Dt:- 2/4/22
5:50pm

Part 02

* Data Preprocessing / Data Preparation

⇒ Feature Engineering

Data Cleaning
- Missing Values
- Outliers

+

Data Wrangling
↳ (Data Transformation)
- Encoding
- Scaling
- Transformation.

Us. (Car Insurance)

Data cleaning (In spyder)

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

why two DaTaSet
Because to check old(and new dataset.

⇒ Load Data Set

```
# df_raw = pd.read_csv ("claimants.csv")
# df = pd.read_csv ("claimants.csv")
```

Out :- df_raw | dataFrame | 1340,7 | column names : CASENUM,
df | dataframe | 1340,7 | column names : "

⇒ Check is there null values.

df.isnull().sum()

Out :-

CASENUM	0
CLMSEX	12
CLMINSUR	41
SEATBELT	48
CLMAGE	189
LOSS	0
ATTORNEY	0

MISSING Values.

Data Understanding

* Case Num :- Number of case (insurance)

Discrete clm sex :- 0, 1 [Male] [Female]

Discrete clm Insur :- 1(Yes), 0 [NO] # having Insurance or not

Discrete Seat belt :- 0 [NO], 1 [Yes] # wearing Seat belt

Continuous clm age :- Age of The person

Loss :- Loss / Damage happened

attorney :- whether They hire a lawyer (or) not, [0] No, [1] Yes.

Person who is Claming Isurance is called "claimants" During The accident

There are Various methods For Filling null Values

* For Continous Data → mean (no outliers) / median (outliers)

Because Mean is Heavily Impacted with Outliers

[Less affected]

* For Discrete Data → Mode

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Replacing null values For discrete Variables

So, we have to Replace Them with "MODE"

df["CLMSEX"].value_counts()

(MODE)

Out :- 1.0 742 → Highest, so we Replace with "1"
0.0 586

df["CLMSEX"].fillna(1, inplace = True)

↓ changing in orginal DaTa = "TRUE"

Out Filled with "1"

df["CLMINSUR"].Value Counts()

Out :- 1.0 1220 → MODE
0.0 120

df["CLMINSUR"].fillna(1, inplace = True)

Out Filled with "1", so, replaced with "1" value

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

df["SEATBELT"].valuecounts()

Out :- 0.0 1270 → Mode.
1.0 22

df["SEATBELT"].fillna(0, inplace = True).

Out :- Filled with "0"

Time 11:30pm

Replacing null Values For Continous Variables By # Pandas

df["CLMAGE"].mean()

Out 28.414

df["CLMAGE"].median()

Out 30.00

df["CLMAGE"].fillna(28.414, inplace True)

Out Filled with 28.414

If we ask again what is median value.

df["CLMAGE"].median()

Out 28.414, so it changes 30.00 To 28.414 Because we Filled with null values 28.414.

2nd method

Sklearn (Scikit Learn) (One stop shop For ML)

* SimpleImputer

Load data set again

Liberary → (module) (Function)

From Sklearn.impute import SimpleImputer

Ex: missing values = "???"

Calculating mean value

mean_imputer = SimpleImputer (strategy = "mean") → $\frac{\Sigma}{n}$

(Function) → (Argument)

name as

Function. (or) store in mean_imputer

Ex:- Core Python [Keyword argument]

If we don't write This line it Gives

Same : a = 4

Converting array To data Frame

To Remove array. From mean value.

df ["CLMAGE"] = Pd. DataFrame (mean_imputer . fit_trans

storing

_Trans Form (df [["CLMAGE"]]))

array ([50.],
[18.],
[5.],
⋮

Fill in This column (or) Replacing

Calculate The mean value

Out

Filled with 28.414

For Median only change mean To median

median_imputer = Simple Imputer (strategy = "Median")

df["CLMAGE"] = pd.DataFrame(median_imputer.fit_Transform(df[["CLMAGE"]])

For Mode [Discrete Data]

mode_imputer = SimpleImputer(strategy="mode")

df["CLMSEX"] = pd.DataFrame(mode_imputer.fit_transform(df[["CLMSEX"]])

df["CLMINSUR"] = pd.DataFrame(mode_imputer.fit_transform(df[["CLMINSUR"]])

df["SEATBELT"] = pd.DataFrame(mode_imputer.fit_transform(df[["SEATBELT"]])

Output

Fills Directly The mode Value in to the dataset.

MODE = it is for Discrete Data.

MEAN
MEDIAN → Continous Data

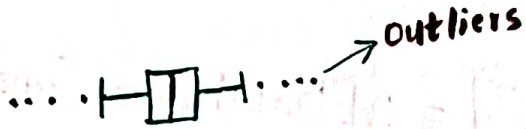
Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

3/4/22 2:08 Am

Outliers → # Applicable Only For Continous Data

1 Que :- What is outlier ?

Ans :- An Outlier is a data point in a data set. that is distant From all other observation which is significantly different From The Remaining Data



(or)

- The data point that Lies outside The over all Distribution of the dataset.

2 Que :- What are Impacts having Outliers in a Data Set ?

Ans :- it will cause Various Problems in Statistical Analysis. (it may cause a significant Impact on mean and Standard deviation)

Ex :- $\bar{x}$ = mean (affect)

$\frac{\Sigma[x-\bar{x}]}{n-1}$ = varience → $\sqrt{\frac{\Sigma(x-\bar{x})^2}{n-1}}$ (Standard deviation)

all are Effected by Outliers

* In addition, Some Machine learning models are Sensitive to Outliers, which may Decrease There Performance.

3 Que : Reasons For Outliers ?

Ans : 1. Data Entry Errors (Ex:- Salary = 100,000, But Entering 10,000)

2. Measurement Errors (Ex :- Measuring in meters instead of KM)

3. Instrumental Error

4 Que :- Types of Outliers ?

Ans :- Univariate Outlier ----> Identifying Outlier For Single Variable

* Bivariate Outlier ----> Identifying as Outlier by Analyzing 2 Variables at a Time

Importing Libraries

```
# Import numpy as np
# Import pandas as pd
# Import matplotlib as plt.
# Import Seaborn as Sns.
```

Real Time Case study

boston - House Price Dataset

```
# boston = pd.read_csv("D:\\data science \\ shivaram
krishna class \\ 6. Data cleaning \\ boston.csv")

boston.head()
```

out

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRTIO
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.090	1	296	15.3
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7

B	LSTAT	PRICE
396.90	4.98	24.0
396.90	9.14	21.6
392.83	4.03	34.7
394.63	2.94	33.4
396.90	5.33	36.2

```
# boston.info()
```

506 rows, 14 columns

out

0	CRIM	Float 64
1	ZN	Float 64
2	INDUS	Float 64
3	CHAS →	int 64
4	NOX	Float 64
5	RM	Float 64
6	AGE	Float 64
7	DIS	
8	RAD →	Int 64
9	TAX →	Int 64
10	PTRTIO	Float 64
11	B	Float 64
12	LSTAT	Float 64
13	PRICE	Float 64

(Float 64 (11), int 64 (3))

Various ways of finding The outlier

Changing Every Value to Zscore.

Ex:-

X	Zscore
6	
4	$\frac{4-5.6}{1}$
9	$\frac{9-5.6}{1}$
7	$\frac{7-5.6}{1}$
2	$\frac{2-5.6}{1}$

$\boxed{Z = \frac{X-\mu}{\sigma}}$, $Z = \frac{6-5.6}{1}$

$\therefore \mu = \frac{6+4+9+7+2}{5}$

$\mu = 28/5 = 5.6$

Ex:- $\sigma = 1$

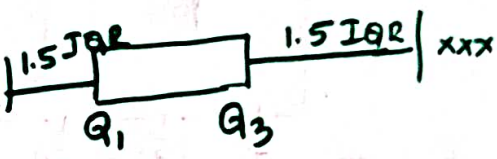
Option (1) Z score

$|Z\text{score}| > 2$

$\left[\begin{array}{l} \therefore Z < -2 \\ Z > +2 \end{array}\right]$ |modulus|

Option (2) IQR (Inter Quartile Range)

- any Value $< Q_1 - 1.5\,IQR$
- any value $> Q_3 + 1.5\,IQR$



any value Less than $Q_1 - 1.5\,IQR$ is Outlier.

any value More Than $Q_3 - 1.5\,IQR$ is Outlier

Option (3) Visualization

* Box plot, Histogram For Univarte Outliers
* Scatter plot For bivariate Outliers.

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

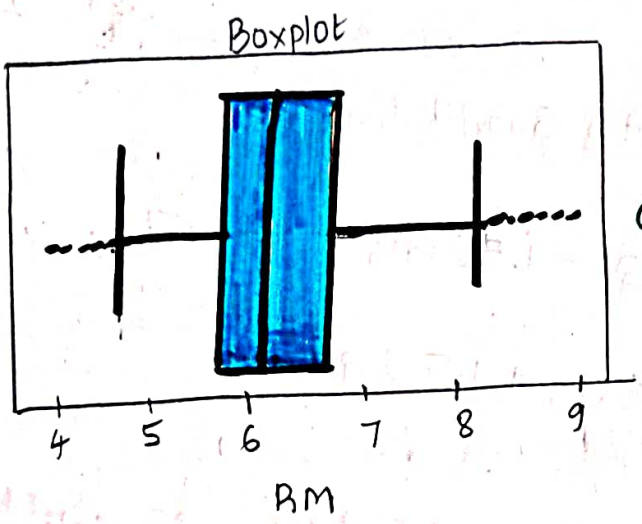
Option ③

Detection of Outliers of "RM" column (Based on Boxplot)

```
# sns.boxplot(boston.RM)
  plt.title("Boxplot")
  plt.show()
```

* Over all view of Outlier

Out :



Outliers in Both Tails of "RM".

Option ①

Detection of Outliers of "RM" column (Based on Zscore)

* For Exact Outliers in Data

$$\boxed{Zscore = \frac{X - \mu}{\sigma}}$$

```
# boston["RM"].mean(), boston["RM"].std()
```

Out (6.284, 0.702)

```
# boston["RM_Zscore"] = (boston["RM"] - boston["RM"].mean()) / boston["RM"].std()
```

creating new column and storing Zscores

X, μ, σ

Output :-

Clarity :-

$$\frac{(\overset{x}{\boxed{boston["RM"]}} - \overset{\mu}{\boxed{boston["RM"].mean()}})}{\underset{\sigma}{\boxed{(boston["RM"].std()}}}$$

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

E Answers Ami

boston["RM_Zscore"] =

Endulo store aye thaye.

	CRIM	ZN	INDUS	CHAS	NOX	RM (original)	AGE	DIS	RAD	TAX	PTRATIO	B	... RM SCORE (Zscore)
0	0.0632	18.0	2.31	.	.	6.575	.	.	.	.	.	.	0.413263
1	0.02731	0.0	7.07	.	.	6.421	.	.	.	.	.	.	0.194082
2	0.02729	0.0	2.18	.	.	7.185	.	.	.	.	.	.	1.281446
⋮													
													-0.362408
505	0.04741	0.0	11.93	.	.	6.030	.	.					

506 x 15 columns

Now, Identify Outliers in this column

OR

Outlier = boston[(boston["RM_Zscore"] > +2) | (boston["RM_Zscore"] < -2)]

2 conditions

Greater Than +2

Less Than -2

Out :- Outlier

	CRIM	...	RM SCORE
97			2.539
98			2.185
162			3.47
163			-2.127
66			-3.055
			⋮

3/4/22 4:30 Am

Time
1:00pm

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Outlier. Shape

Out :- (31, 15)

Option 2

Detection of Outlier of RM (based on IQR)

- Calculate First (q_1) and third quartile (q_3)
- Find interquartile range ($q_3 - q_1$)
- Find Lower bound q_1 1.5 & Find upper bond q_3 1.5

* For Exact no. of range + - of outliers

```
# Q3 = boston["RM"].quantile(0.75)
  Q1 = boston["RM"].quantile(0.25)
```

$Q_1 = 0.25$ (or)
$Q_3 = 0.75$ (or)

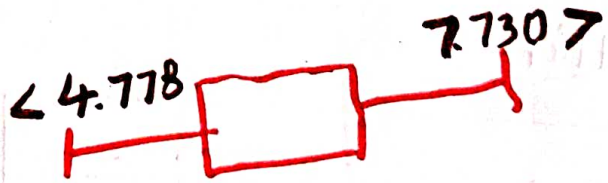
$IQR = Q3 - Q1$

$\boxed{IQR = Q_3 - Q_1}$

Lower_limit = Q_1 - (IQR * 1.5) $\Rightarrow Q_3 + IQR * 1.5$

Upper_limit = Q_3 + (IQR * 1.5) $\Rightarrow Q_1 - IQR * 1.5$

Lower_limit, upper_Limit



Out :- 4.778 , 7.730
Min Max

* which is having less than $\boxed{4.778}$ is outlier
* which is having greater than $\boxed{7.730}$ is outlier

Methods of deal The Outliers

3R Techinque
- * Remove → Deleting The outliers
- * Replace → changing values / Data Manipluation
- * Retain → Treat Them Seperately. Work without outlier / with outlier "Work" seperately.

***

Outliers should be detected and "REMOVED" Only From Training data set, NOT from The Test set. So we Should first divide our data set into train and tests. and remove Outliers in the train set, but Keep those in test set, and Measure how well our model is doing.

Option ① Removing * (Let's trimm The dataset) → 4.77

```
# boston_trimmed = boston[(boston["RM"] > Lower_limit) &
                          (boston["RM"] < upper_limit)]
```

7.73

```
boston_trimmed
```

Out :-

* Which ever Data is Greater Than 4.77 and Less Than 7.73 Consider That data Only. Remove Remaining data.

		RM		RM_Zsco
0		6.575		0.413263
1		6.421		0.194032
2		7.185		1.28 14416
⋮		⋮		⋮
504		6.794		0.724955
505		6.030		-0.362408

476 × 15 columns

∴ From 506 records, Removed 30 records

that Means 476 Having (or) there.

⇒ it's about 6% of the data is Removed

⇒ it is Only in One Variabe (1 column) = 6%.

⇒ If Total Variable (1,2,3,4..... all columns) (14) → [10] continous Data → Remove → May Delete 100 rows ⇓ 20% of Data

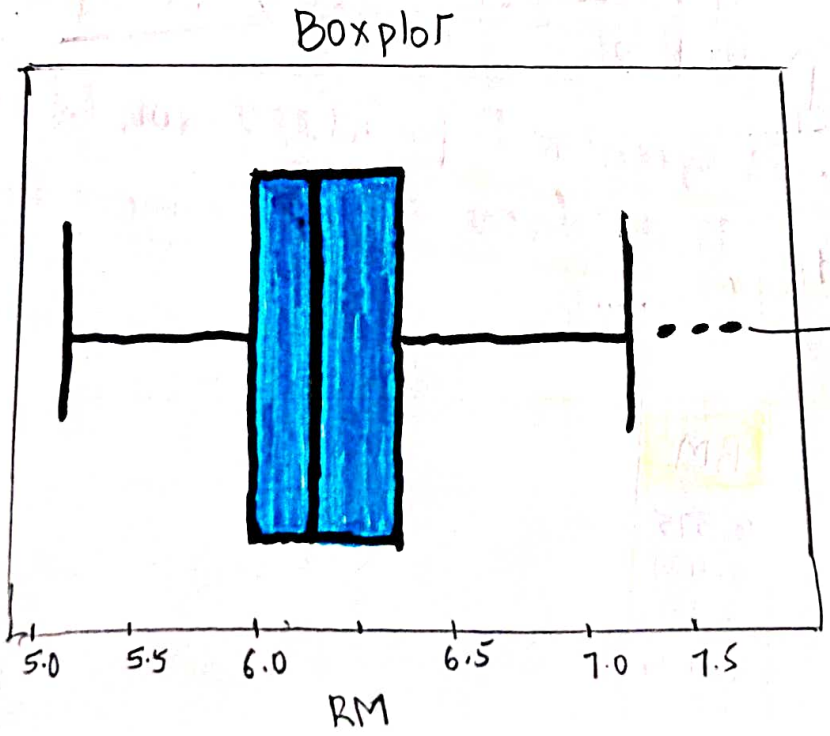
So, In real Life, we don't use outliers "Remove"

Even, Though, If we want Remove, still it Contains Outliers.

⇒ Outliers in The Trimmed dataset

```
# sns.boxplot (boston_trimmed.RM)
plt.title ("Boxplot")
plt.show()
```

Out



still, we have Outliers, after Removing The Outlier records

Q:- Why ? it showing Outliers after Removing records of outliers

A:- When we remove datapoints From Our dataset, all the Parameters of The distribution are re-calculated.

For 476 records,

Again, it's calculating (Q_1, Q_3, IQR)

For 506

$Q_1 = 4.77$, Q_1 = New Value (again)

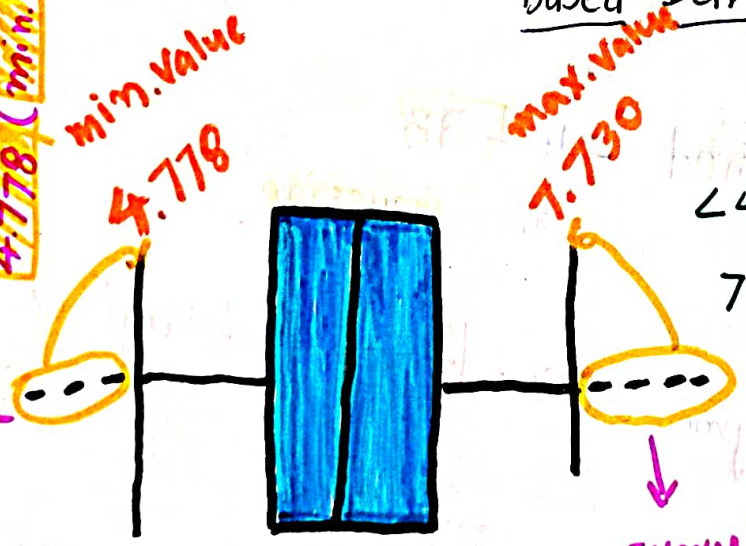
$Q_3 = 7.730$, Q_3 = New value (again)

* Therefore, In The new-trimmed Variables Values that before were not considered Outliers.

That's why, it showing new Outliers after removing some records also.

Option (2). **Replace**

Replace with Upper Limit & Lower Limit (Calculated Based IQR)



$< 4.778 = Q_1 - [IQR * 1.5]$

$> 7.730 = Q_3 + [IQR * 1.5]$

Every Outlier Replacing with 4.778 (min. value)

Every Outlier Replacing with 7.730 (max. value)

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

⇒ More Detailness

Replacing with Q_1, Q_3 Values

	CRIM	. . . . .	RM	. . . . . RAD	TAX
97		More Than 7.730, so, They are outliers. Reduce with Q_3 = 7.730	8.069 7.730		
98			7.820 7.730		
162			7.802 7.730		
163			8.375 7.730		

$Q_3 - IQR * 1.5 < [4.778]$ → less Than 4.778 = outlier (Lower_value)

$Q_1 \quad IQR * 1.5 > [7.730]$ → greater Than 7.730 = outlier (Upper_value)

* Winsorizer

PIP install feature_engine

Winsorzier

From feature_engine . Outliers Import Winsozier

Library, Module, Function

In Winsozier (capping method)

Function, argument

* Capping method = 1. IQR
2. "Gaussian"

In feature_engine . Outliers Import ArbitaryOutlierCapper

Library, module, Function

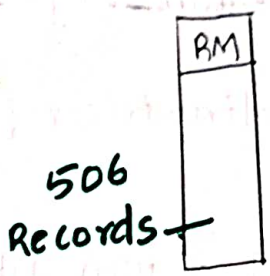
We can give our own Lower, upper Values.

1. By IQR method

In Depth of IQR.

Firstly,

 # we Calculate Q_1 Value



Q_1 = boston["RM"]. quantile (0.25)

Q_1

Out :- 5.855

$\therefore Q_1 = 5.885$

$Q_3 = 6.623$

 # we calculate Q_3 value.

Q_3 = boston ["RM"]. quantile (0.75)

Q_3

Out :- 6.623

$IQR = Q_3 - Q_1$

$= 0.737$

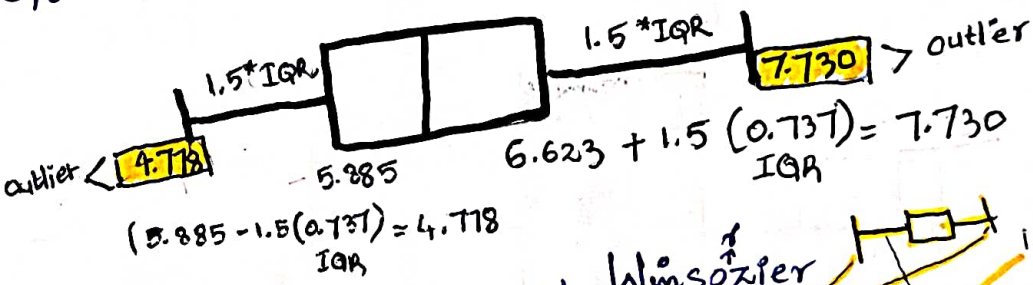
Distance between Q_3 and Q_1

 # IQR Value

IQR = $Q_3 - Q_1$

IQR

Out 0.737



CODE :-

From feature_engine.Outliers Import Winsozier

Win = Winsozier (Capping_method = "iqr", tail = 'both', fold = 1.5, Variables = ["RM"])

Random name

store in

boston_t => Win. fit_transform (boston[["RM"]])

original data

store in

Print (win. left_tail_caps_ , Win. right_tail_caps_)

4.778 7.730

Sns. boxplot (boston_t . RM)

plt. title ("boxplot")

plt. show()

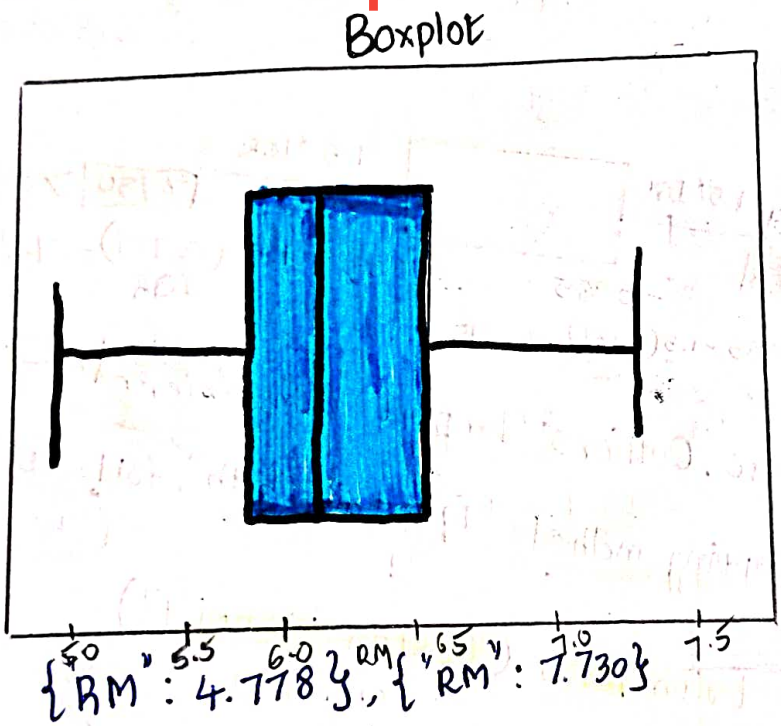
Edited (Replaced IQR). RM

again

```
From feature_engine.Outliers Import Winsorizer

win = Winsorizer (capping_method = "iqr", fold = 1.5, Tail = "both",
                  Variables = ["RM"])
boston_t = win.fit_transform (boston [["RM"]]

Print (win.Left_tail_caps_ , win.right_tail_caps_)

Sns.boxplot (boston_t.RM)
plt.title ("boxplot")
plt.show()
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

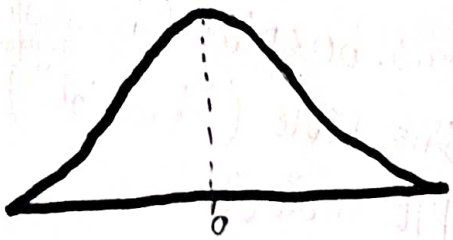
Out



2, Replacing using Winsorizer (min, max, automatically Taken by "Gaussian method")

Gaussian method = "Normal Distribution method"

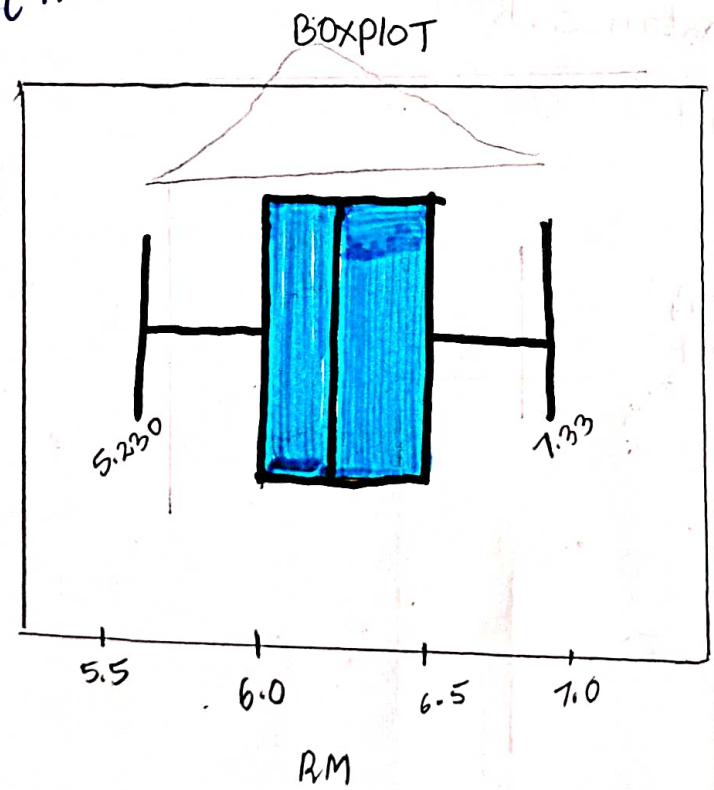
Same



```
# from feature_engine.Outliers import Winsorizer

# win = Winsozier (capping method = "gaussian", tail = "both",
                  fold = 1.5, variables = ['RM'])

boston_t = win.fit_transform (boston[["RM"]])

print (win.left_tail_caps_, win.right_tail_caps)

sns.boxplot (boston_t, RM)
plt.title ("Boxplot")
plt.show ()
```

Out

{'RM' : 5.230} {"RM": 7.33}



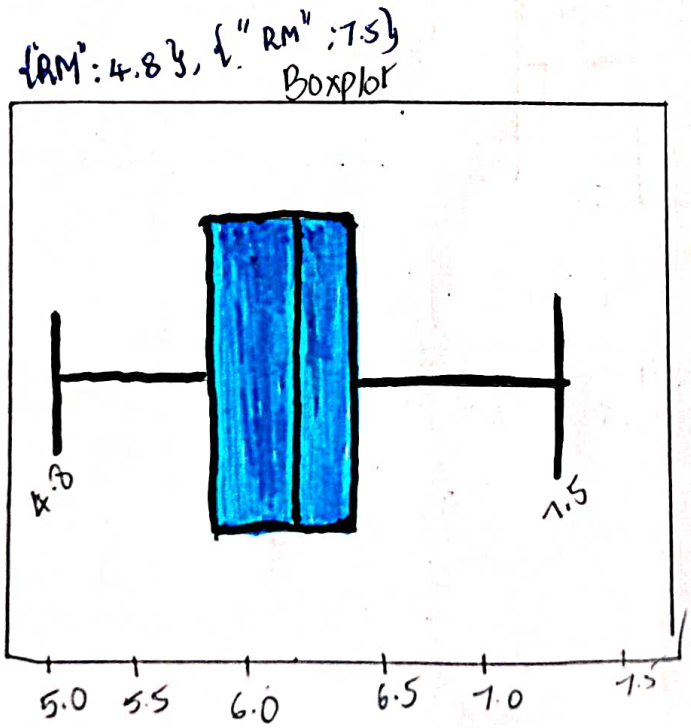
Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

3. Replace arbitary Outlier Capper (The min, max, by user)

we get the min and max values based on "Domain Expert" (or) by our Own "Research"

```
# From feature_engine.Outliers Import ArbitraryOutlierCapper
Capper = ArbitraryOutlierCapper (max_capping_dict = {"RM": 7.5},
                                 min_capping_dict = {"RM": 4.8})
boston_c = Capper.fit_transform(boston[["RM"]])
Print (capper.right_tail_caps_, Capper.left_tail_caps_)
Sns.boxplot (boston_c.RM)
plt.title ("Boxplot")
plt.show()
```

Out



3/4/20
5:20 pm

5/4/22
10:30pm

3 * Discretization *

Converting Continous Data To Discrete Data

also called as "Binning"

Code :-

```
Import pandas as pd

# stroke = pd.read_csv("stroke prediction.csv")

# stroke.head()
```

Out

id	gender	age	hyper Tension	heart-disease	Ever_married	Work type	Residence Type	avg-glocose Level	bmi	Smok status	Stroke
30669	Male	3.0	0	0	No	children	Rural	95.12	18.0	NaN	0
30468	Male	58.0	1	0	Yes	Private	urban	87.96	39.2	Never	0
16523	Female	8.0	0	0	No	Private	urban	110.89	17.6	NAN	0
56543	Female	70.0	0	0	Yes	Private	Rural	69.04	35.9	Formerly smoked	0
46136	male	14.0	0	0	No	Never_Worked	Rural	161.28	19.1	NAN	0
discrete (cat)	continous	discrete (const)	discrete (count)	discrete	discrete	discrete	continous	Contin	discrete	discre	

```
# stroke.shape
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Out : (43400,12)

```
# stroke.info()
```

Out information about stroke data.

⇒ Create Bins

intervals = [0, 12, 19, 30, 60, 90] → # bins ⇒ Each interval

categories = ["child", "teenager", "young adult", "middle aged", "Senior_citizen"]

Creating New column & storing the data of "Age

Storke ["Age_category"] = pd.cut [X = X = stroke ["Age"], bins = intervals, Labels = categories]

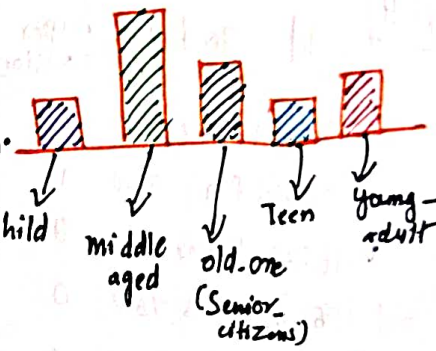
Divide.

The Age column is Continous Data. so, We converting into Discrete Variable

intervals → categoses

* 0-12 → child
* 12-19 → teenager
* 19-30 → young adult
* 30-60 → Middle aged
* 60-90 → Senior citizen

Storke ["Age category"]. hist()



stroke.head()

Changed Continous Data To Discrete Categorical data

Newly-added

Out

Id	gender	age	hyper Tension	Heart_disease	Ever married	work type	Residence Type	avg. glucose Level	bmi	Smoke	Stroke	Age_Category
30669	Male	3.0	-	-	-	-	-	-	-	-	-	child
30468	Male	58.0	-	-	-	-	-	-	-	-	-	middle_aged
16523	Female	8.0	-	-	-	-	-	-	-	-	-	child
56543	Female	70.0	-	-	-	-	-	-	-	-	-	Senior_citizen
46143	Male	14.0	-	-	-	-	-	-	-	-	-	teen

4 * Encoding *

Applicable For Categorical Variables (Discrete)

* Machine Can't Understand Text Data. So, We Convert the "discrete categorical" to "discrete Count" Data

* There are two types of Categorical Data

1. Nominal (No Natural Sequence)
2. Ordinal (Natural sequence)

we should Treat Every variable Equally.

Ex:- Grades

O	10
A+	9
A	8
B+	7
B	6
F	0

⇒ O > A+
10 > 9

odinal Data

cityname	
Hyd	0
Delhi	1
Mumbai	2
Kerala	3
Gujarat	4

In this case, we can't Decide by numbering which is greater given To state.

⇒ To convert Categorical Data To Numeric:

Nominal Data → (dummies)

1. get Dummies (Pandas)
2. One hot Encoding (sklearn)

Odinal Data

1. map (pandas)
2. Label Encoder (sklearn)

# One Hot Encoding	# Label Encoding
* Nominal	* Odinal

```
import Pandas as pd
import Numpy as np
import matplotlib.pyplot as plt

% matplotlib inline
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

```
# df = pd.read_csv("homeprices.csv")

# df.head()
```

Out :-

	Town	area	Price
0	chennai	2600	5500000
1	chennai	3000	5650000
2	chennai	3200	6100000
3	chennai	3600	6800000
4	Banglore	2600	5850000

Town

1. Chennai
2. Banglore
3. hyderabad

Discrete → Categorical Data

Nominal Data

⇒ **Pd.get_dummies** [Nominal Variable Encoding using pandas]

```
# dummies = pd.get_dummies(df["Town"])

# dummies
```

Out :-

	Banglore	Chennai	Hyderabad
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	0	0	1
9	0	0	1
10	0	0	1

Out :-

	Banglore	Chennai	Hyderabad
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	0	0	1
9	0	0	1
10	0	0	1
11	0	0	1

→ Dummies

which Ever Record is there it shows = 1

* For other columns it shows = 0

Adding → original → converted one → by

```
# df_dummies = pd.Concat([df, dummies], axis = "columns")
df_dumies
```

Out :-

	Town	Area	Price	Banglore	Chennai	Hyderabad
0	Chennai	2600	5500000	0	1	0
1	Chennai	3000	5650000	0	1	0
2	Chennai	3200	6100000	0	1	0
3	Chennai	3600	6800000	0	1	0
4	Banglore	3600	5850000	1	0	0
5	Banglore	2600	6150000	1	0	0
6	Banglore	2800	6500000	1	0	0
7	Banglore	3300	610000	1	0	0
8	Hyderabad	3600	7100000	0	0	1
9	Hyderabad	2600	5750000	0	0	1
10	Hyderabad	2900	6000000	0	0	1
11	Hyderabad	3100	6200000	0	0	1

Delete The Original column

```
# df_dummies.drop("town", axis = "columns", inplace = True)

df_dummies
```

Out:

	Area	Price	Banglore	chennai	Hyderabad
0	2600	5500000	0	1	0
1	3000	5650000	0	1	0
2	3200	6100000	0	1	0
3	3800	6800000	0	1	0
4	2600	5850000	1	0	0
5	2800	6150000	1	0	0
6	3300	7100000	1	0	0
7	3600	5150000	1	0	0
8	2600	6000000	0	0	1
9	2900	6200000	0	0	1
10	3100	6900000	0	0	1
11	3600	6500000	0	0	1

after deleting original column in New Data (ie. df_dummies) Old_data (df) is still remains Same.

Dummy Variable Trap

⇒

Bang	chenn	Hyder
0	1	0
0	1	0
1	0	0
1	0	0
0	0	1
0	0	1

→

Bang	chenn
0	1
0	1
1	0
1	0
0	0
0	0

→ hyderabad

If we Remove One of The column also, still we can Identify, col name by (0,0)

Town — 3 caterogies

↓ Dummies

3 columns [Multi-collinearity Problem / colleration]

If column → "n" categories

↓ We convert

(n-1) column

df_dummies.drop("chennai", axis="columns", inplace=True)

df_dummies

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

out

	Area	Price	Benglore	Hyderabad
0	2600	5500000	0	0
1	—	—	0	0
2	—	—	0	0
3	—	—	0	0
4	—	—	0	0
5	—	—	1	0
6	—	—	1	0
7	—	—	1	0
8	—	—	1	0
9	—	—	0	1
10	—	—	0	1
			0	1
11	36 000	6500000	0	1

(— Same) as old data

where ever Both column Having Equal Zero's, it i. another column' data.

Chennai = 0 0

We Can Write One line of Code To do all these

```
# df_dum = pd.get_dummies (df, drop_first = True)

# df_dum
```

Out:

	Area	Price	town_chennai	town_hyderabad
0			1	0
1			1	0
2			1	0
...			1	0
10			1	0
11				

This Diagram is Exact of (Previous) Last page diagram. But deleted banglore Replaced chennai

town_banglore [deleted] → Because, B ✓ (Alphabet order) chennai hyd

Replaced with (0 0)

⇒ One Hot Encoding → Nominal using sklearn

```
From sklearn.Preprocessing import OneHotEncoder

# enc = OneHotEncoder (drop = "First")
```

stores in enc

```
# enc_df = pd.DataFrame (enc.fit_transform
                         (df [["town"]]).
                         to array ())
```

converting

output

enc_df

Out

	0	1
0	1.0	0.0
1	1.0	0.0
2	1.0	0.0
3	1.0	0.0
4	0.0	0.0
5	0.0	0.0
6	0.0	0.0
7	0.0	0.0
8	0.0	1.0
9	0.0	1.0
10	0.0	1.0
11	0.0	1.0

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

merge with main df

```
# df_ohe = df.join(enc_df)
# df_ohe.drop("town", axis="columns", inplace=True)
# df_ohe
```

Out

	area	Price	0	1
0	2600	5500000	1.0	0.0
1	3000	5650000	1.0	0.0
2	–	–	1.0	0.0
3	–	–	1.0	0.0
4	–	–	0.0	0.0
5	–	–	0.0	0.0
6	–	–	0.0	0.0
7	–	–	0.0	0.0
8	–	–	0.0	1.0
9	–	–	0.0	1.0
10	–	–	0.0	1.0
11	–	–	0.0	1.0

Alphabetical Order

⇒ **Label Encoder** → **Ordinal Variable** [sklearn]

Used For binary category Variable

```
# dfnew = df.copy()
```
→ Creating "copy" From original Data.

```
# dfnew
```

Out

	town	Area	Price
0	Chennai	2600	5500000
1	chennai	3000	5650000
2	Chennai	3200	6100000
⋮	⋮	⋮	⋮
11	Hyderabad	3600	6950000

```
From sklearn.preprocessing import LabelEncoder

# Le = LabelEncoder()

# dfnew.town = Le.fit_transform(dfnew.town)

# dfnew
```

Out:

	town	Area	Price
0	1		
1	1		
2	1		
3	1		
4	0		
5	0		
6	0		
7	0		
8	2		
9	2		
10	2		
11	2		

Alphabetical Order

Bangalore [B] — 0
Chennai [C] — 1
Hyderabad [H] — 2

User Desired values we use this.

⇒ **map()** → **Ordinal Variable** [pandas]

map function.

```
# df_m = df.copy()
# df_m ["town"] = df_m ["town"]. map ({ "Chennai :0,
                "Banglore ":2 , "hyderabad": 1})

# df_m
```

Out :-

	town	Area	Price
0	0		
1	0		
2	0		
3	0		
4	2		
5	2		
6	2		
7	2		
8	1		
9	1		
10	1		
11	1		

Taken by our choices

Here

Chennai = 0, 20

Banglore = 2, 15

hyderabad = 1, 25

We can Replace with any Value

which Ever order we write it Takes in the order

⇒ **Ordinal Encoder** → **Ordinal Variable** (Ascending Order)

```
From sklearn. Preprocessing import OrdinalEncoder

# df_new1 = df.copy()
# oe = OrdinalEncoder (categories = [[ "Banglore", "Hyderabad", "Chennai"]])
# df_new1. town = oe. fit_transform (df_new1 [[ "town"]])

# df_new1
```

Out :-

	Town
0	2.0
1	2.0
2	2.0
3	2.0
4	0.0
5	0.0
6	0.0
7	0.0
8	1.0
9	1.0
10	1.0
11	1.0

6/4/22
3:00AM

Dt:- 3/4/22
11:30pm

"1" ⇒ **Feature Transformation** → Only For **Continous Data**

Feature processing : Transformation.

↓

columns / variables / Features (Same)

Ex:

X
2
4
9
12
25

$x^{0.5}$ √

$x^{1/2}$

$x^{1/3}$

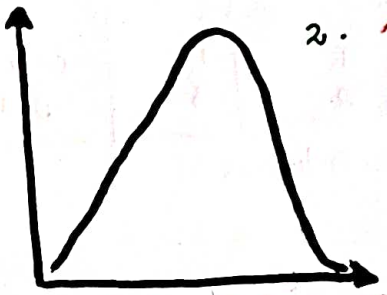
$x^{1/4}$

1. $[x]^{1/2}, [x]^{1/3}, [x]^{1/4}$
2. np.log

Right skewed

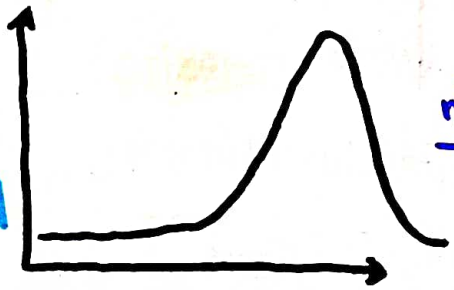
+1.5
+2
+6.9 .....

n^{th} root (or) log(x) →

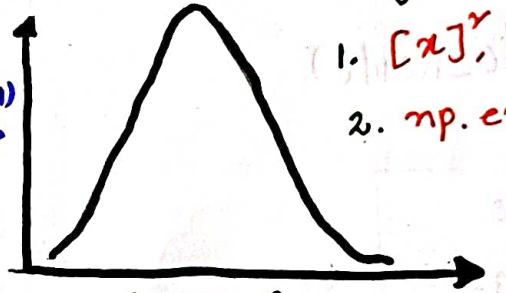


−1 to +1 (normal Distribution)

Left skewed



n^{th} power (or) exp →



−1 to 1

1. $[x]^2, [x]^3, [x]^4$
2. np.exp

import Liberaris / packages.

```
# import numpy as np
# Import pandas as pd
# Import matplotlib.pyplot as plt
# % matplotlib inline
# Import Seaborn as Sns
```

1. Root Transformations
2. log Transformations
3. Reciprocal Transformat
4. Boxcox Transformatio

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

titanic = pd.read_csv("titanic_csv")

location of csv

titanic

	Passenger Id	Survied	Pclass	Name	Sex	Age	Sibsp	Parch	Ticket	Fare
0	1	0	3	Owen Harris	male	22.0	1	0	A/52771	7.2500
1	2	1	1	cumring	Female	38.0	1	0	.	71.2833
2	3	1	3	Laina	Female	26.0	0	0	.	7.9250
3	4	1	1	Futrelle	Female	35.0	1	0	.	53.1000
	⋮									
889	890	1	1	Behr	male	26.0	0	0	.	30,0000
890	891	0	3	Dooley	male	32.0	0	0		7.7500

891 rows X 12 columns.

Cabin	Embarked
NAN	S
C85	C
NAN	S
C123	S
⋮	C
C148	

titanic = pd.read_csv("titanic.csv", Usecols= ["Fare"])

Particular column

Usecols = ["Fare"]

col. Name.

titanic_head()

Out

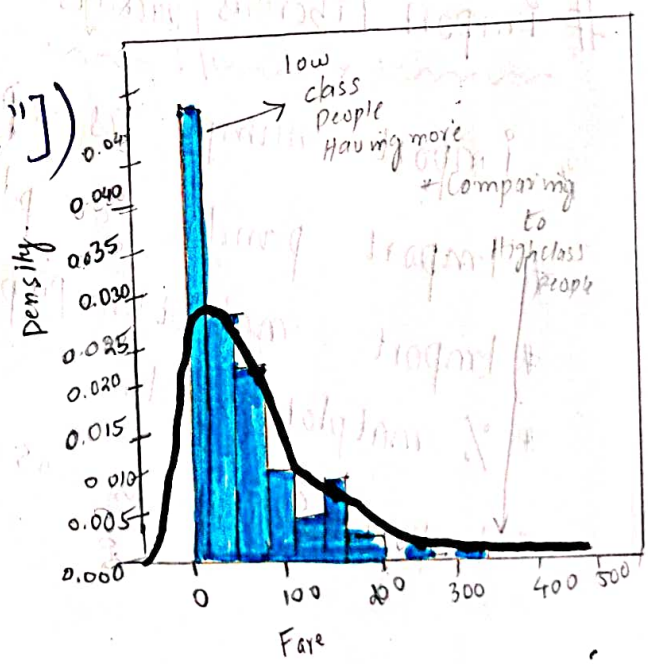
	Fare
0	7.2500
1	71.2833
2	7.9250
3	53.1000
4	8.0500

sns.distplot(titanic["Fare"])

plt.show()

titanic["Fare"].skew()

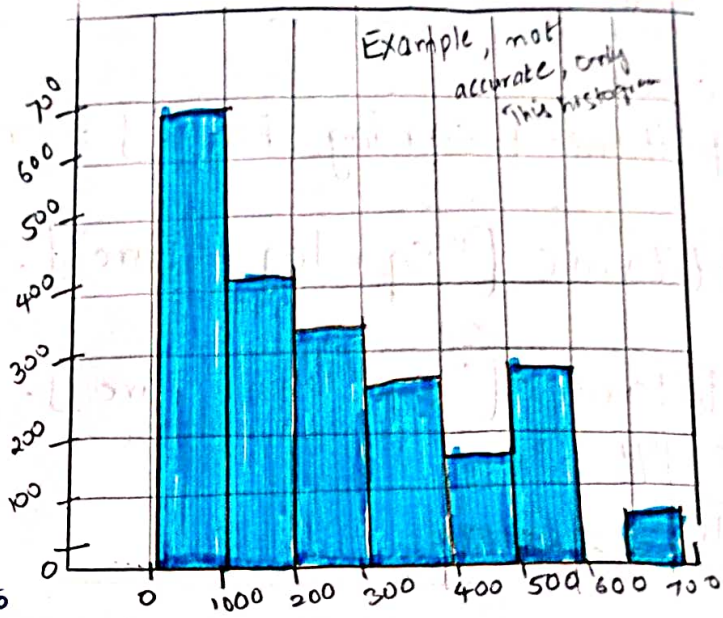
Out: + 4.7873



another way

```
t - ["Fare"].hist()
plt.show()
titanic["Fare"].skew()
```

Out :- +4.7873



Values

Skew = +2.085	(2th root)
Skew = +0.5	(4th root)
Skew = +0.21	(5th root)
Skew = -0.95	(6th root)

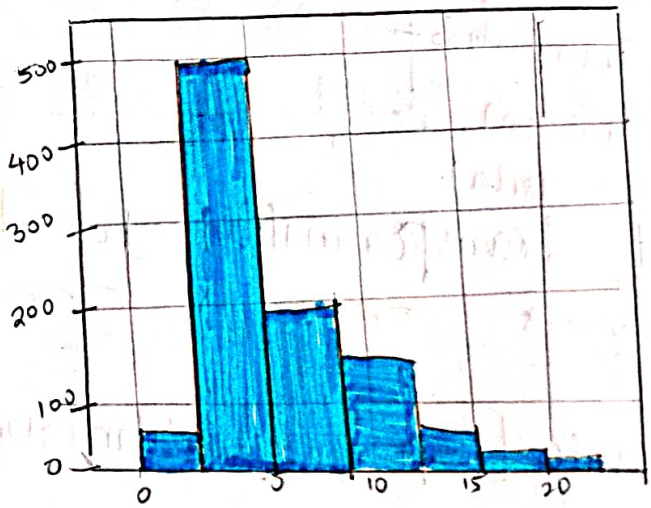
Consider, This which is close To "0".

* Root Transformation # For Right skewed

```
# titanic["sqr_Fare"] = titanic["Fare"]**(1/2)
# titanic["sqr-Fare"].skew()
# titanic["sqr-Fare"].hist()
```

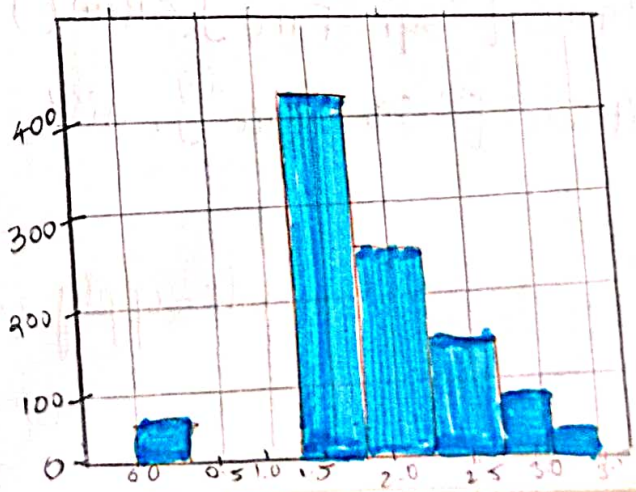
→ For Right skewed we use n^{th} root $\sqrt{\ }, \sqrt[3]{\ }, \sqrt[4]{\ }, \sqrt[5]{\ }$

Out :- +2.085 More Than +1 or -1



* Try with [1/5], [1/6]...

```
# titanic["sqr_Fare"] = titanic["Fare"]**(1/5)
# titanic["sqr_Fare"].hist()
# titanic["sqr_Fare"].skew()
```



Out :- -0.212676 Final

less Than -1 to +1

→ log Transformation

Q. Is, This mandatory, To write Every time?
- No, Only if we "0" in The Data.

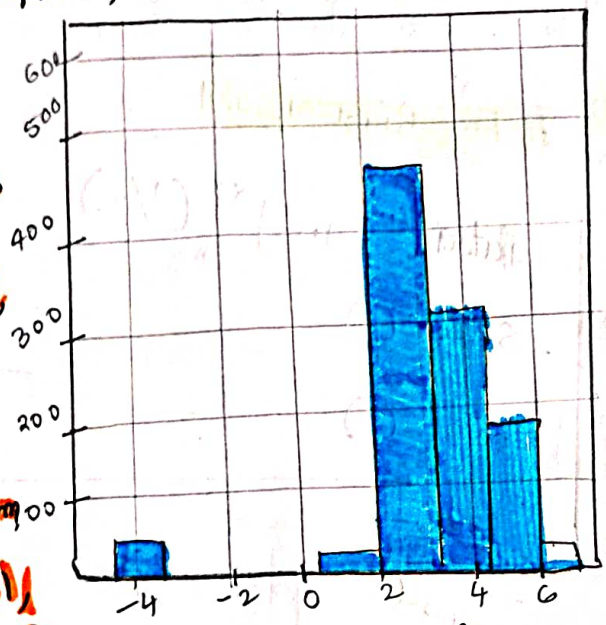
```
#titanic ["sqr log_"Fare"] = np.log(titanic["Fare"] +0.01)
#titanic ["sqr log_"Fare"].skew()
#titanic ["sqr log_"Fare"].hist()
# plt.show()
```

Why?
Because In Data "Fare" column, we have Some records "0" Values. So, If we calculate with log. it give ∞ (infinite) Value, So, we add +0.01 to "0" column.

Out

-2.41004

Here, The Value, in This Have more, So we don't use log Transform For This col, Data)



we use "n" power

Root Transformation For (Left Skewed) → n^{th} power → $(x)^2, (x)^3, (x)^4$ → left skewed
- n^{th} root $(x)^{1/2}, (x)^{1/3}, (x)^{1/4}$ → Right skewed

Fare data is not Left skewed, But To understand, Left skewed

```
titanic ["sqr_Fare"] = titanic["Fare"]**(2)
titanic ["sqr_Fare"].skew()
titanic ["sqr_Fare"].hist()
```

it can be any Value (2) (3) (4) . . . .

* when The Data is left skewed.

Out :

* We apply n^{th} power To The Data (or) column.

* Exponential

```
titanic["sqr_exp Fare"] = np.exp(titanic["Fare"])
titanic["sqr_exp Fare"].skew()
titanic["sqr_exp Fare"].hist()
plt.show
```

another method For Left Skewed Distribution

* Evaluates $e^{\wedge}x$ For Each Element in The given Input.

* Reciprocal Transformation

```
titanic["Rec_r"] = 1/(titanic["Fare"]+0.01)
titanic["Rec_Fare"].skew()
titanic["Rec_Fare"].hist()
plt.show()
```

"Reciprocal" is that We divide Every value with (1)/[column] +0.0

of The data.

to Elimi "0" value In The Data.

* Box Cox Transformation

Fare
7.250
71.283
-
-
-
-

$\rightarrow \frac{7.250^{\lambda}-1}{\lambda}$

$- \frac{71.283^{\lambda}-1}{\lambda}$

$$T[X] = \frac{X^{\lambda}-1}{\lambda}$$

∴ λ[lamba] Varies From -5 to +

∴ Where X is The response Variable and

⇒ In This Transformation, all Values o "λ" are Considered and The optimal Value for a given Variables are Selected.

from scipy import stats

stats.boxcox (titanic["Fare"] +0.01 / +1)

If we don't write This Value
Error: Data must be +ve
+0.01 (or) +1 (User defined)

Out :- (array([2.3844558, 6.43357655, 2.512739737, 2.607914, 5.7648427, 4.06768781 ...]))

Which ever gives best result

From help

we see Two return values

1. array values (boxcox)
2. optimal Lamda parameter (max log)

Fitting data, Fitting Lambda

titanic["Fare_boxcox"], param = stats.boxcox (titanic["Fare"]+1)

1 2

Multiple Variable assignment

Out :- a, b = _ _

stores

print ("λ=", param) (or) param

Out :- -0.09

titanic["Fare_boxcox"].skew()

Out :- -0.04

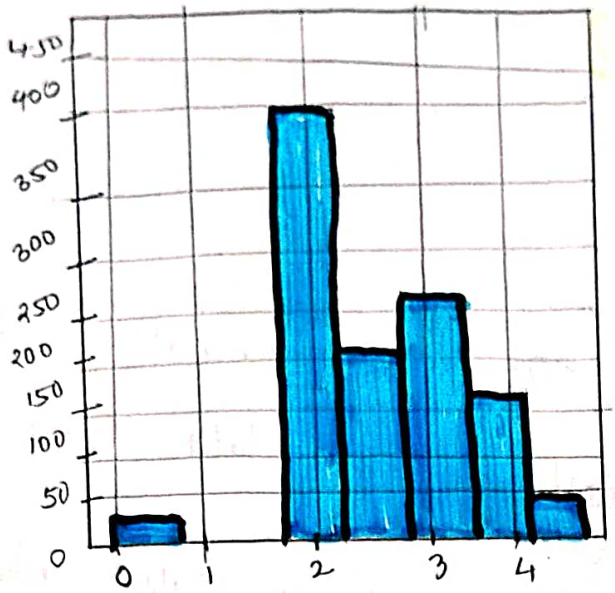
lowest values while Compare with other Transformations

Code

```
# titanic["Fare_boxcox"], Param = stats.boxcox (titanic.Fare +1)
# titanic["Fare_boxcox].hist()
# titanic["Fare_boxcox].skew()
```

Out :- -0.04

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666



4/4/22
5:30pm.

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G

5/4/22
11:00 am

"2" ⟹ Feature Scaling ⟹ Applicable only For Continous Data

Feature Scaling ?

Ans:- it refers (or) Techniques used to normalize The ranges of independent Variables in our data. (or)
input Variables

* The methods to set The Features value range within a similar Scale.
same scale

* Variables with bigger magnitude / Larger value range dominate over those with smaller magnitude / value range.

Ex:- 10,000,00 , 10
If we take 10 ——— 10 → Both are Equal Import

* Scale of The features is an important consideration when building Machine Learning models

* Feature Scaling is generally the last step in the data pre processing pipe line, Performed Just Before Training the machine Learning algorithms.

5. Gradient descent converges Faster when Features are on Similar scales

Example :-

C.C	Milage
1600	14
1800	15
2200	16.5
2100	18
2600	22.5
1800	19.4
1900	25.4

of a car

Continous Data

* Feature Scaling *

Q. When we ask Machine which is important In The both columns ?

* Machine is Giving Importance To C.C Because it is Having Large Values.

* But, We know both are importa

So, we have train model in Initially such that, it should consider "Both The columns Similar

So, we Reduce The Value

$$= \frac{800}{1000} = \frac{80}{100} = \frac{8}{10} = \frac{4}{5}$$

with 200

with 100

(Every one is Same)

We are scaling down Dividing Every value with "10" In this case.

* To Reduce The Value.

* Feature Scaling Importance in Some ML Algorithms

1. Linear Regression & logistic Regression

* The regression coefficients of linear models are Directly influenced by Scale of The Variable.

2. Support Vector Machines :-

* Feature scaling helps Decrease The Time to Find Support Vectors For SVM's

3. K-means clustering

* Euclidean distances are Sensitive to Feature magnitude

4. Principal component analysis (PCA) :-

* PCA requires The Features to be Centered at "0".

* Various Feature Scaling Techniques :-

1. Standardisation
2. Normalisation Technique
   - → Scaling to minimum and Maximum values [Min Max Scaling]
   - → Scaling to Maximum value [Max Abs Scaling]
   - → scaling to quantiles and Median [Robust Scaling]

```
# import pandas as pd
# import matplotlib.pyplot as plt
%. matplotlib inline.
```

→ Using (titanic Data set)

code:

```
# titanic = pd.read_csv("titanic.csv", usecols=["Age"
titanic.head()
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Out

Age
22.0
38.0
26.0
35.0
35.0

check Null Values in "Age" column.

```
# titanic.isnull().sum()
```

Out Age 177 → Null values

Replacing with "Median Value" For Null values

```
# titanic["Age"].fillna(titanic["Age"].median(),
                        inplace = True)

# titanic["Age"].isnull().sum()
```

Out : Age 0 → Zero Null Values

I. Standardisation

↓

Converting Each Value To Zscore

Ex :-

X	$\frac{x-\mu}{\sigma}$ = Zscore
23	$\frac{23-60}{2}$ = -18.5
45	$\frac{45-60}{2}$ = -7.5
67	$\frac{67-60}{2}$ = 3.5
89	$\frac{89-60}{2}$ = 14.5
78	$\frac{78-60}{2}$ = 9

These Five values converting To Zscores is called as "Standard" Scaling

Avg/mean = 60.

(Std) σ = 2

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Code :-

```
From sklearn.Preprocessing import StandardScaler (SC)
```

call The Function (Short Cut)

```
# SC = StandardScaler ()
```

main (don't Forget)

```
# titanic ["Age_SC"] = SC. fittranform (titanic [["Age"]
```

fit_transform

creating new column

Calculation

Converting The value

stores in

```
# titanic ["Age_SC"]
```

Out :-

	Age_SC
0	-0.5657
1	0.6638
2	-0.2583
3	0.43312
⋮	⋮
890	0.20272

Here Every Value is Converted To Zscore.

$$\left[\frac{X-\mu}{\sigma}\right] = Z \text{ score.}$$

Ex :- Age [22] # First record X=22

titanic.mean()

μ = 29.361

titanic.std ()

σ = 13.01

$$\left[\frac{X-\mu}{\sigma}\right] = \left[\frac{22-29.361}{13.01}\right]$$

$\Downarrow$

"Age_SC" → [-0.5657]

```
# titanic
```

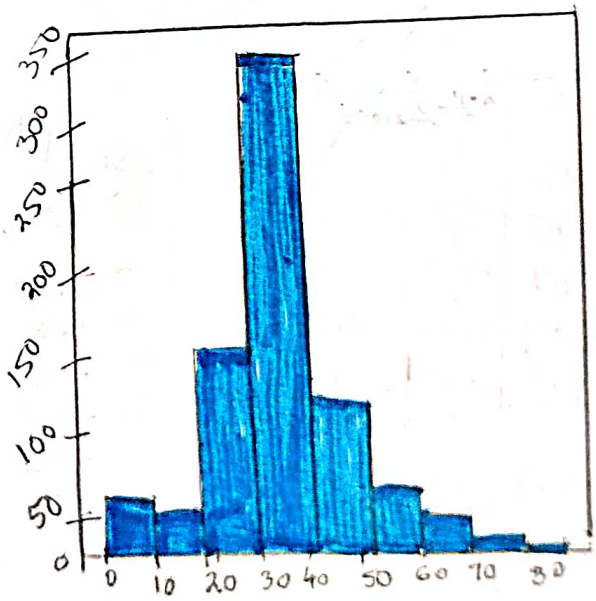
Out

	Age	Age_SC
0	22.0	-0.5657
1	38.0	-0.6638
2	25.0	-0.2583
⋮	⋮	
889	26.0	-0.2583
890	32.0	0.20272

Original Data

Converted To Z_Score

histogram of Original ["Age"]

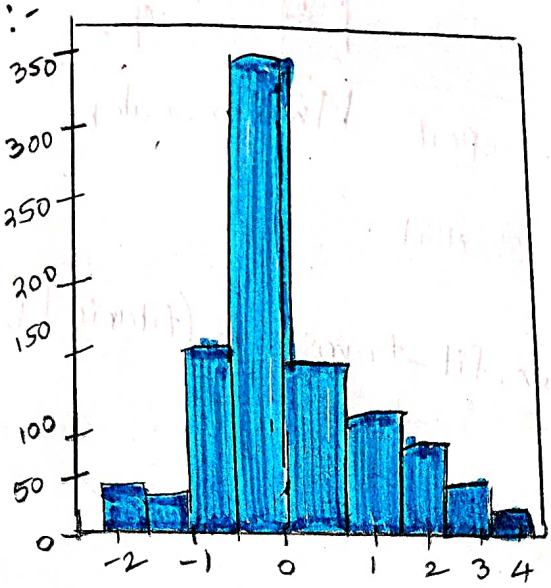



```
# titanic ["Age"]. hist()
plt. show
```

Histogram of Converted "Age"

```
# titanic["Age_sc"].hist()
# plt.show()
```

Out :-



Here, we see, Histogram is not going to change, But, Values does.

⇒ Example :-

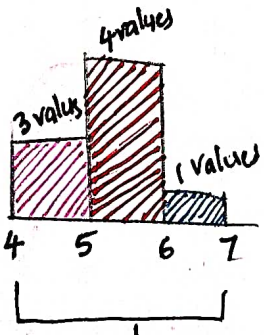
* How ? Histogram Remains Same, But changed Values

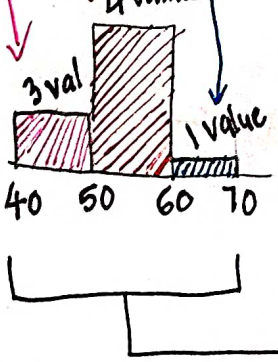
Ans :-

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Ex :-

X	Changed x
40	4
50	5
42	4.2
43	4.3
54	5.4
58	5.8
59	5.9
62	6.2





Here only "x" values Only changes But Histogram Remains Same.

II # Normalisation

* MinMax Scaling

⇒ MinMax Scaling Scales, The Values between "0" to "1" (Every Thing is (+ve) value No -ve value)

Ex:-

X	
65	$\frac{65-34}{53}$
45	$\frac{45-34}{53}$
34 (min. value)	$\frac{34-34}{53}=0$
67	$\frac{67-34}{53}$
67	
87 (Max. value)	$\frac{87-34}{53}=1$

$$\frac{X - X_{min}}{X_{max} - X_{min}}$$

Code: From sklearn.preprocessing import MinMaxScaler

```
# min_max = MinMaxScaler()  # Shortcut
# titanic["Age_mm"] = min_max.fit_transform(titanic[["Age"]])
```

creating new column

fit_transform

```
# titanic (calling/showing)
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Out :-

	Age	Age_SC	Age_mm
0	22.0	-0.565	0.271
1	38.0	0.663	0.472
2	26.0	-0.258	0.32
⋮	⋮	⋮	⋮
889	26.0	-0.25	0.32
890	32.0	0.20	0.39

* Robust Scaling

$$\frac{X - X_{median}}{IQR}$$

$\therefore IQR = Q_3 - Q_1$

Code :

```
From sklearn.preprocessing import RobustScaler

# rs = RobustScaler() → shorcut()
# titanic["Age_rs"] = rs.fit_transform(titanic[["Age"]])
```

titanic ["Age_rs"] (or) = RobustScaler().fit_transform(...)

If not using any shortcut like rs = RobustScaler()

```
# titanic
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Out

Age	Age_SC	Age_Mm	Age_rs
22.0	0.565	0.461	0.4615
38.0	0.66	0.769	0.769
26.0	-0.25	-0.153	-0.153
⋮			
26.0	-0.25	-0.153	-0.153
32.0	0.2	0.396	0.3076

---

* Max Abs Scaling

$$\frac{X}{X_{max}}$$

Ex:-

X	
68	68/80
44	44/80
52	52/80
69	69/80
80	80/80 = 1

Max value

code

```
From sklearn.preprocessing import MaxAbsScaler

# mas = MaxAbsScaler()

# titanic["Age_mas"] = mas.fit_transform(titanic[["Age"]])

# titanic
```

Out :-

Original Variable
Standard Scaler
Min max Scaler
Robust scaler
Max Abs Scaler

	Age	Age-sc	Age-mm	Age-rs	Age-mas
0	.	.	.	.	0.2750
1	.	.	.	.	0.4950
2	.	.	.	.	0.3250
					⋮
889	.	.	.	.	0.3250
890	.	.	.	.	0.4000

891 × 5 col

5/04/22
4:58 pm.

Example :- Discretization

Continous Data ⇒ Milage of a Car (Discrete. categorical) Data

12.5
22.8
25.2
18.3
12.4
15.8
23.3

<15 → low. milage
15-20 → Avg. milage
20-25 → good. milage
>25 → Very good

→ By our Own choice To Divide them into parts

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G

5/4/22
11:00 am

"2" ⟹ Feature Scaling ⟹ Applicable only For Continous Data

Feature Scaling ?

Ans:- it refers (or) Techniques used to normalize The ranges of independent Variables in our data. (or)
input Variables

* The methods to set The Features value range within a similar Scale.
same scale

* Variables with bigger magnitude / Larger value range dominate over those with smaller magnitude / value range.

Ex:- 10,000,00 , 10
If we take 10 ——— 10 → Both are Equal Import

* Scale of The features is an important consideration when building Machine Learning models

* Feature Scaling is generally the last step in the data pre processing pipe line, Performed Just Before Training the machine Learning algorithms.

5. Gradient descent converges Faster when Features are on Similar scales

Example :-

C.C	Milage
1600	14
1800	15
2200	16.5
2100	18
2600	22.5
1800	19.4
1900	25.4

of a car

Continous Data

* Feature Scaling *

Q. When we ask Machine which is important In The both columns ?

* Machine is Giving Importance To C.C Because it is Having Large Values.

* But, We know both are importa

So, we have train model in Initially such that, it should consider "Both The columns Similar

So, we Reduce The Value

$$= \frac{800}{1000} = \frac{80}{100} = \frac{8}{10} = \frac{4}{5}$$

with 200

with 100

(Every one is Same)

We are scaling down Dividing Every value with "10" In this case.

* To Reduce The Value.

* Feature Scaling Importance in Some ML Algorithms

1. Linear Regression & logistic Regression

* The regression coefficients of linear models are Directly influenced by Scale of The Variable.

2. Support Vector Machines :-

* Feature scaling helps Decrease The Time to Find Support Vectors For SVM's

3. K-means clustering

* Euclidean distances are Sensitive to Feature magnitude

4. Principal component analysis (PCA) :-

* PCA requires The Features to be Centered at "0".

* Various Feature Scaling Techniques :-

1. Standardisation
2. Normalisation Technique
   - → Scaling to minimum and Maximum values [Min Max Scaling]
   - → Scaling to Maximum Value [Max Abs Scaling]
   - → scaling to quantiles and Median [Robust Scaling]

```
# import pandas as pd
# import matplotlib.pyplot as plt
%. matplotlib inline.
```

→ Using (titanic Data Set)

code:

```
# titanic = pd.read_csv("titanic.csv", usecols=["Age"
titanic.head()
```

Out

Age
22.0
38.0
26.0
35.0
35.0

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

check Null Values in "Age" column.

```
# titanic.isnull().sum()
```

Out Age 177 → Null values

Replacing with "Median Value" For Null values

```
# titanic["Age"].fillna(titanic["Age"].median(),
                        inplace = True)

# titanic["Age"].isnull().sum()
```

Out : Age 0 → Zero Null Values

I. Standardisation

↓

Converting Each Value To Zscore

Ex :-

X	$\frac{x-\mu}{\sigma}$ = Zscore
23	$\frac{23-60}{2}$ = -18.5
45	$\frac{45-60}{2}$ = -7.5
67	$\frac{67-60}{2}$ = 3.5
89	$\frac{89-60}{2}$ = 14.5
78	$\frac{78-60}{2}$ = 9

These Five values converting To Zscores is called as "Standard" Scaling

Avg/mean = 60.

(Std) σ = 2

Exp

Code :-

```
From sklearn.Preprocessing import StandardScaler (SC)
```

call The Function (Short Cut)

```
# SC = StandardScaler ()
```

main (don't Forget)

```
# titanic ["Age_SC"] = SC. fittranform (titanic [["Age"]
```

fit_transform

creating new column

Calculation

Converting The value

stores in

```
# titanic ["Age_SC"]
```

Out :-

	Age_SC
0	-0.5657
1	0.6638
2	-0.2583
3	0.43312
⋮	⋮
890	0.20272

Here Every Value is Converted To Zscore.

$$\left[\frac{X-\mu}{\sigma}\right] = Z \text{ score.}$$

Ex :- Age [22] # First record X=22

titanic.mean()

μ = 29.361

titanic.std ()

σ = 13.01

$$\left[\frac{X-\mu}{\sigma}\right] = \left[\frac{22-29.361}{13.01}\right]$$

$\Downarrow$

"Age_SC" → [-0.5657]

```
# titanic
```

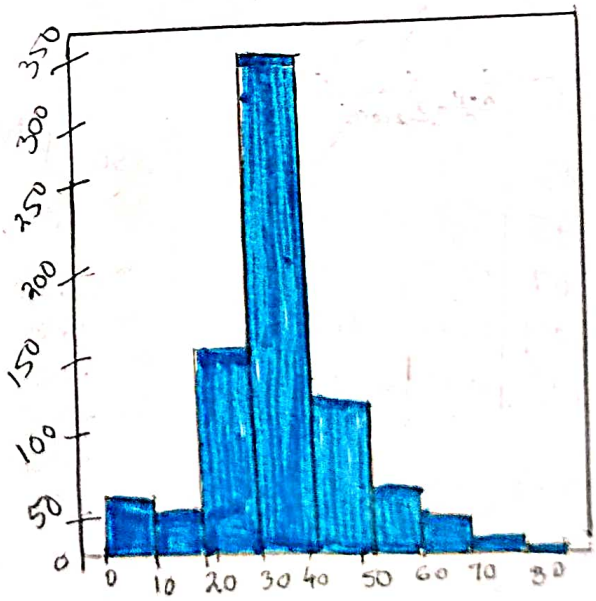
Out

	Age	Age_SC
0	22.0	-0.5657
1	38.0	-0.6638
2	25.0	-0.2583
⋮	⋮	
889	26.0	-0.2583
890	32.0	0.20272

Original Data

Converted To Z_Score

histogram of Original ["Age"]

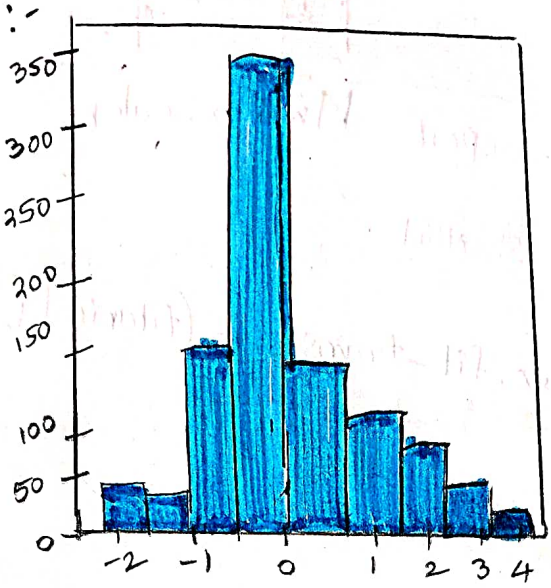



```
# titanic ["Age"]. hist()
plt. show
```

Histogram of Converted "Age"

```
# titanic["Age_sc"].hist()
# plt.show()
```

Out :-



Here, we see, Histogram is not going to change, But, Values does.

⇒ Example :-

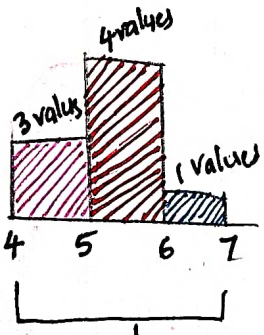
* How ? Histogram Remains Same, But changed Values

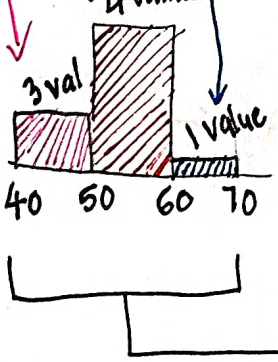
Ans :-

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Ex :-

X	Changed x
40	4
50	5
42	4.2
43	4.3
54	5.4
58	5.8
59	5.9
62	6.2





Here only "x" values Only changes But Histogram Remains Same.

II # Normalisation

* MinMax Scaling

⇒ MinMax Scaling Scales, The Values between "0" to "1" (Every Thing is (+ve) value No -ve value)

Ex:-

X	
65	$\frac{65-34}{53}$
45	$\frac{45-34}{53}$
34 (min. value)	$\frac{34-34}{53}=0$
67	$\frac{67-34}{53}$
67	
87 (Max. value)	$\frac{87-34}{53}=1$

$$\frac{X - X_{min}}{X_{max} - X_{min}}$$

Code: From sklearn.preprocessing import MinMaxScaler

```
# min_max = MinMaxScaler() # Shortcut
# titanic["Age_mm"] = min_max.fit_transform(titanic[["Age"]])
```

creating new column

fit_transform

```
# titanic (calling/show Data)
```

Out :-

	Age	Age_SC	Age_mm
0	22.0	-0.565	0.271
1	38.0	0.663	0.472
2	26.0	-0.258	0.32
⋮			
889	26.0	-0.25	0.32
890	32.0	0.20	0.39

* Robust Scaling

$$\frac{X - X_{median}}{IQR}$$

$\therefore IQR = Q_3 - Q_1$

Code :

```
From sklearn.preprocessing import RobustScaler

# rs = RobustScaler() → shorcut()
# titanic["Age_rs"] = rs.fit_transform(titanic[["Age"]])
```

titanic ["Age_rs"] (or) = RobustScaler().fit_transform(...)

If not using any shortcut like rs = RobustScaler()

```
# titanic
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Out

Age	Age_SC	Age_Mm	Age_rs
22.0	0.565	0.461	0.4615
38.0	0.66	0.769	0.769
26.0	0.25	-0.153	-0.153
⋮			
26.0	-0.25	0.153	0.153
32.0	0.2	0.396	0.3076

* Max Abs Scaling

$$\frac{X}{X_{max}}$$

Ex:-

X	
68	68/80
44	44/80
52	52/80
69	69/80
80	80/80 = 1

Max value

code

```
From sklearn.preprocessing import MaxAbsScaler

# mas = MaxAbsScaler()

# titanic["Age_mas"] = mas.fit_transform(titanic[["Age"]])

# titanic
```

Out :-

Original Variable
Standard Scaler
Min max Scaler
Robust scaler
Max Abs Scaler

	Age	Age_sc	Age_mm	Age_rs	Age_mas
0	.	.	.	.	0.2750
1	.	.	.	.	0.4950
2	.	.	.	.	0.3250
					⋮
889	.	.	.	.	0.3250
890	.	.	.	.	0.4000

891 × 5 col

5/04/22
4:58 pm.

Example :- Discretization

Continous Data ⇒ Milage of a Car (Discrete. categorical) Data

12.5
22.8
25.2
18.3
12.4
15.8
23.3

<15 → low. milage
15 - 20 → Avg. milage
20 - 25 → good. milage
>25 → Very good

→ By our Own choice To Divide them into parts

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Dt:- 3/4/22
11:30pm

"1" ⇒ **Feature Transformation** → Only For **Continous Data**

Feature processing : Transformation.

↓

columns / variables / Features (Same)

Ex:

X
2
4
9
12
25

$x^{0.5}$ √

$x^{1/2}$

$x^{1/3}$

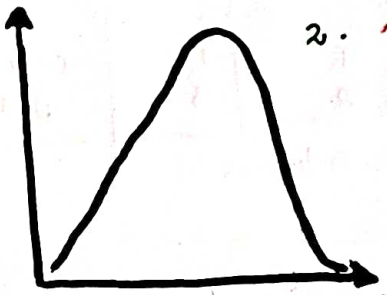
$x^{1/4}$

1. $[x]^{1/2}, [x]^{1/3}, [x]^{1/4}$
2. np.log

Right skewed

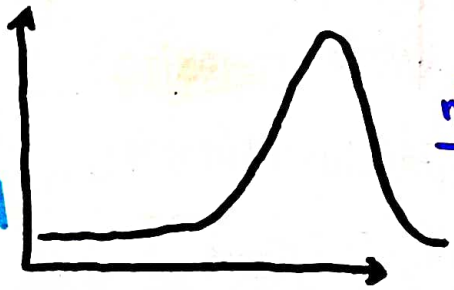
+1.5
+2
+6.9 .....

n^{th} root (or) log(x) →

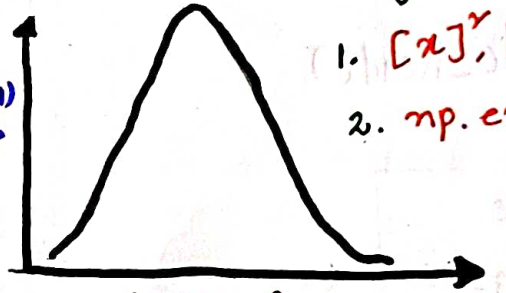


−1 to +1 (normal Distribution)

Left skewed



n^{th} power (or) exp →



−1 to 1

1. $[x]^2, [x]^3, [x]^4$
2. np.exp

import Liberaris / packages.

```
# import numpy as np
# Import pandas as pd
# Import matplotlib.pyplot as plt
# % matplotlib inline
# Import Seaborn as Sns
```

1. Root Transformations
2. log Transformations
3. Reciprocal Transformat
4. Boxcox Transformatio

titanic = pd.read_csv("titanic_csv")

location of csv

titanic

	Passenger Id	Survied	Pclass	Name	Sex	Age	Sibsp	Parch	Ticket	Fare
0	1	0	3	Owen Harris	male	22.0	1	0	A/52771	7.2500
1	2	1	1	cumring	Female	38.0	1	0	.	71.2833
2	3	1	3	Laina	Female	26.0	0	0	.	7.9250
3	4	1	1	Futrelle	Female	35.0	1	0	.	53.1000
	⋮									
889	890	1	1	Behr	male	26.0	0	0	.	30,0000
890	891	0	3	Dooley	male	32.0	0	0		7.7500

891 rows X 12 columns.

Cabin	Embarked
NAN	S
C85	C
NAN	S
C123	S
⋮	C
C148	

titanic = pd.read_csv("titanic.csv", Usecols= ["Fare"])

Particular column

Usecols = ["Fare"]

col. Name.

titanic_head()

Out

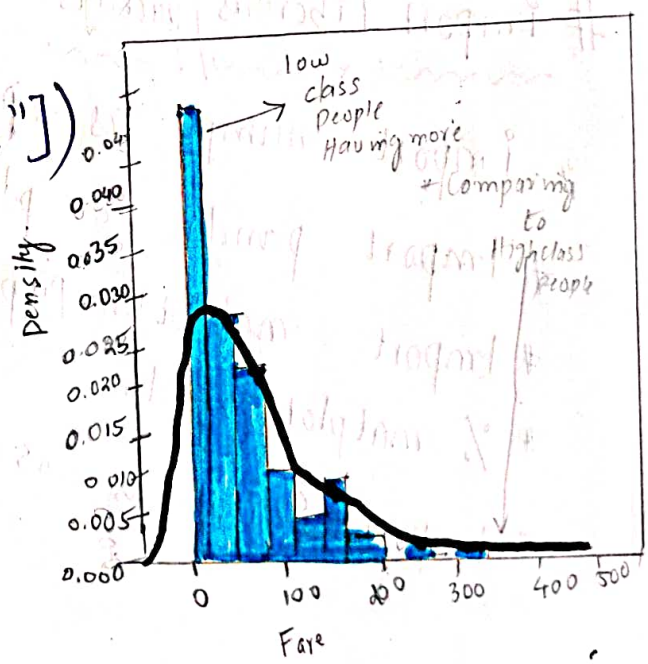
	Fare
0	7.2500
1	71.2833
2	7.9250
3	53.1000
4	8.0500

sns.distplot(titanic["Fare"])

plt.show()

titanic["Fare"].skew()

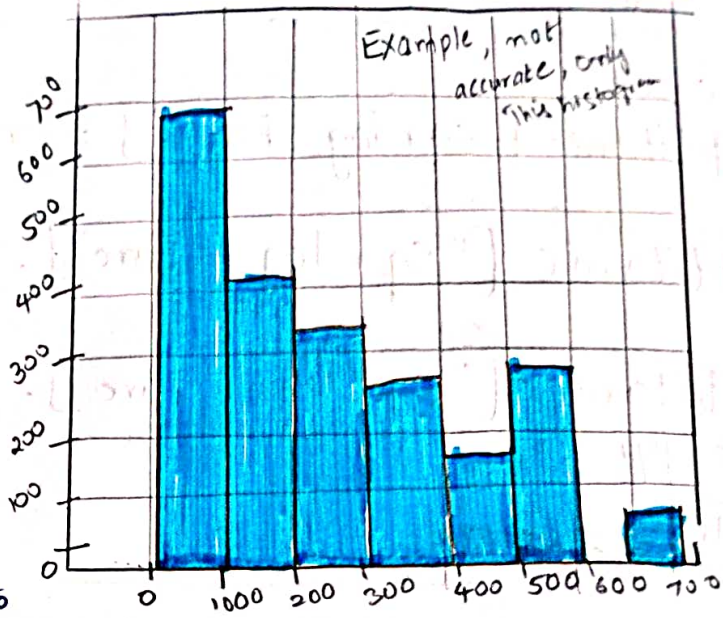
Out: + 4.7873



another way

```
t - ["Fare"].hist()
plt.show()
titanic["Fare"].skew()
```

Out :- +4.7873



Values

Skew = +2.085	(2th root)
Skew = +0.5	(4th root)
Skew = +0.21	(5th root)
Skew = -0.95	(6th root)

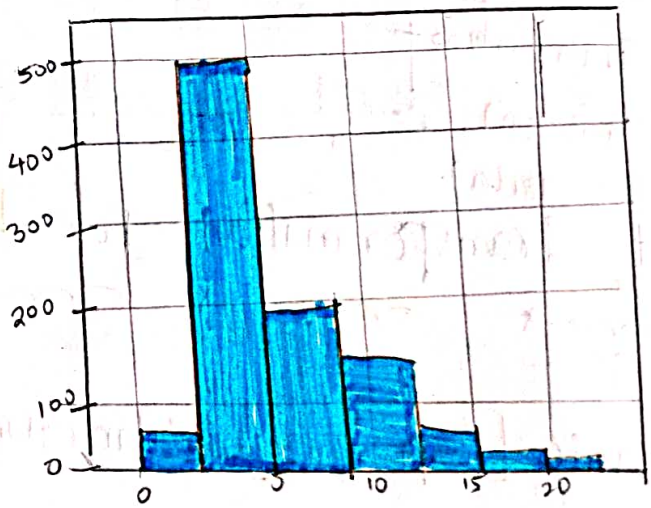
Consider, This which is close To "0".

* Root Transformation # For Right skewed

```
# titanic["sqr_Fare"] = titanic["Fare"]**(1/2)
# titanic["sqr-Fare"].skew()
# titanic["sqr-Fare"].hist()
```

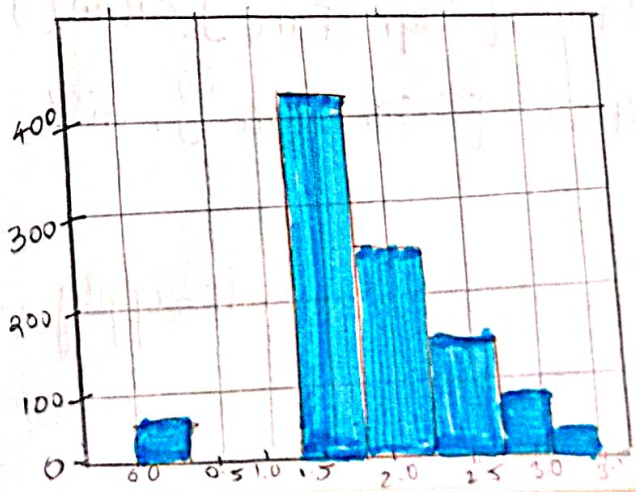
→ For Right skewed we use n^{th} root $\sqrt{\ }, \sqrt[3]{\ }, \sqrt[4]{\ }, \sqrt[5]{\ }$

Out :- +2.085 More Than +1 or -1



* Try with [1/5], [1/6]...

```
# titanic["sqr_Fare"] = titanic["Fare"]**(1/5)
# titanic["sqr_Fare"].hist()
# titanic["sqr_Fare"].skew()
```



Out :- -0.212676 Final

less Than -1 to +1

→ log Transformation

Q. Is, This mandatory, To write Every time?
- No, Only if we "0" in The Data.

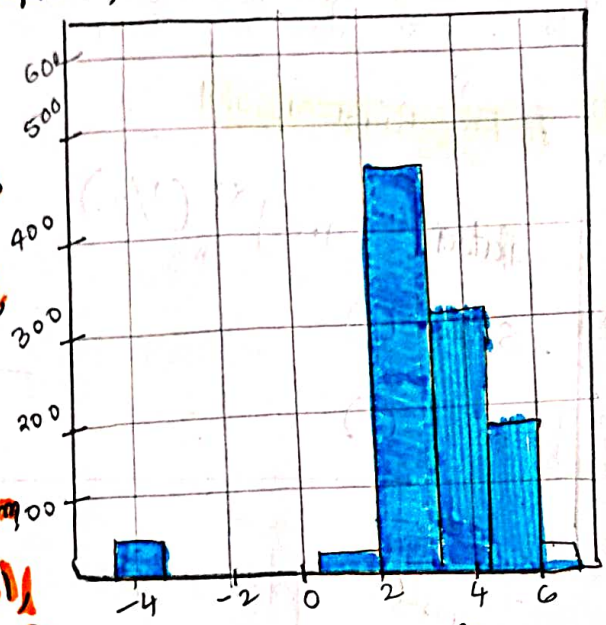
```
#titanic ["sqr log_"Fare"] = np.log(titanic["Fare"] +0.01)
#titanic ["sqr log_"Fare"].skew()
#titanic ["sqr log_"Fare"].hist()
# plt.show()
```

Why?
Because In Data "Fare" column, we have Some records "0" Values. So, If we calculate with log. it give ∞ (infinite) Value, So, we add +0.01 to "0" column.

Out

-2.41004

Here, The Value, in This Have more, So we don't use log Transform For This col, Data)



we use "n" power

Root Transformation For (Left Skewed) → n^{th} power → $(x)^2, (x)^3, (x)^4$ → left skewed
- n^{th} root $(x)^{1/2}, (x)^{1/3}, (x)^{1/4}$ → Right skewed

Fare data is not Left skewed, But To understand, Left skewed

```
titanic ["sqr_Fare"] = titanic["Fare"]**(2)
titanic ["sqr_Fare"].skew()
titanic ["sqr_Fare"].hist()
```

it can be any Value (2) (3) (4) . . . .

* when The Data is left skewed.

Out :

* We apply n^{th} power To The Data (or) column.

* Exponential

```
titanic["sqr_exp Fare"] = np.exp(titanic["Fare"])
titanic["sqr_exp Fare"].skew()
titanic["sqr_exp Fare"].hist()
plt.show
```

another method For Left Skewed Distribution

* Evaluates $e^{\wedge}x$ For Each Element in The given Input.

* Reciprocal Transformation

```
titanic["Rec_r"] = 1/(titanic["Fare"]+0.01)
titanic["Rec_Fare"].skew()
titanic["Rec_Fare"].hist()
plt.show()
```

"Reciprocal" is that We divide Every value with (1)/[column] +0.0

of The data.

to Elimi "0" value In The Data.

* Box Cox Transformation

Fare
7.250
71.283
-
-
-
-

$\rightarrow \frac{7.250^{\lambda}-1}{\lambda}$

$- \frac{71.283^{\lambda}-1}{\lambda}$

$$T[X] = \frac{X^{\lambda}-1}{\lambda}$$

∴ λ[lamba] Varies From -5 to +

∴ Where X is The response Variable and

⇒ In This Transformation, all Values o "λ" are Considered and The optimal Value for a given Variables are Selected.

from scipy import stats

stats.boxcox (titanic["Fare"] +0.01 / +1)

If we don't write This Value
Error: Data must be +ve
+0.01 (or) +1 (User defined)

Out :- (array([2.3844558, 6.43357655, 2.512739737, 2.607914, 5.7648427, 4.06768781 ...]))

Which ever gives best result

From help

we see Two return values

1. array values (boxcox)
2. optimal Lamda parameter (max log)

Fitting data, Fitting Lambda

titanic["Fare_boxcox"], param = stats.boxcox (titanic["Fare"]+1)

1 2

Multiple Variable assignment

Out :- a, b = _ _

stores

print ("λ=", param) (or) param

Out :- -0.09

titanic["Fare_boxcox"].skew()

Out :- -0.04

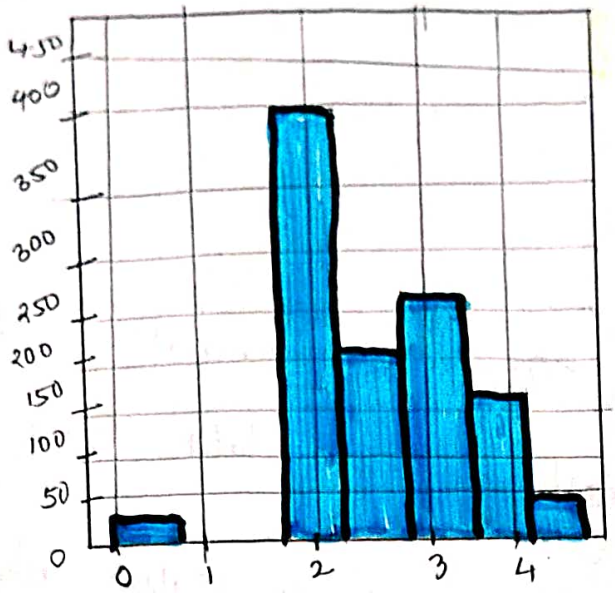
lowest values while Compare with other Transformations

Code

```
# titanic["Fare_boxcox"], Param = stats.boxcox (titanic.Fare +1)
# titanic["Fare_boxcox].hist()
# titanic["Fare_boxcox].skew()
```

Out :- -0.04

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666



4/4/22
5:30pm.

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G

Dt 26/04/22 10:06pm

Ensemble Learning

Ex :- Avengers.

Boosting Technique :- it is methodology, not a m.L Algorithm, it is applied to an Existing m.L, mostly applied on Decision Tree.

Ex :- Student

* Exam

⇒ not prepared well, Focus on wrong answers only

* Attempt Exam again

⇒ rectified but missed another section

* Again Exam

⇒ Total correctly done.

* Boosting :-

⇒ Decision Tree 1 :- misclassified records

⇒ Decision Tree 2 :- misclassified records

⇒ Decision Tree 3 :-

* $\boxed{2^{nd}}$ Weak learner is dependent on 1st weak learner

algorithm doesn't give more accuracy.

* Formula for Boosting :-

Sumation of t=1 to "T"

$$\boxed{F_T(x) = \sum_{t=1}^{T} f_t(x)} \rightarrow f_t(x) = \alpha_t \, h(x)$$

Learning rate

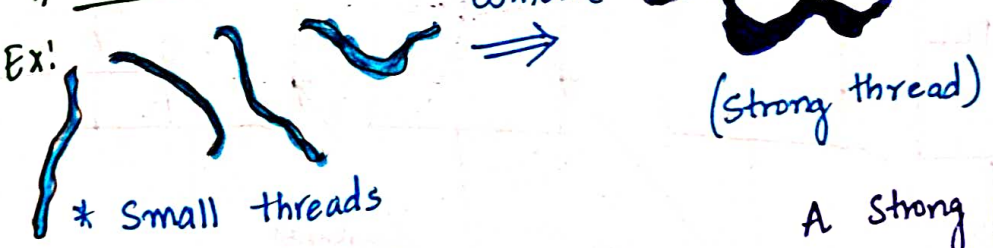
[Each model]

* boosting is the $\boxed{\text{Combination}}$ of weak learners:

* Implies that Combination of Estimators (models) with an applied Coefficient Could act as an effective Ensemble Estimator.

Boosting

Ex:

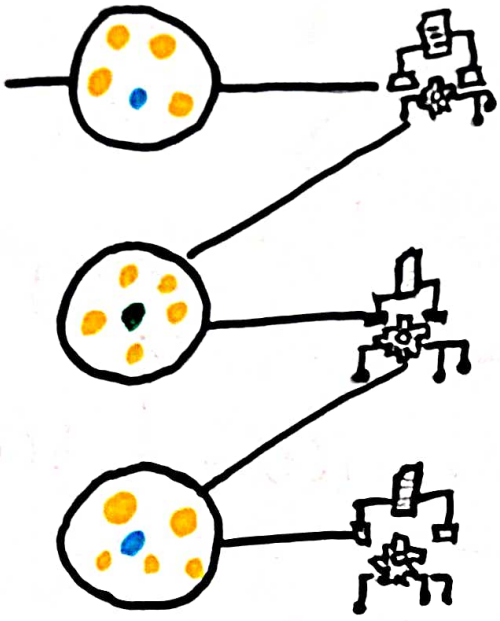


* Small threads

A Strong Classifier

* Multiple Weak classifer (Stumps)

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666



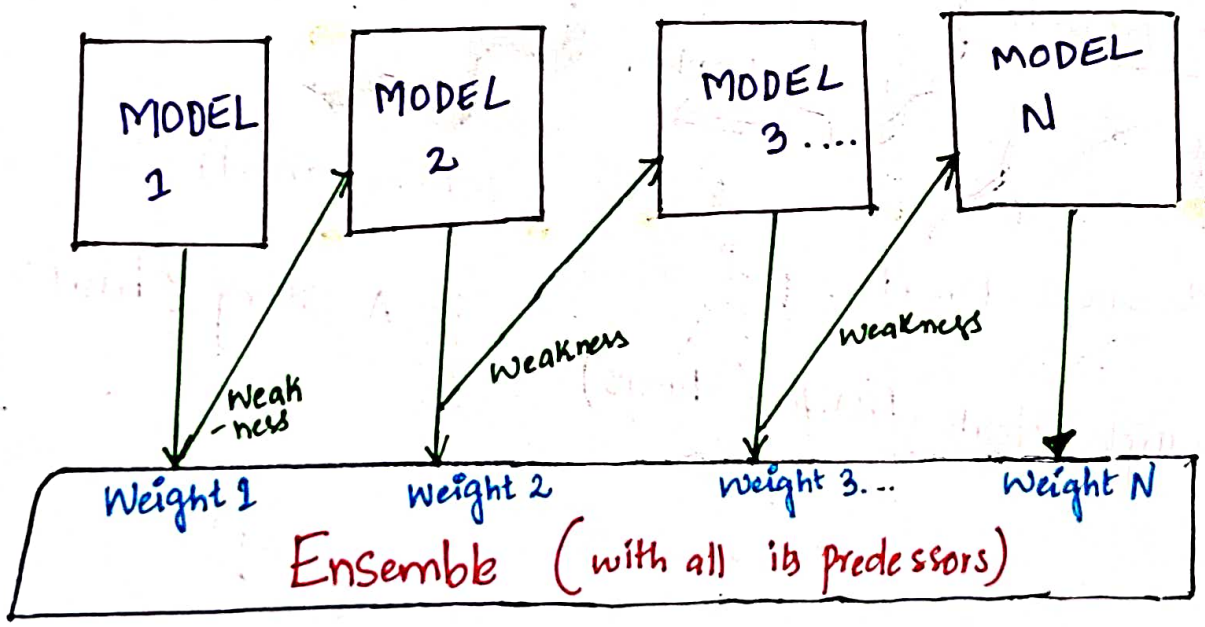
"Sequential learning":

* From the first algorithm whatever mistakes was done
* we share to the second algorithm.
* again it shared to third algorithm. - - -
* it is in "Sequential order"

Boosting - "Sequential" Learning.

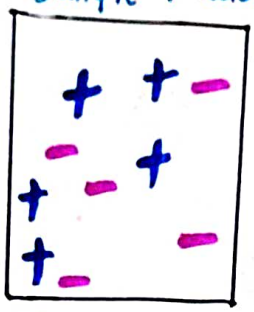
1. Ada Boost → Boosting.
   |
   Adaptive

Ex:- Model, 1, 2, - - - N are individual models (eg. Decision Tree)



Ex:- covid (Positive / negative)

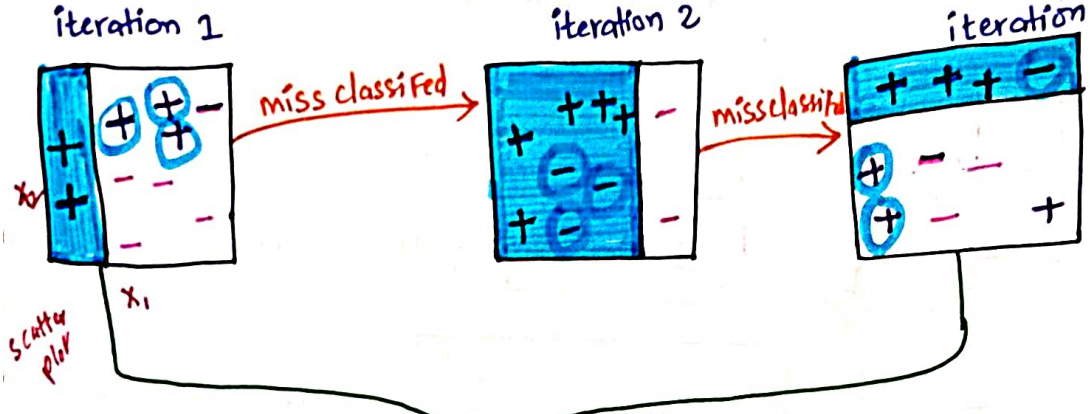
Sample DataSet

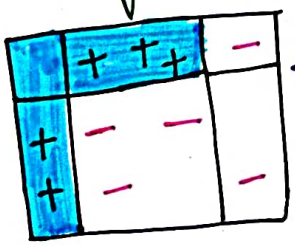


Weak Learner 1 — iteration 1

Weak learner 2 — iteration 2

Weak learner 3 — iteration 3.

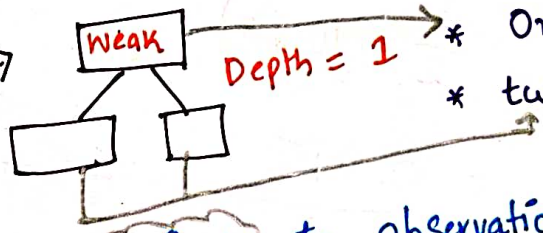




Final classifier / strong classifier.

* In Adaboost :- Each weak learner is called "stump"

* Stump ⇒



* One root node
* two Leaf nodes.

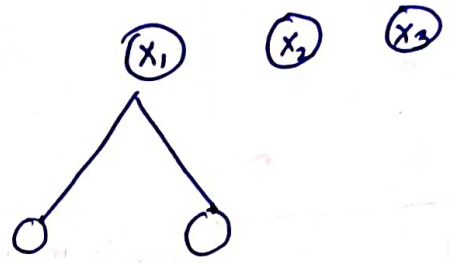
* Step 1 :- Adaboost assigns weights to observations which are incorrectly predicted

* Step 2 :- Next model works to predict these Values correctly which are wrongly predicted before.

Ex:-

Taken by Entropy / Information gain

	X_1	X_2	X_3	X_4	O/P	weights
1	–	–	–	–	Yes	1/7 ✓
2	–	–	–	–	No	1/7 ✓
3	–	–	–	–		1/7 ✓
×4	–	–	–	–		1/7 ×
5	–	–	–	–		1/7 ✓
6	–	–	–	–		1/7 ✓
7	–	–	–	–		1/7 ✓



{weak Learner}

*1. Total Error [TE] :- $\frac{1}{7}$:- $\boxed{\varepsilon_t = \varepsilon_w}$ ∴ ε_t = "Episinol" T

2. Performance of stump :- $\boxed{\frac{1}{2}\log_e\left[\frac{1-TE}{TE}\right]}$ ∴ TE = Total Error

$$= \frac{1}{2}\log_e\left[\frac{1-\frac{1}{7}}{\frac{1}{7}}\right]$$

$$= \frac{1}{2}\log_e\left(\frac{6/\cancel{7}}{1/\cancel{7}}\right)$$

$$= \frac{1}{2}\log_e(6)$$

$$= 0.895$$

New Sample weight $= \boxed{\text{Weight} * e^{-Ps}}$ → correct records ∴ Ps = Performance Stump

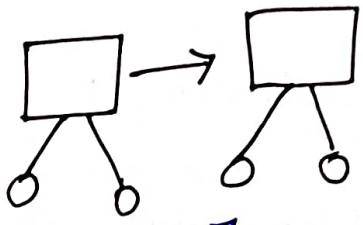
$$= \frac{1}{7} * e^{-0.895} = 0.05$$

+ incorrect record $= \boxed{\text{weight} * e^{Ps}}$ → incorrect records.

$$= \frac{1}{7} * e^{0.895} = 0.349$$

is it 1 ??

Weights	New weights	Normalized weights	Buckets
$\frac{1}{7}$	0.05 ÷ 0.649	0.077	(0 - 0.07)
$\frac{1}{7}$	0.05 ÷ 0.649	0.077	(0.07 - 0.14)
$\frac{1}{7}$	0.05	0.077	(0.14 - 0.21)
$\frac{1}{7}$	0.349	0.537	(0.21 - 0.747) ✓
$\frac{1}{7}$	0.05	0.077	[1.747 - 0.751) ✓
$\frac{1}{7}$	0.05	0.077	—
$\frac{1}{7}$	0.05	0.077	—
	0.649	~ 1	



$\therefore$ 1.537 + 0.21 = 1.747

* Steps :-

⇒ weight $= \frac{1}{n}$

⇒ Total Error_t $= \frac{[\text{correct Prediction} - n]}{n}$

⇒ Total weighted error_t $= \frac{\text{Sum}(\text{weight}(i) * \text{Error}(i))}{\text{Sum}(\text{weight})}$

⇒ $\alpha_t = \frac{\ln[1 - TE]}{TE\,[\text{Total Error}]}$

⇒ weight = weight * exp (α_t * Total error)

⇒ total error = 0 if (y == y^{hat}), else 1.

* Parameters

* n_estimors :- no. of weak learners
* Cretierion :- Gini / Entropy
* Max_Feature :- all Feature / selected ones
* Max_depth :
* min_samples_split
* n_jobs :- System is quadcore :- no. of cores = 4
  dual core :- no. of cores = 2
  octo core : no. of cores = 8
  -1 (all cores)
* Random_state :
* Learning rate :

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G

Dt: 11:30 Am

CODE :- AdaBoost

Data :-

This Data set includes descriptions of hypothetical Samples Corresponding to 23 species of gilled mushrooms in agaricus and Lepiota Family (pp 500-525). Each species in indentified as definetly edible, definitely poisonous, or of unknown edibility and not recommended. this latter class was was combined with the poisinous one. the Guide Clearly states that there is no simple rule for determing the edibility of a mushroom; no rule like "leaflets three, Let it be" for poisinous Oak and Ivy.

→ Mushroom

Attribute Information

1. cap_shape : bell = b, Conical = c, Convex = x, flat = f, Knobbed = k, Sunken = s

. Cap_Surface : fibrous = f, grooves = g, Scaly = y, Smooth = s

Cap_colour : brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, Yellow = y.

ruises ? : bruises = t, no F

dor : almond = a, anise = l, creoste = c, fishy = y, foul = f. musty = m, none = n, punget = p, spicy = s

6. gill-attachment : black = k, brown = n, buff = b, chocolate = h, gray = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y

7. gill-spacing : close = c, crowded = w, distant = d

8. gill-size : broad = b, narrow = n

1. gill-color : black = k, brown = n, buff = b, chocolate = h, grave = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y.

. stalk-shape : enlarging = e, tapering = t

stalk-root : bulbous = b, club = c, cup = u, equal = e, rhizomorphs = z, rooted = r, missing = ?

Stalk Surface-above-ring : fibrous = f, scaly = y, silky = k, smooth = s

Stalk Surface-below-ring : fibrous = f, scaly = y, silky = k, Smooth = s

Stalk-colour-above-ring : brown = n, buff = b, Cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y.

stalk-colour-below-ring : "

Veil-type : Partial = p, universal = u

Veil-color : brown = n, Orange = o, white = w, Yellow = y

18. ring-number :- none = n, One = o, two = t

19. ring-type :- cobwebby = c, evanescent = e, flaring = f, large = l, none = n, Pendent = p, sheathing = s, zone = z,

20. Spore-print colour :- black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, Purple = u, white = w, Yellowy

21. Population :- abundant = a, clustered = c, numerous = n, Scattered = s, Several = v, solitary = y

22. habitat :- grasses = g, Leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d.

23. class (Output)

Bussiness Problem

Goal here is to see if we can harness the power of machine Learning and boosting to help create not just a predective model, but general Guideline for features People should look out for when picking mushrooms.

```
# df = Pd.redd_csv("mushrooms.csv")
# df.head()
```

class	cap shape	cap surface	cap color	bruises	odor	gill attachment	gill spacing	gill size	gill color ...	stalk surface above Ring	stalk color above ring	stalk color below ring	veil type	veil color	ring number	ring type	Spore print color	Population	habitat
p	x	s	n	t	p	f	c	n	k	s	w	w	p	w	o	p	k	s	u
e	x	s	y	t	a	f	c	b	k	s	w	w	p	w	o	p	n	n	g
e	b	s	w	t	l	f	c	b	n	s	w	w	p	w	o	p	n	n	m
p	x	y	w	t	p	f	c	n	n	s	w	w	p	w	o	p	k	s	u
e	x	s	g	f	n	f	w	b	k	s	w	w	p	w	o	e	n	a	g

df.shape

Out : [8124, 23]

df.info()

Out : Range Index : 8124 Entries, 0 to 8123

Column	Non-null Count	Dtype.
* class	8124 Nonnull	Object
* cap-shape	"	"
* cap Surface	"	"
⋮		
* habitat	"	"

df.isnull().sum()

Out : 0.

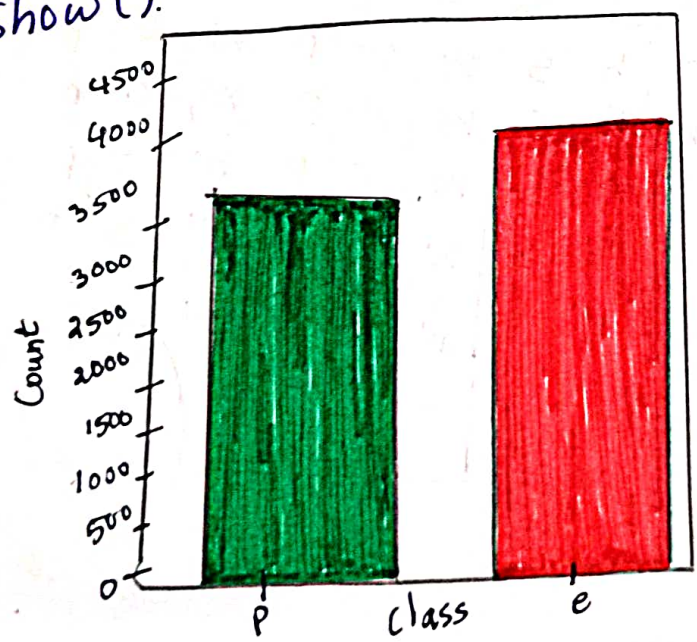
Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

EDA

: sns.countplot(data=df, X="class", palette="Dark2")
output

plt.show().

Out



Transpose. gives total no. columns. without....

```
# df.describe().transpose()
```

Out

	Count	Unique	top	Freq
(o/p) Class	8124	2	e	4208
cap Shape	8124	6	x	3656
cap- Surface	8124	4	y	3244
cap- color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill attachment	8124	2	f	7914
gill spacing	8124	2	c	6812
gill size	8124	2	b	5612
gill - color	8124	12	b	1728
Stalk- Shape	8124	2	t	4608
Stalk - root	8124	5	b	3776
Stalk - Surface - above ring	8124	4	s	5176
Stalk - Surface - below ring	8124	4	s	4936
Stalk - color above ring	8124	9	w	4464
Stalk - color below ring	8124	9	w	4384
Veil - type	8124	1	p	8124
Veil - color	8124	4	w	7924
ring - number	8124	3	o	7488
ring - type	8124	5	p	3968
Spore - Print color	8124	9	w	2388
Population	8124	6	v	4040
habitant	8124	7	d	3148

* posinous
* non posinous

$\boxed{X \,\&\, Y}$

Nominal ... (nominal data) ⇒ One hot Encoding

```
# X = pd.getdummies(df.drop("class", axis=1), drop_first = True)

# y = df["class"]
```

```
# x
```

Out

cap shape -c	cap shape -f	cap shape -k	cap shape -s	cap shape -x	cap surface -g	cap surface -s	cap surface -y	cap color -c	cap color -e	...	Population -a	Population -s	Population -v	Population -y	habitat -g	habitat -l	habitat -m	habitat -p	habitat u
0	0	0	0	1	0	1	0	0	0		0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0		1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0		1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	0		0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0		0	0	0	0	1	0	0	0	0

8124 rows x 95 columns

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

```
# y
```

Out:

0	p
1	e
2	e
3	p
4	e

```
# x.shape, y.shape
```

Out: (8124, 95), (8124,)

train / test split

```
from sklearn.model_selection import train_test_split
# x_train, x_test, y_train, y_test = train_test_split (x, y,
                                test_size = 0.2, random_state=29)
```

MODELLING

```
from sklearn.ensemble import AdaBoostClassifier
# model = AdaBoostClassifier ()
```

→ n_estimator = 100 (stumps)

```
# model.fit (x_train, y_train)
```

out : AdaBoostClassifier()

PREDICTION

```
# ypred_train = model.predict (x_train)
# ypred_test = model.predict (x_test)
```

Evaluation

* **Accuracy**

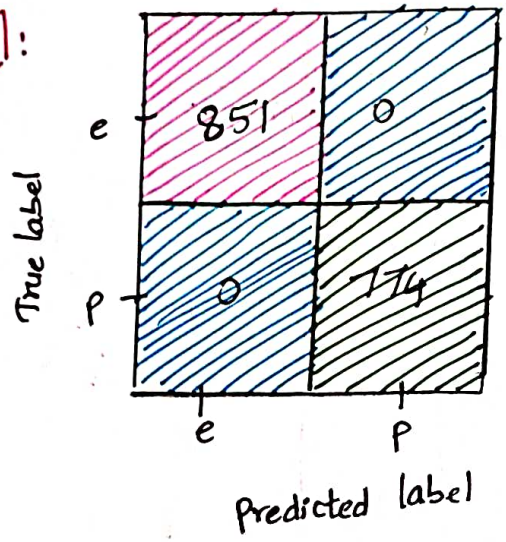
```
from sklearn.metrics import accuracy_score
# Print ("Train accuracy" : , accuracy_score (y_train, ypred_train)
# print ("Test accuracy": , accuracy_score (y_test, ypred_test)
```

out : Train Accuracy : 1.0
Test Accuracy : 1.0

* Confusion Matrix

```
from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix (model, x_test, y_test)
# plt.show()
```

Out:



* Classification Report

```
from sklearn.metrics import classification_report
# print (classification_report (y_test, y_pred_test))
```

Out:

	Precision	recall	f1 Score	Support
e	1.00	1.00	1.00	851
p	1.00	1.00	1.00	774
Accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
weighted Avg	1.00	1.00	1.00	1625

* **Cross-Validation-Score**

```
from sklearn.model_selection import cross_val_score.

# scores = cross_val_score (model, x, y, cv=5)

# print (" cross validation score:", scores.mean())
```

Out : cross validation score : 0.925

* **model.feature.importances**

```
# model.feature_importances_
```

Out array([0. , 0. , 0. 0, 0. , 0, 0. , 0. ,
0. . 0, 0. , 0.02, 0. 0.04,
0.06, 0.02, 0.0, 0, 0.1, 0.12,
0]

```
# f_imp = pd.DataFrame (index = X.columns, data = model.feat
                -re = importances_ columns = ["Feature Import
                                                ances"
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

```
# f_imp
```

Out:

	Feature Importances
cap_shape_C	0.00
cap_shape_f	0.00
cap_shape_K	0.00
cap_shape=s	0.00
cap_shape_x	0.00
⋮	⋮

95 rows x 1 columns

Data Extraction (Pandas)

```
# f_imp [ f_imp ["feature importance"] > 0 ]
```

Out :

	Feature Importance
cap_color w	0.02
odor_c	0.04
odor_f	0.04
odor_n	0.06
odor_p	0.02
gill_spacing w	0.10
⋮	⋮

Hyper Parameter Tuning

```
from sklearn.model_selection import GridSearchCV
# estimator = AdaBoostClassifier()
# param_grid = {"n_estimators": list(range(1,101))}
# grid = GridSearchCV(estimator, param_grid, cv=5, scoring="Accuracy")
# grid.fit(x_train, y_train)
# grid.best_params_
```

Out : {"n_estimator : 20}

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

final model

```
# final_model = AdaBoostClassifier(n_estimators = 20)
# final_model.fit(x_train, y_train)

# Pred_train = final_model.predict(x_train)
# Pred_test = final_model.predict(x_test)

# print("train accuracy:", accuracy score(y_train, Pred_train))
# print("test accuracy:", accuracy score(y_test, Pred_test))
```

Out: train accuracy : 1.0
Test accuracy : 1.0

28/04/22
10:00pm

27/4/22
4:30pm.

```
# final model.feature_importances_
```

Out: array([0. , 0. , 0. , 0. 0.
0.0434 0. 0, 0, 0.
0 0 0 0 0.434...])

```
# f_imp2 = pd.DataFrame(index = X.columns, data = final model
.feature_importances_, columns = ["importance_
Feature"])
```

f_imp2

Out :

	importance_Feature
cap_shape_c	0.0
cap-shape-f	0.0
⋮	
habitat_p	0.0
habitat_u	0.0

important = f_imp2 [f_imp2 ["importance_feature"] > 0]

important . Sort_Values ("importance_feature")

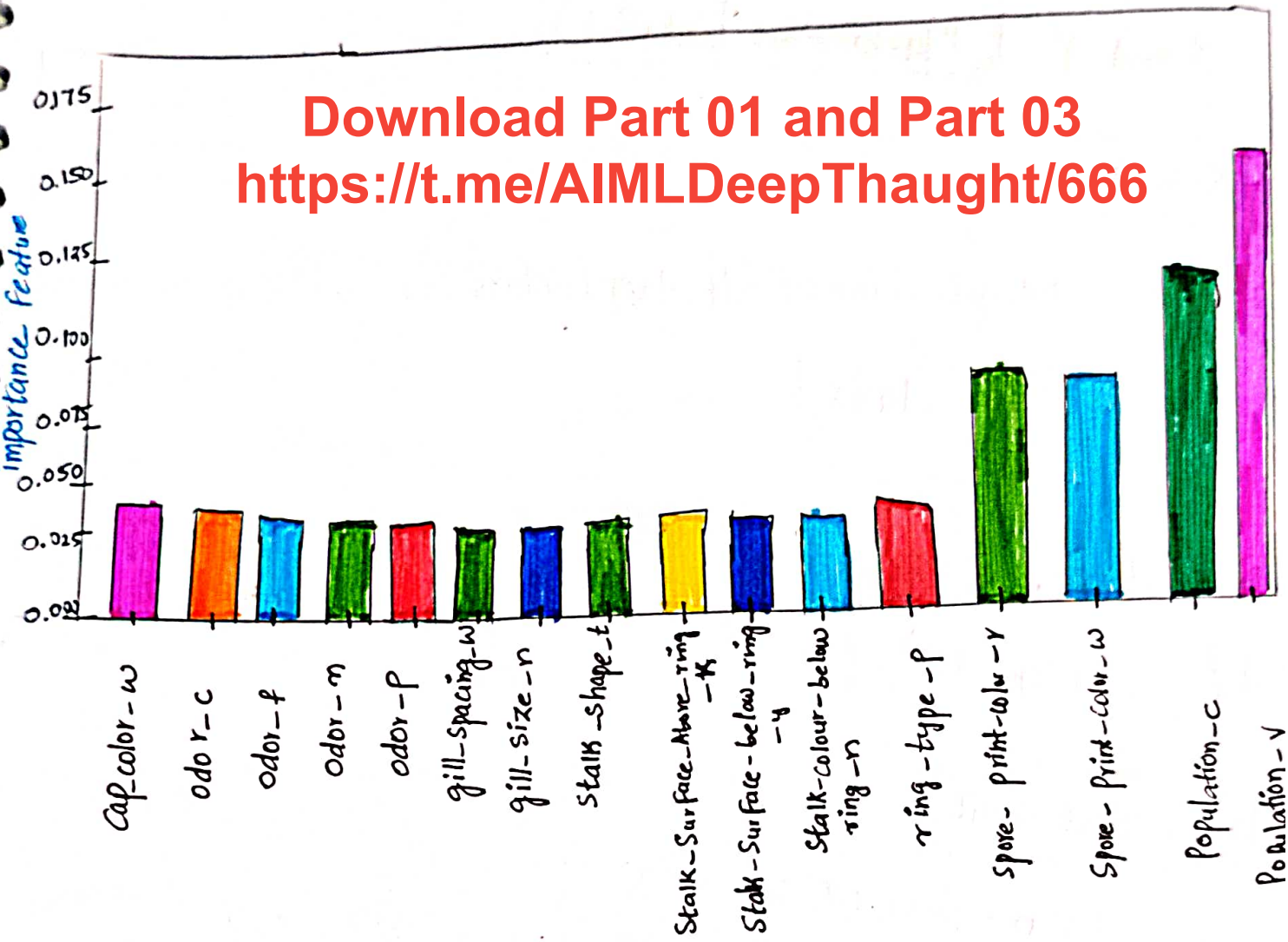
Out :

	importance_Feature
cap_color_w	0.043478
odor_c	0.043478
odor_f	0.043478
odor_p	0.043478
⋮	
order_n	0.130435
gill_size_n	0.173913

plt. figure (fig_size = (14,6), dpi = 200)

sns. barplot (data = important. Sort_values ("importance_feature"), x = important. index, y = "importance_Feature")

```
# plt.xticks (rotation = 90)
# plt. show ()
```



Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G

***

Gradient Boosting

* with in Gradient Boost, The Decision Trees are Pruned To Maximum Leaf nodes from 8 to 32.

Ex:-

X_1 Age	X_2 City	y Income		Prediction $\hat{y}$ (using D.T-1)	Residuals Error	Prediction (D.T-2)	Residuals error	y-Error
32	A	51000		53500	-2500	-5500	3000	48000
30	B	78000	model:	61000	17000	8000	9000	69000
21	A	20000	1 →	28500	-8500	-5500	-3000	model 2 → 23000
27	B	44000		61000	-17000	-4300	-12700	56700
36	B	89000		90500	-15000	8000	-9500	98500
25	A	37000		28500	8500	8000	5800	36500
							3000 − 51000 = 48000	

Target (error)min

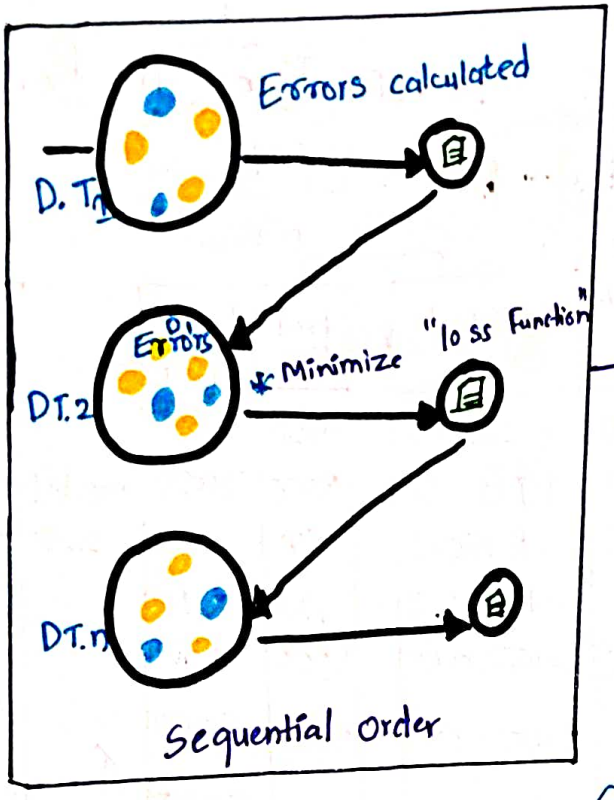
Decision Tree - 2

MODEL 2 Income = MODEL 1 income + Predicted Errors

	Model 0	M_1	M_2	.....	M_n
Feature	X	X	X		X
Target	Y	e_0	e_1	.....	e_{n-1}
	$H_0(x,y)$	$H_1(x,e_0)$	$H_2(x,e_1)$		$H_n(x,e_n)$

$y - y^1$ $y - y^0$

* it Continues, till it gets "Minimum Error".



→ GBM

"Gradient Boosting method".

How it works

1. A loss function to be optimized
2. A weak learner to make predictions
3. An additive mode to add weak learners to minimum (or) minimize the loss function.

* **Loss Function** :- $(\text{Sum of Error})_{min}$

(or)

(Overall Error should be Minimum)

Que :- What is difference between cost function & loss function ?

Ans :- Ex :-

Total = 100

	Exam	Error
Student 1	70	30
Student 2	85	15
Student 3	60	40

→ cost function. (Focusing on only one student)

→ loss function

: Cost function :- Error of individual record

Loss function :- Overall Error

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Que :- What is difference b/w "Adaboost" and "gradient boost"?

A:- * Adaboost :- We consider only stump, depth = 1

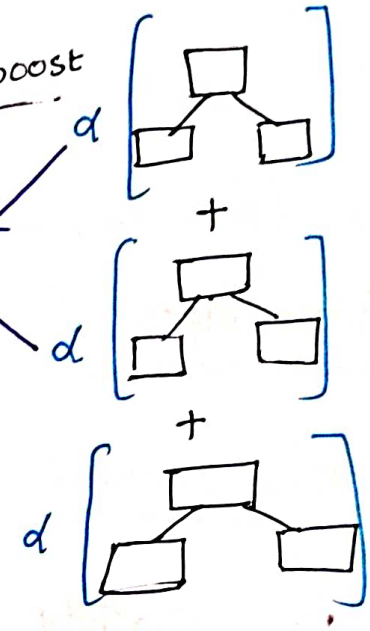
* Gradient boost :- We consider some what grown The Tree, Max. leaf nodes = 8 to 32

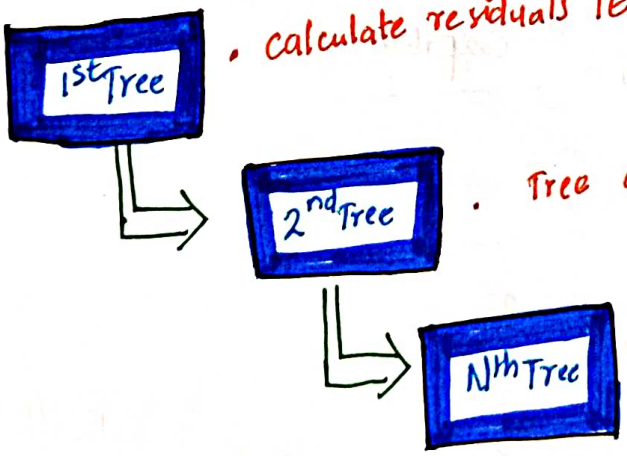
How it works

* $\hat{y}$ hat
* [y_actual - y_hat]^2
* y = m(x) + error 1
* error 1 = G[x] + error 2
* error 2 = H[x] + error 3
* y = m(x) + G[x] + H[x] + error 3
* y = alpha * M(x) + beta * G(x) + gamma * H(x) + error 4

Gradient Boost

* Learning rate





- calculate residuals /Errors
- Tree after adjusting for Residuals /Errors
- Final Tree after adjusting Errors multiple times.

Data :-

This Data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in agaricus and Lepiota Family (pp 500-525). Each species in indentified as definetly edible, definitely poisonous, or of unknown edibility and not recommended. this latter class was combined with the poisinous one. the Guide clearly states that there is no simple rule for determing the edibility of a mushroom; no rule like "leaflets three, Let it be" for poisinous Oak and Ivy.

→ Mushroom

Attribute → Information

1. cap_shape : bell = b, Conical = c, Convex = x, flat = f, Knobbed = k, sunken = s

. Cap_Surface : fibrous = f, grooves = g, Scaly = y, smooth = s

Cap_colour : brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, Yellow = y.

ruises ? : bruises = t, no = F

dor : almond = a, anise = l, creoste = c, fishy = y, foul = f, musty = m, none = n, punget = p, spicy = s

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

6. gill-attachment : black = k, brown = n, buff = b, chocolate = h, gray = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y

7. gill-spacing : close = c, crowded = w, distant = d

8. gill-size : broad = b, narrow = n

1. gill-color : black = k, brown = n, buff = b, chocolate = h, grave = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y.

. stalk-shape : enlarging = e, tapering = t

stalk-root : bulbous = b, club = c, cup = u, equal = e, rhizomorphs = z, rooted = r, missing = ?

Stalk Surface-above-ring : fibrous = f, scaly = y, silky = k, smooth = s

Stalk Surface-below-ring : fibrous = f, scaly = y, silky = k, Smooth = s

Stalk-colour-above-ring : brown = n, buff = b, Cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y.

stalk-colour-below-ring : "

Veil-type : Partial = p, universal = u

Veil-color : brown = n, Orange = o, white = w, Yellow = y

18. ring-number :- none = n, One = o, two = t

19. ring-type :- cobwebby = c, evanescent = e, flaring = f, large = l, none = n, Pendent = p, sheathing = s, zone = z,

20. Spore-print colour :- black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, Purple = u, white = w, Yellowy

21. Population :- abundant = a, clustered = c, numerous = n, Scattered = s, Several = v, solitary = y

22. habitat :- grasses = g, Leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d.

23. class (Output)

Bussiness Problem

Goal here is to see if we can harness the power of machine Learning and boosting to help create not just a predective model, but general Guideline for features People should look out for when picking mushrooms.

```
# df = Pd.redd_csv("mushrooms.csv")
# df.head()
```

class	cap shape	cap surface	cap color	bruises	odor	gill attachment	gill spacing	gill size	gill color ...	stalk surface above Ring	stalk color above ring	stalk color below ring	veil type	veil color	ring number	ring type	Spore print color	Population	habitat
p	x	s	n	t	p	f	c	n	k	s	w	w	p	w	o	p	k	s	u
e	x	s	y	t	a	f	c	b	k	s	w	w	p	w	o	p	n	n	g
e	b	s	w	t	l	f	c	b	n	s	w	w	p	w	o	p	n	n	m
p	x	y	w	t	p	f	c	n	n	s	w	w	p	w	o	p	k	s	u
e	x	s	g	f	n	f	w	b	k	s	w	w	p	w	o	e	n	a	g

df.shape

Out : [8124, 23]

df.info()

Out : Range Index : 8124 Entries, 0 to 8123

column	Nonnull Count	Dtype.
* class	8124 Nonnull	Object
* cap-shape	"	"
* cap Surface	"	"
⋮		"
* habitat	"	

df.isnull().sum()

Out : 0.

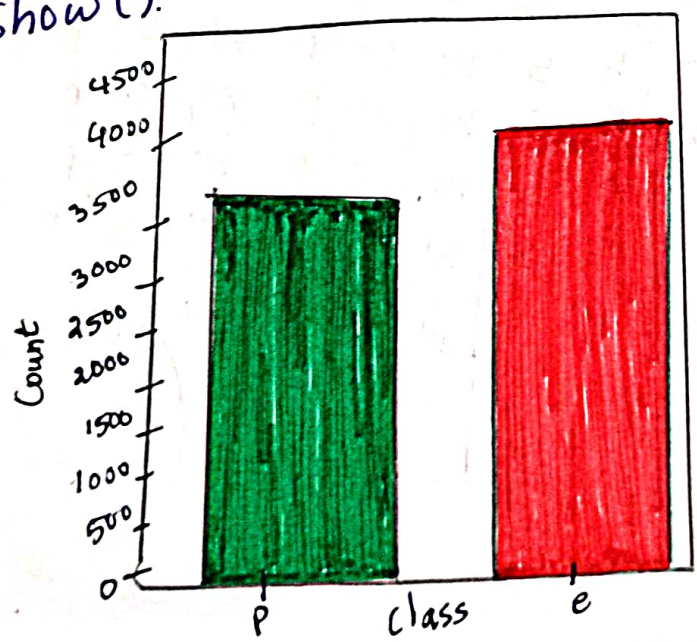
Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

EDA

: sns.countplot(data=df, X="class", palette="Dark2")
output

plt.show().

Out



Transpose. gives total no. columns. without....

```
# df.describe().transpose()
```

Out

	Count	Unique	top	Freq
(o/p) Class	8124	2	e	4208
cap Shape	8124	6	x	3656
cap- Surface	8124	4	y	3244
cap- color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill attachment	8124	2	f	7914
gill spacing	8124	2	c	6812
gill size	8124	2	b	5612
gill - color	8124	12	b	1728
Stalk- Shape	8124	2	t	4608
Stalk - root	8124	5	b	3776
Stalk - Surface - above ring	8124	4	s	5176
Stalk - Surface - below ring	8124	4	s	4936
Stalk - color above ring	8124	9	w	4464
Stalk - color below ring	8124	9	w	4384
Veil - type	8124	1	p	8124
Veil - color	8124	4	w	7924
ring - number	8124	3	o	7488
ring - type	8124	5	p	3968
Spore - Print color	8124	9	w	2388
Population	8124	6	v	4040
habitant	8124	7	d	3148

* posinous
* non posinous

CODE :

Gradient Boost :-

Same Data. and Same Procedure from import to X and Y (Mushroom DataSet)

X & Y

```
# x = pd.get_dummies (df.drop("class",axis=1 drop_first=True)
# y = df ["class"]
```

```
# X.shape, y.shape
```

Out : (8124, 95), (8124,))

train Test split

```
from sklearn.model_selection import train_test_split
# x_train, x_test, y_train, y_test = train_test_split (x, y, test_size=0.
                                     random_state = 29)
```

modelling

```
from sklearn.ensemble import Gradient Boosting Classifier
# gradmodel = Gradient Boosting classifier ()
# gradmodel . fit (x_train , y_train)
```

Out : Gradient Boosting classifier ()

Prediction

```
# ypred_train = gradmodel.predict (x_train)
# ypred_test = gradmodel.predict (x_test)
```

Evaluation

* **Accuracy**

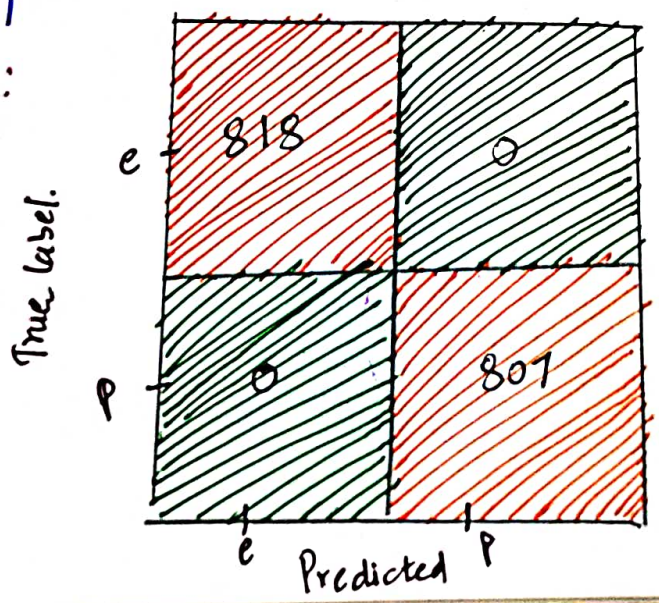
```
from sklearn.metrics import accuracy_score
# print ("train accuracy" :, accuracy_score (y_train, ypred_train))
# print ("test accuracy:" ; accuracy_score (y_test, ypred_test)
```

Out : train accuracy : 1.0
Test accuracy : 1.0

* **Confusion Matrix**

```
from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix (gradmodel , X_test , y_test)
# plt.show()
```

Out :



* Classification report

```
from sklearn.metrics import Classification_report
# Print (Classification_report (y_test, ypred_test))
```

Out :

	Precision	Recall	f1-score	Support
e	1.00	1.00	1.00	818
p	1.00	1.00	1.00	807
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
Weighted avg	1.00	1.00	1.00	1625

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

* Cross-validation Score

```
from sklearn.model_selection import Cross_Val_Score
# scores = cross_val_score (gradmodel, x, y, cv=5)
# Print ("cross_val_score :", scores.mean())
```

Out : cross_val_score : 0.9192

* feature importance

```
# gradmodel.feature_importances_
```

Out : array([0.00e+00, 1.0465, 1.93018, 0.000e+0, 6.19492, 1.36263, 4.49731 -3.5778, ...])

Hyper Parameter Tunning

Important Parameter for Gradient Boost

```
from sklearn.model_selection import GridSearchCV
# estimator = GradientBoostingClassifier()
# param_grid = {"n_estimators": [1,5,10,20,40,100], "learning rate":
                                [0.1, 0.2, 0.5, 0.9]}
# grid = GridSearchCV(estimator, param_grid, Scoring="accuracy")
                                , cv=5
# grid.fit (x_train, y_train)
# grid.best_params_
```

Out: { "learing_rate": 0.2, "n_estimators": 100)

Final Model

```
# final model = GradientBoostingClassifier (n_estimators =100,
                                            Learning_rate =0.2)
# final model. fit (x_train, y_train)
# ypred_train = final model. predict (x_train)
# ypred_test = final model. predict (x_test)
# print ("train_accuracy", accuracy_score (y_train, ypred_train)
# print ("test accuracy", accuracy_score (y_test, ypred_test)
```

Out: train accuracy: 1.0
test accuracy : 1.0

```
# final model.feature_importances_
```

Out: array([0.000e+00, 2.457745e-16, 1.27008, 4.735
1.137228 1.485440 1.923168, 3.209765,
0.000bev])

```
# important = pd.DataFrame(index = X.columns, data = final model.
feature_importances, columns = ["importance_feature
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

```
# important
```

Out:

	importance_feature
cap_shape_c	0.0000e+00
cap_shape_f	2.457745e-16
cap_shape_k	1.270085e-23
cap_shape_s	4.735679e-08
⋮	⋮
habitat_u	9.502012e-05
habitat_w	0.00000e+00

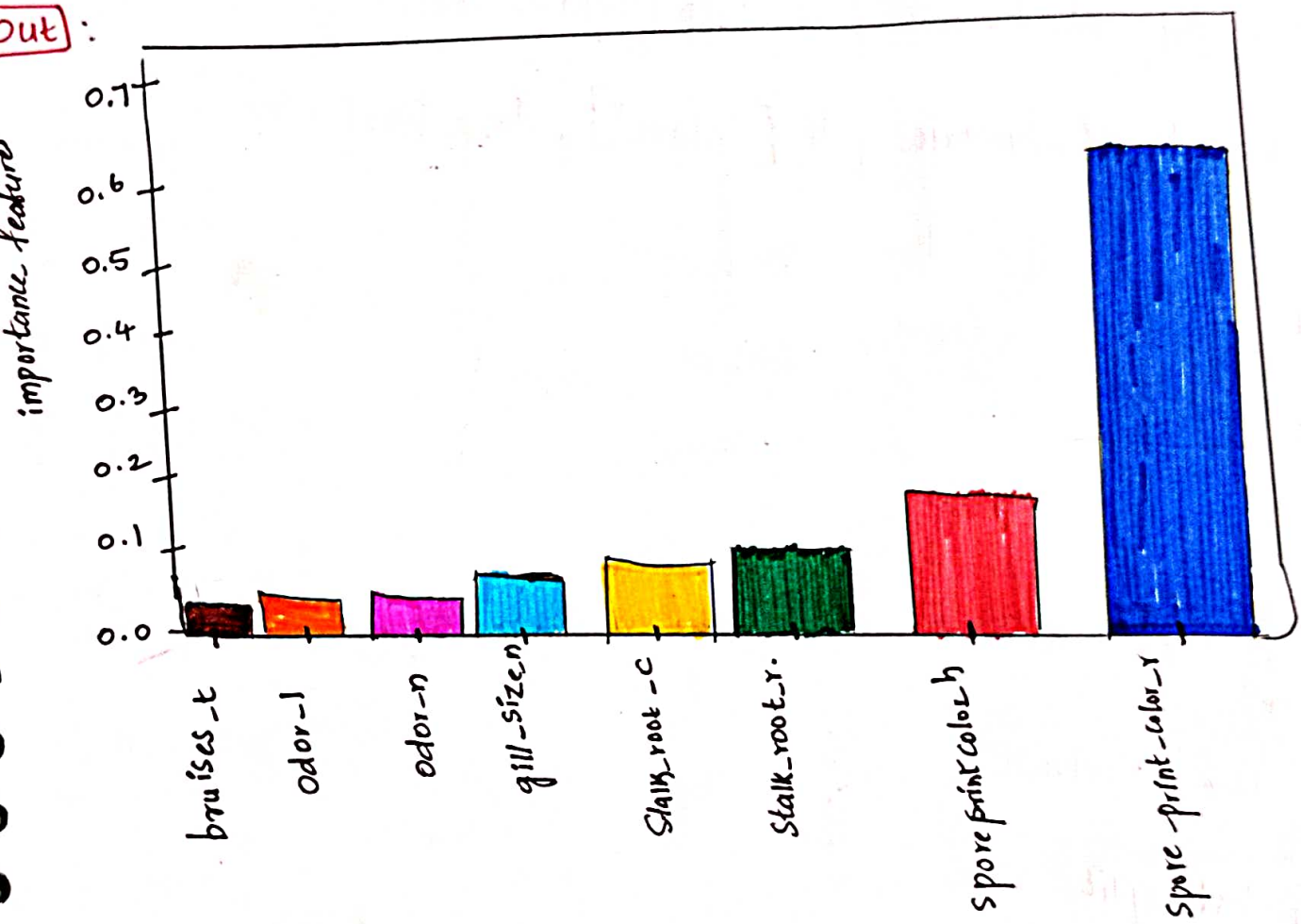
```
# Jack = important[important["importance_feature"] > 0.01]
# Jack.sort_values("importance_feature")
```

Out:

	importance_feature
gill_size_n	0.010084
Spore print color_h	0.014901
odor_l	0.019874
Spore print color_r	0.032914
Stalk root_r	0.052216
bruises_t	0.060678 . . . .

```
# plt.figure (figsize=(14,6), dpi=200)
# sns.barplot (data = Jack.sort_values ("importance_feature"),
                x = Jack.index , y = "importance_feature")

# plt.xticks( rotation = 90)
# plt.show()
```

Out:



Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

**** XG BOOST

X → Gradient

Xtreme

* Adding "Penality" term Reduce over all Error

* "10" times Faster than "Gradient Boosting".

* "XG" Boost also called as "Regularized Boosting".

* Like L_1, L_2 Regularization

* Advantages

* Tree Pruning Using depth first approach

* High-flexibity :- Catche awarness and out of core computing. (-1) it uses all cores in system

* Reduce Overfitting :- "Regularization" for avoiding overfitting

Efficient Handling missing data

Inbuilt "cross-validation" Capability

XG-boost

error + Penality Term

L_2 : α (slope)^2

L_1 : α |slope|

with in "X"G Boost :-
α = gamma

* **XG boost classifier** Black box Model.

Base model = Probability = 0.5

Ex:- DataSet (Pr - Approval)

Salary	Credit	Approval	Residuals
<= 50k	Bad	0	0 - 0.5 = -0.5
<= 50k	Good	1	0.5
<= 50k	Good	1	0.5
> 50k	Bad	0	-0.5
> 50k	Good	1	0.5
> 50k	Normal	1	0.5
<= 50k	Normal	0	-0.5

* **Steps** :-

1. Create Binary Decision Tree using Features

2. Calculate Similarity weight

$$= \frac{\Sigma[\text{Residual}]^2}{\Sigma[\text{Probability}[1-\text{Probability}]+\lambda]}$$

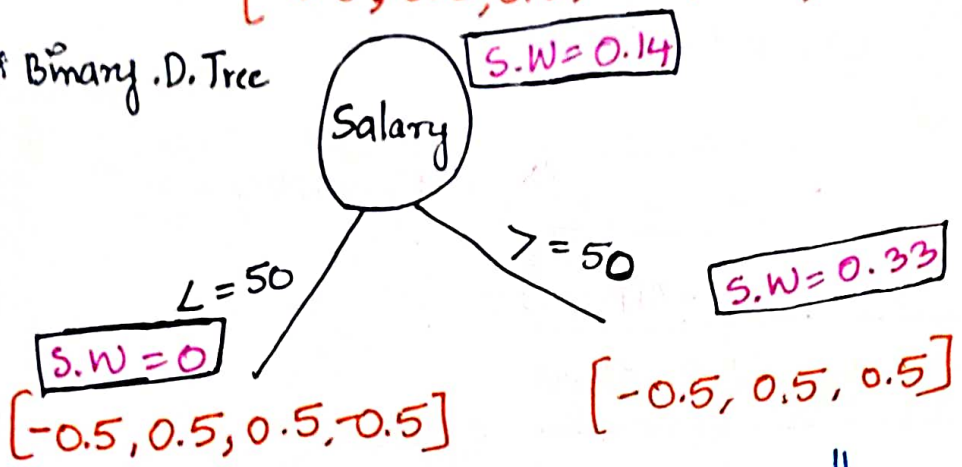
$\rightarrow +\lambda$ (hyper parameter)

3. Information Gain.

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Errors :- $[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]$

* Binary D. Tree



$\Downarrow$

Similarity weight $\Rightarrow \boxed{\dfrac{\Sigma(res)^2}{\Sigma p(1-p)\lambda}}$

$$= \frac{[-0.5+0.5+0.5-0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0} \quad \because \lambda = 0$$

$$= 0$$

$\Downarrow$

Similarity weight $= \boxed{\dfrac{\Sigma(res)^2}{\Sigma p(1-p)\lambda}}$

$$= \frac{[-0.5+0.5+0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+\frac{0}{\lambda}}$$

$$= \frac{0.25}{0.75} = \frac{1}{3} \Rightarrow 0.33$$

For Root node Similarity weight $= \boxed{\dfrac{\Sigma(res)^2}{\Sigma p(1-p)\lambda}}$

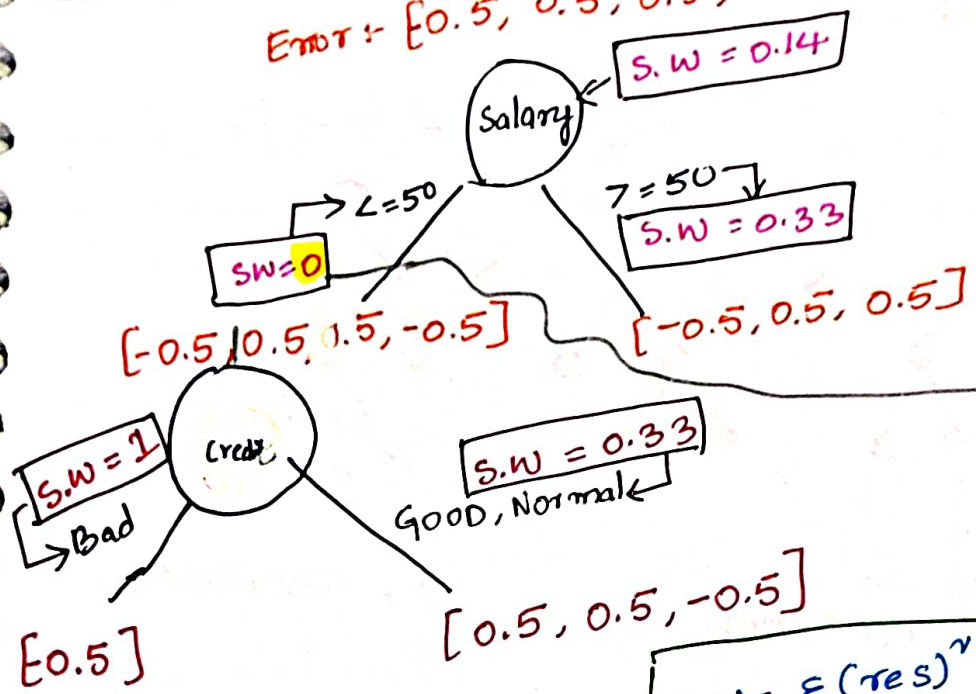
$$= \frac{[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]}$$

$$= \frac{0.25}{1.75} = \frac{1}{7} = 0.14$$

* Information Gain = 0 + 0.33 − 0.14

↳ = $\boxed{0.19}$

Error :- [0.5, 0.5, 0.5, −0.5, 0.5, 0.5, −0.5]



* $S.W = \frac{\varepsilon(res)^2}{\varepsilon p(1-p)\lambda}$

$= \frac{[-0.5]^2}{0.5[1-0.5]} = \frac{0.25}{0.25}$

$= 1$

* $S.W = \frac{\varepsilon(res)^2}{\varepsilon p(1-p)+\lambda}$

$= \frac{[0.5, \cancel{0.5}, \cancel{-0.5}]^2}{0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5]}$

$= \frac{0.25}{0.75} = \frac{1}{3} = 0.33$

* Information Gain :- 1 + 0.33 − 0 → ?

= 1.33

For classification

Salary	Credit	Approval
<=50	→ B	0

Base model = 0.5

∴ Real probability = $\log\left(\frac{p}{1-p}\right) = \log\left(\frac{0.5}{1-0.5}\right) = 0$

Goes to binary

activation function → sigmoid

$\sigma[0 + \alpha(1)]^{\alpha=0.01}$ ⇒ Based on "α" value → o/p $\boxed{0-1}$

Learning rate

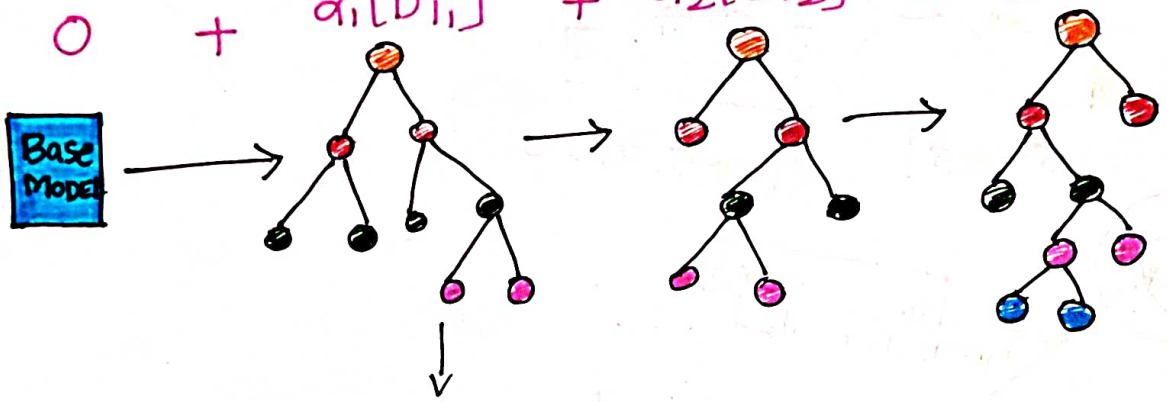
S.W

Output :-

$$\left[\sigma\left[0+\alpha_1[DT_1]+\alpha_2[DT_2]+\alpha_3[DT_3]+\cdots\alpha_n[DT_n]\right]\right]$$

For any new record

$$0 + \alpha_1[DT_1] + \alpha_2[DT_2] + \alpha_3[DT_3] + \cdots$$



Based on independent Features.

$\therefore \lambda$ = Cross validation.

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G

* **XG boost regressor**

* Base model = 51K

Ex:-

x_1	x_2	y	
Exp	Gap	Salary	Residual
2	Yes	40 K	-11K
2.5	Yes	42 K	9
3	No	52 K	1
4	No	60 K	9
4.5	Yes	62 K	11

Avg = 51.

S.W = $\frac{1}{6}$

[-11, -9, 1, 9, 11]

$$\frac{[-\cancel{11}\ \cancel{9}+1+\cancel{9}+\cancel{11}]^2}{5+1} = \frac{1}{6}$$

Experience

S.W = 65.5

≤ 2

[-11]

SW = 28.8

> 2

[-9, 1, 9, 11]

$$\frac{[-\cancel{9}+1+\cancel{9}+11]^2}{4+1} \qquad \boxed{\lambda = 1}$$

$$= \frac{[12]^2}{5} = \frac{144}{5} = 28.8$$

* Similarity weight = $\dfrac{\Sigma[\text{Residual}]^2}{\text{no. of residuals} + \lambda}$

$$= \frac{[-11]^2}{1+1} = \frac{121}{2} = 60.5$$

$\boxed{\therefore \lambda = 1}$

* Information Gain = $60.5 + 28.8 - \frac{1}{6} = 89.13$

S.W 1/6

Experience

S.W = 133.33 ≤ 2.5 $(-11, -9)$

S.W = 110.25 > 2.5 $(1, 9, 11)$

$$S.W = \frac{[-11-9]^2}{2+1} = \frac{[-20]^2}{3} = \frac{400}{3} \Rightarrow 133.33$$

$$= \frac{[1+9+11]^2}{3+1} = \frac{[21]^2}{4} = 110.25$$

any records goes from base model = 51

≤ 2.5

Avg. of S.W

$$51 + \alpha_1 \left[\frac{[-11-9]}{2}\right] = (-10)$$

$$* \; 51 + \alpha_1 [-10]$$

Avg. of S.W

$$51 + \alpha_1 \left[\frac{1+11+9}{3}\right] \Rightarrow \frac{21}{3} = 7$$

$$* \; 51 + \alpha_1 [7]$$

$$* \; 51 + \alpha_1 (-10) + \alpha_2 [DT_2] + \alpha_3 [DT_3] \ldots \alpha_n [DT_n]$$

↳ Output.

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

27/04/22
6:00 Am.

Data :-

This Data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in agaricus and Lepiota Family (pp 500-525). Each species in indentified as definetly edible, definitely poisonous, or of unknown edibility and not recommended. this latter class was combined with the poisinous one. the Guide clearly states that there is no simple rule for determing the edibility of a mushroom; no rule like "leaflets three, Let it be" for poisinous Oak and Ivy.

→ Mushroom

Attribute	Information
1. cap_shape :	bell = b, Conical = c, Convex = x, flat = f, Knobbed = k, sunken = s
.. Cap_Surface :	fibrous = f, grooves = g, Scaly = y, smooth = s
Cap_colour :	brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, Yellow = y.
ruises ? :	bruises = t, no = F
dor :	almond = a, anise = l, creoste = c, fishy = y, foul = f, musty = m, none = n, punget = p, spicy = s

6. gill-attachment : black = k, brown = n, buff = b, chocolate = h, gray = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y

7. gill-spacing : close = c, crowded = w, distant = d

8. gill-size : broad = b, narrow = n

1. gill-color : black = k, brown = n, buff = b, chocolate = h, grave = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y.

. stalk-shape : enlarging = e, tapering = t

stalk-root : bulbous = b, club = c, cup = u, equal = e, rhizomorphs = z, rooted = r, missing = ?

Stalk Surface-above-ring : fibrous = f, scaly = y, silky = k, smooth = s

Stalk Surface-below-ring : fibrous = f, scaly = y, silky = k, Smooth = s

Stalk-colour-above-ring : brown = n, buff = b, Cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y.

stalk-colour-below-ring : "

Veil-type : Partial = p, universal = u

Veil-color : brown = n, Orange = o, white = w, Yellow = y

18. ring-number :- none = n, One = o, two = t

19. ring-type :- cobwebby = c, evanescent = e, flaring = f, large = l, none = n, Pendent = p, sheathing = s, zone = z,

20. Spore-print colour :- black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, Purple = u, white = w, Yellowy

21. Population :- abundant = a, clustered = c, numerous = n, Scattered = s, Several = v, solitary = y

22. habitat :- grasses = g, Leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d.

23. class (Output)

Bussiness Problem

Goal here is to see if we can harness the power of machine Learning and boosting to help create not just a predective model, but general Guideline for features People should look out for when picking mushrooms.

```
# df = Pd.redd_csv("mushrooms.csv")
# df.head()
```

class	cap shape	cap surface	cap color	bruises	odor	gill attachment	gill spacing	gill size	gill color ...	stalk surface above Ring	stalk color above ring	stalk color below ring	veil type	veil color	ring number	ring type	Spore print color	Population	habitat
p	x	s	n	t	p	f	c	n	k	s	w	w	p	w	o	p	k	s	u
e	x	s	y	t	a	f	c	b	k	s	w	w	p	w	o	p	n	n	g
e	b	s	w	t	l	f	c	b	n	s	w	w	p	w	o	p	n	n	m
p	x	y	w	t	p	f	c	n	n	s	w	w	p	w	o	p	k	s	u
e	x	s	g	f	n	f	w	b	k	s	w	w	p	w	o	e	n	a	g

df.shape

Out : [8124, 23]

df.info()

Out : Range Index : 8124 Entries, 0 to 8123

column	Nonnull Count	Dtype.
* class	8124 Nonnull	Object
* cap-shape	"	"
* cap Surface	"	"
⋮		
* habitat	"	"

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

df.isnull().sum()

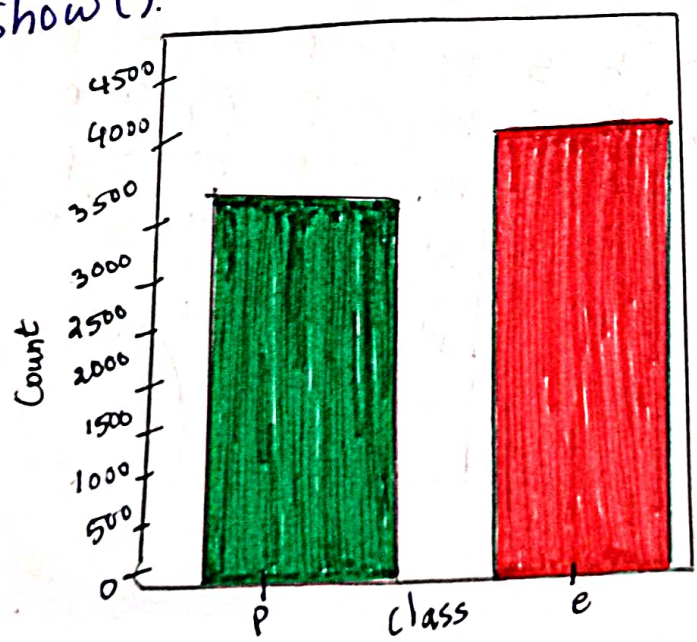
Out : 0.

EDA

: sns.countplot(data=df, X="class", palette="Dark2")
output

plt.show().

Out



Transpose. gives total no. columns. without....

```
# df.describe().transpose()
```

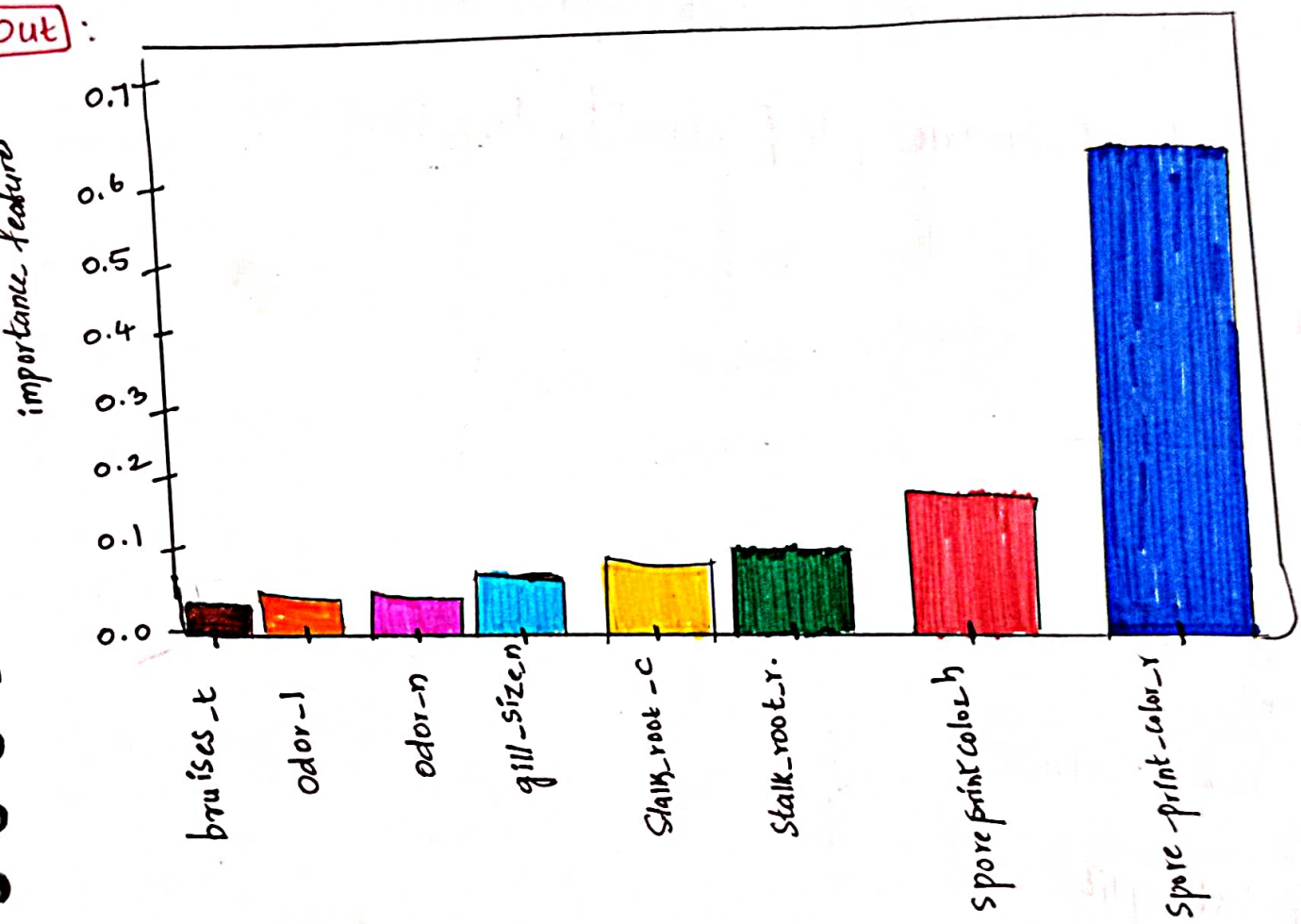
Out

	Count	Unique	top	Freq
(o/p) Class	8124	2	e	4208
cap Shape	8124	6	x	3656
cap- Surface	8124	4	y	3244
cap- color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill attachment	8124	2	f	7914
gill spacing	8124	2	c	6812
gill size	8124	2	b	5612
gill - color	8124	12	b	1728
Stalk- Shape	8124	2	t	4608
Stalk - root	8124	5	b	3776
Stalk - Surface - above ring	8124	4	s	5176
Stalk - Surface - below ring	8124	4	s	4936
Stalk - color above ring	8124	9	w	4464
Stalk - color below ring	8124	9	w	4384
Veil - type	8124	1	p	8124
Veil - color	8124	4	w	7924
ring - number	8124	3	o	7488
ring - type	8124	5	p	3968
Spore - Print color	8124	9	w	2388
Population	8124	6	v	4040
habitant	8124	7	d	3148

* posinous
* non posinous

```
# plt.figure (figsize=(14,6), dpi=200)
# sns.barplot (data = Jack.sort_values ("importance_feature"),
                x = Jack.index , y = "importance_feature")

# plt.xticks( rotation = 90)
# plt.show()
```

Out:



Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Code :

XG boost Classifier

Same Example, Same Question of Mushroom dataset.

import to X & Y.

X & Y

```
# X = pd.get_dummies(df.drop("class", axis=1), drop_first=True)
# y = pd.get_dummies(df["class"], drop_first=True)

# y
```

out :

	p
0	1
1	0
2	0
3	1
4	0
⋮	⋮

8124 x 1 column

train test split

```
from sklearn.model_selection import train_test_split

# x_train, x_test, y_train, y_test = train_test_split(x, y,
                                     test_size = 0.2, random_state=29
```

modelling

```
! Pip install xgboost
from xgboost import XGBClassifier
# xgb_model = XGBClassifier ()
# xgb_model. fit (x_train, y_train)
```

Out : XGBClassifier (base core = 0.5 , booster = "gbtree". . . .
- - - - - - - - - - -)

Prediction

```
# ypred_train = xgb_model. Predict (x_train)
# ypred_test = xgb_model. Predict (x_test)
```

Evaluation

* Accuracy

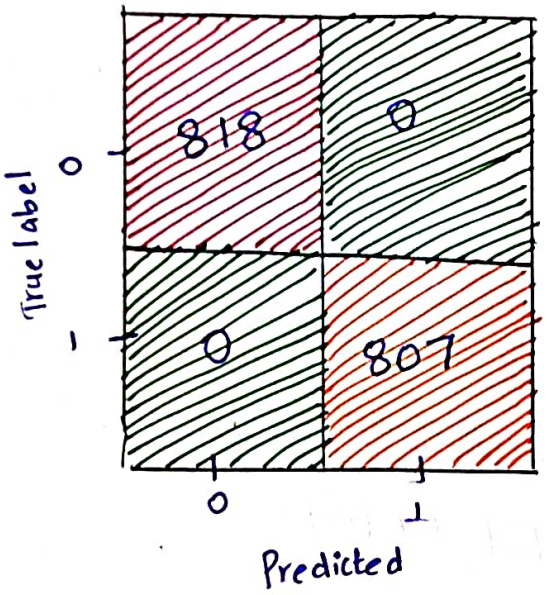
```
from Sklearn . metrics import accuracy_score
# print ("train accuracy:", accuracy_score (y_train, ypred_train.
# print ("Test accuracy:", accuracy_score (y_test , ypred_tes
```

Out : Train accuracy : 1.0
Test accuracy : 1.0

* Confusion Matrix

```
from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix (xgb_model , x_test , y_test)
# plt.show ()
```

Out :



* classification report

```
from sklearn.metrics import Classification_report
# Print (classification_report (y_test , ypred_test)
```

Out :

	Precision	recall	f1 score	Support
0	1.00	1.00	1.00	818
1	1.00	1.00	1.00	807
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
Weighted Avg	1.00	1.00	1.00	1625

* Cross Validation Score

```
from Sklearn . model_selection import Cross_val_score

# scores = Cross_val_score (xgb_model , x, y, cv=5)

# print (" cross_val_score :", scores. mean())
```

Out: cross-val-score : 0.935

* Feature importance

```
# xgb_model . feature_ importances_
```

Out: array ([0.0000e+0 , 0.0000e+00 , 0.0000e+0 , 0.0000e+0
3.961512e5 0.00000e+0 , 9.23.225 ---])

Hyper Parameter tuning

Important Parameters in XGboost

```
from sklearn.model_selection import GridSearchCV

# estimator = XGBClassifier ()

# Param_grid = { "n_estimators": [1,5,10,20,40,100], "gamma" : [0,1,
                , 0.5 , 0.9] , "max_depth": [2,6

# grid = GridSearchCV (estimator, param_grid, Scoring = "accuracy
                CV = 5)

# grid .fit (x_train, y_train)

# grid. best_params
```

Out: { "gamma": 0.1 , "max_depth": 2, "n_estimators": 20}

final Model

```
# final_model = XGBClassifier (n_estimators = 20, gamma = 0.1, max_
                                depth = 2)
# final_model . fit (x_train, y_train)

# ypredictions_test = final_model. Predict (x_test)
# ypreditions_train = final_model. Predict (x_train)

# Print ("train accuracy:" ; accuracy_score (yprediction_train , y_train))
# print ("test accuracy:", accuracy_score (yprediction_test , y_test))
```

Out: Test accuracy : 0.99815
Train accuracy : 0.9979

```
# final_model. feature_importances_
```

Out: array ([0. , 0. , 0., 0.,
0., 0., 0. ..])

```
# important = pd. DataFrame ( index = X. columns , data = final_
model . feature_importances_ , columns = ["importance_
Features"])
```

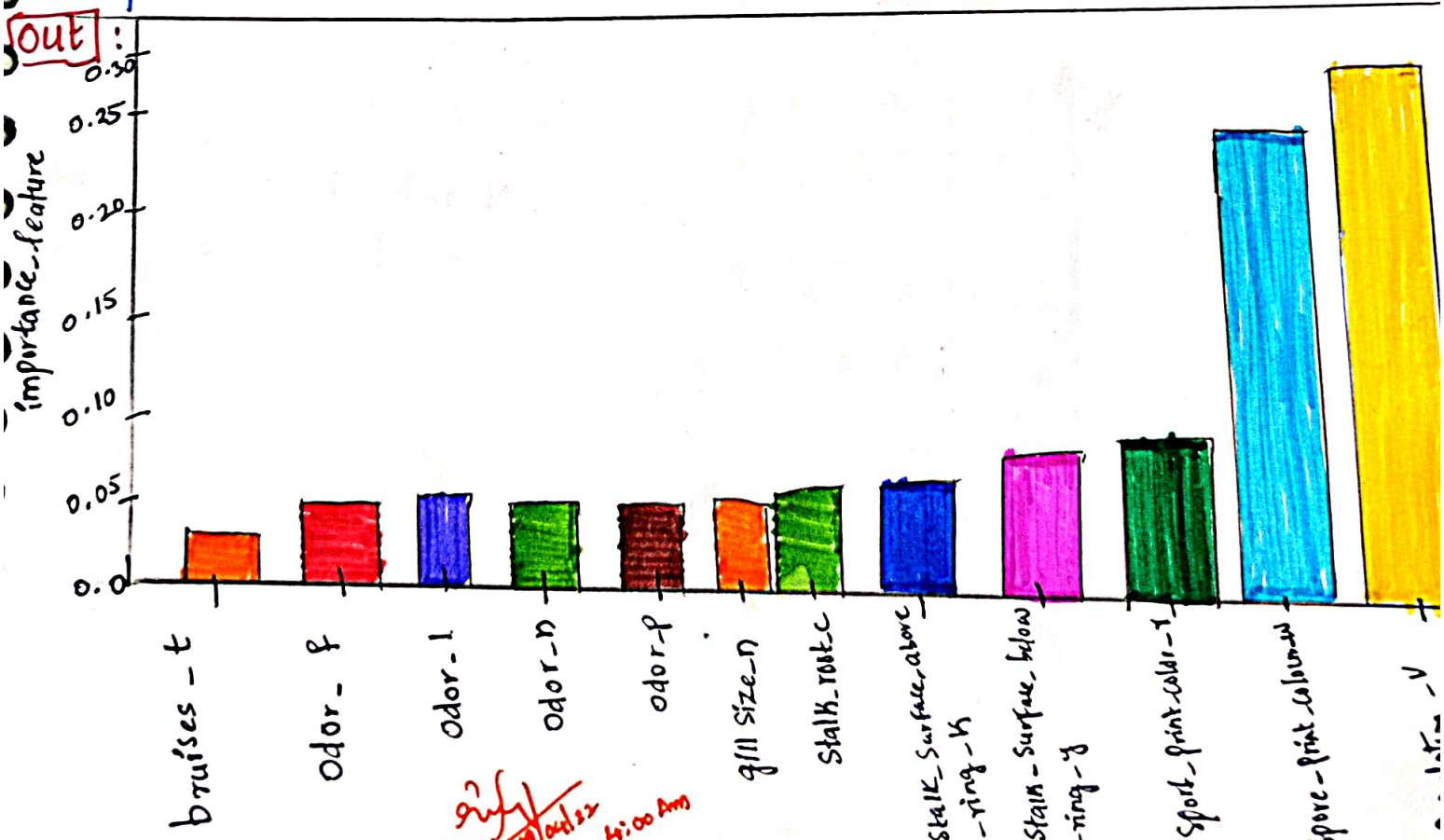
```
# Jack = important[important["importance_feature"] > 0.01]
# Jack.sort_values("importance_feature")
```

Out:

	important_features
odor-p	0.015831
spore_print_color_w	0.020152
⋮	⋮
odor_n	0.290767

```
# plt.figure(fig_size=(14,6), dpi=200)
# sns.barplot(data = Jack.sort_values("importance_feature"),
                x = Jack.index, y= "importance_feature")
# plt.xticks(rotation =90)
# plt.show()
```

Out:



Dt:- 18/4/22
2:30 pm

* Supervised Algorithm :-
  it has Label Data | class (Yes/NO)

* Predicting :- Discrete (categories)

  Ex :-
  * Binary classification [0, 1]
  * Multi classe classification [drug x, drug, drug

Classification [Algorithms]

→ Logistic Regression
→ KNN [K-Nearest Neighbours]
→ Decision Tree
→ Random Forest
→ Ada Boost
→ Gradient Boost
→ XG Boost
→ SVM [Support vector Machine]
→ Navies Bayes

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

⇒ Classification Use Cases :

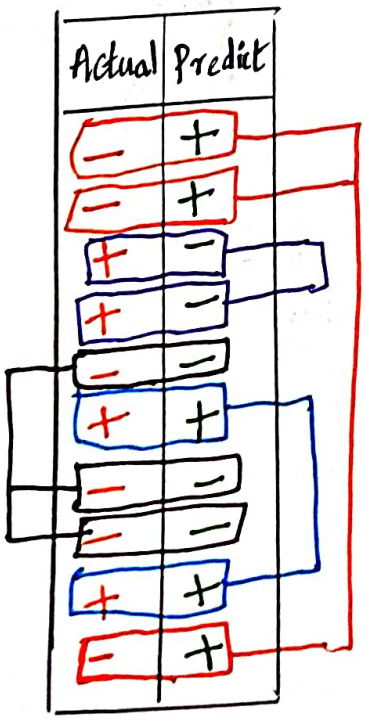
1. Which Category a customer belongs to ?
2. Whether customer Switches to another provider / brand ?
3. Whether customer responds to a particular advertising Campaign ?

⇒ Evaluation Metrics :

⇒ Confusion Matrix

Ex:- Whether The person is covid: (+) positive
(−) negative.

covid
(+) positive
(−) negative



Actual / True

Prediction	+ve	−ve
+ve	T 2 P True positive	F 3 P False positive
−ve	F 2 N False Negative	T 3 N True Negative

⇒ T/F − Correct Prediction | incorrect Prediction

⇒ P/N − Predicted Positive | Predicted Negative

* Another Type

Predict

		+ve	-ve
Actual	+ve	TP	FN
	-ve	FP	TN

Predict

		+ve	-ve
Actual	+ve	2 [TP]	2 [FN]
	-ve	3 [FP]	3 [TN]

* Classification [Accuracy] = $\frac{\text{Correct Predictions}}{\text{total predictions}} = \frac{5}{10}$

* Mis-classification [Error] = $\frac{\text{Incorrect prediction}}{\text{total predictions}} = \frac{5}{10}$

Que :- which is more important "False positive", "False Negati

Ans :- it is dependent on "Bussiness problem" → FN ↓, FP ↓, Error ↓, Accuracy ↑

⇒ Person is Having covid *

Actual +ve

Predicted -ve

Always Predicted Comes last (Second Letter) *

[FN] More important

Because. if person is Having covid, You predicted negative, Problem (i) die (ii) Spread

FN

⇒ Person is not having covid *

Actual -ve

Predicted +ve

[FP] More important

But, Person is Havin cancer, But predict positive.

(i) Kimotherapy

FP

[FP]

If person is not having covid, You predicted covid, (i) quarantine (ii) Antibotics (medicine)

Que :- In bussiness problem, They said to Focus of "FN" should be low ?

	Actual	
Pred	1 TP	1 FP
	8 FN	90 TN

Accuracy = 91 %.
FN = 8

	Actual	
Pred	4 TP	6 FP
	3 FN	87 TN

If client, wants Accuracy and FN
↓ Best

Accuracy = 91 %.
False Negative = 3

If client, wants Only FN should be low, Not Accuracy
↓ best

	Actual	
Pred	38 TP	24 FP
	0 FN	38 TN

Accuracy = 76 %
FN = 0

18/4/22
4:30pm.

Dt :- 18/4/22
10:00pm

	Actual	
	+ve	-ve
+ve	TP	FP
-ve	FN	TN

Pred

Recall

Precisely (or) Correctly.

$$\frac{TN}{FN+TN}$$

Precision (-ve)
⇓
$$\frac{\text{Correctly predicted (-ve)}}{\text{Total predicted (-ve)}}$$

Precision (+ve)
⇓
$$\frac{\text{Correctly predicted (+ve)}}{\text{Total predicted (+ve)}} = \frac{TP}{TP+FP}$$

2) Recall (+ve) = $\frac{\text{Correctly Predicted (+ve)}}{\text{total Actual positive (+ve)}}$

$$= \boxed{\frac{TP}{TP + FN}}$$

* Recall (-ve) = $\frac{\text{correctly predicted (-ve)}}{\text{total Actual negative (-ve)}}$

$$\boxed{\frac{TN}{FP + TN}}$$

** Que :- What is difference between Recall and Precision ?

Ans :- Precision :- Comparing with The Predicted Values

Preci	Recall
Predicted	Actual

comparing with

Recall :- Comparing with Actual values.

* denominator changes, where numerator is Same

* denominator instead of prediction, Take actual One, instead of predicted Positive, Take actual positive

True class

Predictions	+	-
+	True +	False +
-	False -	True -

Type - I (Error)

Type - II Error

* **True positive [TP]** :- when classifer predicted True and correct class was True

Ex: They have disease, predicted correctly.

* **True Negative [TN]** : when model predicted False and correct class was False

Ex:- No disease, But predictive not have disease.

* **False positive [FP] [Type-I Error]** : classifier predicted True, but correct class was False.

Ex:- Patient did not have disease.

* **False Negatives [FN] [Type-II Error]** : classifier predicted False, but they actually do have disease.

Ex:- Patient do not have disease.

** **Key Performance Indicators [KPI]**

classification accuracy = $\dfrac{TP + TN}{(TP + TN + FP + FN)}$

Mis classification rate (error rate) = $\dfrac{FP + FN}{(TP + TN + FP + FN)}$

Example

		Actual +	Actual −
Pred	+	TP = 1	FP = 1
	−	FN = 8	TN = 90

classification accuracy $= \dfrac{TP + TN}{TP + FP + FN + TN} = 91\%$

precision (+ve) $= \dfrac{TP}{TP + FP} = \dfrac{1}{2} = 50\%$

Recall (+ve) $= \dfrac{TP}{TP + FN} = \dfrac{1}{1+8} = \dfrac{1}{9} = 11\%$

precision (−ve) $= \dfrac{TN}{FN + TN} = \dfrac{90}{8+98} = \dfrac{90}{98} = 0.91\%$

Recall (−ve) $= \dfrac{TN}{FP + TN} = \dfrac{90}{1+90} = \dfrac{90}{91} = 0.98\%$

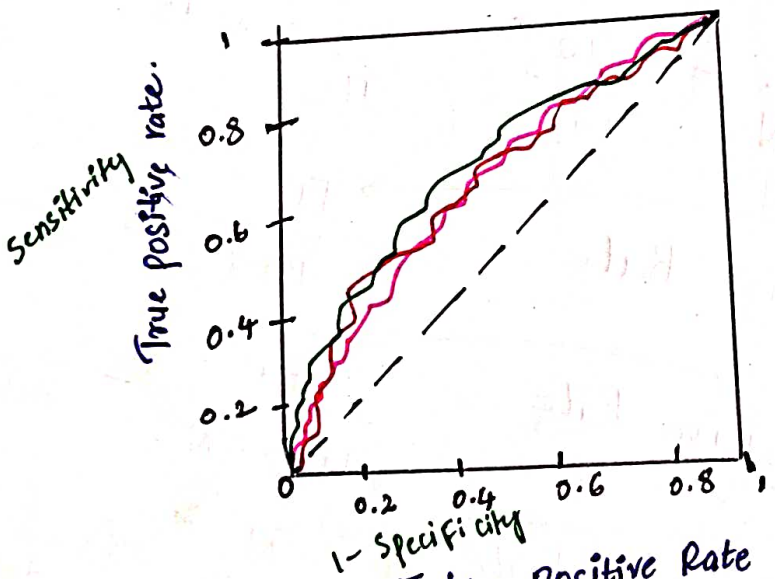
* TPR (recall) = True Positive Rate $= \dfrac{TP}{TP + FN}$ ⇒ Actual (+ve)
* FPR = False Positive Rate $= \dfrac{FP}{TP + FN}$ ⇒ Actual (+ve)
* TNR = True Negative Rate $= \dfrac{TN}{TN + FP}$ ⇒ Actual (−ve)
* FNR = False Negative Rate $= \dfrac{FN}{FP + TN}$ ⇒ Actual (−ve)

* Specificity = TNR ⇒ 1 − FPR

= True Negative Rate = $\frac{\text{True Negatives}}{\text{True negatives + False positives}}$

* Sensitivity ⇒ TPR ⇒ Recall ⇒ $\frac{TP}{TP+FN}$

ROC [Reciver operating characteristic Curve]



False Positive Rate = 1 − specifity = 1 − TNR

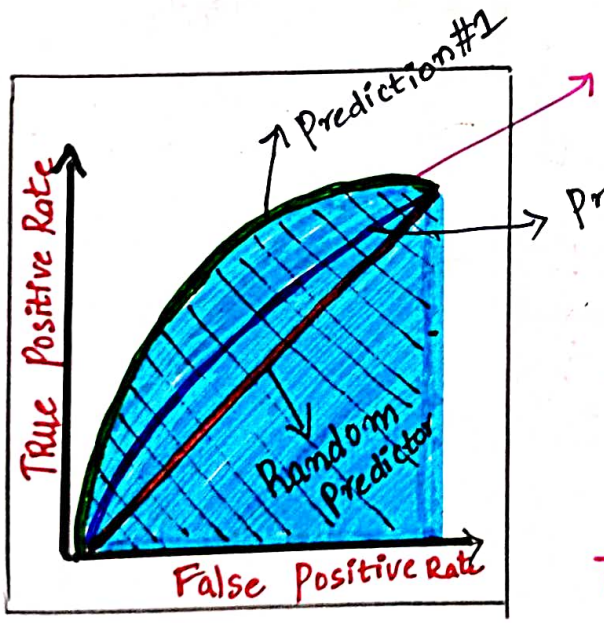
* ROC curve is a metrics that asses model ability to distinguish between binary (0 (or) 1) classes

* ROC curve is created by plotting the true positive rate (TPR) against the False positive Rate [FPR] at various threshold settings

* True positive rate is also known as Sensitivity, recall (or) probability of detection in Machine Learning.

* False positive rate is also known as the probability of False alarm and can be calculated as [1 - specificity]

AUC [Area Under Curve]



the light Blue area represents area under curve of receiver operating characteristic [AUROC]

Higher the AUC, the better the model is at predicting

18/04/22
11:30 pm

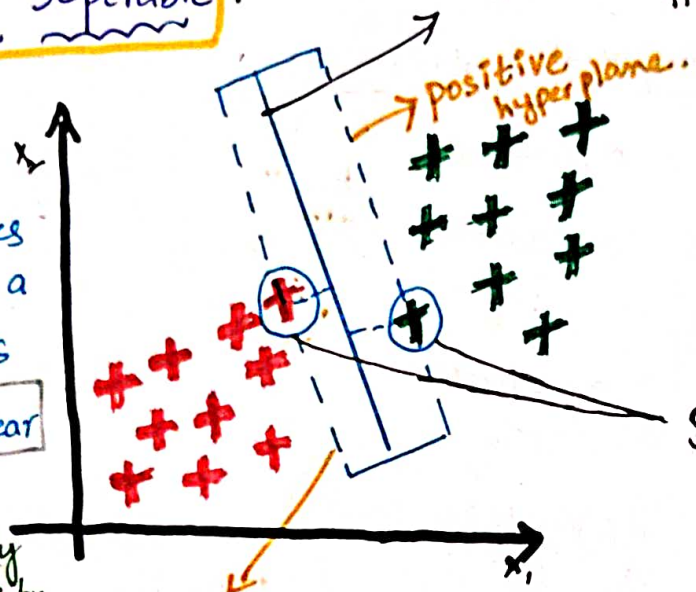
29/04/22
1:00pm

SVM (Support Vector Machines)

* **Linear Seperable** :-

Seperating The classes by using a Straight is called "Linear SVM"

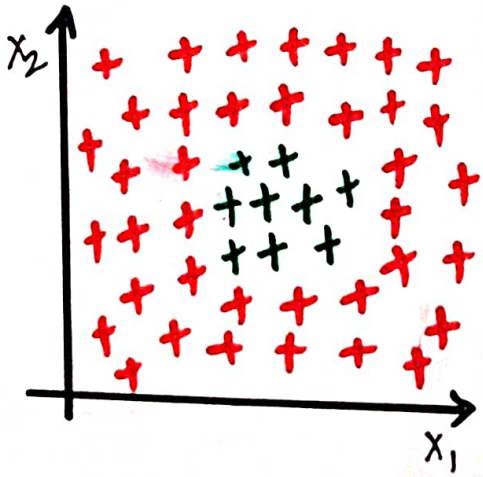
it indentify Best fit line by "Maximum margin classifier".



it is used in 3 Dimensional where as, straight is consider in 2D, when it in 3 dimensional view it is called "Hyperplane"

The classes (or) points which are very close to "difference".

* **Non-linear seperable (or) Kernal SVM** *



* In This sitution, we use "Non-linear SVM" (or) Kernal SVM

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

Non-linear

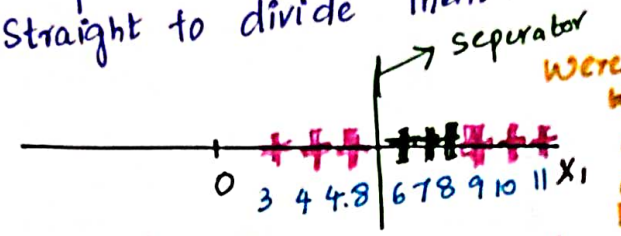
Ex:- Mapping to a higher Dimension

x	y
3	Pink
4	Pink
4.8	Pink
6	green
7	green
8	green
9	Pink
10	Pink
11	Pink

* Earlier, it is degree of '1'

Now, we convert it into "quadratic Equation" and we seperate the data, By writing single condition

These Points are not linearly seperable. we Can't draw a Straight to divide Them.



were we have different class, Before to that is called "Seperator".

So, we Convert Every Data Point to

$$\boxed{f = x-5}$$

Why $f=(x-5)$?

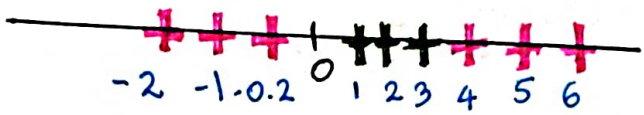
Because, we identify "Seperator."

Ex: 9 → seperator
if it is 9, $\boxed{f = x-9}$

$f = 3-5 = -2$
$f = 4-5 = -1$
$f = 4.8-5 = -0.2$

$f = 6-5 = 1$
$f = 7-5 = 2$
$f = 8-5 = 3$

$f = 9-5 = 4$
$f = 10-5 = 5$
$f = 11-5 = 6$



Next, we square the data points.

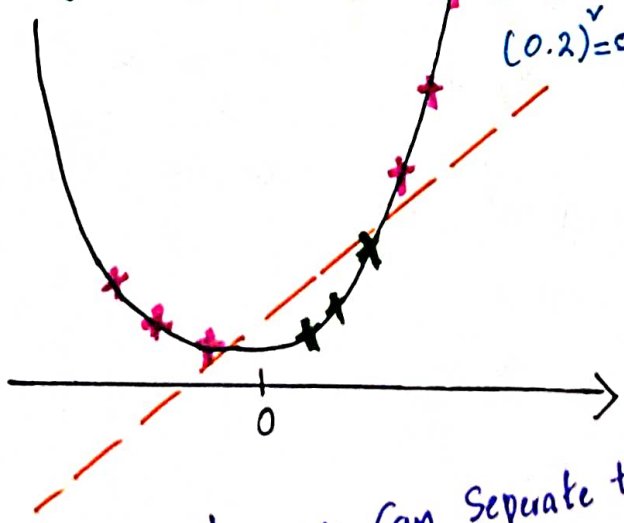
$$\boxed{f = (x-5)^2}$$

$(-2)^2 = 4$
$(-1)^2 = 1$
$(0.2)^2 = 0.04$

$1^2 = 1$
$2^2 = 4$
$3^2 = 9$

$4^2 = 16$
$5^2 = 25$
$6^2 = 36$

Here, we are increasing "Higher" dimension.



Now, we Can Seperate them by "Linear" line

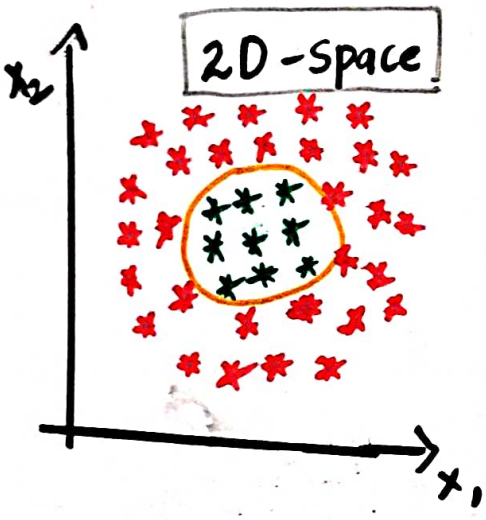
Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

By Table

x	y	x-5	(x-5)²
3	pink	-2	4
4	pink	-1	1
4.8	pink	-0.2	0.04
6	green	1	1
7	green	2	4
8	green	3	9
9	pink	4	16
10	pink	5	25
11	pink	6	36

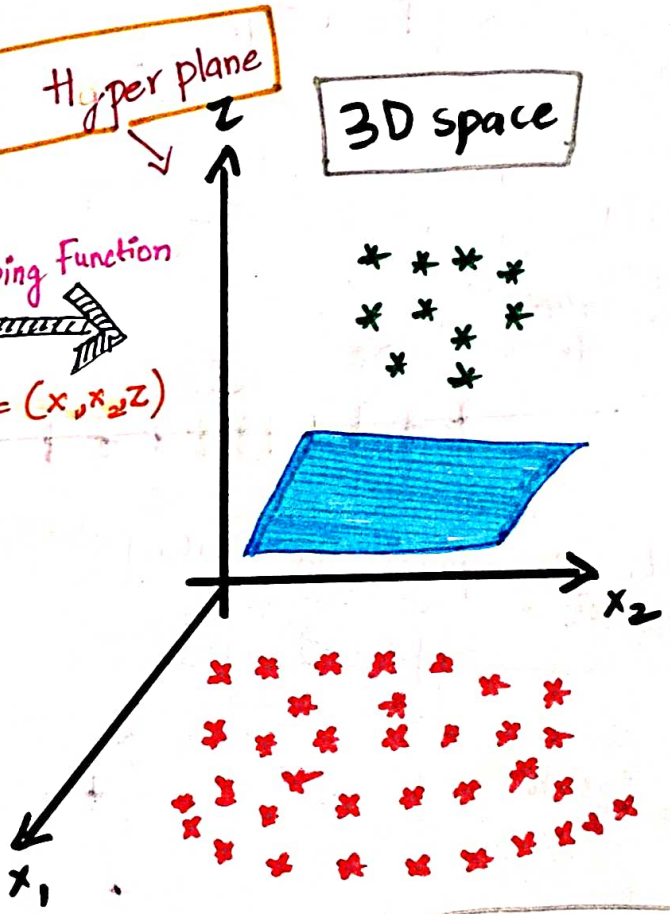
⇒ SVM-Kernal :-

By Hyper plane



Mapping Function

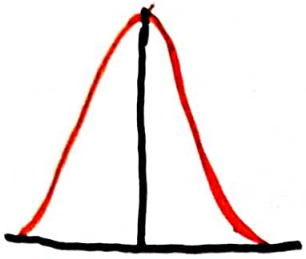
$\varphi(x_1, x_2) = (x_1, x_2, z)$



* Mapping to higher dimensional space can be Highly Compute intensive. More RAM needed.

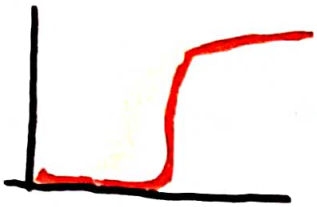
* Types of Kernel functions

Radial Basis function :- Shape of Kernal : R.B.F



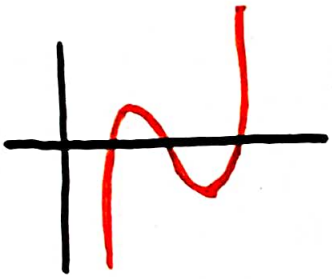
Gaussian RBF Kernal

$$K[\vec{x}, \vec{l_i}] = e^{-\frac{\|\vec{x}-\vec{l_i}\|^2}{2\sigma^2}}$$



sigmoid Kernal

$$K(x,y) = \tanh(\gamma \cdot x^T y + r)$$

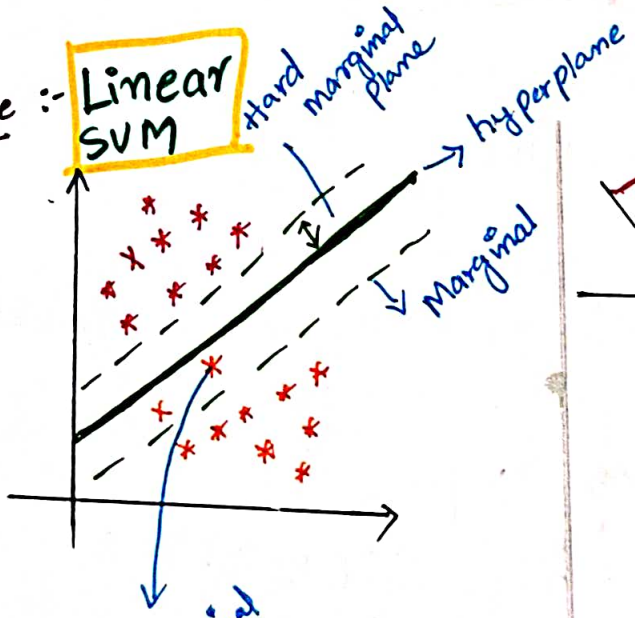


Polynomial Kernal

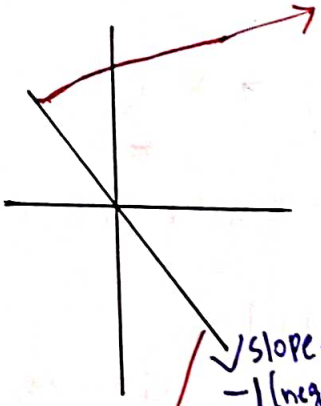
$$K(x,y) = (\gamma \cdot x^T y + r)^d, \ \gamma > 0$$

apply Higher The "degree"

Example :- **Linear SVM**



***



$y = mx + c$ (slope, intercept)

$\Downarrow$

Equation For Straight line

$ax + by \pm c = 0$

$y = \frac{-c}{b} - \frac{ax}{b}$ } same

$\therefore m = \frac{-a}{b}, \ c = \frac{-c}{b}$

slope = -1

-1 (negative direction)

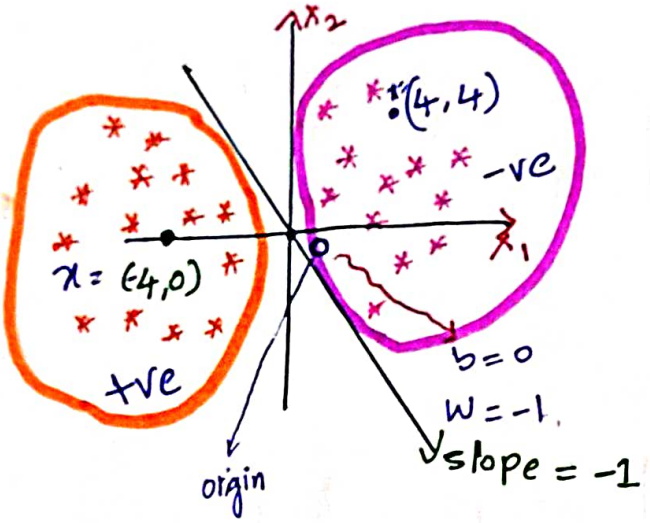
$y = mx + c$

written as

$y = w_1 x + w_2 x + w_3 x + \dots + c$

$$\boxed{y = w^T x + b}$$

Transpose



Equation :- $\boxed{y = W^T x + b}$

$$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [-4, 0]$$

By multiplication

$= 4 \Rightarrow$ (+ve) value.

Any point under the line is Positive value.

New Data point (4,4)

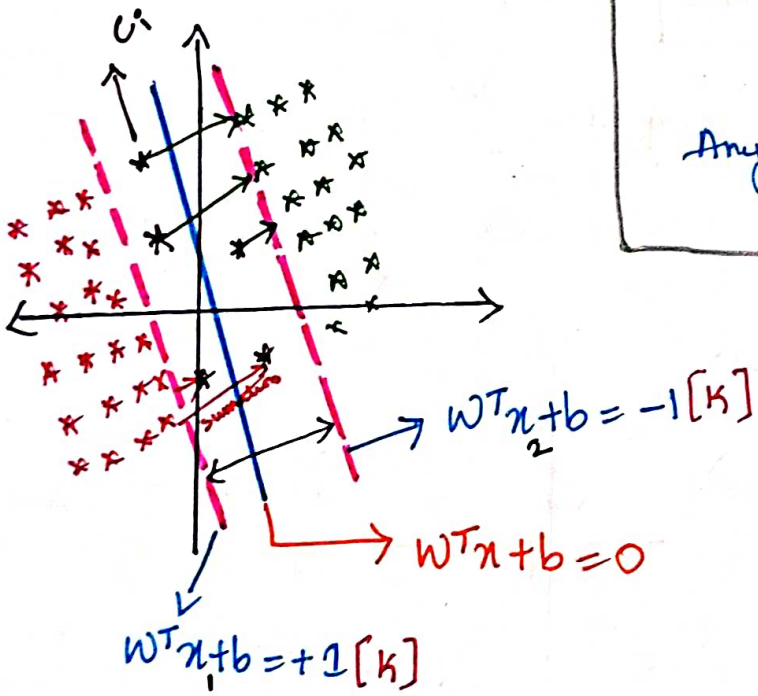
$$\boxed{y = W^T x + b}$$

$$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4 \ 4]$$

$= -4 + 0 = -4$

Any point above the line is negative value.

⇒ Marginal plane



Our Aim should increase distance, to perform model well → Maximize

$$W^T x_1 + \cancel{b} = +1$$

$$W^T x_2 + \cancel{b} = -1$$

$$W^T (x_1 - x_2) = 2$$ (difference between above & below plane)

Vectors + magnitude

$\vec{W}$ $\quad$ $\|W\|$

$$\frac{W^T (x_1 - x_2)}{\|W\|} = \frac{2}{\|W\|}$$

Maximize cost function

$$\underset{(w,b)}{\text{Maximize}} \quad \frac{2}{\|w\|}$$

(Marginal plane will be bigger)

Such that,

$$\boxed{y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}}$$

$y_i * (w^T x_i + b) \geq 1$

⇓

For Correct point

Condition

$(w^T x + b)$ greater than $= 1 = +1$

$(w^T x + b)$ Lesser than $= -1 = -1$

$$\boxed{\underset{(w,b)}{\text{Maximize}} = \boxed{\frac{2}{\|w\|}} \xRightarrow{\text{Both are Equal}} \underset{(w,b)}{\text{Minimize}} = \boxed{\frac{\|w\|}{2}}}$$

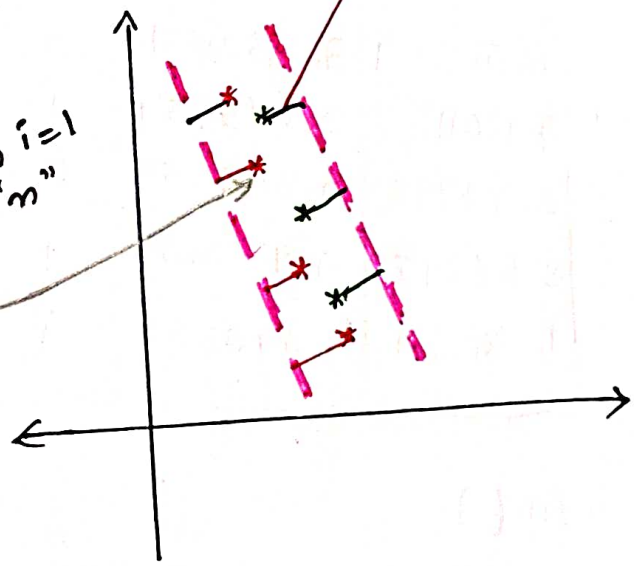
For [SVR] parameter changes.

$$\underset{(w,b)}{\text{Min}} \quad \frac{\|w\|}{2} + \overbrace{c_i \sum_{i=1}^{n} \xi_i}^{\text{"Hypertuning Parameter"}}$$

ξ_i → Eta (summation of the distance of the wrong data points)

c_i → How many wrong prediction (or) Errors, we can have

$\sum_{i=1}^{n}$ → Sumation $i=1$ to "n"



29/04/22 4:30pm

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

29/9/23 11:00pm

CODE :-

Data :-

The data shown here simulates a medical study in which infected with a virus were given various doses of two medicines and then checked 2 weeks later to see if they were still infected. Given this data, Our Goal is to create a classification model than predict (given two dosage measurements) If they mouse will still be infected with virus.

load data, with import libraries.

df = pd.read_csv ("mouse_viral_study.csv")

df.head()

Out :

med_1_mL	Med_2_mL	Virus_Present
6.508231	8.582531	0
4.126116	3.073459	1
6.427870	6.369758	0
3.672953	4.905215	1
1.580321	2.440562	1

df.info()

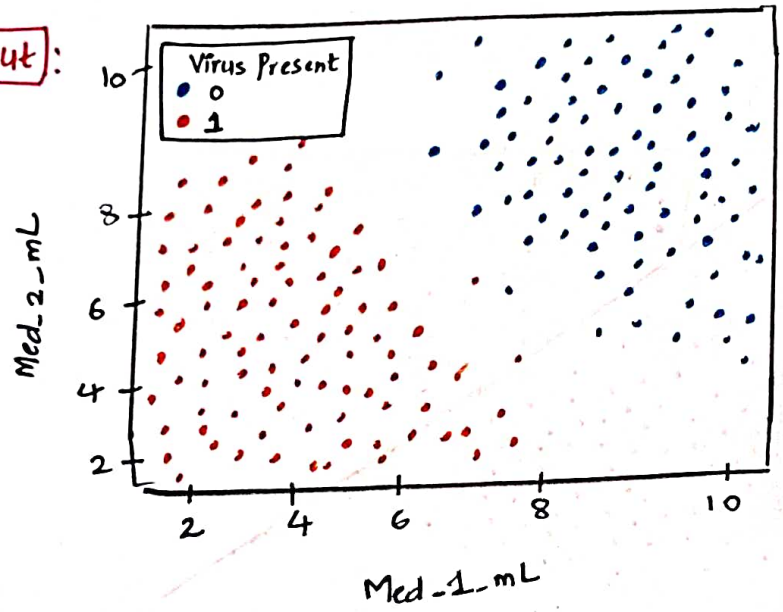
Out : Range Index 400 entries 0 to 399

0	Med_1_mL	400 non null	Float64
1	Med_2_mL	400 non null	Float64
2	Virus_Present	400 non null	int64

EDA

```
# sns.scatterplot(x="Med_1_mL", y="Med_2_ml", hue="Virus present
                  data=df)

# plt.show()
```

Out:



X & Y

```
# x = df.drop("Virus Present", axis=1)
# y = df["Virus present"]
```

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

MODELLING

```
from sklearn.svm import SVC

# model = SVC(Kernel = "linear")

# model.fit(x,y)
```

Out: SVC(Kernal = "Linear")

30/4/22
12:30 pm

30/4/22
11:00Am

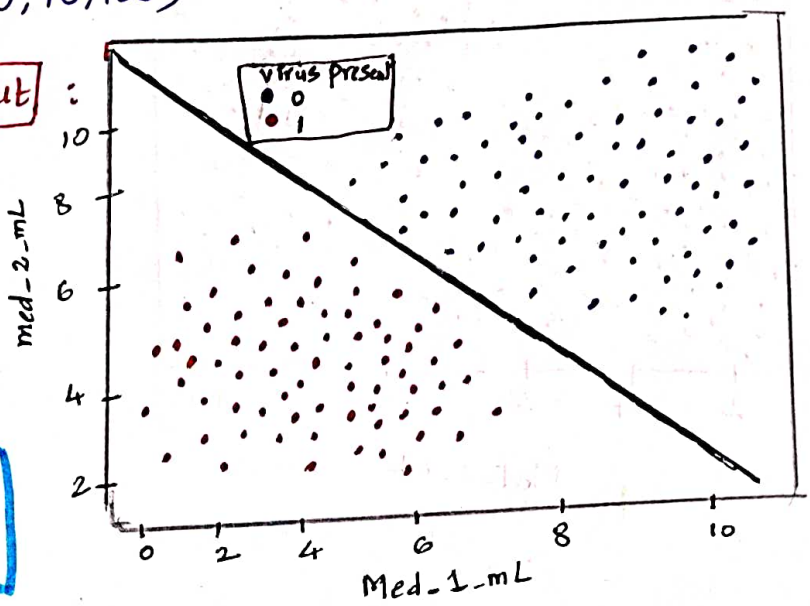
Seperating Hyper plane

```
#sns. scatterplot( x = "Med_1_ML", y = "Med_2_mL", hue = "Virus Pres.
                   Palette = "seismic", data = df)
```

we want to some how automatically Create a Seperating hyperplane (a line in 2D)

```
# x = np.linspace (0,10,100)
#   m = -1
#   b = 11
#   y = m*x +b
# plt.plot (x,y,"k")
```

Out :

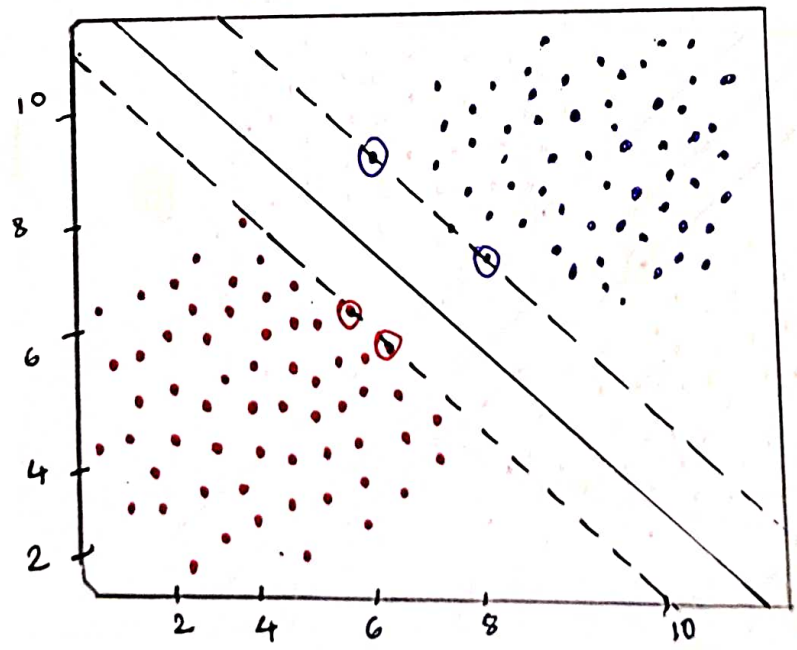


user defined function

```
# def plot_svm_boundary (model, x,y):
        x = x.Values
        y = y.Values
        # scatter plot
        plt.scatter (x[:,0] , x[:,1], c=y, s=30,
                     cmap = "seismic")
        # plot the decision function.
        ax = plt.gca()
        xlim = ax.get_xlim()
        ylim = ax.get_ylim()
```

```
# Creating grid to Evaluate model.
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = model.decision_function(xy).reshape(XX.shape
# plot decision boundary and margins.
ax.Contour(XX, YY, Z, colors="k", Levels=[-1,0,1
           alpha=0.5, line styles=["--", "-", "--"
# plot Support vectors.
ax.scatter(model.Support_vectors_[:,0], model.
           support_vectors_[:,1], s=100, linewidth=1,
           face colours="none", edge colors="k")
plt.show()
# plot_svm_boundary(model, x,y)
```

Out:



Our goal with SVM is to create the best seperating hyperplane with Maximum margin classifier.

In 2 dimensions, The hyperplane is simply a line.
straight

In 3 dimensional → plane

Hyper Parameters

#C :- Regularization parameter.

The strength of the regularization is inversely proportional to "C". Must be strictly positive. The penality is a Squared L2 penalty.

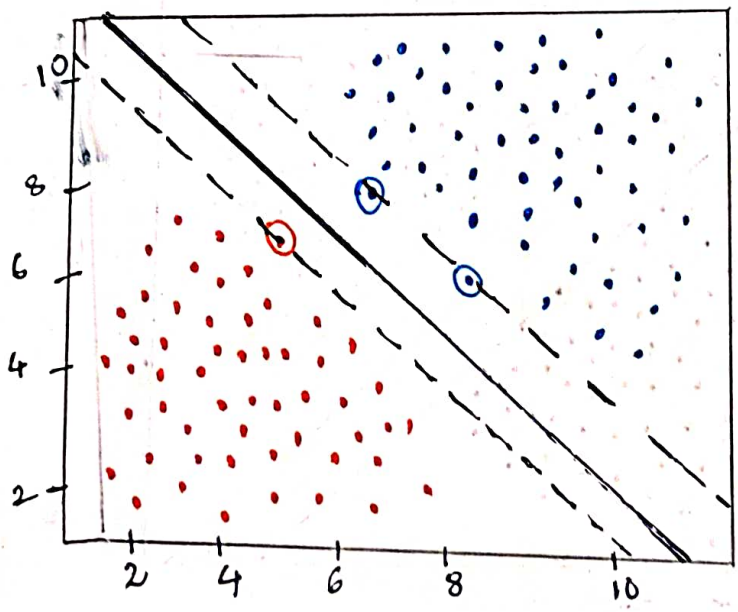
c ↑ (Less Sensitility).

Higher The "c" value - Lesser The missclassification

* c ↓ (# lower The "c" value - in correct answer

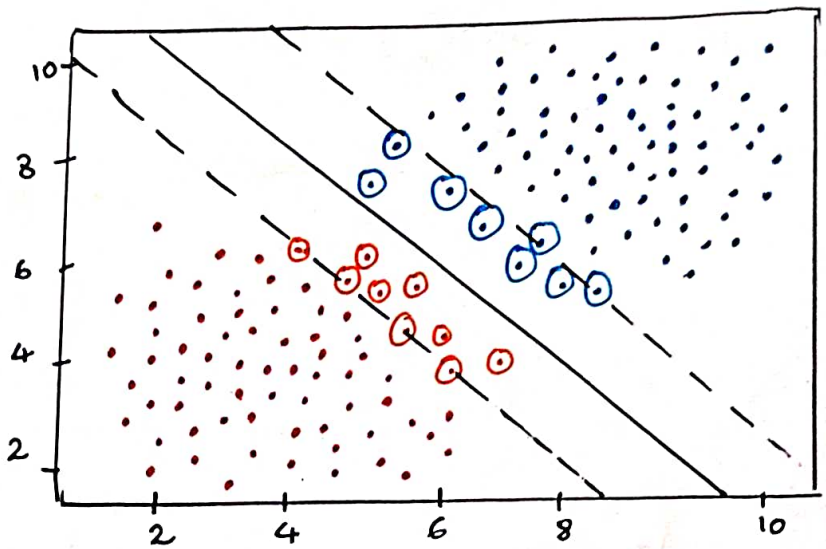
```
# model = SVC (kernel = "Linear", C = 1000)
# model.fit (x, y)
# plot_svm_boundary (model, x, y)
```

out :-



```
# model = SVC (Kernel = "linear", C = 0.05)
# model. fit (x,y)
# plot_svm_boundary (model, x,y)
```

out :

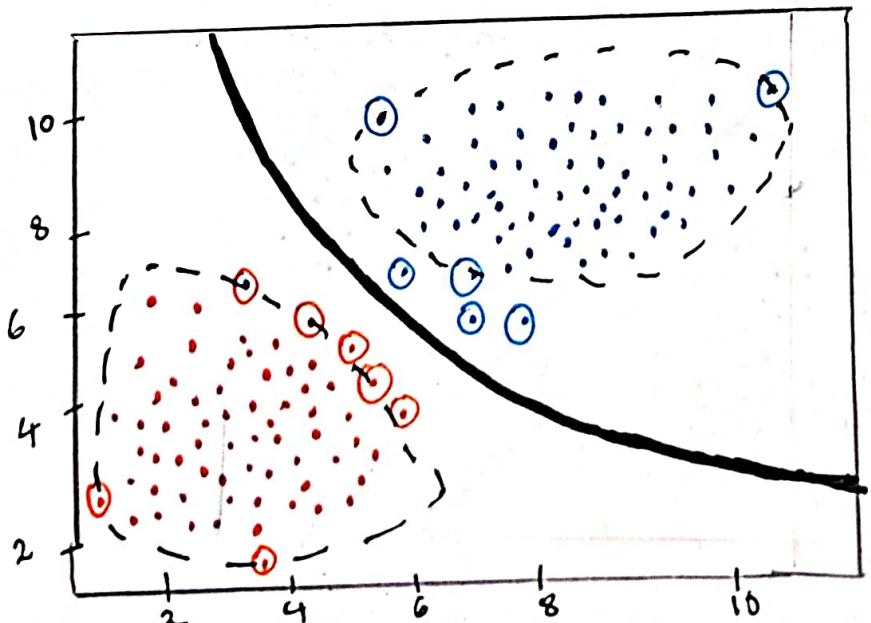


Kernal

1. RBF

```
# model = SVM (Kernel = "rbf", C = 1)
# model. fit (x,y)
# plot_svm_boundary (model, x,y)
```

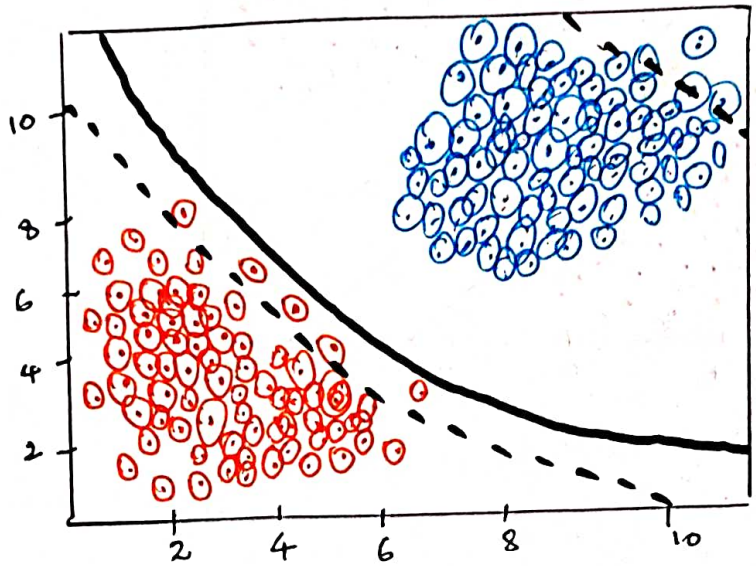
out :



2. Sigmoid

```
# model = SVC (kernel = "sigmoid")
# model.fit (x,y)
# plot_svm_boundary (model, x,y)
```

Out:

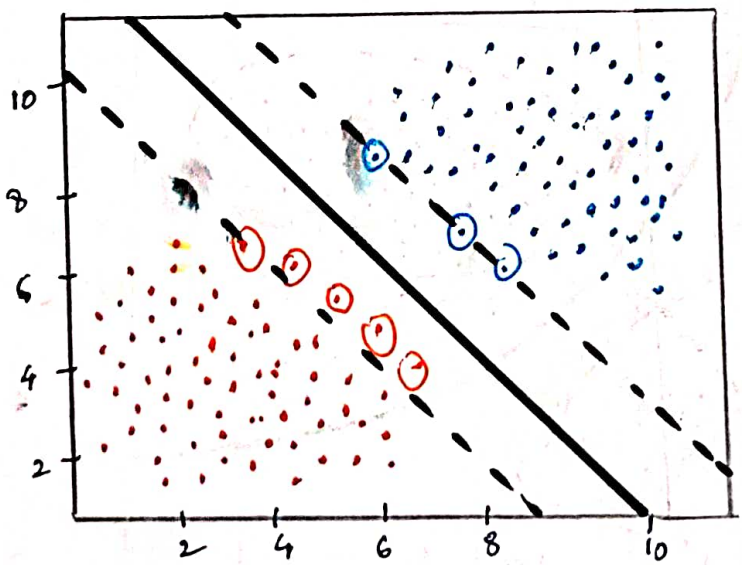


3. Polynomial

"Linear" Both are same.

```
# model = SVC (kernel = "poly", C=1, degree=1)
# model.fit (x,y)
# plot_svm_boundary (model, x,y)
```

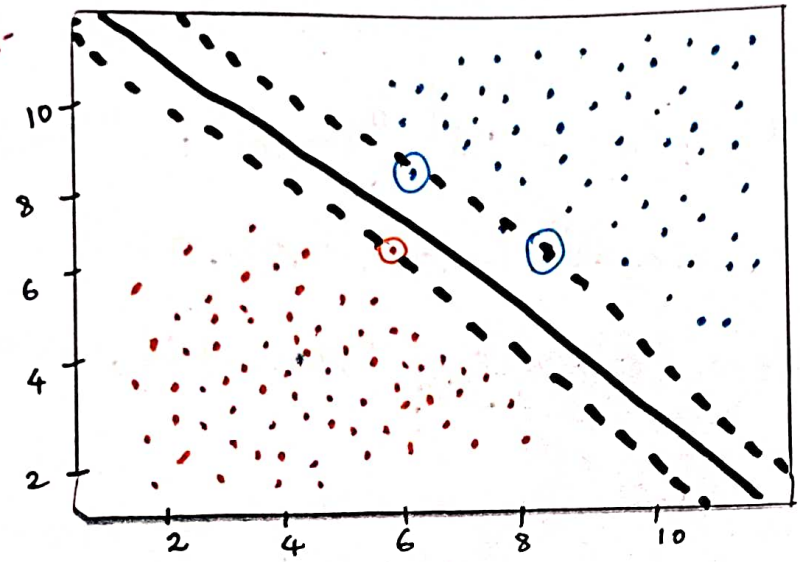
Out:



```
# model = SVC ( Kernel = "Poly" , C = 1, degree = 2)
# model . fit (x,y)
# plot_svm_boundary (model , x,y)
```

"Quadratic curve."

out :-



gamma

gamma :- { scale , auto } or float

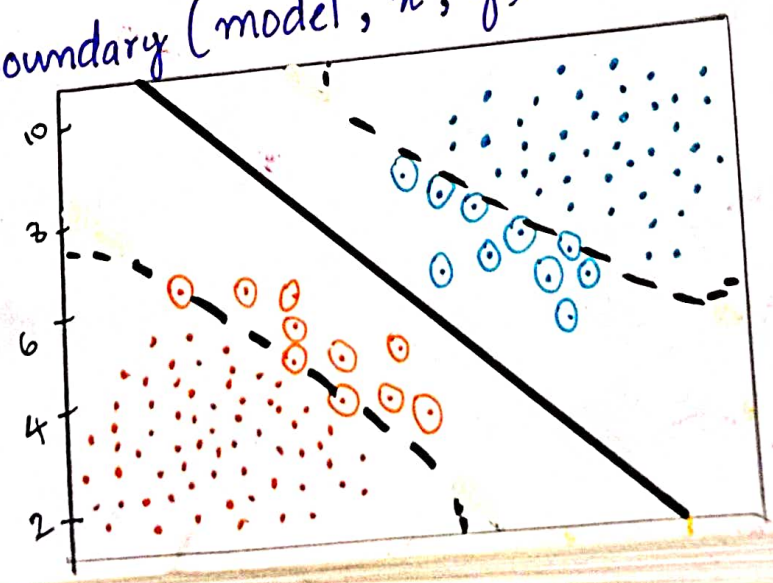
Default = "scale" Kernel Coefficient for "rbf", "Poly" & "sigmoi

"gamma = scale" ⇒ $\frac{1}{\text{num. of feature}} \times x.\text{Var}()$

"gamma = auto" ⇒ $\frac{1}{\text{n_features}}$.

```
# model = SVC ( Kernel = "rbf" , C = 1, gamma = 0.01)
# model . fit (x,y)
# plot_svm_boundary (model , x , y)
```

out :



Hyper parameter Tuning

```
from sklearn.model_selection import GridSearchCV

# estimator = SVC()
# Param_grid = {"c":[0,0.01,0.1], "Kernel":["linear","rbf"]
                , "gamma":[0.01, 0.02, 0.1]}
# grid = GridSearchCV(estimator, Param_grid, cv=5)
# grid.fit(x,y)
```

Out: GridSearchCV(cv=5, estimator=SVC(), Param_grid=
{"c":[0,0.01,0.1], "Kernel":("linear","rbf"), gamma
[0.01, 0.02, 0.1]}

```
# grid.best_Params_
```

Out: {"c": 0.01, "gamma": 0.01, "Kernel": "rbf"}

```
# grid.best_score_
```

Out: 1.0

If we don't write gamma, Kernel = "linear" gives best prediction and accuracy.
: Only for this problem

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

train test split

```
from sklearn.model_selection import train_test_split
# x_train, x_test, y_train, y_test = train_test_split (x,y,
                test_size = 0.25, random_state = 29)
```

Scaling

```
from sklearn.preprocessing import StandardScaler
# sc = StandardScaler ()
# x_train = sc.fit_transform (x_train)
# x_test = sc.fit_transform (x_test)
```

model

```
# model = SVC (Kernel = "rbf", C = 0.01, gamma = 0.01)
                      "Linear"
# model.fit (x_train, y_train)
```

Out

```
SVC [C = 0.01, gamma = 0.01, Kernel = "Linear")
```

Prediction

```
# ypred_train = model.Predict (x_train)
# ypred_test = model.Predict (x_test)
```

accuracy

```
from sklearn.metrics import accuracy_score

# accuracy_score (y_test, ypred_test)
```

Out: 1.0

30/4/22
5:00 pm.

Que:- What is "model Selection"?

Ans:- Identify The best Algorithm with best Hyperparamet

By applying hyperparameter "tunning"

Download Part 01 and Part 03
https://t.me/AIMLDeepThaught/666

•• Join me LinkedIn for the latest updates on Machine Learning:
https://www.linkedin.com/groups/7436898/

•• Join my WhatsApp Channel for the latest updates on Machine Learning:
https://www.whatsapp.com/channel/0029VavNSDO9mrGWYirxz40G