean	-28	25.667
Resistant	median	26	28.25
Resistant	mode	31	31
Affected	range	381	16.
	↳ max - min		

* Histogram



* Frequency distribution Table:

weight	Frequency
100-110	8
110-120	16
120-130	12
130-140	8
140-150	6

Use of df.describe

1) 5 number summary
2) mimum, 1st Quartile, median, 3rd Quartile, maximum.

Quantile value 1st (Q_1)
Quantile value 3rd (Q_3)

4 8

10 11 12 25 25 27 31 33 34 34 35 36 4

Q_1 mean Q_3

left side of mean (L) = 7.

$\therefore Q_1 = \frac{L+1}{2} = \frac{8}{2} = 4.$

$\therefore Q_1 = 25.$

Right Side of mean (R) = L

$\therefore Q_3 = \frac{R+1}{2} = \frac{8}{2} = 4.$

$Q_3 = 36$

* Frequency distribution

weight	frequency	Cal
100-110	8	8÷50
110-120	16	16÷50
120-130	12	12÷50
130-140	8	8÷50
140-150	6	6÷50
	sum = 50	

Relative frequency distribution

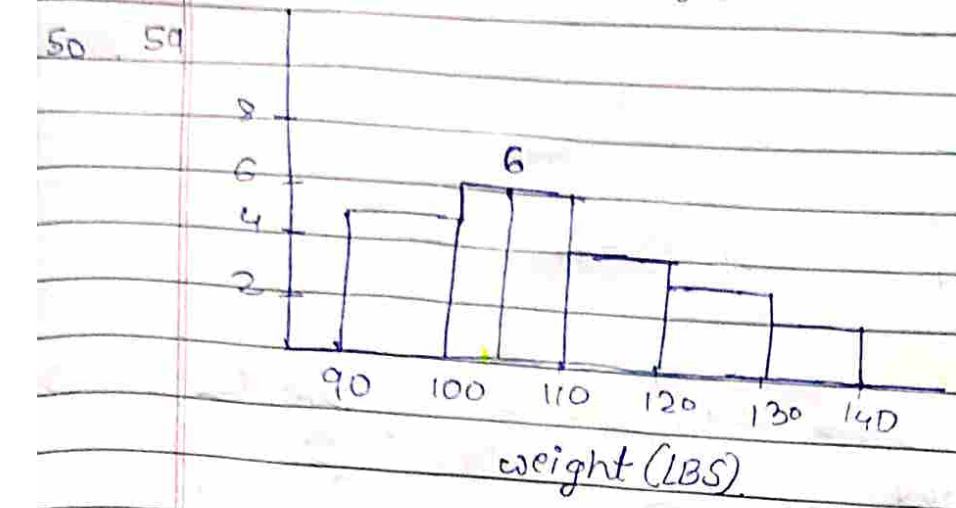
weight	relative frequency	relative freq (%)
100-110	0.16	16%
110-120	0.32	32%
120-130	0.24	24%
130-140	0.16	16%
140-150	0.12	12%
	sum = 1	100%

relative frequency = 8 ÷ sum

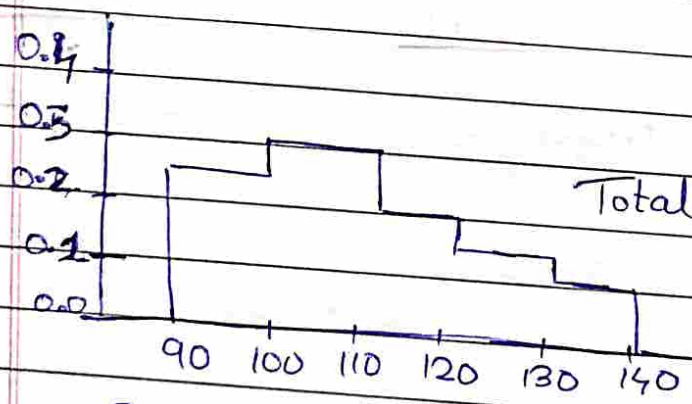
16 ÷ sum

Sum must be all ways 1 or 100%.

* 'Regular' frequency distribution ~~table~~ graph.

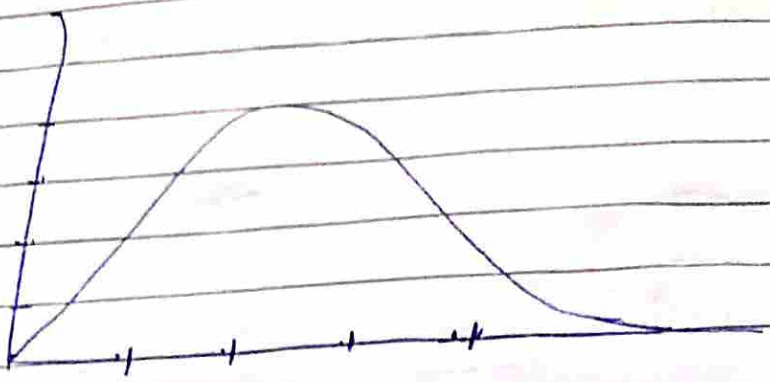


In regular frequency distribution, if we want corresponding values of weights ~~it will~~ b/n 100 & 110 you can easily plot that values as shown in fig. But This is not true with relative frequency distribution.



for example If we want to find corresponding weight in relative ~~fre~~ frequency distribution bet^n 100 & 110 then In that case we will have to find out area under the curve.

Relative ~~freque~~ frequency distribution is transformed in equivalent density curve

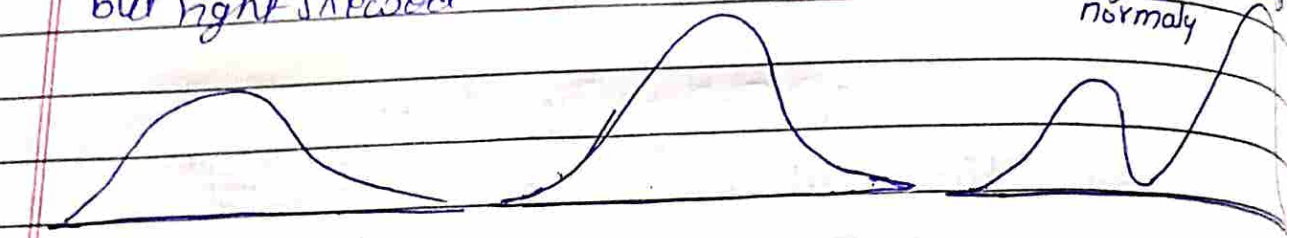


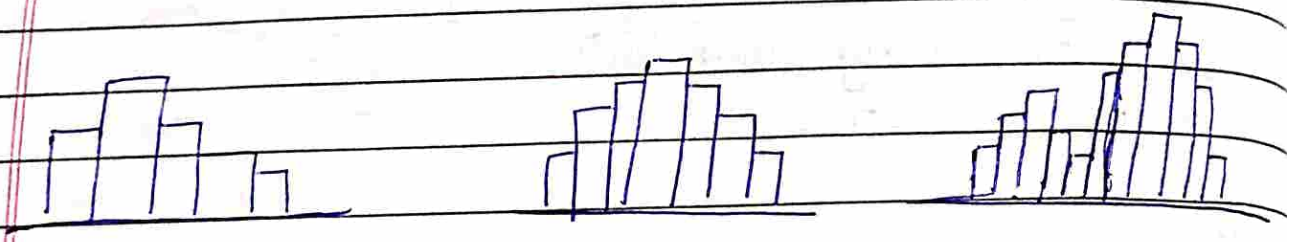
* Various forms of density curves.

Normaly distributed but right skewed

symetrically normal distributed

Bimodel -cally distributed normaly





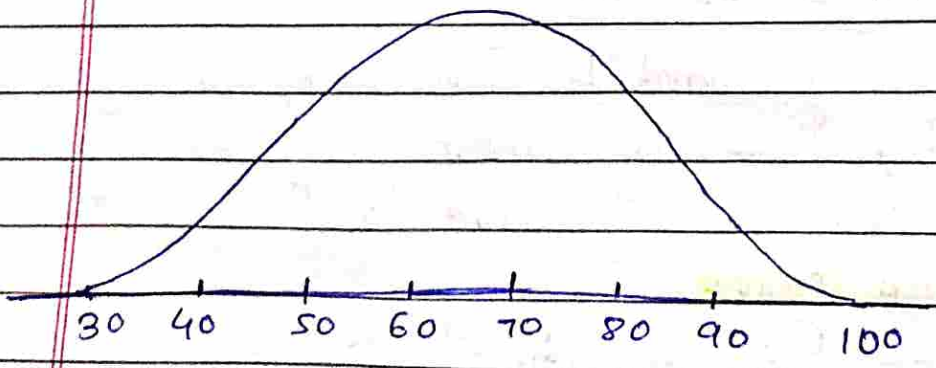
Total area = 1

Total area = 1

Total area = 1

* How to read density curves

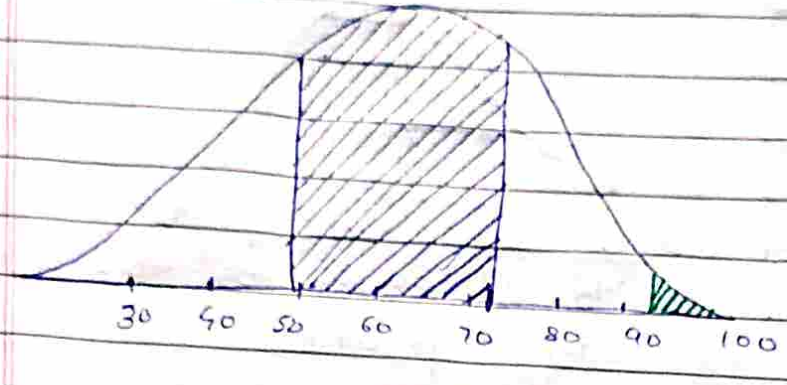
Test scores.
n = 1000000



If n (number of entries) are more than curve is defined) but if n is less then there are empty spaces in curves.

Page No.
Date

large majority of students scored score between 50 - 70



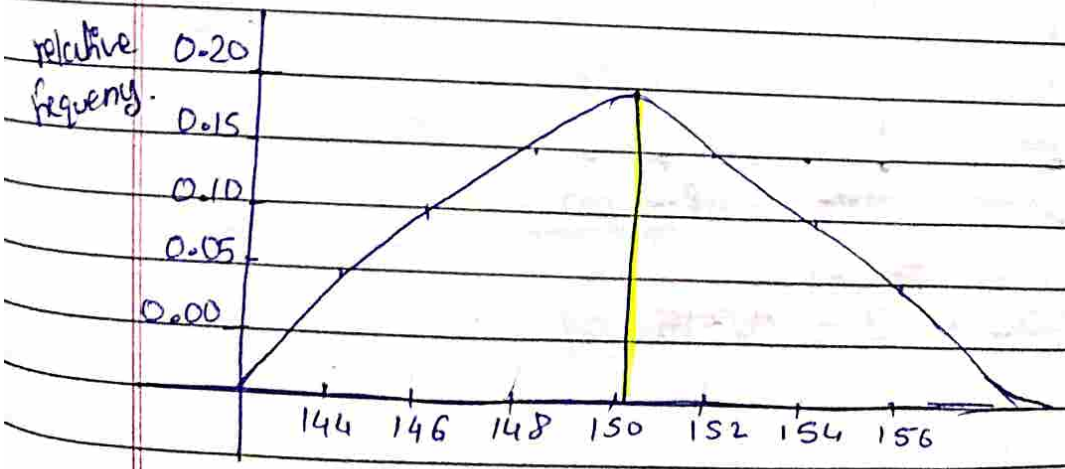
Very few student score bet^n & 95 to 100

* Properties of density curve.

1) A density curve must lie on x axis or above horizontal axis.

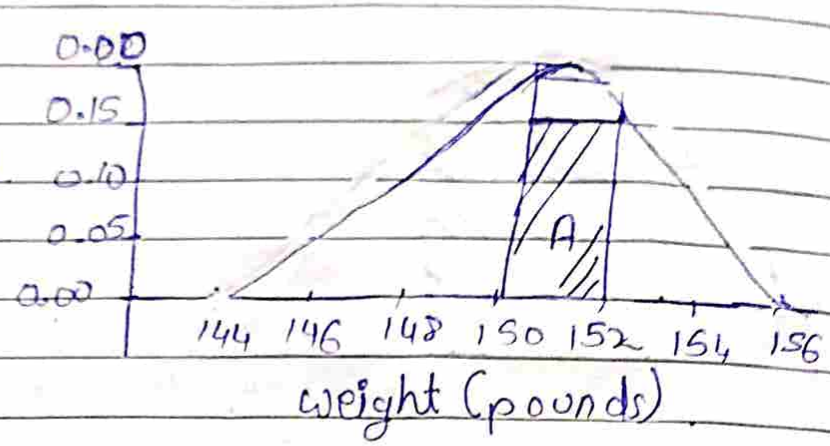
2) The total area under the curve is always 1.

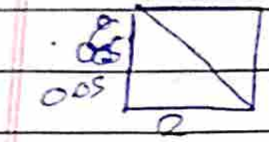
Q. for the density curve below, approx what is percentage of people weight exact 150 pounds?



area of line is 0 ∴ 0 people weight 150 pound.

* What percentage of people weight bet^n 150 & 152 pounds?



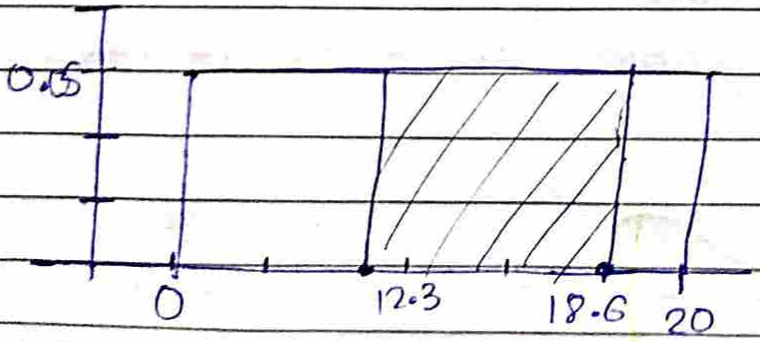


$A' = 1 \quad A/2 = 0.05$

$A \quad 2 \times 0.15 = 0.30.$

$A_{Total} = 0.30 + 0.05 =$

* For the ~~Area~~ uniform distribution below. what proportion of values are located bet^n 12.3 & 18.6?



$18.6 - 12.3 = 6.3.$

$A = 0.05 \times 6.3 = 0.315$ or 31.5%

Normal distribution/Gaussian distribution.

- Parameter
  A number that describes the data from a population.

- Statistics
  A number that describes the data from a sampale.

	Sample	population.
mean	$\bar{x}$	μ
tandard viation	s	σ
	Statistic	Parameter.

If σ is small then bell shape ~~curve~~ curve is taller (^)
If spread is higher than curve will be wider.

- The normal distribution is unimodal.
- The normal curve is symmetric about its mean
- The parameters μ and σ completely characteri-ze the normal distribution.

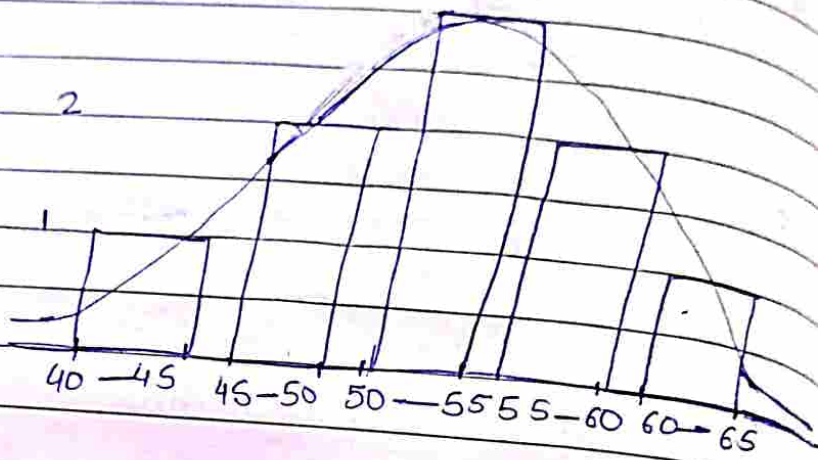
$X \sim N(\mu, \sigma)$

Variable — normal distribution. — mean

when data follows symmetric distribution normaly then means median are same.

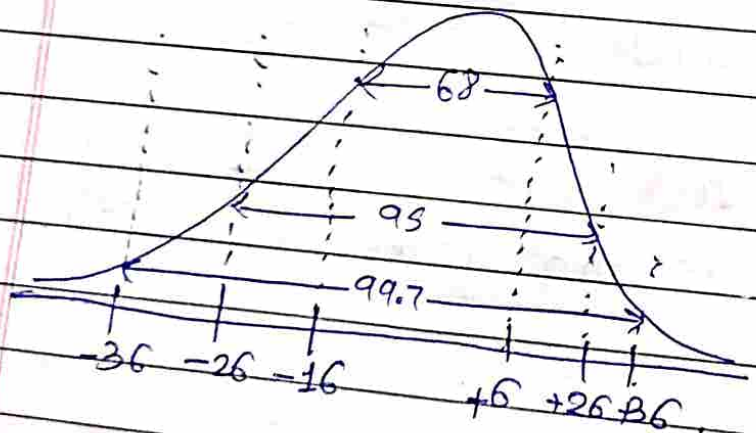
Name	height
a	6.2
b	5.7
c	4.6
d	5.4
e	5.9
f	4.3
g	5.1
h	5.2
i	4.9

count



* What formula we use to remove outliers with normal distribution.

68-95-99.7 RULE



* normal distribution (file).

54.2631 63.50 66.36 69.17 78.99

min Q1 mean Q3 max.

* To eliminate the some outlayer.

1) Z score.

$$\text{ZSCORE} = \frac{x - \text{mean}(\mu)}{\text{standard deviation}(\sigma)}$$

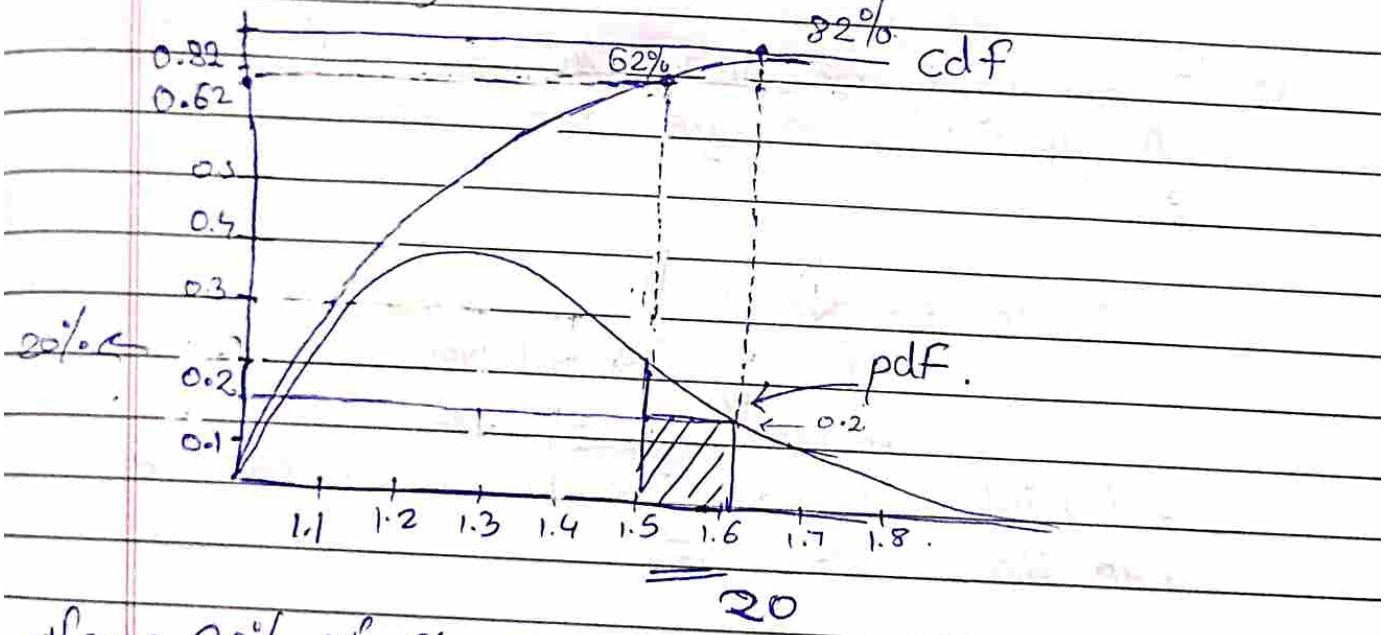
* Cummulative distribution (CDF)

Cummulative distribution function (CDF) for Random variable (x) is denoted by.

$$F_x(x) = p(X \leq x)$$

$f(-\infty) = 0$

$f(+\infty) = 1$



pdf says. 20% of flower having length 1.5 to 1.6.

- 82% of setosa flower having petal length <= 1.6
- There are 62% of setosa flower having petal length <= 1.6
- 30% of setosa flower having petal length having length <= 1.1

cdf Calculation

1) There are total 50 setosa flowers & there are total 41 flowers having length =< 1.6.

$\frac{41}{50} = 0.82\%$

2) You have histogram of setosa flowers cdf value of corresponding of 1.6 will be cummulative sum of 1.1, 1.2, 1.3, 1.4, 1.5, 1.6 histogram.

Q. If We want to find the count of variable In perticular range then we use pdf or cdf

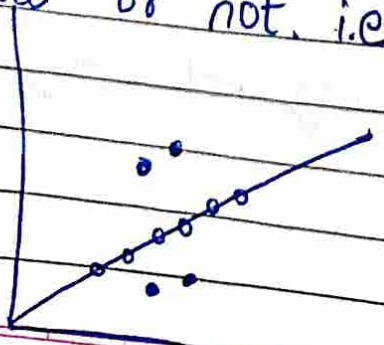
Variable ~ N(μ, σ)
- μ: mean
- σ: std deviation.

normaly distibuted

If Variable is Gussian distibuted then we can find CDF & PDF.

* Summary:-

To understand any dataset we should have Q-Q plot of that dataset. from Q-Q plot we can check whether the data is normally distributed or not. i.e



- Normally distributed - ma points lie on the s line only few of them lie outside line.

- find cdf & pdf from Q-Q plot

* **Z Score :-** (Standard Normal Distribution)

$$Z = \frac{x - \mu}{\sigma}$$

How many standard deviation away a datapoint is from mean.

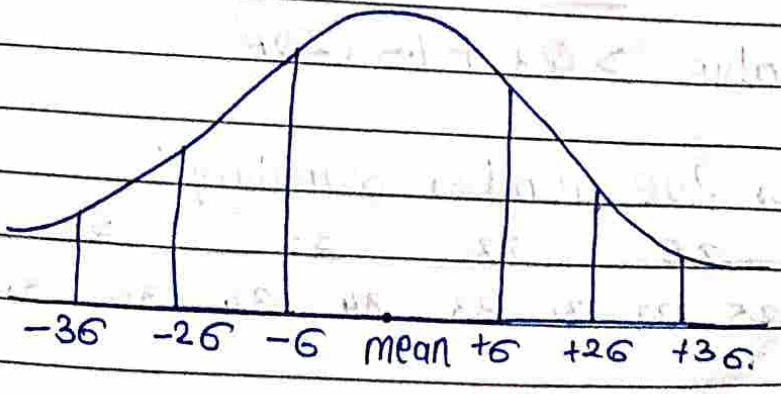
Name	Height.	Z score $= (x-\mu)/\sigma$
A	6.2	$(6.2-5.25)/0.61 = 1.53$
B	5.7	$(5.7-5.25)/0.61 = 0.72$
C	4.6	$(4.6-5.25)/0.61 = 1.06$
D	5.4	$(5.4-5.25)/0.61 = 0.23$
E	5.9	$(5.9-5.25)/0.61 = 1.04$
F	4.3	$(4.3-5.25)/0.61 = 1.55$
G	5.1	$(5.1-5.25)/0.61 = 0.25$
H	5.2	$(5.2-5.25)/0.61 = 0.09$
I	4.9	$(4.9-5.25)/0.61 = 0.58$

average = 5.25

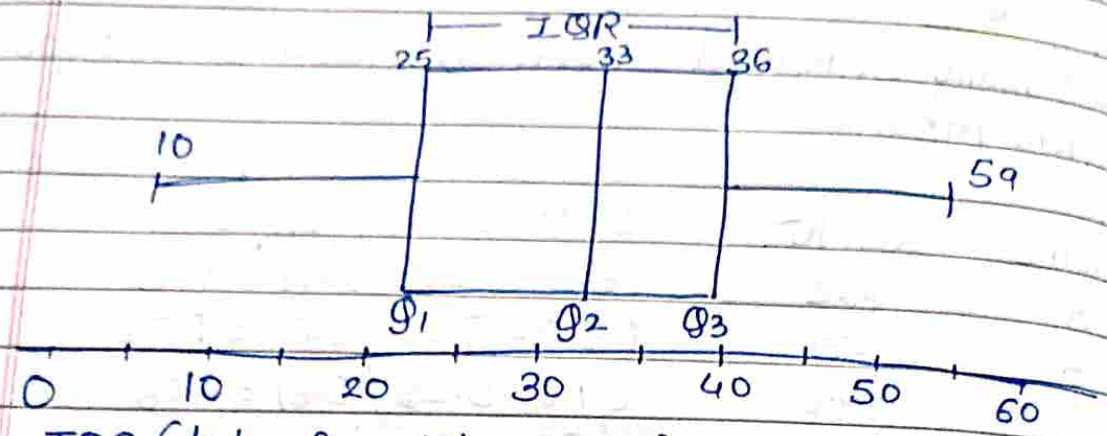
standard deviation = 0.61

$$Z = \frac{x - \mu}{\sigma}.$$

- Z score is also called as standard Normal Distribution.

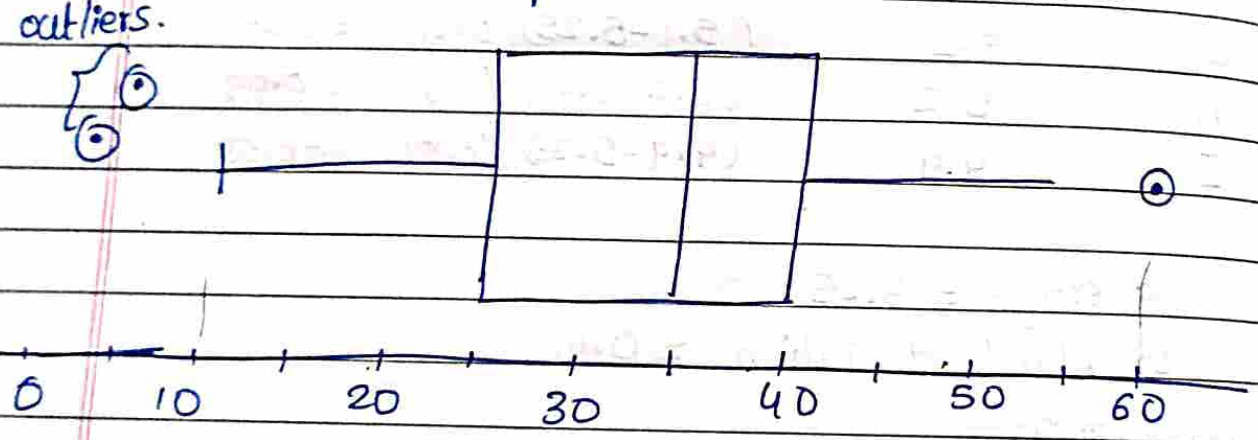


* Box plot :
Gives us a visual representation of the fi
number summary.



IQR (Inter Quartile Range) = $Q_3 - Q_1$

* Modified Boxplot



- A data value is considered to be an outlier if
Data Value $< Q_1 - 1.5(IQR)$
OR
Data Value $> Q_3 + 1.5(IQR)$

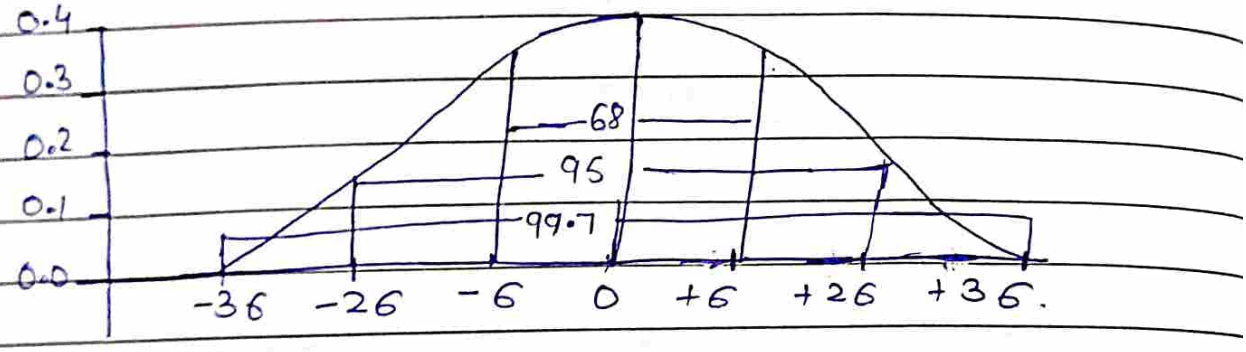
* Example for five number summary :

10 25 33 36 59

10 11 25 25 27 31 33 34 34 35 36 43

* Chebyshev's Inequality Eqn.

We know that If data is normally distributed we have 68, 95, 99.7 Rule i.e 68% of data points lie in the Range -1σ to 1σ & 95% of data points lie in the Range -2σ to $+2\sigma$ & Similarly 99.7% data point lie in the Range -3σ to $+3\sigma$ as shown in the fig.



Q let us assume height of students, data follows normal distribution.

here x follows normal distribution given that μ & σ

given that μ is 150cm & σ is 10cm. then probability of

~~$P(\mu-\sigma \leq x \leq \mu+2\sigma) = 95\%$~~

$\therefore P(\mu-2\sigma \leq x \leq \mu+2\sigma) = 95\%$.

[130,170]

This is posible bcoz we know that data is normally distributed. What if we don't know the distribution of data but we know μ & σ where μ is finite & σ is nonzero & finite. can we find % of data betn the range

If we don't know distribution like normal distribution we did?

In this case chebyshev's inequality equation will help to solve this problem.

Chebyshev inequality equation is given by

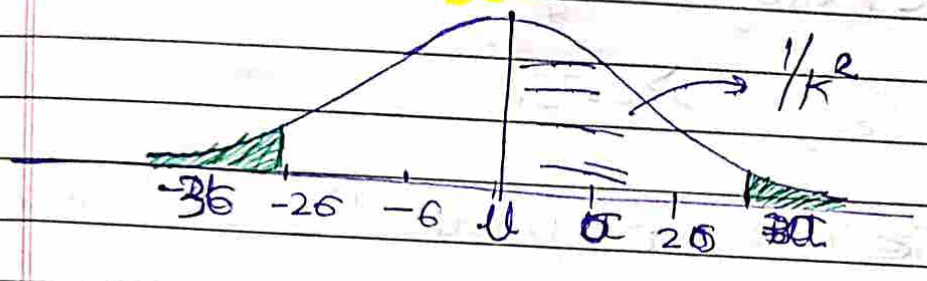
$$P(|x-\mu| \geq k\sigma) \leq 1/k^2$$

where $k = -1, -2, -3, 1, 2, 3$

This eqn can be further simplified

$x \geq \mu + k\sigma$ &

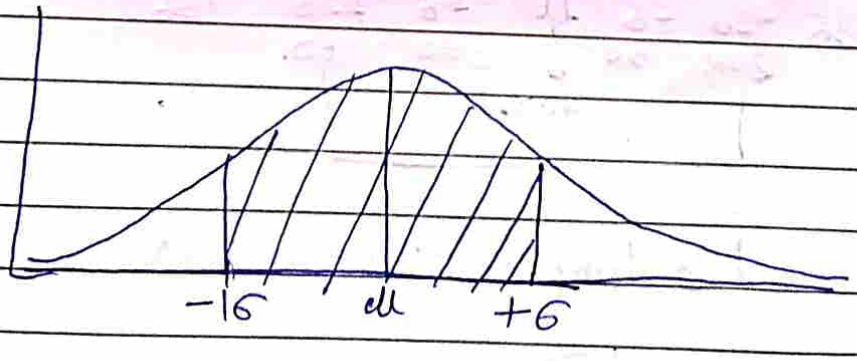
$x \leq \mu - k\sigma$



If $k=2$ the data points will be as shown in fig

2nd eqn of chebyshev eqn :-

$$P(\mu - k\sigma < x < \mu + k\sigma) > 1 - 1/k^2.$$



68-95-99 - rule.

Page No.
Date

Don't Know distribution.

Q $\mu = 40$, $\sigma = 10k$ $\quad -26$ to $+26$.
salary bet^n 20k, 60k = ?

$20k = \mu - 26$

$40k \; \mu$

$60k = \mu + 26$

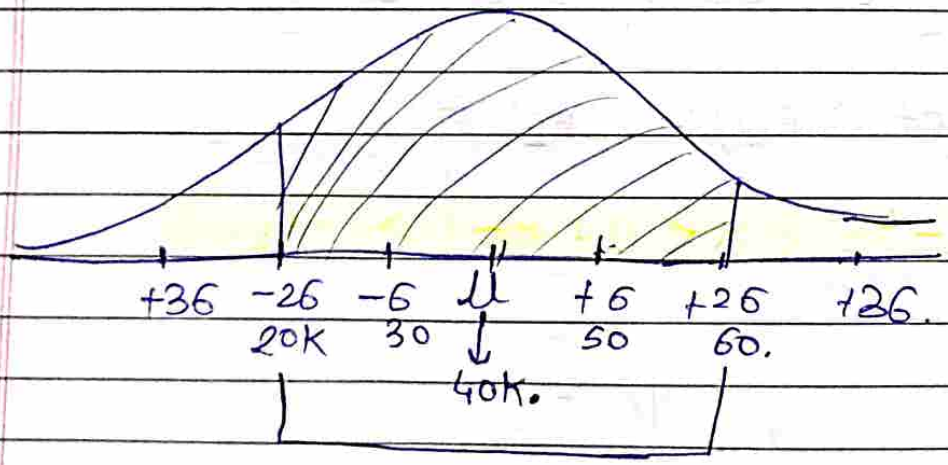
$P(\mu - 26 < X < \mu + 26) > 1 - \frac{1}{k^2}$ → 2nd eq chev

$P(40 - 20 < x < 40 + 20) > 1 - \frac{1}{2^2}$

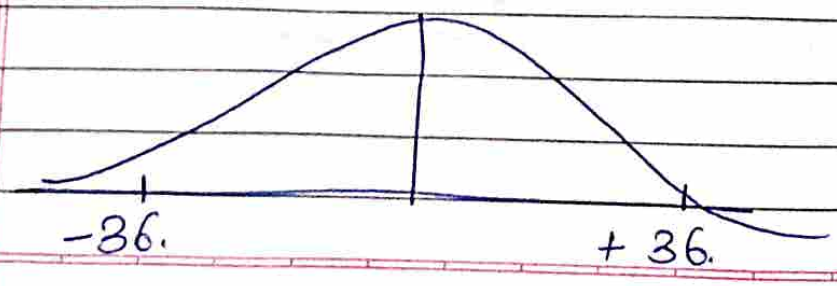
$P(20 < x < 60) > 1 - \frac{1}{4}$

$P(20 < x < 60) > 0.75.$

i.e 75% data point lies bet^n 20k-60k



Q If we want salary bet^n 10k to 70k.



$P(\mu - 3\sigma < x < \mu + 3\sigma) > 1 - \frac{1}{3^2}$.

$P(40 - 30 < x < 40 + 30) > 1 - \frac{1}{9}$

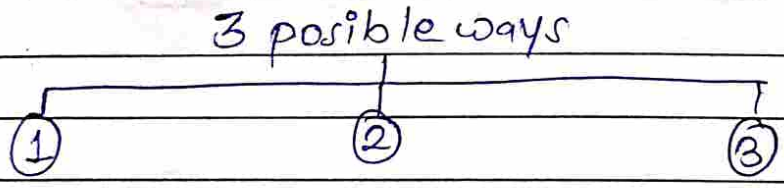
$P(10 < x < 70) > 0.89$

89%

* BINOMIAL PROBABILITY DISTRIBUTION

- The number of trials (n), must be fixed.
- There are only two possible outcomes for each trial
  - success
  - failure
- The probability (p) of success must be constant for every trial
- Each trial must be Independent

Q If you flip a regular coin three times. what is probability of getting exactly one head and this a binomial experiment?



H T T T H T T T H

P(HTT) = 0.125 P(THT) = 0.125 P(TTH) = 0.125

P(exactly one head) = 0.125 × 0.125 × 0.125

= 0.375

1) The number of trials (n) must be fixed
2) There are 2 possible outcomes for each trial
   - success = Heads
   - failure = Tails.

3) The probability (p) of success must be constant for every trial

$p = 0.5$

4) Each trial must be Independent

5) Binomial Experiment

outlayer identify → boxplot → median

curtosis 3 > 3 < 3

* 1st movement bussiness decision.
① calculate mean value, Identify outlayer
It measures ~~se~~ central tendency of data!

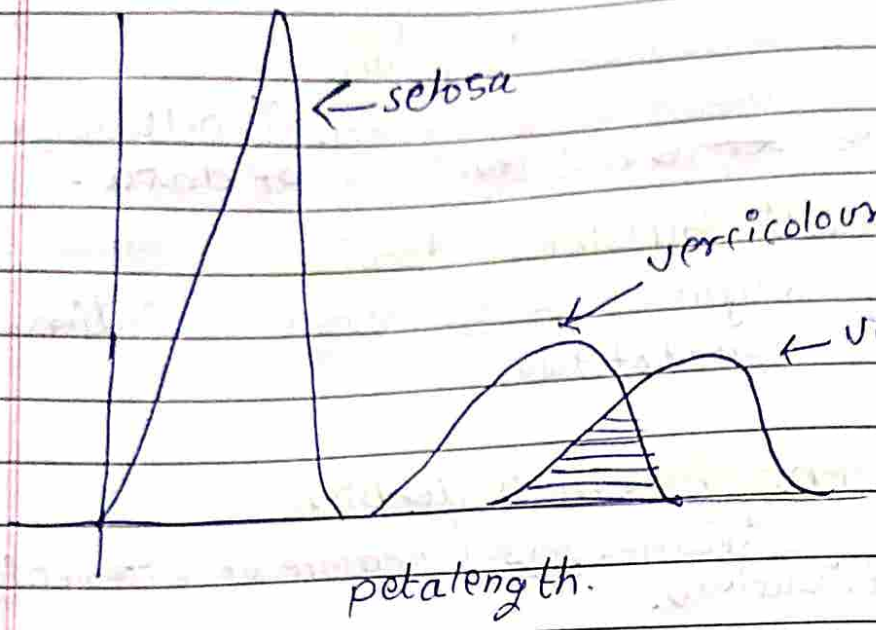
* 2nd movement bussiness decision
① finding variance & standard deviation.
spread of data.

* 3rd movement bussiness decision.
① finding skewness. (measure of assemetric in the distribution).

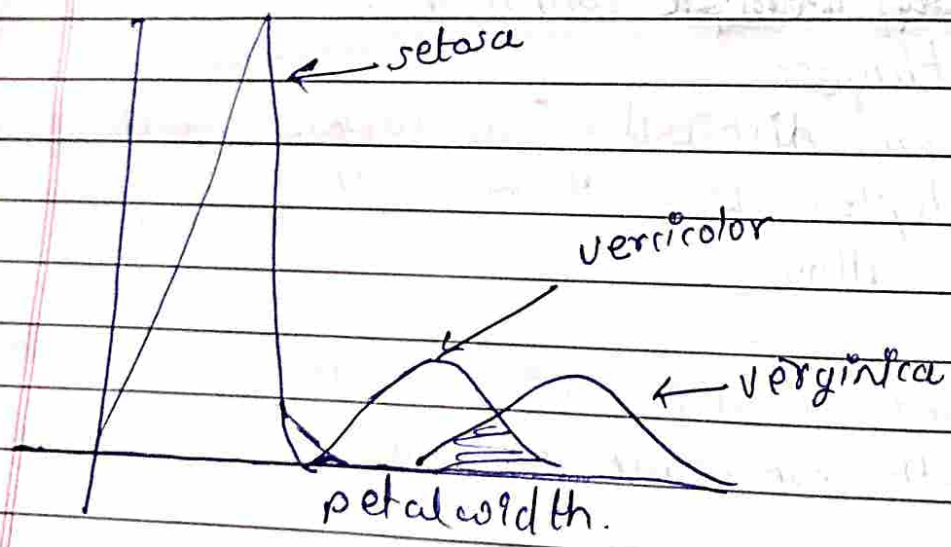
* 4th movement bussiness decision.
Finding curtosis (measuring fakedness of distribution)
Curtosis of normall distribution should be 3.
If given distribution of curtosis < 3 then it is said platy kurtic which means ↓ reproduce less outlayer
If given distribution as curtosis > 3 it is called leptocurtics It mean that it produces more outlayer.

* Univarient analysis using pdf,
① why we want to study univarient analysis?

In univarient analysis using pdf one can understand which feature contitatively or which features are linearly separable or which features are not linearly separable.



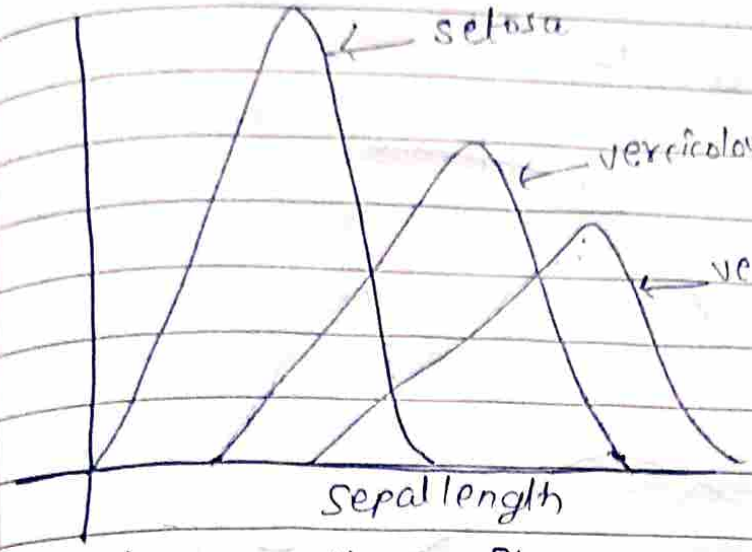
In above diagram we can ~~cle~~ conclude that setosa ~~fot~~ flower are linearly ~~spe~~ separable where as vercicolor & vergenica are not ~~be~~ linearly separable



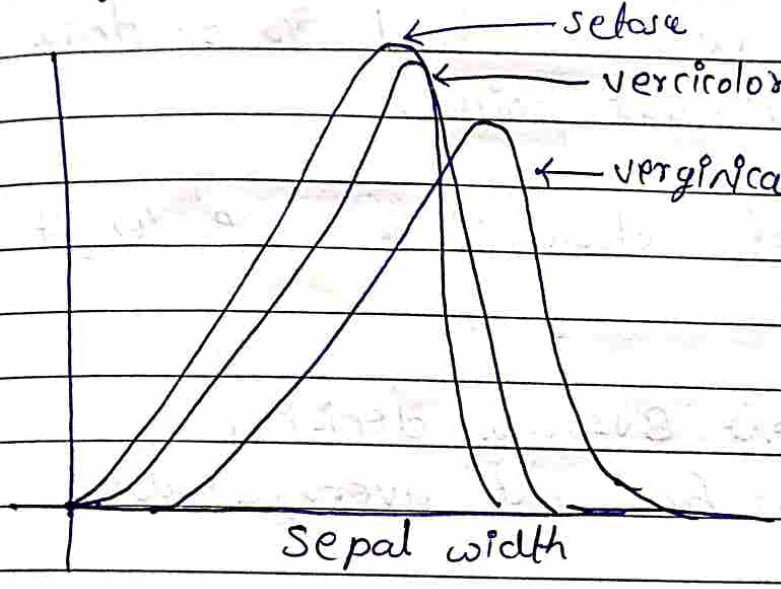
In above diagram we can see that setosa is linearly separable were as vercicolour

Compare to petal length, petal width is less linearly seperable, ~~however~~

pdf gives point on that point.



here, all 3 flower are not linearly seperable. If you are going to use sepal length.

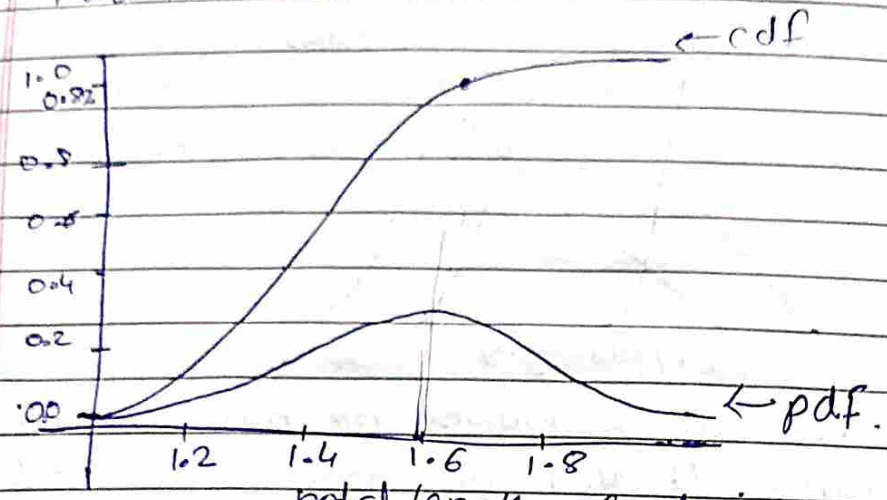


here, all 3 flower are not linearly ~~see~~ seperable. If you are going to use sepal length

* ploting histogram & pdf.
* ploting of cdf

From pdf we can visually see that what percentage of setosa/verginica/virsicolor flower have petal length less that some value

* how to read cdf



~~We~~ we can find % of data ~~in~~ till that particular point.

82% of setosa flower have length less than 1.6.

* 1st movement Business decision
  - we can find out average value.

* 2nd movement BD.
  Instead of variation we will used standard deviation bcoz $Var = \frac{\Sigma(x-\mu)^2}{N}$ where $(x-\mu)^2$ gives the higher value ∴ we use $SD = \sqrt{\frac{\Sigma(x-\mu)^2}{N}}$.

* equation of Variance for population (complete dataset) & Sample Statistics.

$$(Var)\sigma^2 = \frac{\Sigma(x-\mu)^2}{N} \quad \text{(population parameter.)}$$

$$\sigma = \sqrt{\frac{\Sigma(x-\mu)^2}{N}}$$

$$Var\ s^2 = \frac{\Sigma(x-\bar{x})^2}{n-1} \quad \text{Sample Statistics.}$$

$$s = \sqrt{\frac{(x-\bar{x})^2}{n-1}}$$

* Bar plots: (Univarient analysis).
  - data less - Bar plot
  - data more - histogram, - (frequency distribution table)

* 5 Number summary.
minimum, 1st Quartile, median, 3rd Quartile, max
df.describe().

* Box plot
1) Understanding distribution
2) finding outliayers

Data Preparation
sweetviz, autoviz, Dtale, dataprep.

Page No.
Date

Data Analytics, analyse past data to make conclusion.

stages of Analytics.

prescriptive analysis
predictive analysis
diagnostic analysis
descriptive analysis

* Data types → ratio, interval.

1) Continuous
2) discrete
3) Qualitative & Quantitative
4) structure, semi, non
5) Big Data, not big data
6) Cross sectional vs Time series vs longitudinal / panel data
7) Balanced / Imbalanced Data
8) offline / live data.

* Business problem
  - " object
  - " successcriteria
  - ML "
  - economic "

* Data preparation.

Data cleaning/ preparation/ man -aging/ organizing

1. Type casting
2. Handling Duplicates
3. Outlier treatment
4. Zero & near zero variable Features
5. Discretization / Grouping
6. Dummy variable creation
7. Missing values
8. Transformation
9. Feature scaling/ feature shrinking
10. string manipulation (Unstructure Textual Data)

IQR = Q3 - Q1 then outlier
< Q1 - 1.5 (IQR) or > Q3 + 1.5 (IQR

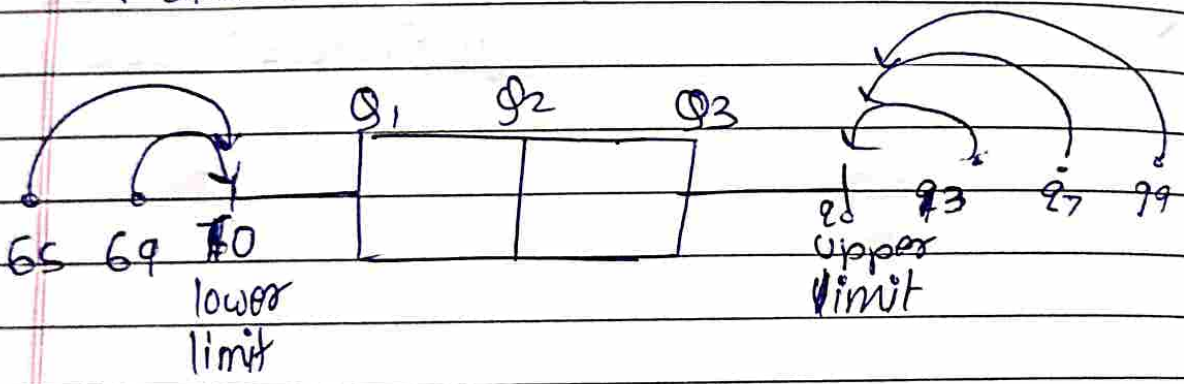
Page No.
Date

Outliers treatment.

- 3R rule
- Rectify
- Retain
- Remove
- Trimming: Removal
- Winsorization: Rounding up / masking

- Rectify
correcting the data / asking data entry operator to check.

- Retain



This process is called masking in which we convert outlier into its respective lower or upper limit.

lower limit = Q1 − 1.5 * iqr.
UL = Q3 + 1.5 * iqr.

- pip install feature_engine.

- In Trimming we remove the outlier, so instead of that we use replace.

* Winsorizer

```
from feature_engine.outliers import Winsorizer
winsor = winsorizer(capping_method='iqr',
                    tail = "both",
                    fold =1.5,
                    Variables ["Salaries"])
df_t = winsor.fit_transform(df[["Salaries"]])
sns.boxplot (df_t["salaries"])
```

* Discretization (otherwise known as quantization or binning) provides a way to partition continuous features into desrib value.

* necessary to have binning?
Whenever data is fall in small quantity it is very easy to visualize & overall compitional complexity is less. however if ~~data~~ data quantity is less then it is very difficult to visualize using bar graph. then It is suggested to divide data into quantums/ Sins so as to make it visuable and apply to ML model.

* What are bins?

Count 0 1 1 2 5 10 11

Notation:

[0,2] → 0, 1, 1, 2
(0,2) → 1, 1
(0,2] → 1, 1, 2
[0,2) → 0, 1, 1.

* ~~Trag~~ Strategies use for binning for discretization

1) Uniform strategy
2) Quantile Strategy
3) Kmean Strategy.

Data Transformation

Standardization & normalization

- Why it is necessary.

x_1	x_2	y
6	160	21
6	160	21
4	108	22.8
6	258	21.4
8	360	18.7

When we are appling data to ML model all features should have appropriate mean value & standard deviation If there is drastic variation of x_1 & across all features then accura -cy we will be deteriated

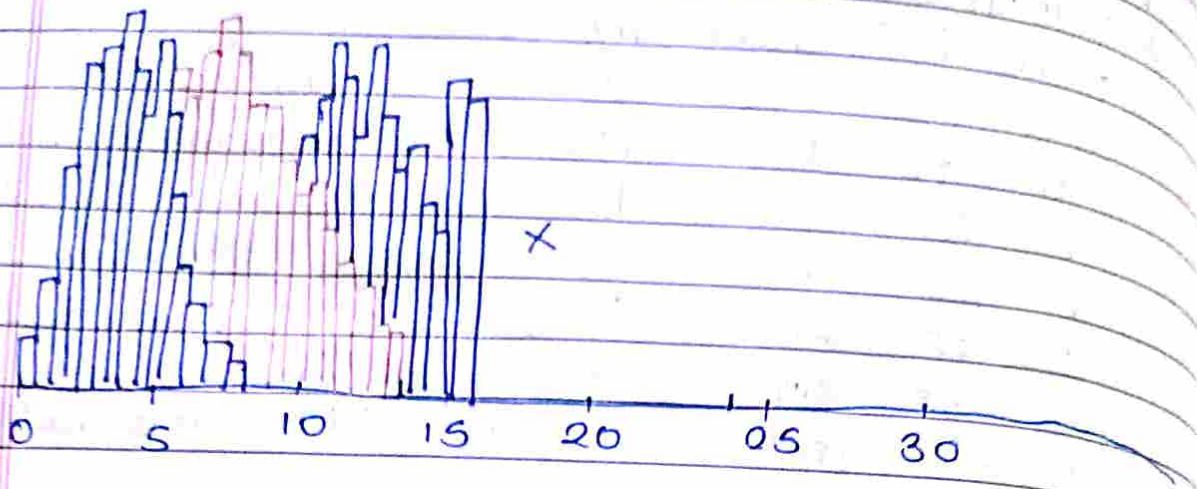
* If data is continuous than distribution is normal and having following properties
  1) Bell shape curve will be symetric
  2) range will be $-\infty$ to $+\infty$
  3) Area under curve is 1
  4) probablity of x will be equal to zero

* Transformation Techniques

1) Normalization] popular
2) standardization]
3) Quantile Transformation
4) log Transformation
5) Binserization
6) Power transformations
7) Unit Vectors scaling.

necessary of Trasformation

- Faster conversion & computing distance appropriatly



min-max Normalization $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$

Standardization $x = \frac{x - \mu}{\sigma}$

- If we apply standard normalization of skewed data Resultant will be skewed
- In standard normal distribution $\mu = 0$ and $\sigma = 1$

* Winsorizer

```
from feature_engine.outliers import Winsorizer
winsor = winsorizer(capping_method='iqr',
                    tail = "both",
                    fold = 1.5,
                    variables ["Salaries"])
df_t = winsor.fit_transform(df[["Salaries"]])
sns.boxplot(df_t["salaries"])
```