Machine Learning

Basics

1) Lists

i) Define a list:

```
x = [1, 2, 3, 4, 5, 6]
```

ii) To print the length of a list

```
print(len(x))
```

iii) slicing

```
x[:3]
```

Output: [1, 2, 3]

iv) Negative - slicing

```
x[-2:]
```

Output: [5, 6]

v) Append a list to another list

```
x.extend([7, 8])
x
```

Output: [1, 2, 3, 4, 5, 6, 7, 8]

vi) Append an element to the existing list

```
x.append(9)
x
```

output: [1, 2, 3, 4, 5, 6, 7, 8, 9]

vii) y = [10, 11, 12]

listoflists = [x, y]
listoflists

[[1, 2, 3, 4, 5, 6, 7, 8, 9], [10, 11, 12]]

y[1]
output: 11

viii) #Sorting a list (Ascending order by default)
z = [3, 2, 1]
z.sort()
z
output: [1, 2, 3]

ix) Sorting a list (reverse = True) [Descending order]

z.sort(reverse = True)
z
[3, 2, 1]

* List items are ordered, changeable and allow duplicate values.
* List items are indexed.

x.) list1 = list(("abc", 1, "cherry"))
list1
Outputs the list. Here, the list is created using a list constructor.

2.) Tuples

↳ A tuple is a collection which is ordered & unchangeable.

```
tuple = ("apple", "banana")
print (tuple)
```

i.) To print the length of the tuple :

```
x = (1,2,3)
len(x)
```

Output : 3

ii.)

```
y = (4,5,6)
y[1]
```

output : 5

iii.)

```
listofTuples = [x, y]
List of Tuples
```

Output: [(1,2,3), (4,5,6)]

iv.)

```
(age, income) = "32,12000".split(',')
print (age)
print (income)
```

output:

32

12000

3.) Dictionary

* Dictionary is a map / hash table in other languages.

i.) captains = {}
Captains ["Enterprise"] = "Kirk"
Captains ["Enterprise D"] = "Picard"
Captains ["Deep space Dive"] = "Sisko"
Captains ["Voyager"] = "Janeway"

print (captains ["Voyagers"])
Output:
Janeway

ii.) print (captains.get ("Enterprise"))
Output:
Kirk

iii.) print (captains.get ("NX-02"))
None

iv.) for ship in captains:
print (ship + ":" + captains [ship])

v.) for ship in captains:
print (ship)
output:
Enterprise
Ent D
DSN
voyager

4) Functions

```
# def sq(x):
    return x*x
print (sq(2))
```

Output:
4

```
# def doSomethn(p,x):
    return p(x)
print (doSomethn(sq, 4))
```

Output:
16

```
# print (doSomethn( lambda x: x*x*x, 3))
```

Output:
27

5) Boolean Expressions

```
# print (1==3)
```
False

```
# print (True or False)
```
True

```
# print (1 is 3)
```
False

```
# if 1 is 3:
    print ("How")
elif 1>3:
    print ("yikes")
else:
    print ("All good")
```

5.) Looping

```
# for x in range(10):
    print(x)
```

output: 0 1 2 3 4 5 6 7 8 9

```
# for x in range(10):
    if (x is 1):
        continue
    if (x > 5):
        break
    print(x)
```

output:
0
2
3
4
5

```
# x=0
while (x<10):
    print(x)
    x += 1
```

output:
0 1 2 3 4 5 6 7 8 9

Pandas:

> Pandas is a python library that makes handling tabular data easier.

> Introduces "Data frames" and "Series" that allow you to slice and dice rows & columns of information.

Numpy:

> Numpy arrays are multi-dimensional array objects. It's easy to create a pandas dataframe from a numpy array and pandas Dataframes can be cast as Numpy arrays.

Scikit_Learn:

> Load, clean and manipulate your input data using pandas. Then convert pandas Dataframe into a Numpy array as it's being passed into some Scikit_Learn function. that Conversion can happen automatically.

Exercise 1

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

df = pd.read_csv("PastHires.csv")
df.head()  # prints top 5 rows of the dataframe

# df.head(10)  # prints the top 10 rows

# df.tail(4)  # prints the least 4 rows

# df.shape  # (prints (rows, cols))

# df.size  # (prints rows * cols)

# len(df)  # prints the no. of rows

# df.columns  # prints the Column names of the dataframe

# df['Level of Education']
  # prints all the data present at that particular column.

# df['Employed?'][:3]
  # prints (0-3) rows data of given column.
```

```
# df ['Hired'][4]
# prints data present at 4th row of column 'Hired'.
# df [['Years Exp', 'Hired']]
# prints data of both the columns given
# df [['Years Exp', 'Hired']] [:3]
# prints data of both the columns [0-3 rows]
# df.sort_values(['Years Exp'])
# prints data present at years Exp column by sorting
# df.sort_values(['Years Exp'], ascending = False)
# prints data present at years Exp column sorting in descending.
# Edcnt = df['Level of Education'].value_counts()
# Outputs each unique value present at column given and the frequency of that value.
# Edcnt.plot(kind = 'bar')
# Outputs a bar graph of the given dataframe
# prevEmpHir = df[['prev', 'Hired']][5:10]
# prevEmpHir.plot(kind = 'hist')
# Outputs a histogram of the given dataframe.
```

Types of Data

i) Numerical
ii) Categorical
iii) Ordinal

Numerical

* Represents some sort of quantitative measurement
Ex: Heights of people, page load times, stock prices etc..

- Discrete Data

Integer based; often counts of some event.
- How many purchases did a customer make in a years?
- How many times did I flip "heads"?

- Continuous Data

Has an infinite number of possible values
- How much time did it take for a user to check out?
- How much rain fell on a given day?

Categorical

* Qualitative data that has no inherent mathematical meaning
- Gender, Yes/no (binary data), Race, State of Residence, product category, political party etc..

* you can assign numbers to categories in order to represent them more compactly, but the numbers don't have mathematical meaning.

Ordinal

* A mixture of numerical and categorical
* Categorical data that has mathematical meaning.

Ex: Movie ratings on a 1-5 scale.
Ratings must be 1, 2, 3, 4 (or) 5
But these values have mathematical meaning
1 means it's a worse movie than a 2.

Quiz!!

* How much gas is in your gas tank?
> Numerical

* A rating of your overall health (choices - 1, 2, 3, 4) corresponding to ("poor", "moderate", "good", "excellent")
> Categorical Ordinal

* The races of your classmates
Categorical

* Ages in years
> (Numerical)

* Money spent in a store
> (Numerical)

Mean

> AKA Average
> Sum (no. of samples
> Ex:

No. of children in each house of the street:

0, 2, 3, 2, 1, 0, 0, 2, 0

$$\text{Mean} = (0+2+3+2+1+0+0+2+0)/9 = 1.11$$

Median

> Sort the values and take the value at the midpoint

Ex: 0, 2, 3, 2, 1, 0, 0, 2, 0

Sort: 0, 0, 0, 0, 1, 2, 2, 2, 3

↑

Median = 1

* If you have an even no. of samples take the avg of the two in the middle.

> Median is less susceptible to outliers than the mean.

Ex: the mean household income in the US is $ 72,641
but the median is only $51,939 — because the mean is skewed by a handful of billionaires.

** Median better represents the "typical" American in this example.

Mode

> The most common value in a dataset.
Not relevant to continuous numerical data.
Works only for discrete numerical data.

0, 2, 3, 2, 1, 0, 0, 2, 0

> How many of each value are there?

0 : 4 1 : 1 2 : 3 3 : 1

Mode = 0

Normal Distribution

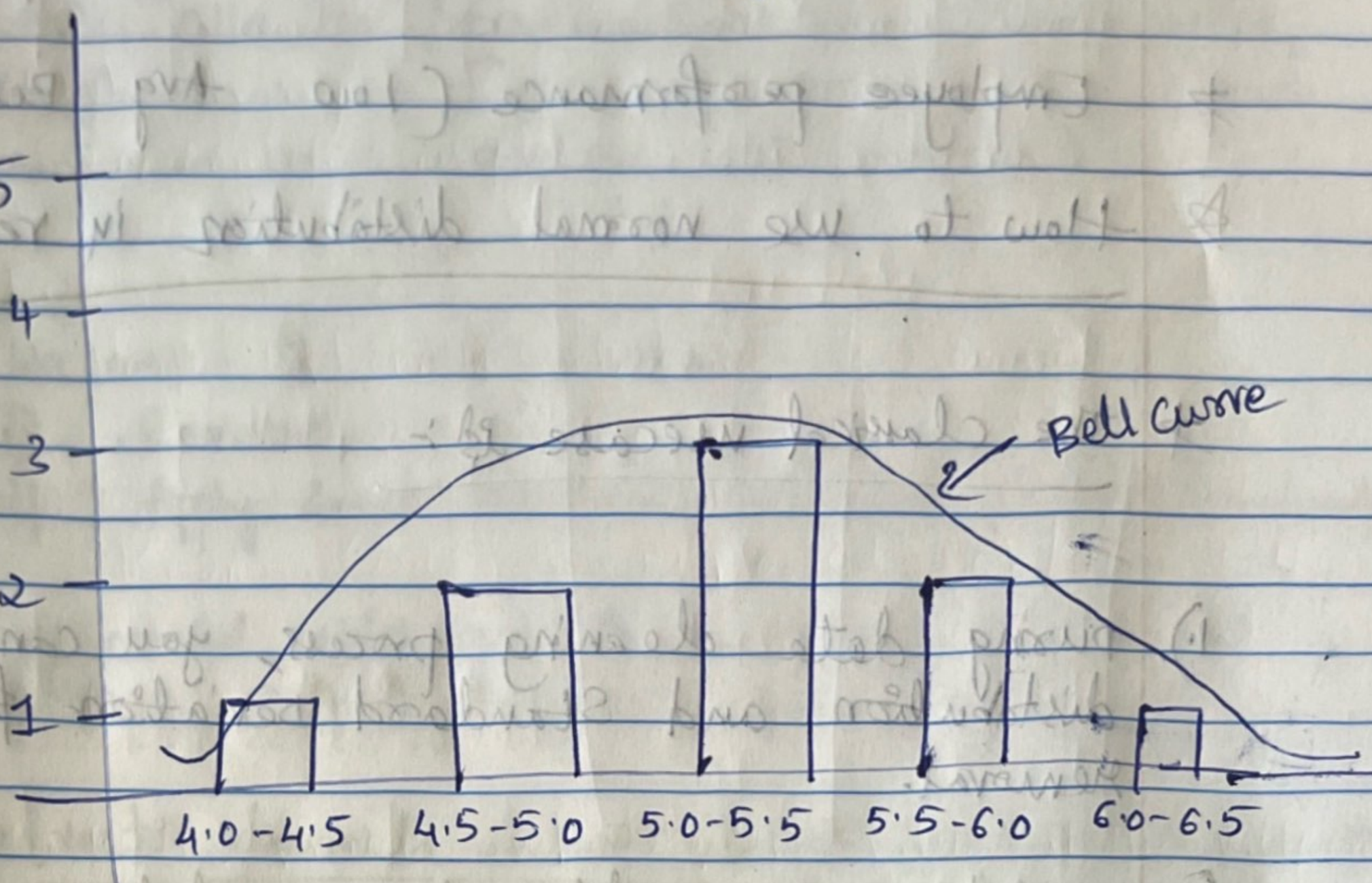
* Let's say you want to do data analysis on people's height

Ex:

Name	Height (ft)
Rob	6.2
Thomas	5.7
Nina	4.6
Mit	5.4
Sofia	5.9
Mohag	4.3
Tao	5.1
Deepika	5.2
Rafiq	4.9

* When you plot this data on a histogram, it will look like:

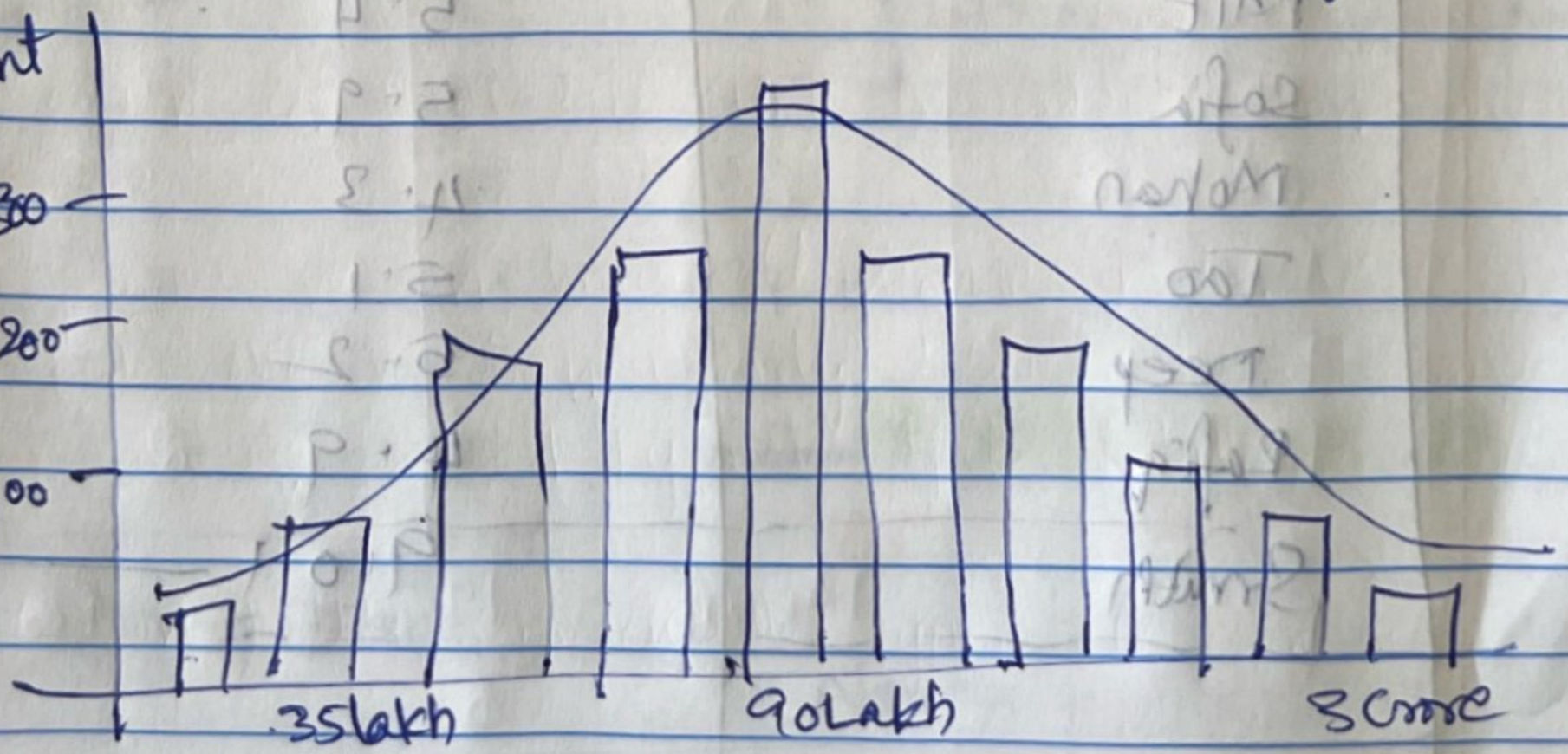
(Histogram is a freq distribution)



* When you draw a curve that passes through this histogram it looks like a bell.

* This is called a normal distribution; Most of your data samples are around avg value. Some far away from data samples on the left & right.

Ex: 2 bedroom apt price in Bangalore city



* We can even consider (Test Score out of 100)
* Employee performance (Low Avg Best performance)
* How to we normal distibution in real life??

* The classical usecase is:-

1) During data cleaning process, you can use normal distabution and Standard Deviation for Outlier removal.

Ex: Adding an Oultier to our data

Outlier is a very different value from our data.

Name	Height (ft)
Rob	6.2
Thomas	5.7
Nina	4.6
Mit	5.4
Sofia	5.9
Mohan	4.3
Tao	5.1
Deep	5.2
Rafiq	4.9
Smith	9.0 → outlier

* This kind of outliers can occur because of an error in data collection / transmission process / sometimes might be valid [max in the history [They stand out even on a histogram when plotted.]

* We need to treat outliers
  i) Remove them
  ii) Apply some methods to treat them (Transformation)

* ML model might get skewed if you don't treat outliers. So, it's better to treat them.

** what formula do we use to remove Outliers?

1) First, let's learn about:

Mean Absolute Deviation

Standard Deviation

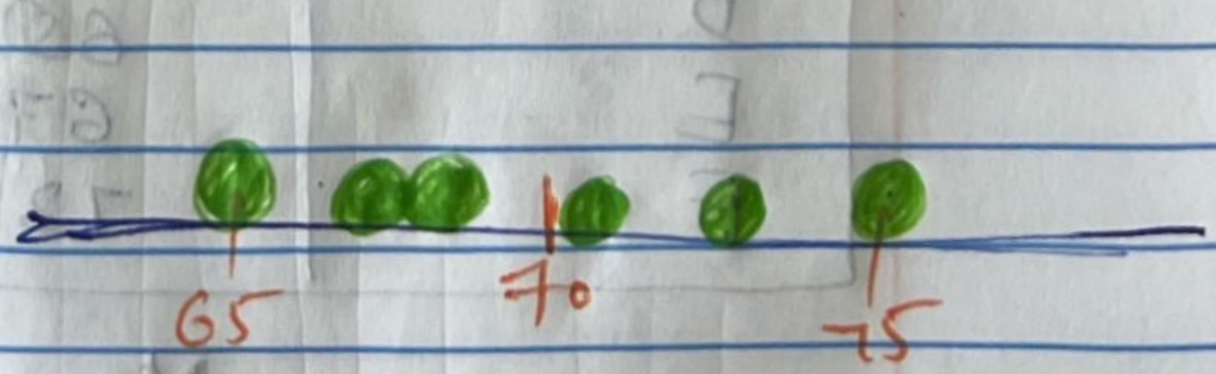
Ex:-

History Test

Name	Score
A	75
B	72
C	68
D	65
E	67
F	73

Avg = 70

Plotting on a Line chart



[Nearer to average]

→ we are trying to know how far are our data points from the average. [How spread out].

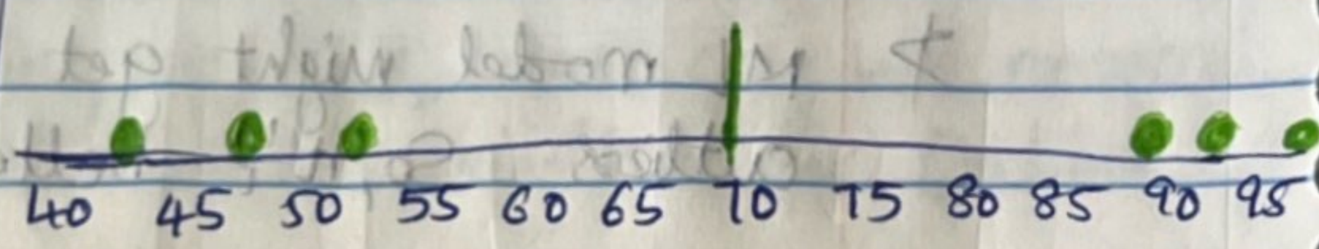
Ex: Consider Other Sets

Math Test

Name	Score
A	93
B	96
C	43
D	47
E	51
F	90

Avg = 70

plotting on a line chart



[Quite Spreadout from average]

* Let's Compute Absolute for each individual score

Ex:

History Test

Name	Score	Abs(Score − Avg) (Avg = 70)
A	75	5
B	72	2
C	68	2
D	65	5
E	67	3
F	73	2

$$\text{Mean} = \frac{19}{6} = 3.16$$

Math Test

Name	Score	Abs (score - Avg)
A	93	23
B	96	26
C	43	27
D	47	23
E	51	19
F	90	20
	Mean	23

$$\text{Mean} = \frac{161}{6} = 23.$$

* This mean is known as Mean Absolute Deviation(or MAD.

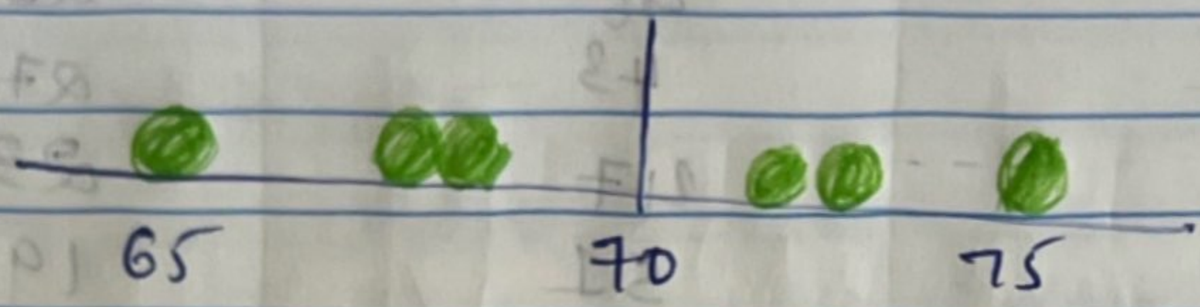
* So, this is a useful metric to represent how spread out the data points are.
* Higher the MAD, more the data points are spread ou

** There could be a scenario where using MAD might not be enough.

Ex:

Name	Score	Abs (score - Avg)
A	75	5
B	72	2
C	68	2
D	65	5
E	67	3
F	73	3
	Mean	3.33

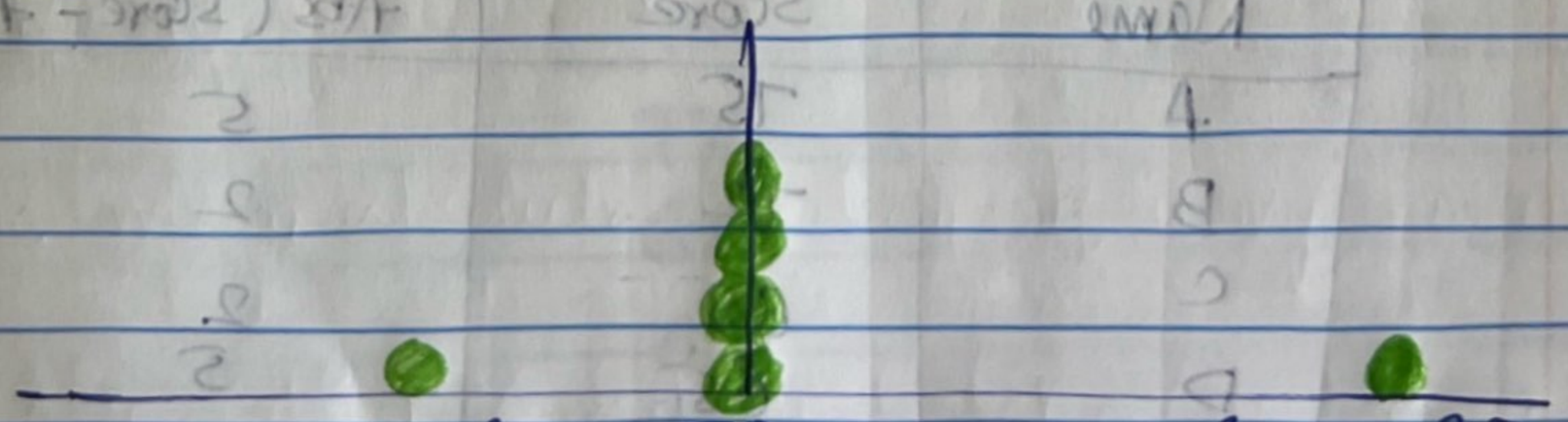
plotting:



Let's take another example:

Name	Score	Abs (Score − Avg)
A	83	13
B	70	0
C	70	0
D	63	7
E	70	0
F	70	0
	MAD	3.33

Plotting:



→ So, in this case MAD is not very useful. we need to come up with something else.

(Avg = 70)

Name	Score	Abs(Score - Avg)	(Score - Avg)^2
A	75	5	25
B	72	2	4
C	68	2	4
D	65	5	25
E	67	3	9
F	73	3	9
		Avg =	12.66
		√Avg =	3.55

(Avg = 70)

Name	Score	Abs (Score - Avg)	(Score - Avg)^2
A	83	13	169
B	70	0	0
C	70	0	0
D	63	7	49
E	70	0	0
F	70	0	0
		Avg	36.33
		√Avg	6.02

Higher the (√Avg), more the data points are spread out.

* this ($\sqrt{Avg}$) is known as Standard deviation.

Formula

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

σ = Population Standard deviation
N = Size of population
x_i = Each value from the population
μ = The population mean

Notes:

1.) MAD and SD (Both measure the dispersion of your data by computing the distance of the data to its mean).

* The mean absolute deviation is using norm L1

(Also called Manhattan distance / rectilinear distance)

* The standard deviation is using norm L2

(Also called Euclidean distance)

* The difference between the two norms is that the standard deviation is calculating the square of the differences whereas the mean absolute deviation is only looking at the absolute difference.

* Hence, large outliers will create a higher dispersion when using the standard deviation instead of the other method. The Euclidean distance is indeed also more often used. The main reason is that the standard deviation have nice properties when the data is normally distributed. So, under this assumption, it's recommended to use it.

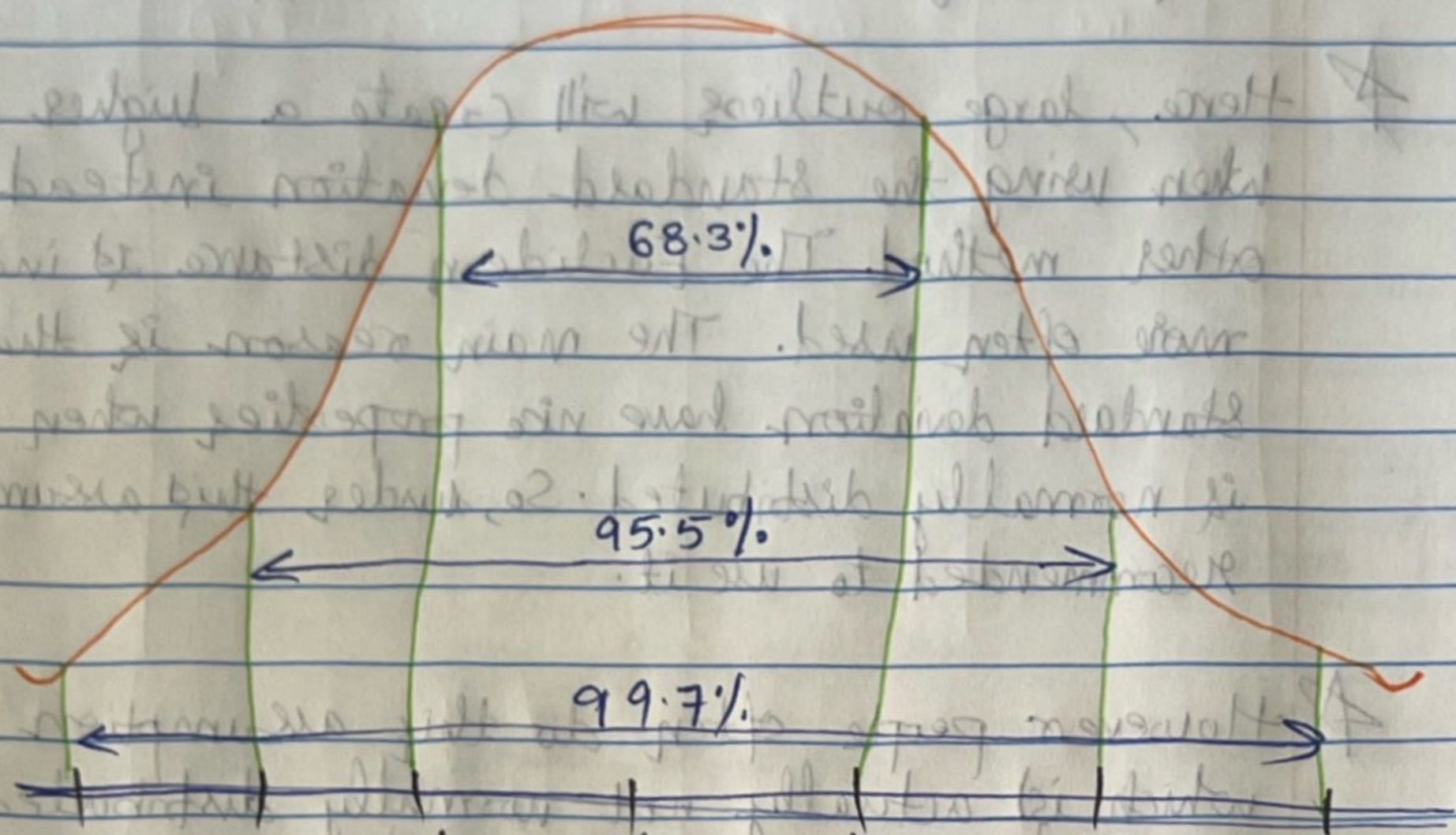
* However people often do this assumption for data which is actually not normally distributed which creates issues. If your data is not normally distributed, you can still use the standard deviation, but you should be careful with the interpretation of the results.

Important:

* Identifying whether data is normally distributed is an essential step in many statistical analyses because many techniques assume normality. The methods to determine if your data are normally distributed:

1) Visual Inspection Methods (Histogram, Q-plot)
2) Statistical Tests
3) Descriptive Statistics

* Using Standard Deviation to remove Outliers:



* By conducting several tests on normally distributed data mathematicians found that (68.3% of data in any normal distribution comes in (-1σ to +1σ).
95% of data points fall under (-2σ to +2σ) and 99.7% of data under (-3σ to +3σ).

* We can use this knowledge to remove outliers
If the data point is beyond (+3σ (or) -3σ) can be treated as an outlier (This is the general guideline).

* Z Score :

* How many standard deviation away a datapoint is from mean.

Ex:

Name	Height (ft)	Z Score
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

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

x = Each data point
μ = Average (Mean)
σ = Standard deviation