Machine Learning (ML)

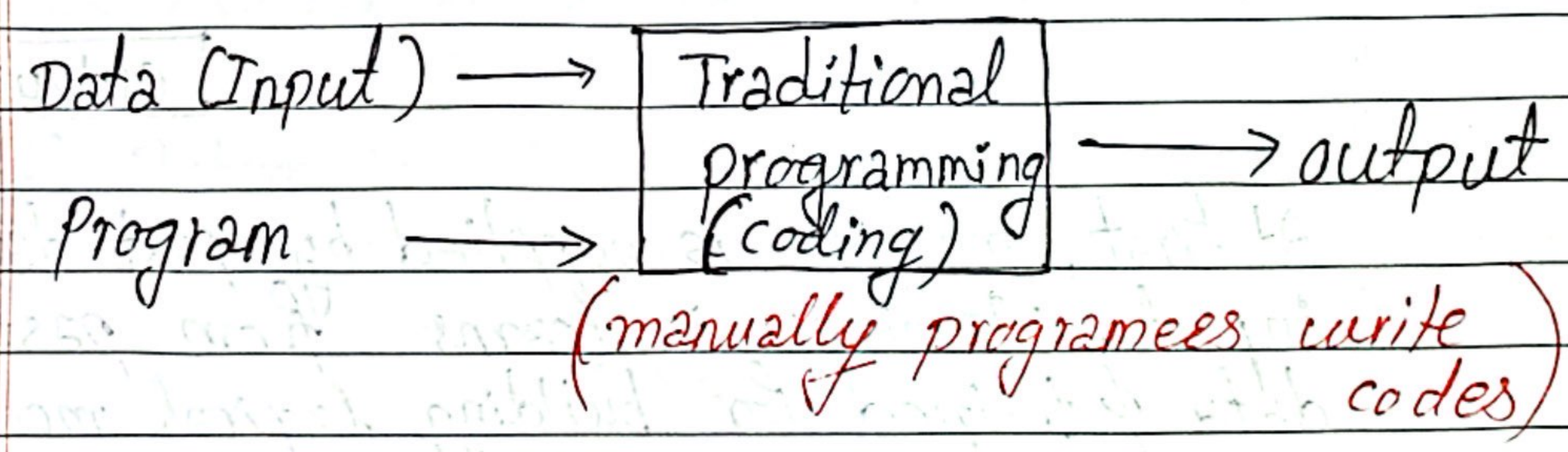
Date _____
Page _____

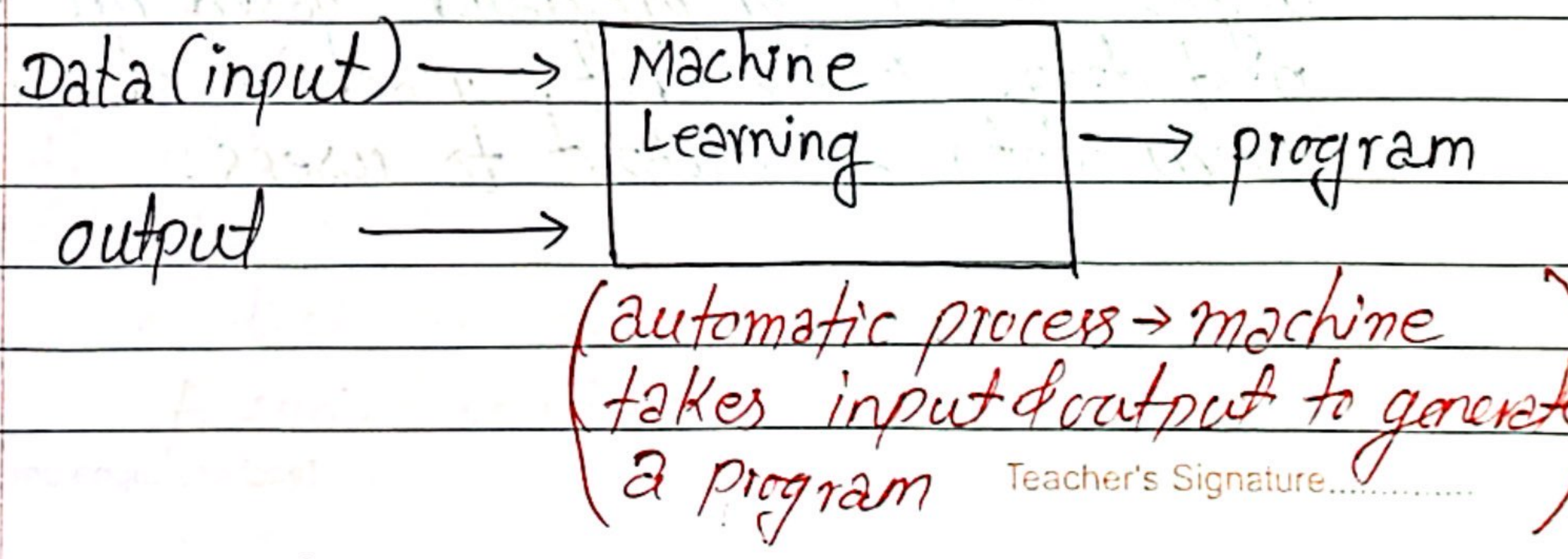
1) What is machine learning?

⇒ Machine learning is the science of getting computers to learn and act like humans do and improves their learning overtime in autonomous fashion, by feeding them data and information in the form of observation and real-world interactions.

2) How ML is different from Traditional Programming?

⇒



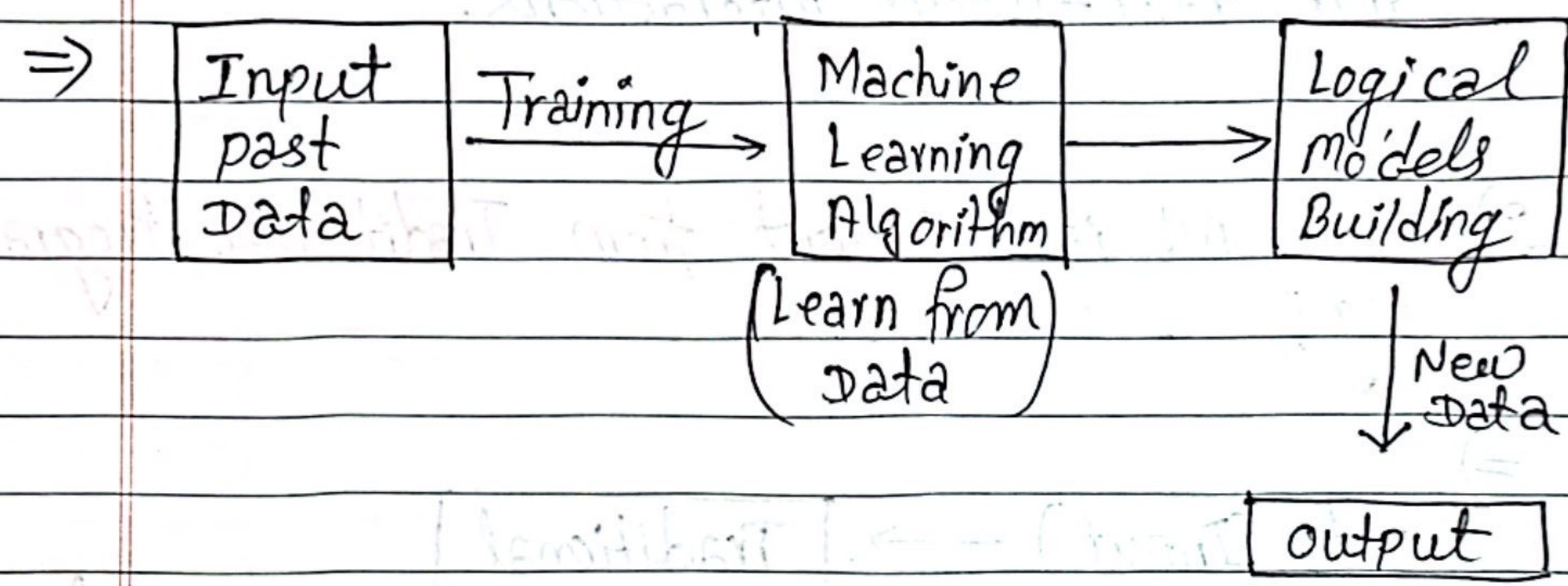


Teacher's Signature.....

Date
Page

examples:- Recommendation in youtube, online shopping websites.
- Automatic recognizing spam emails

3) How it works?



At first, machine is supplied by past data as input. Machines learns from past data & trained for building logical models. Then Machine Learning (ML) algorithm builds logical models or programs based on input past data. After that, machine shows new data as output to users.

Teacher's Signature.....

Date
Page

4) Characteristics of Machine Learning

⇒ a) The potential to perform automated Data Visulization.

b) Precise Data Analysis

c) Business Intelligence at it's Finest.

5) Advangates of Machine learning

⇒ a) Resolving Complex Problems
→ Capable of solving complex problems.

b) Automation for everything.
→ Powerful technique to automate anything like decision making (eg: chat bot)

c) Trends and Pattern Identification
→ learn trends & patterns from user data

d) Wide Range of Application
→ Use in every kind of machines like cars, medical sector, image/video processing, financial services, cyber fraud detection etc.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

e) Data Analysis:

6) Disadvantages :

a) Data Acquisition
→ Need more & quality data. High amt of large data is required & correct data is required.

b) Highly Error-Prone

c) Algorithm Selection

d) Time Consuming

7) Classification of ML

⇒ Classification of ML → Supervised Learning
→ UnSupervised Learning
→ Reinforcement Learning

Teacher's Signature.....

Scanned with CamScanner

Date
Page

a) Supervised Learning

→ In supervised learning, the model is trained using labeled data, which means each input has an associated correct output.

→ Goal: To learn a mapping from inputs to outputs so the model can predict the label of unseen data.

→ eg:- emails as spam or not;
predicting house prices.

b) Unsupervised Learning

→ The model is trained on unlabeled data, meaning the data has no associate output labels.

→ To discover patterns, grouping or relationships within the data.

→ eg:- • customer segmentation in marketing
• reducing the number of features for visualization or efficiency.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

c) Reinforcement learning

→ An agent interacts with an environment and learns to make desisions by receiving rewards or penalties.

→ Goal: To learn a policy or strategy that maximizes cumulative reward overtime

→ eg:-
- Games like chess or Ludo or Go
- Robotics & autonomous driving.

8) Career Opportunities

a) Machine Learning Engineering

b) Business Intelligence Developer

c) Data Scientist

d) Human Centered Machine Learning Designer

e) NLP Scientist.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

9) Prerequisites

a) Basic Knowledge of python programming language

b) Fundamental Knowledge of statistics & probability

c) Understanding of linear algebra & calculus

Teacher's Signature.....

Scanned with CamScanner

Date
Page

Machine learning Life cycle:

⇒ a) Data Collection
b) Data preparation
c) Data Wrangling
d) Data modeling
e) Model Training
f) Model Testing
g) Deployment.

→ The predefine path to use data for machine learning is machine learning life cycle

1) Data Collection

⇒ Goal: Over here is to gather as much as relevant data as possible

- identify various source of information

- Gather data

- Combine the data acquired from various data source.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

2) Data Preparation :

⇒ Data preparation deals with exploration of your data to generate for better results.
It is study of data.

③ Data Wrangling :

⇒ Data Wrangling is the process of cleaning & converting raw data into a useable format.
Steps :
- Filtering / cleaning up of raw data
- Filtering Noise
- Recognizing and removing outliers.
- Removing or filling the missing values.

④ Data Modeling :

⇒ Data Modeling is the step in which we take the data and select a machine learning algorithm to built a model.
Steps:
- Selecting machine learning algorithm
- Building the models.
- Validating the results.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

5) Model Training:

⇒ A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.

⇒ Model understands patterns in data, identify features, understands rules in it.

6) Model Verification

⇒ This stage of the Machine learning life cycle involves checking for the accuracy of the model by providing with the inputs that are unseen.

7) Model deployment:

⇒ This is the final step in the machine learning life cycle where we have a brilliant model ready to go to production.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

Dataset & Type of Dataset :

1) Dataset

=) Dataset is a collection of records usually presented in tabular form.

example:-

Name	Age	City	Department
Shrawan	25	Pokhara	Digital Marketing
Bishnu	30	Jaipur	Developer
Jacky	29	Dhaka	SEO
Sabina	26	Chitwan	Content Writer
Bimala	24	Tikapur	Trainer

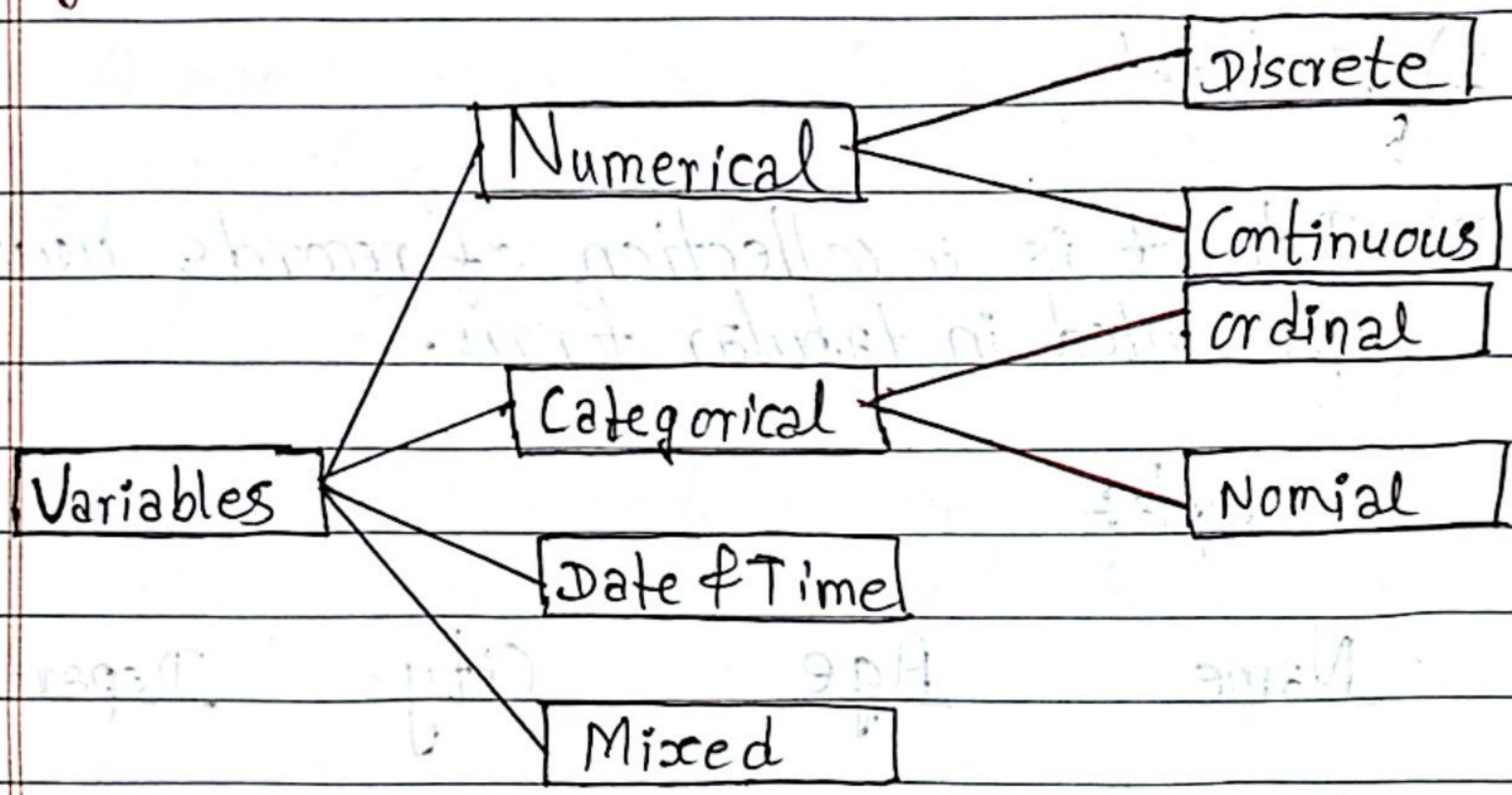
Teacher's Signature.....

Scanned with CamScanner

Date
Page

2) Types of DataSets:

=)



a) Numerical

→ It is numeric numbers.
→ Integer, Floating numbers

i) Discrete

→ Data that repeat itself.
→ Repeating data (or number)

ii) Continuous

→ Real numbers in given range.

Teacher's Signature.....

Date
Page

b) Categorical

⇒ Categorical data are String type data.

i) Ordinal

→ In ordinal variable, there is a kind of order follow.
eg: (S, M, L, XL, XXL, XXXL in Clothes).

→ We can see interconnection between datas.

ii) Nominal

→ Randomly data is entered i.e. it has not order to follow.

→ eg:- (Python, HTML, Java, C++ in Languages)

→ Data are not interconnected.

c) Data & Time

→ Date Time can contain date only, time only.

d) Mixed Variables:

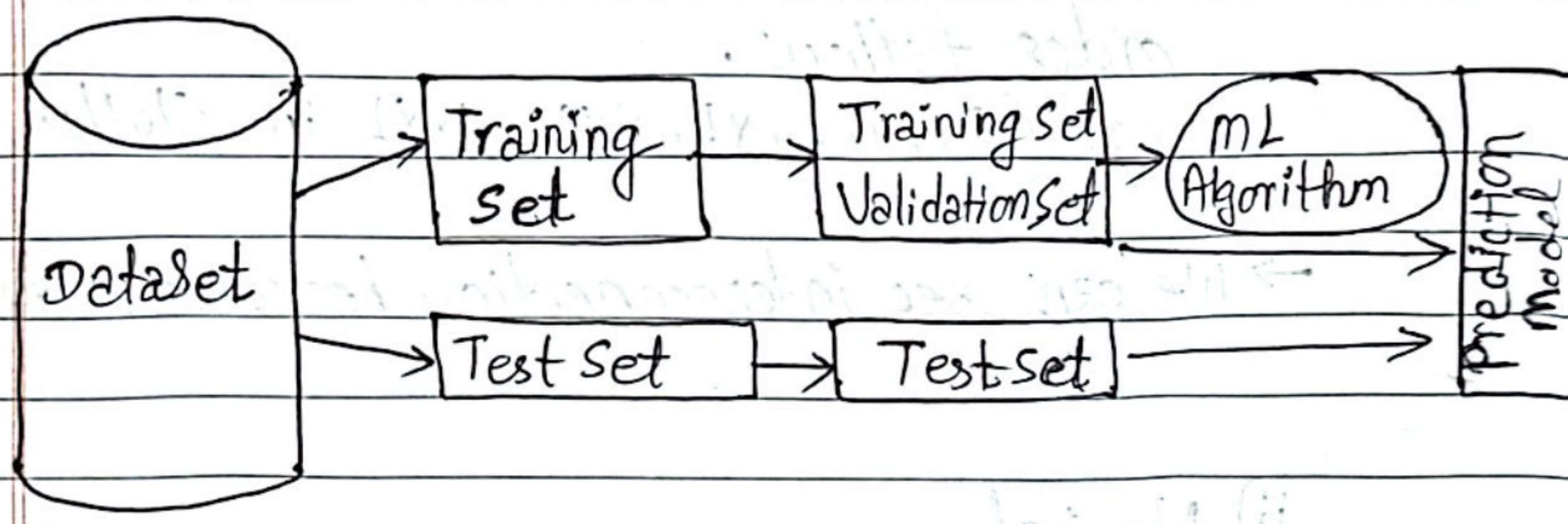
→ Variables which contains numbers & Categories data.

Teacher's Signature.

Date
Page

3) Need of Datasets

=) In the development phase of ML projects, datasets are classified as Training dataset & Test Dataset.



→ Dataset is required to train the machine, and further for self improvement by machine itself.

→ We can bring datasets from following Sources
i) Kaggel Datasets (website)

ii) Amzon Datasets (website)

iii) UCI Machine Learning Repository (site)

iv) Google's Data Search Engine (site)

Teacher's Signature.....

Date
Page

v) Microsoft Datasets (site)

vi) Awesome Public Datasets Collections (site). (github)

vii) Scikit-learn Datasets. (site)

viii) Computer Vision Datasets. (site)

ix) Government Datasets.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

Data Cleaning / Data pre-processing :

⇒ Data Preprocessing is a process of converting your raw data into suitable form.

⇒ Data cleaning is the process of preparing data for analysis /ML/DL by removing or ~~modify~~ modifying data that is incorrect, incomplete, irrelevant, duplicated or improperly formatted.

eg :-

1) dataset.isnull().sum()
↳ gives no. of null value in each column.

2) dataset.shape[0]
↳ returns total no. of rows

3) (dataset.isnull().sum() / dataset.shape[0]) x 100
→ returns null value in percentage

4) dataset.isnull().sum().sum()
→ shows total null value in data.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

5) ((dataset.isnull().sum().sum())/(dataset.shape[0]*
dataset.shape[1])) *100

→ returns total null value in total data in %

6) dataset.notnull().sum().sum()

→ shows total filled data.

7) to show graph :-

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(dataset.isnull())
plt.show()
```

→ this shows graph of null value.

8) delete column having null value more than 50%.

⇒ dataset.drop(columns=["Credit_History"],
inplace=True)

⇒ dataset.dropna()

↑ Remove row having null value.

Teacher's Signature.....

Date
Page

filling/handling missing values

1) dataset.isnull().sum()

2) dataset.fillna(10) → fill '10' in missing content (But not right way)

3) dataset.fillna(method="ffill", axis=1)
→ forward filling
→ fill null value/content by copying just previous column's value.

4) dataset.fillna(method="bfill", axis=1)
→ backward filling.
→ fill null value content by copying just backward column's value.

5) dataset.fillna(method="ffill")
→ forward filling in row.
→ fill null value/content by copying just above row's value.

6) dataset.fillna(method="bfill")
→ backward filling in row
→ fill null value/content by copying just below row's value.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

7) for i in dataset.select_dtypes(include="object").columns:
dataset[i].fillna(dataset[i].mode()[0], inplace=True)
↳ This fill all categorial data.

# for numeric value:
import sklearn.impute import SimpleImputer

8) dataset.info()

9) data.select_dtypes(include="float64").columns

10) si=SimpleImputer(strategy="mean")
ar=si.fit_transform(dataset[['Amount','Amount1','Amount2']])

11) new_data=pd.DataFrame(ar, columns=dataset.select_dtypes(include="float64").columns)

12) new_data.isnull().sum()

out 12): Amount 0
Amount1 0
Amount2 0
dtype: int64

Teacher's Signature.....

Date
Page

Encoding (categorial data to numerical value)

1: from sklearn.preprocessing import LabelEncoder

2: label_encoder_x = LabelEncoder()

3: label_encoder_x.

1) #filldata.
dataset["Month"].fillna(dataset["Month"].mode()[0], implace=True)

2) #seperating
* en_data = dataset[["Month","Monthofpayment"]]
* endata

3) pd.get_dummies(en_data).info()
print columns & info

4) from sklearn.preprocessing import OneHotEncoder

5) ohe = OneHotEncoder()
ar = ohe.fit_transform(en_data).toarray()
pd.DataFrame(ar, columns=["columns_names"])

Teacher's Signature.....

Date _____
Page _____

* Label encoding:

```
[1]: import pandas as pd

[2]: dict1={
        "Animals": ['Cow', 'Horse', 'Dog', 'Tiger', 'Rabbit'],
        "Shelter":['Shed', 'Stable', 'Kennel', 'Cave', 'Hutch']
     }

[3]: df = pd.DataFrame(dict1)
```

out [3] df:

	Animals	Shelter
0	Cow	Shed
1	Horse	Stable
2	Dog	Kennel
3	Tiger	Cave
4	Rabbit	Hutch.

```
[4]: from sklearn.preprocessing import LabelEncoder

[5]: le = LabelEncoder()
[6]: le.fit_transform(df["Animals"])
```

out [6]: array([0, 2, 1, 4, 3])

```
[7]: df["en_name"] = le.fit_transform(df["Animals"])
```

shows another column for ~~actual~~ encoded numbers

```
[8]: df
```

[8]:

	Animals	Shelter	en_name
0	Cow	Shed	0

Teacher's Signature.....

Date
Page

* ordinal encoding:

```
[1]: import pandas as pd
[2] dict1 = {
        "Size":['s','l','m','xl','s','l','xl','l','m','s','xxl','l'
    }
[3]: df = pd.DataFrame(dict1)
     df
```

out[3]:

	size
0	s
1	l
2	m
3	xl
4	s
5	l
6	xl
7	l
8	m
9	s
10	xxl
11	l

```
[4]: ordered_data = [['s','m','l','xl','xxl']]
[5]: from sklearn.preprocessing import OrdinalEncoder
[6]: oe = OrdinalEncoder(categories = ordered_data)
[7]: oe.fit(df[["size"]])
```

Teacher's Signature.....

Date
Page

[8]: oe.transform(df[["size"]])

out[8] array([[0],
[2],
[1],
[3],
[0],
[2],
[3],
[2],
[1],
[0],
[4],
[2]]).

[9]: df[["en_size"]]= oe.transform(df[["size"]])
df

[9]

	Size	ensize
0	s	0.0
1	l	2.0
2	m	1.0
3	xl	3.0
4	s	0.0
5	l	2.0
6	xl	3.0
7	l	2.0
8	m	1.0
9	s	0.0
10	xxl	4.0
11	l	2.0

Teacher's Signature.....

Scanned with CamScanner

Date
Page

using map function

```
[10]: ord_data = {"s": 1, "m": 2, "l": 3, "xl": 4, "xxl": 5}
```

```
[11]: df["en_size_map"] = df["size"].map(ord_data)
```

```
[12]: df
```

out[12]:

	size	en_size	en_size_map
0	s	0.0	1
1	l	2.0	3
2	m	1.0	2
3	xl	3.0	4
4	s	0.0	1
5	l	2.0	3
6	xl	3.0	4
7	l	2.0	3
8	m	1.0	2
9	s	0.0	1
10	xxl	4.0	5
11	l	2.0	3

Teacher's Signature.....

Scanned with CamScanner

Date
Page

* Outliers

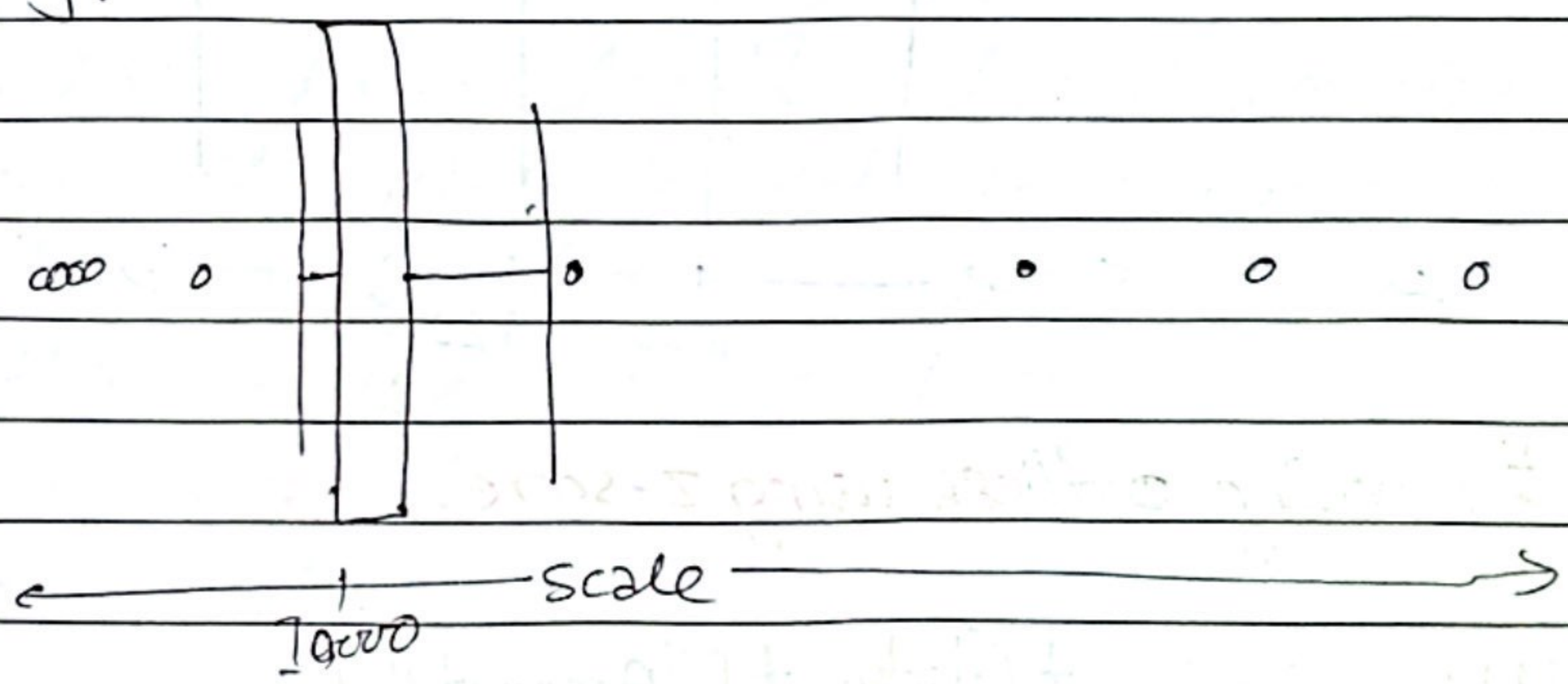
[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
dataset=pd.read_excel("Data2.xlsx")
```

[3]:
```
dataset.describe()
```

[4]:
```
sns.boxplot(x="Amount1", data=dataset)
plt.show()
```

out[4]:



[5]:
```
q1 = dataset["Amount1"].quantile(0.25)
q3 = dataset["Amount1"].quantile(0.75)
```

[6]: $IQR = q_3 - q_1$

Teacher's Signature.....

Date _____
Page _____

[7]: min_range = q1-(1.5 * IQR)
max_range = q3 + (1.5 * IQR)

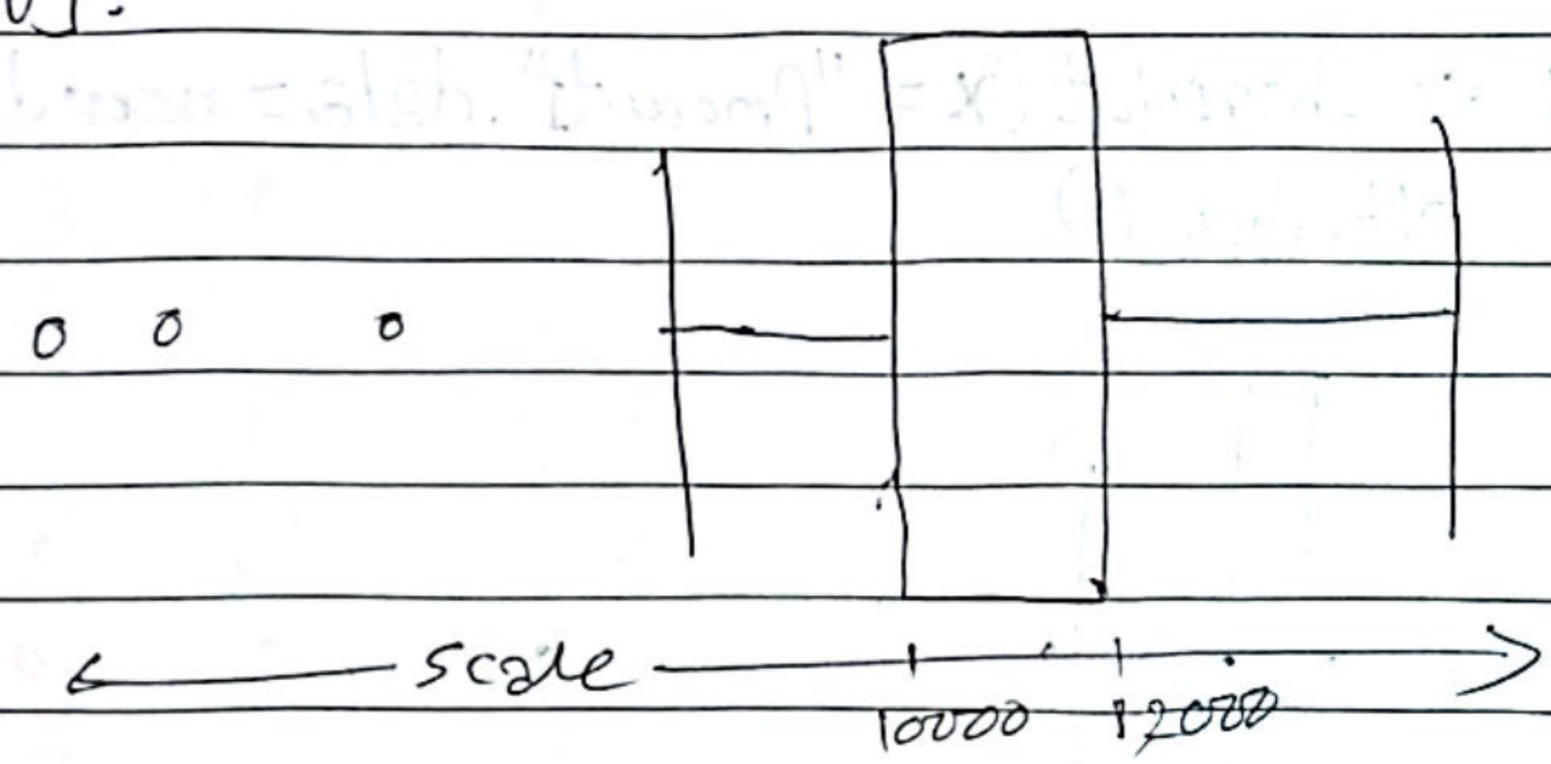
store value less than maxrange & remove

[8]: new_dataset = dataset[dataset["Amount1"] <= max_ran

[9]: new_dataset.shape

[10]: sns.boxplot(x="Amount1", data=new_dataset)
plt.show()

[10]:



removing outliers using z-score...

[11]: sns.displot(dataset["Amount1"])

[12]: min_rang = dataset["Amount1"].~~name()~~mean() -
(3*dataset["Amount1"].std())
max_rang = dataset["Amount1"].mean() +
(3*dataset["Amount1"].std())

Teacher's Signature.....

calculating z-score $\left[Z = \frac{x - \mu}{\sigma} \right]$

Date
Page

```
[13]: z_score = (dataset["Amount1"] - (dataset["Amount1"].mean()
        ))/(dataset["Amount"].std())
```

```
[14] z_score>3
```

```
[15] dataset["Z-Score"] = z_score
```

storing 'z_score' value in table.

Feature Scaling:

⇒ During Model training, we have both datas which have larger value & smaller value. When this kind of data is given to machine it dominate larger value (having higher value than smaller data) and neglegible value of smaller data. So we do feature scaling on our data to take both data's values on same level.

Types:
- Standardization
- Normalization

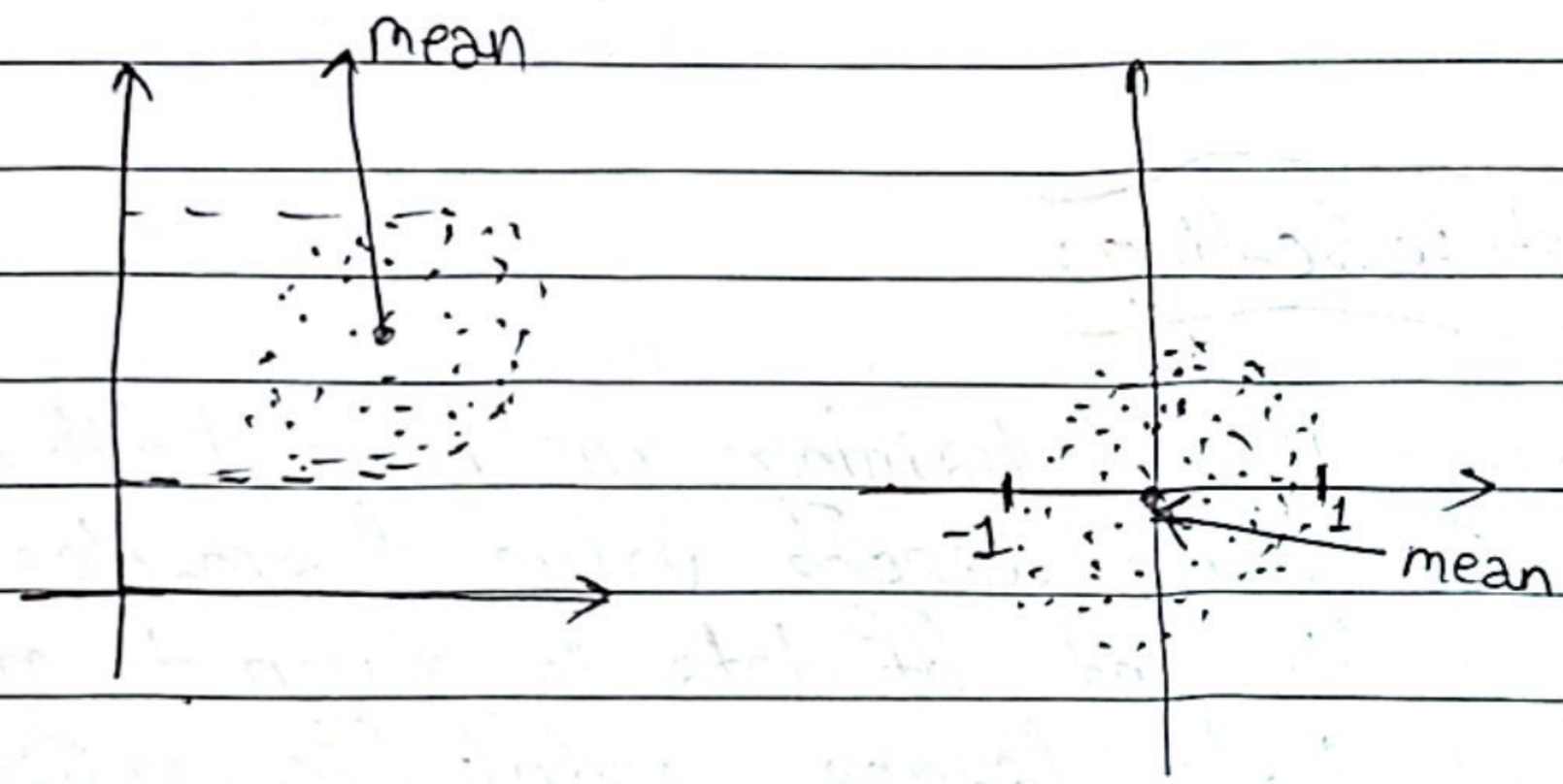
Teacher's Signature.....

Date
Page

1) Standardization:

⇒ It is a very effective technique which re-scales a features value so that it has distribution with 0 mean value and variance equals to 1.

$$X_{new} = \frac{X_i - X_{mean}}{\text{Standard Deviation}}.$$



- Scaling doesn't remove outliers.

eg:-

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(dataset[["Number3"]])

ss.transform(dataset[["Number3"]])
dataset["Number3_scaled = ss.transform(dataset[["Number
```

Teacher's Signature.....

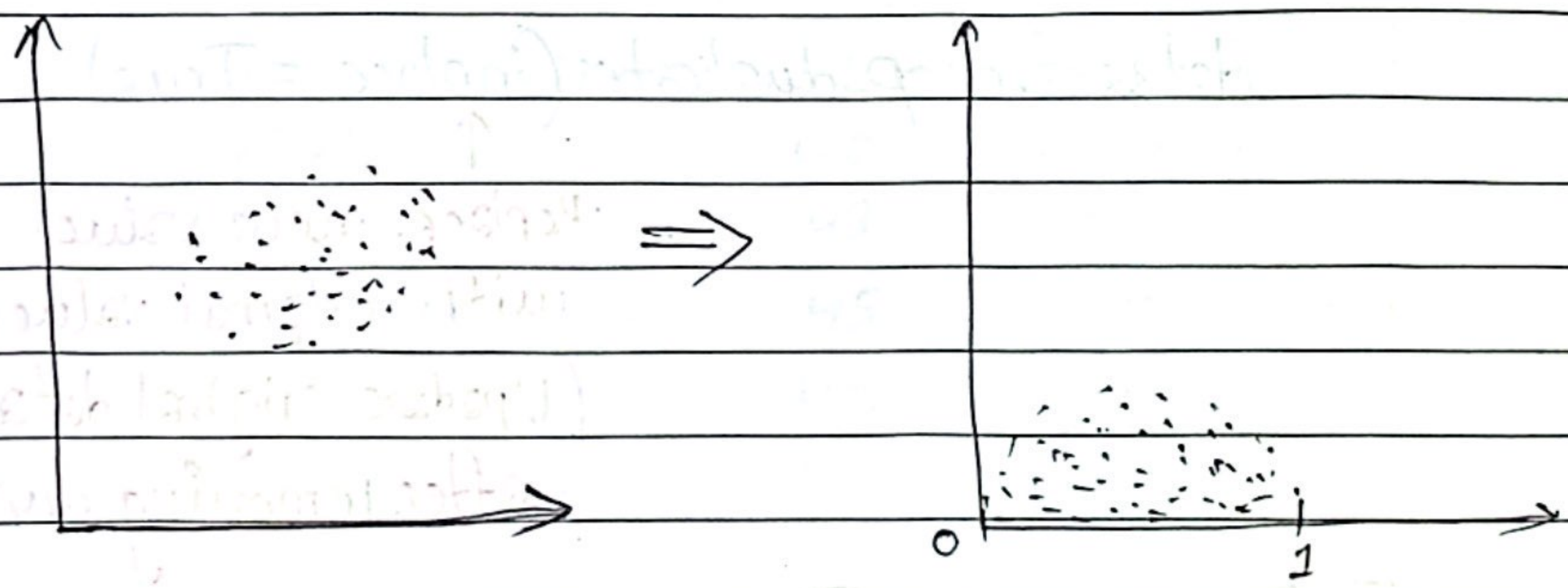
Date _____
Page _____

2) Normalization

→ Min-Max Techniques (scaler)

→ It is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

$$X_{new} = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$



eg:-

```
~~import~~from sklearn.preprocessing import MinMaxScaler.

ms = MinMaxScaler()
ms.fit(dataset[["Number2"]])

ms.transformer(dataset[["Number2"]])
dataset["Number2_scaled"] = ms.transform(dataset[[
                                        "Number2"]])
```

Teacher's Signature .....

Scanned with CamScanner

Date
Page

Duplicate Data:

⇒ To train machine for all data ~~are~~ equally we need to remove duplicates data.

eg:- dataset.duplicated()
↳ show duplicate data in 'True' or 'False'

dataset.drop_duplicates()
↳ To remove duplicates data

dataset.drop_duplicates(inplace = True)
↑
· Replace new values with original values. (Update original data after removing duplicate);

Replace & change datatypes:

⇒ Here, data is

	Name	Nepali	English	Data
0	Ram	40	45	4+ → categorical
1	Hari	45	48	45
2	Krishna	47	43	65
3	Sita	48	45	3+ → categorical

Teacher's Signature.....

4	Gita	44	49	36
5	Krishna	46	43	55

(*) Now, replacing:

```
dataset["Data"].replace("4+", "46", inplace=True)
dataset["Data"].replace("3+", "3", inplace=True)
```

After replacing:

Name	Nepali	English	Data
Ram	40	45	46 → datatype obj.
Hari	45	48	45 → (int)
Krishna	47	43	65.
Sita	48	45	3 → datatype object
Gita	44	49	36.
Krishna	46	43	55.

Changing datatypes:

```
dataset["Data"]=dataset["Data"].astype("int64")
```

→ Now all data in "Data" is 'int'.

Teacher's Signature.....

Date
Page

Supervised learning

- Regression learning
- Classification learning.

Regression Analysis:

⇒ When outcomes from prediction of datasets is continuous in nature, then we use regression analysis. For example, car's selling price prediction.

Some examples of regression can be as:

- Prediction of rain using temperature & other factors.
- Determining Market trends.
- Prediction of road accidents due to rash driving.

Types of Regression Analysis Algorithm.

1) Linear Algorithm:
   a) Linear Regression
   b) Multi-Linear Regression
   c) Lasso Regression

Teacher's Signature.....

Date
Page

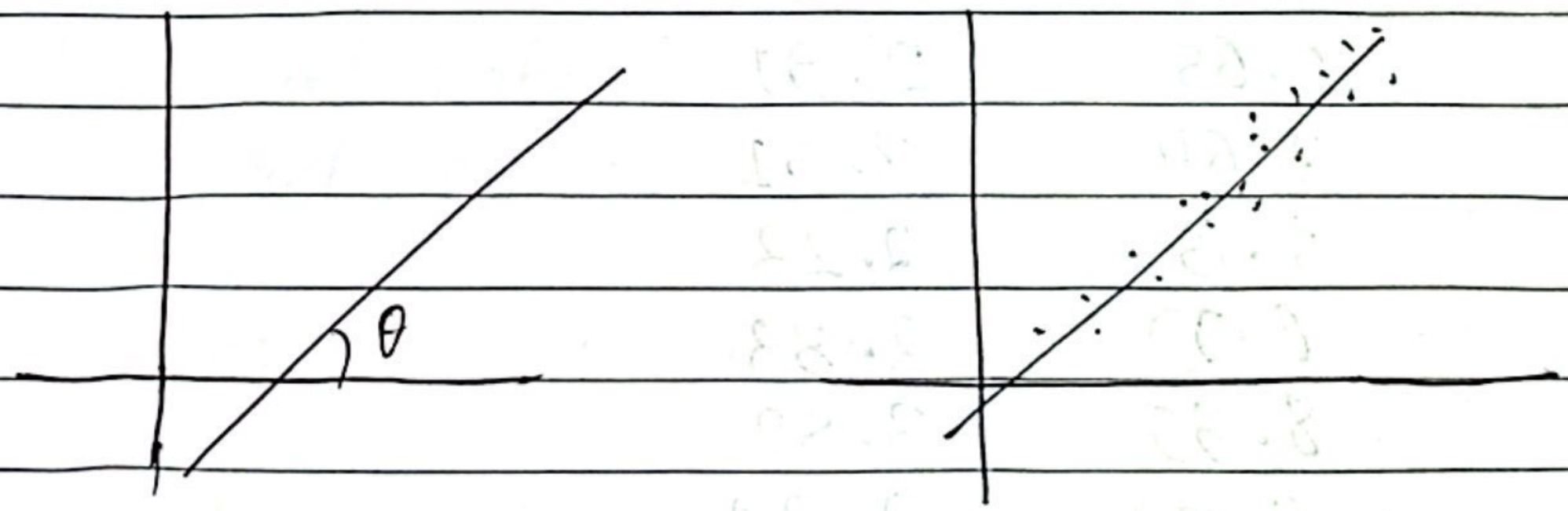
d) Ridge Regression.

2) Non-Linear Algorithm:

a) Polynomial Regression
b) Decision tree Regression
c) Random forest Regression.
d) Support Vector Machine
e) K- Nearest Neighbour

Simple Linear Regression:

→ We use Simple linear regression here,
→ eqn is $y = mx + c$.



⇒ Predecting values based on old values.

Scanned with CamScanner

Date
Page

=) Here, we predict packages based on previous records of cgpa & packages.

Dataset is;

cgpa	Packages
6.89	3.26
5.12	1.98
7.82	3.25
7.42	3.67
6.94	3.57
7.89	2.99
6.73	2.6
6.75	2.48
6.09	2.31
8.31	3.51
5.32	1.86
6.61	2.66
6.65	2.91
7.64	3.51
5.15	2.22
6.29	2.83
8.35	3.52
6.89	3.39
7.34	2.99
8.23	3.34
⋮	⋮

Teacher's Signature.....

Date
Page

steps :-

[1] import pandas as pd

[2] dataset = pd.read_excel ("package_prediction")

[3] dataset.isnull().sum()

# here, x = cpgp => independent variable (input)
y = package => dependend variable (output)

[4] x = dataset[["cgpa"]] → 2D
[5] y = dataset["package"] → 1D

[6]: import seaborn as sns
import matplotlib.pylot as plt

[7]: sns.scatterplot(x = "CGPA", y = 'PACKAGES', data = dataset)
plt.show() =>

[8]: from sklearn.model_selection import train_test split.

Teacher's Signature.....

Date
Page

[9]: x_train, x_test, y_train, y_test =
train_test_split (x, y, test_size=0.2, random_state=

[10]: from sklearn.linear_model import LinearRegression

[11]: lr = LinearRegression ()
lr. fit (x_train, y_train)

[12]: lr. predict ([[6.89]])
out [12]: array ([2.9959986])

[13]: lr. score (x_test, y_test) * 100
[13]: 87.8160.

[14]: lr. predict ([[8]])
out [14]: array ([3.5296513])

[15]: lr. coeff_ → y = mx + c
array ([0.48076788])

[16]: lr. intercept_ → y = mx + c
np. float (-0.316491)

[17]: y_prd = lr. predict (x)

Teacher's Signature.....

Date
Page

[9]: x_train, x_test, y_train, y_test =
train_test_split (x, y, test_size = 0.2, random_state=

[10]: from sklearn.linear_model import LinearRegression

[11]: lr = LinearRegression ()
lr. fit (x_train, y_train)

[12]: lr. predict ([[6.89]])
ut [12]: array([2.9959986])

[13]: lr. score (x_test, y_test) * 100
[13]: 87.8160

[14]: lr. predict([[8]])
t [14]: array([3.5296513])

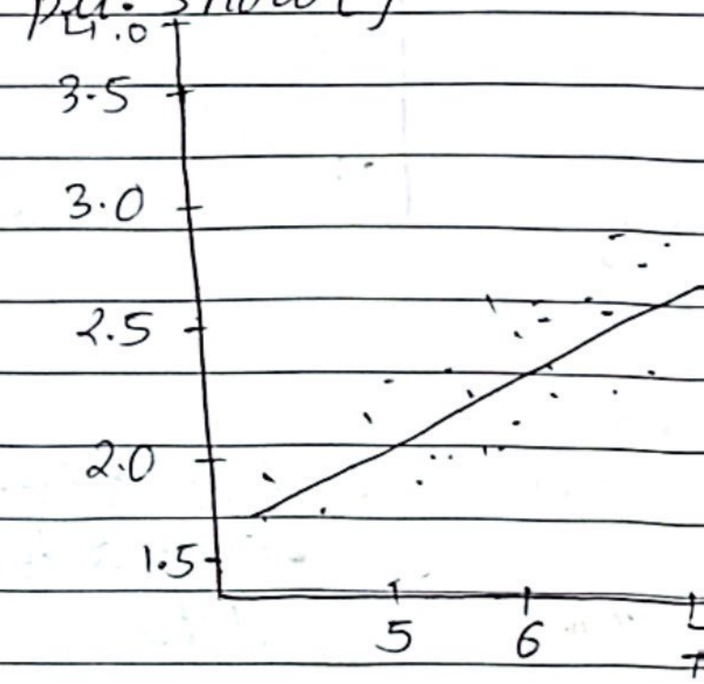
[15]: lr.coeff_ → y = mx + c
array ([0.48076788])

[16]: lr. intercept_ → y = mx + c
np. float (-0.316491)

[17]: y_prd = lr. predict (x)

Teacher's Signature.....

[18]: sns. scatterplot(x='CGPA',
data =
plt. plot (dataset ["CGP
plt. legend ([" org data
plt. show ()

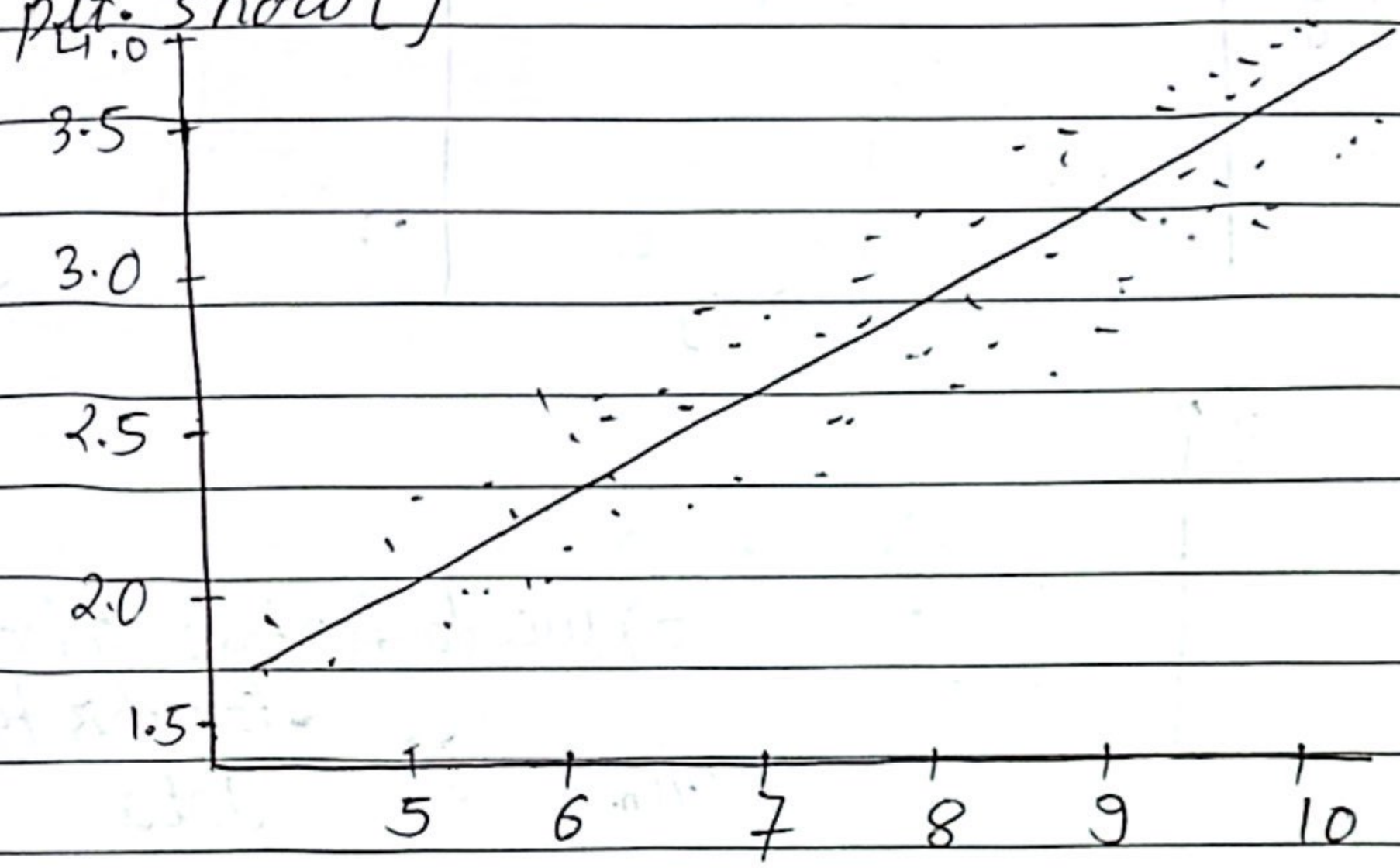


Multiple linear Regression :-

⇒ Multiple linear Regressi
of simple Linear Regre
than one predictor v
response variable.
eg:- showing salary
experience.

Date
Page

```
[18]: sns.scatterplot(x='CGPA', y='PACKAGES',
                      data = dataset)
      plt.plot(dataset["CGPA"], y_prd, c="red")
      plt.legend(["org data","predict line"])
      plt.show()
```



Multiple linear Regression:-

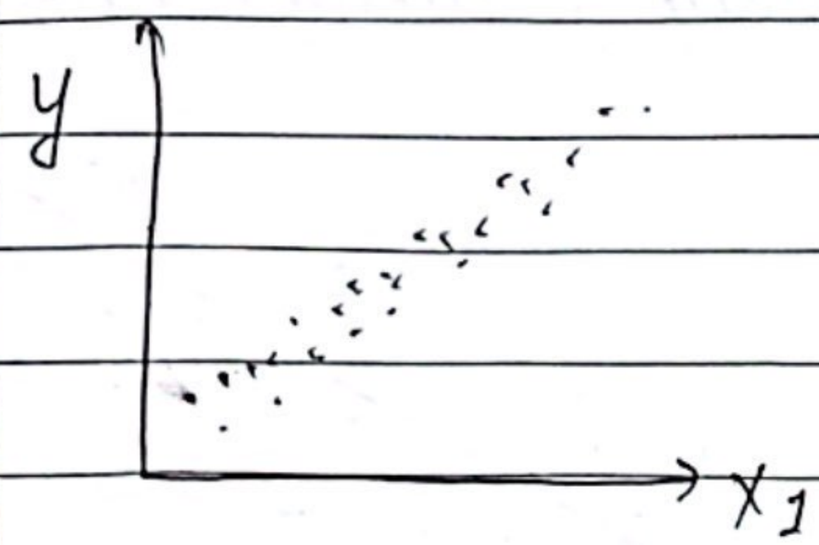
⇒ Multiple linear Regression is an extension of simple Linear Regression as it takes more than one predictor variable to predict the response variable.

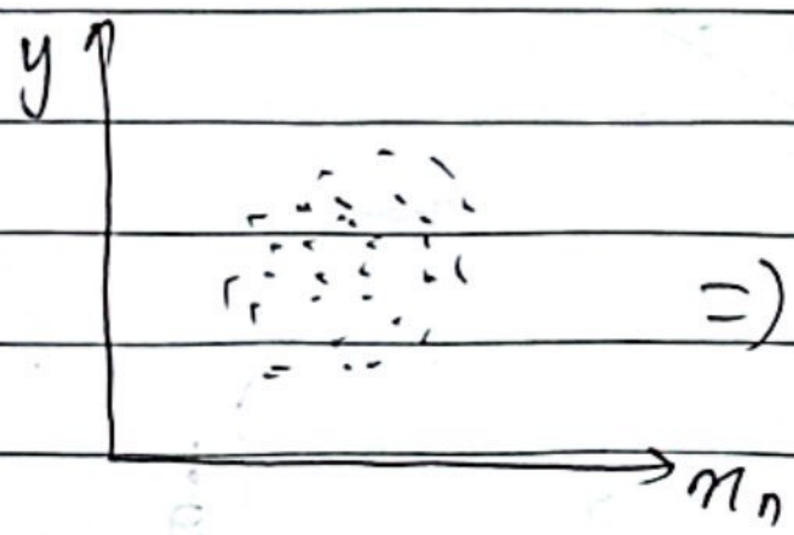
eg:- showing salary on inputs of age & experience.

Teacher's Signature.....

Date
Page

x_1	x_2	... x_n	y





=) we do feature selection for this type of data

$y = mx + c$ → for linear regression

$$\boxed{y = m_1x_1 + m_2x_2 + \cdots + m_nx_n + c}$$

→ for multiple linear regression.

Teacher's Signature.....

Date
Page

Here,

age	experience	Salary
50	13	42,717
36	19	99676
29	2	67952
42	2	82460
40	8	112847
44	5	58449
32	12	73536

Code:

```
[1]: import pandas as pd
[2]: dataset = pd.read_excel('multiple_linear regression.xlsx')
     dataset.head(5)
out[2]: *output*

[3]: dataset.shape
[3]: (60,3)

[4]: dataset.isnull().sum()
[4]: age          0
     experience   0
     salary       0
     dtype: int64
```

Teacher's Signature.....

Date
Page

[5]: import seaborn as sns
[6]: import matplotlib.pyplot as plt.

[7]: sns.pairplot (data = dataset)
plt.show()
out [7]: *graphs that show connection between columns*

[8]: sns.heatmap (data=dataset.corr(), annot=True)
plt.show ()
out [8]: *graph that show, is there linear relation or not*

[9]: x = dataset.iloc[:, :-1]
y = dataset["salary"]

[10]: from sklearn.model_selection import train_test_split

[11]: x_train, x_test, y_train, y_test =
train_test_split (x, y, test_split=0.2, random_state=42)

[12]: from sklearn.linear_model import LinearRegression

[13]: lr= LinearRegression ()

Teacher's Signature.....

Scanned with CamScanner

Date
Page

~~[14]: lr.f~~

[14]: lr.fit(x_train, y_train)

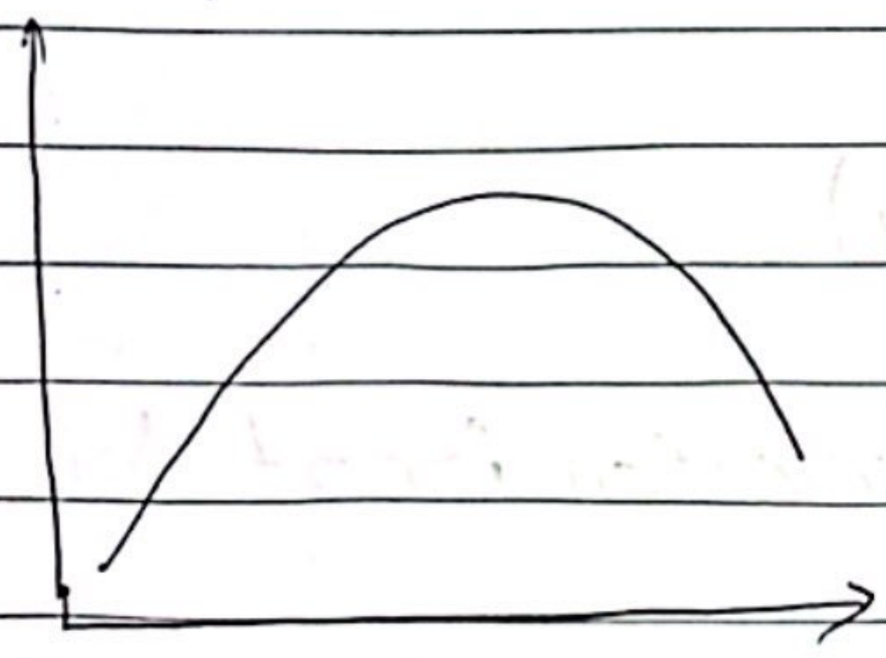
[15]: lr.score(x_test, y_test)*100
out[15]: 100.0

[16]: lr.predict(x_test)
out[16]: array([82,000., 68000., - - - - -])

Polynomial Regression:

⇒ Polynomial regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial.

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + b_2 x_1^3 + \text{-----} b_n x_1^n$$



Teacher's Signature.....

Scanned with CamScanner

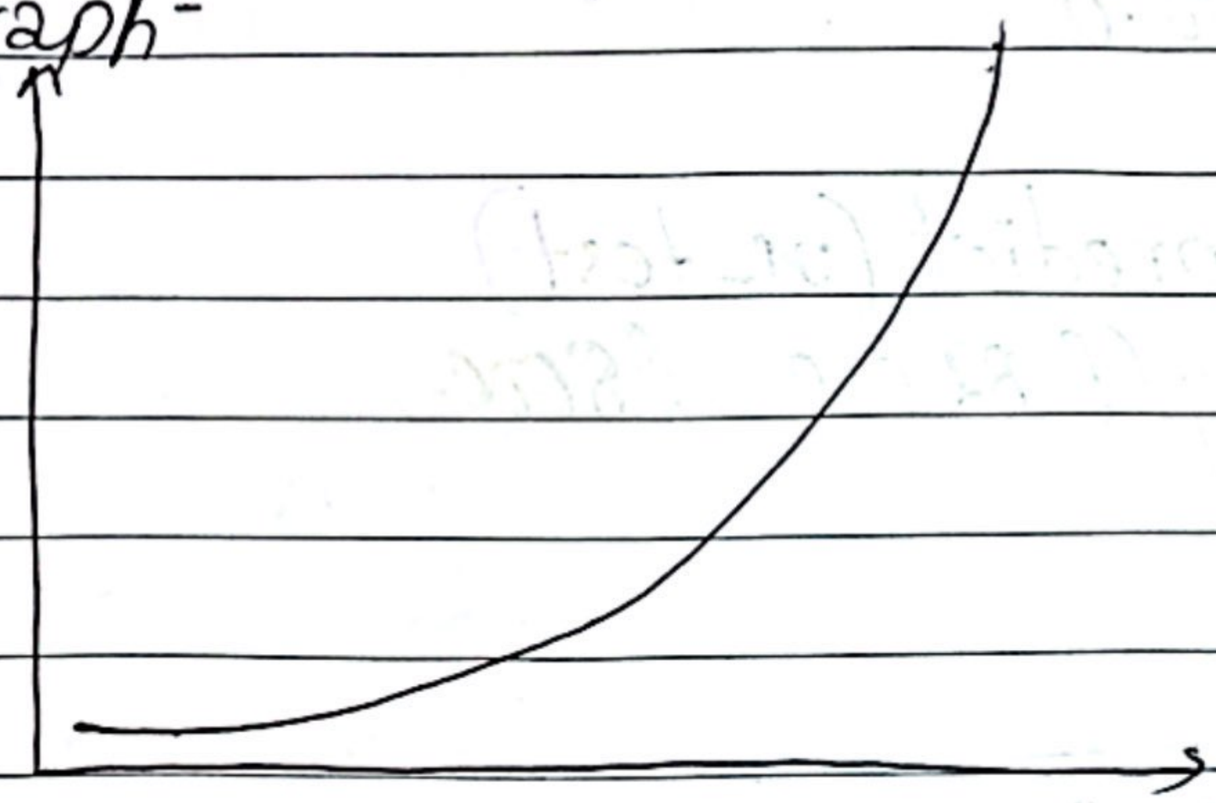
level
salary > are column name

Date
Page

code:

[1]: import pandas as pd.
[2]: dataset = pd.read_excel ("filename.xlsx")

[3] plt.scatter(dataset["level"], dataset["salary"])
plt.show()
out[3]: -graph-



[4]: x = dataset [["Level"]]
y = dataset ["salary"]

[5]: from sklearn.preprocessing import PolynomialFeatures

[6]: pf = PolynomialFeatures (degree = 2)
pf.fit (x)
pf.transform (x)

[7]: x = pf.transform (x)

[8]: from sklearn.model_selection import train_test_split

Date
Page

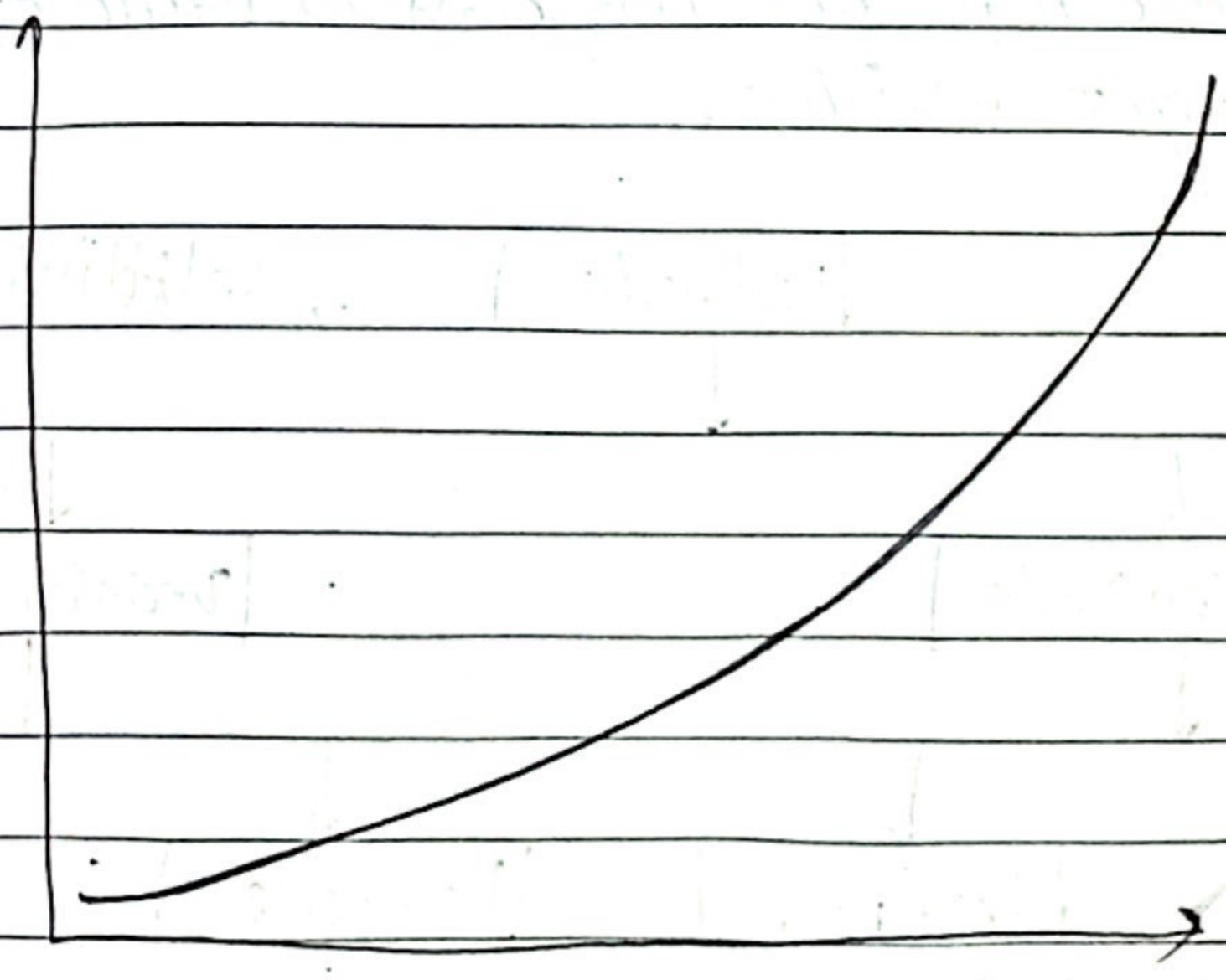
[9]: x_train x_test y_train y_test =
train_test_split (x, y, test_size=0.2, random_state=
42)

[10] from sklearn.linear_model import LinearRegression

[11] lr = LinearRegression ()
lr.fit(x_train, y_train)

[12] lr.score (x_test, y_test) * 100
[13] y_Prd = lr.predict(x)
[14] plt.scatterplot (dataset["level"], dataset["salary"])
plt.plot (dataset["level"]), y_prd, c="red")
plt.xlabel("level")
plt.ylabel("salary")
plt.show()

graph[14]



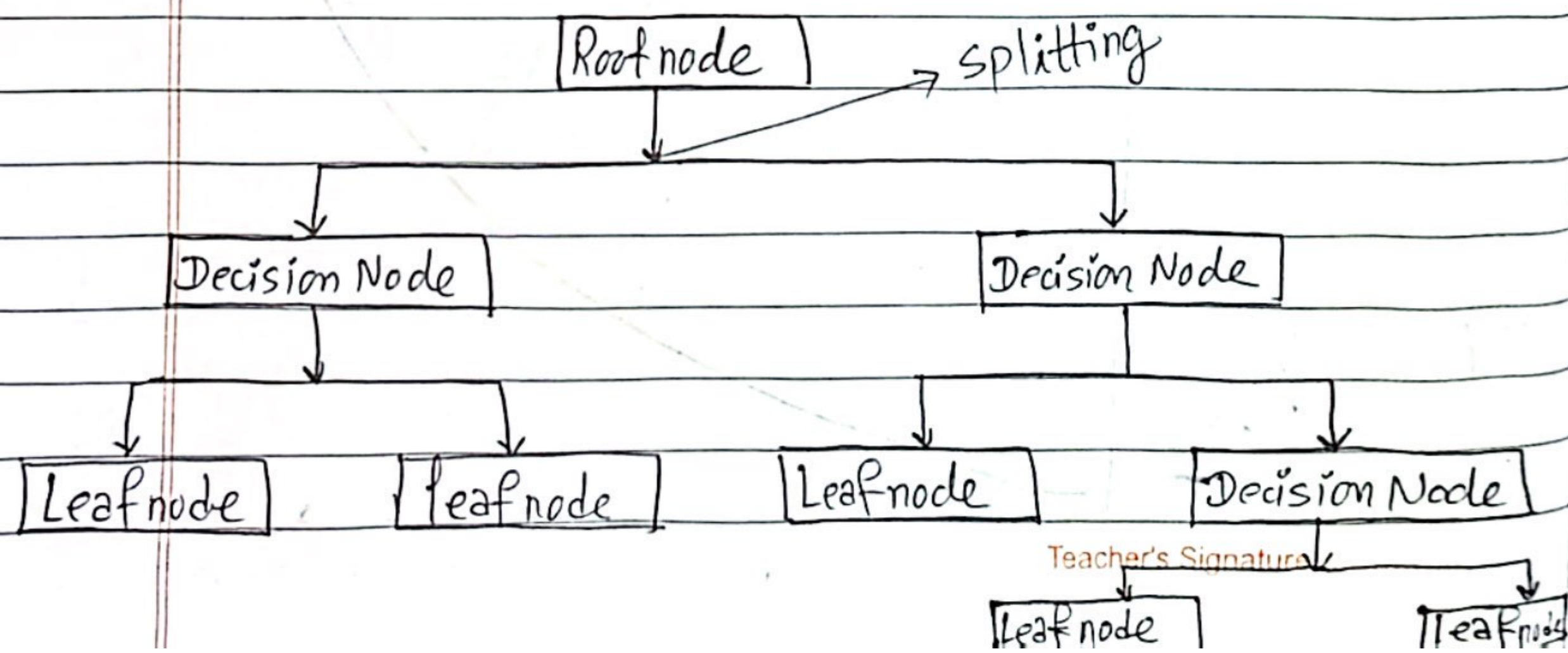
both lines overlaps

[15]: test = pf.transform([[45]])
~~test~~

[16]: lr.predict(test) # on level 45,
array([1057494.48]) salary is 1057494.48

Decision Tree

⇒ Decision Tree is a machine learning algorithm based on superwised learning, that can be used for both regression and classification problems.
The goal is to build up a classification model that can predict the class or value of the target variable.



Scanned with CamScanner

Date
Page

Decision Tree Real life examples:

1. In medical field.
2. In education sector.
3. In banking sector

code:

```
[1]: import pandas as pd
[2]: data = pd.read_csv(r'C:\users\OneDrive\Desktop\file.csv')
[3]: data
```

out [3]:

	outlook	humidity	windy	play
0	sunny	high	weak	no
1	sunny	high	strong ~~weak~~	no
2	overcast	high	weak	yes
3	rainy	high	weak	yes
4	rainy	normal	weak	yes
5	rainy	normal	strong	no
6	overcast	normal	weak	yes
7	sunny	high	weak	no
8	sunny	normal	weak	yes
9	rainy	normal	weak	yes
10	sunny	normal	strong	yes
11	overcast	high	strong	yes
12	overcast	normal	weak	yes
13	overcast	high	strong	no

Teacher's Signature.....

Scanned with CamScanner

Date
Page

[4]: from sklearn.preprocessing import LabelEncoder

[5]: le = LabelEncoder()

[6]: dataset['outlook'] = le.fit_transform(dataset['outlo
dataset['humidity'] = le.fit_transform(dataset['outlook'])
dataset['windy'] = le.fit_transform(dataset['windy'])
dataset['play'] = le.fit_transform(dataset['play])
dataset

out[6]:

	outlook	humidity	windy	play
0	2	0	1	0
1	2	0	0	0
2	0	0	1	1
3	1	0	1	1
4	1	1	1	1
5	1	1	0	0
6	0	1	1	1
7	2	0	1	0
8	2	1	1	1
9	1	1	1	1
10	2	1	0	1
11	0	0	0	1
12	0	1	1	1
13	0	0	0	0

Teacher's Signature.....

Scanned with CamScanner

Date
Page

[7]: features_cols = ['outlook', 'humidity', 'windy']
x = dataset[features_cols]
y = dataset.play

[8]: from sklearn.model_selection import train_test_split.

[9]: x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2,

[10]: from sklearn.tree import DecisionTreeClassifier

[11]: classifier = DecisionTreeClassifier(criterion='gini')

~~[11]: classifier~~

[12]: classifier.fit(x_train, y_train)

[13]: classifier.predict(x_test)
out [13]: array([1,0.0])

Teacher's Signature.....

Scanned with CamScanner

Date
Page

Random Forest Algorithm

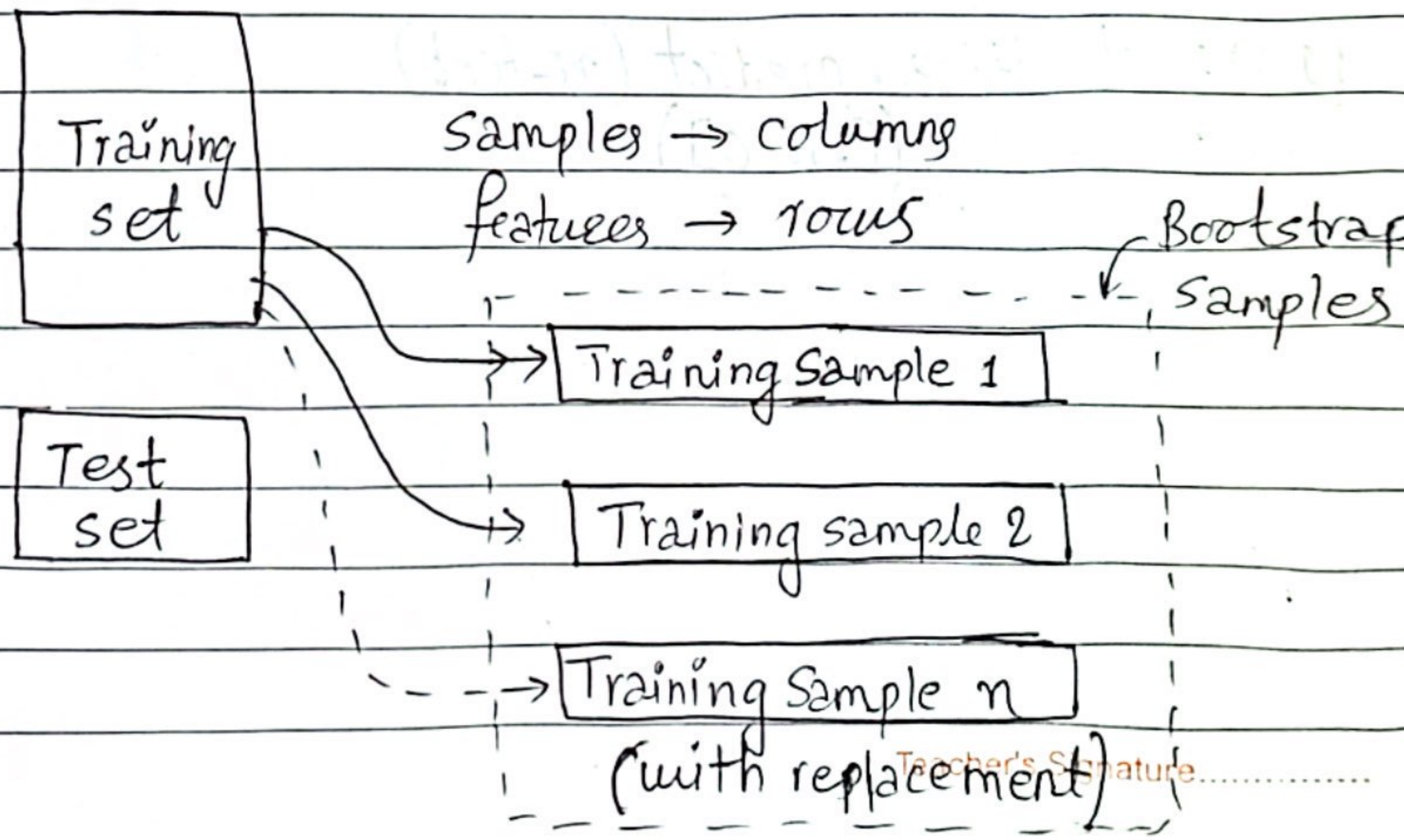
⇒ Random Forest is a machine learning algorithm based on supervised learning, that can be used for both regression & classification problems.

→ It is a collection of multiple random decision trees, which is called the forest.

eg:- For buying a house, we take decisions of different persons (eg; agent, friends, relatives). Here, each person is a decision tree and group of these people is forest or random forest.

*How does the algorithm works?

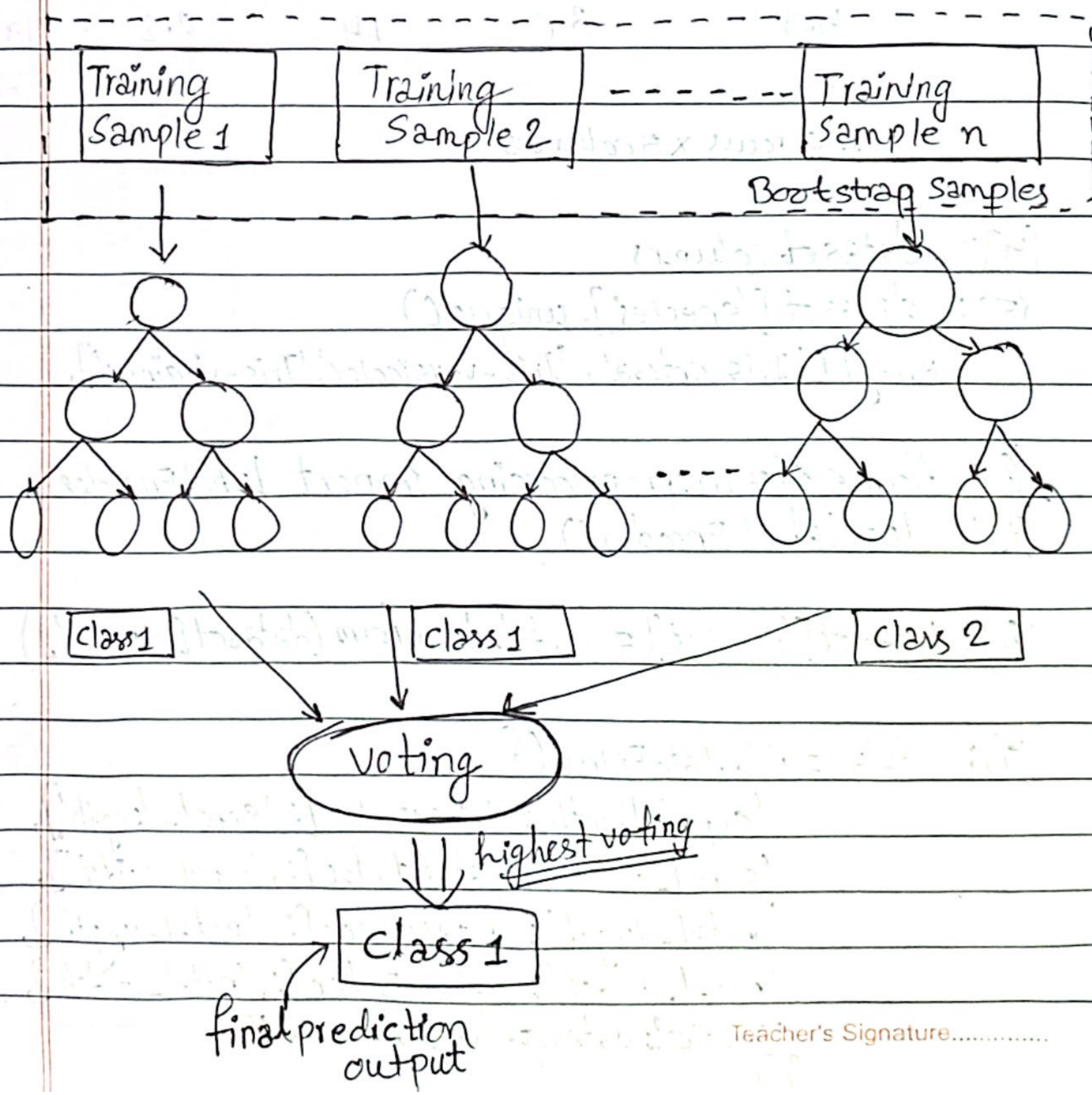
⇒



Teacher's Signature

with replacement means, 'any few records can present in TrainingSample1 and TrainingSample2'.
Especially,
Any record can present in more than one Training Sample.

Bootstrap samples is important to make different decision trees.



Date _____
Page _____

My Code:-

[1]: import pandas as pd

[2]: from sklearn import datasets.

[3]: dataset = pd.read_csv("IRIS.csv")
dataset

Out[3]:

Sepal.length	Sepal.width	petal.length	petal.width	species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
. . .	. . .	. . .	. . .	. . .

150 rows x 5 columns

[4]: dataset.columns

[5]: dataset['species'].unique()
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],

[6]: from sklearn.preprocessing import LabelEncoder.

[7]: le = LabelEncoder()

[8]: dataset['species'] = le.fit_transform(dataset['species'])

[9]: data = pd.DataFrame({
'sepal_length': dataset.loc[:, 'sepal_length'],
'sepal_width': dataset.loc[:, 'sepal_width'],
'petal_length': dataset.loc[:, 'petal_length'],
'petal_width': dataset.loc[:, 'petal_width'],
'species': dataset.loc[:, 'species']
})

Teacher's Signature.....

Scanned with CamScanner

Date
Page

[10]: x = data['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
y = data['species'].

[11]: x

out [11]:

sepal_length	sepal_width	petal_length	petal_width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
- - - -	- -	- - - -	

150 rows x 4 columns

[12]: y

out [12]:

0
0
1
1
2
3

Name: Species, Length: 150, dtype: int64

[13]: from sklearn.model_selection import train_test_split

[14]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =0.3)

[15]: len(x_train)
=105

[16] len(x_test)
=45

Teacher's Signature.....

Scanned with CamScanner

Date
Page

[17] from sklearn.ensemble import RandomForestClassifier

[18] clf = RandomForest Classifier (n_estimators= 100,
criterion='gini')

[19]: clf. fit (x_train, y_train)

[20] clf. predict(x_test)

[21]: clf. score (x_test, y_test) *100

out [21]: 95.555

[22]: clf. predict([[3,5,4,2]])

out [22]: array([1])

Teacher's Signature.....

Scanned with CamScanner

Regularization :

→ This is a form of regression, that constraints or regularizes or shrinks the coefficient estimates towards zero.

→ This technique discourage learning a more complex or flexible model, so as to avoid the risk of overfitting.

→ Regularization can achieve this motive with 2 techniques :

- Ridge Regularization / L2
- Lasso Regularization / L1

1) Lasso Regularization (L1) :

→ This is a regularization technique used in feature selection using a shrinkage method also refereed to as the penalized regression method.

→ Lasso Regression magnitude of coefficients can be exactly zero.

Date
Page

$$\text{cost function} = \text{Loss} + \lambda \Sigma \|w\|$$

- loss = sum of squared residual
- λ = penalty
- w = slope of the curve

Teacher's Signature.....

Date
Page

Logistic Regression:

→ Logistic regression is one of the most popular Machine Learning algorithms, which comes under the superwised Learning technique

→ It is used for predicting the categorical dependent variable using a given set of independent variables.

→ Therefore, the outcome must be a categorical or discrete value. It can be either yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 & 1 it gives the probabilistic values which lie between 0 and 1.

* Types:

a) Binomial:

→ In binomial logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail etc.

b) Multinomial:

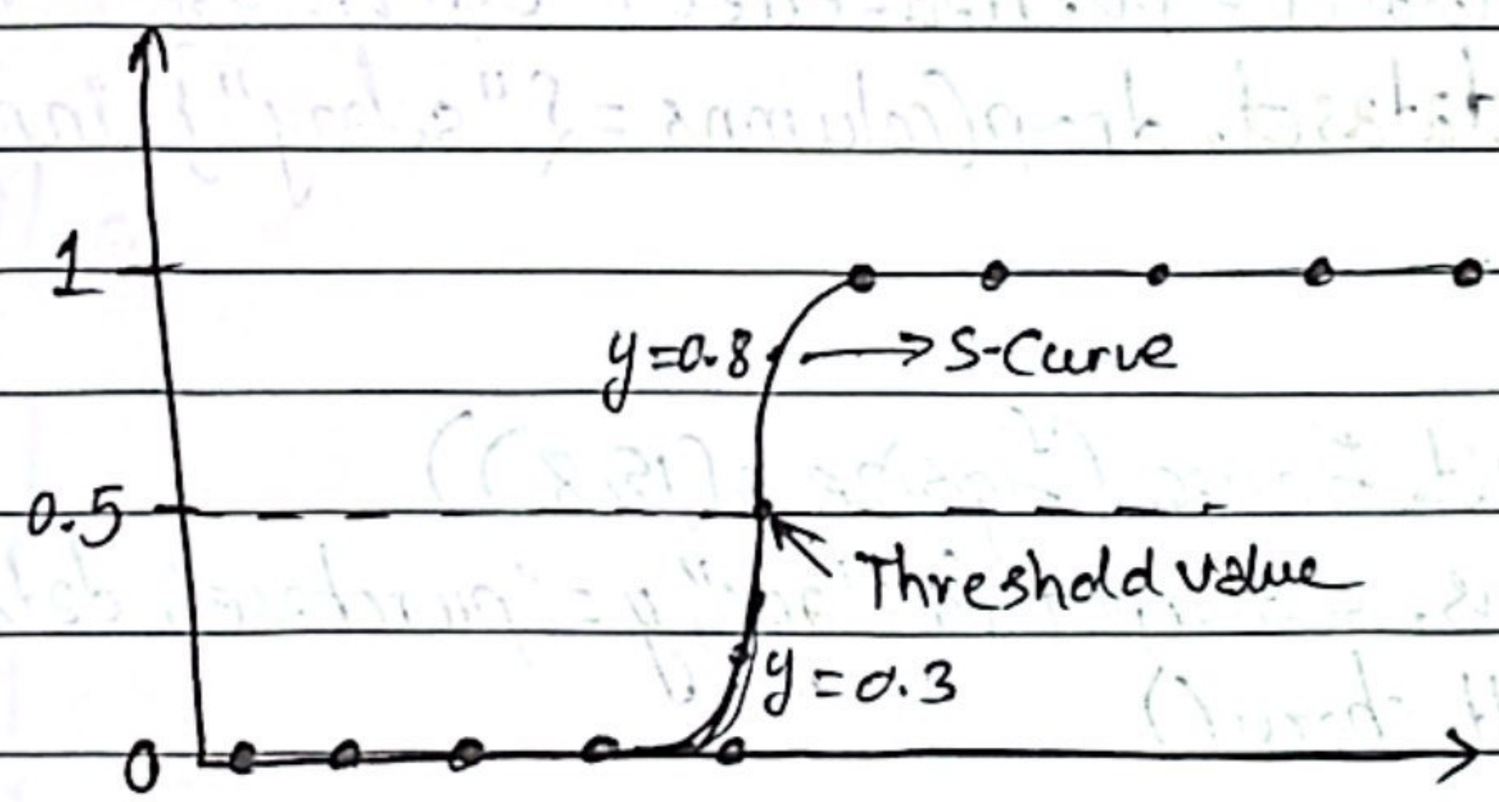
→ In multinomial logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", "sheep".

Teacher's Signature.....

Scanned with CamScanner

c) ordinal:

→ In ordinal logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "medium" or "high".



$$\boxed{y = \frac{1}{1+e^{-x}}}$$

$$x = m_1 x_1 + m_2 x_2 + \dots + b$$

Teacher's Signature.....

Date
Page

* Logistic binomial:

[1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

[2]: dataset = pd.read_excel ("c:/ml course /Dataset / logistic.x
dataset.drop(columns ={"salary"}, inplace=True)

[3]: plt.figure (figsize =(15, 8))
sns.scatterplot (x="age", y="purchase", data=dataset)
plt.show()

[4]: x = dataset [["age"]]
y = dataset ["purchase"]

[5]: from sklearn.model_selection import train_test_split

[6]: x_train, x_test, y_train, y_test = train_test_split (x, y,
test_size=0.2, random_state
=42)

[7]: from sklearn.linear_model import LogisticRegression

[8]: lr = LogisticRegression ()
lr.fit (x_train, y_train)

Teacher's Signature.....

Scanned with CamScanner

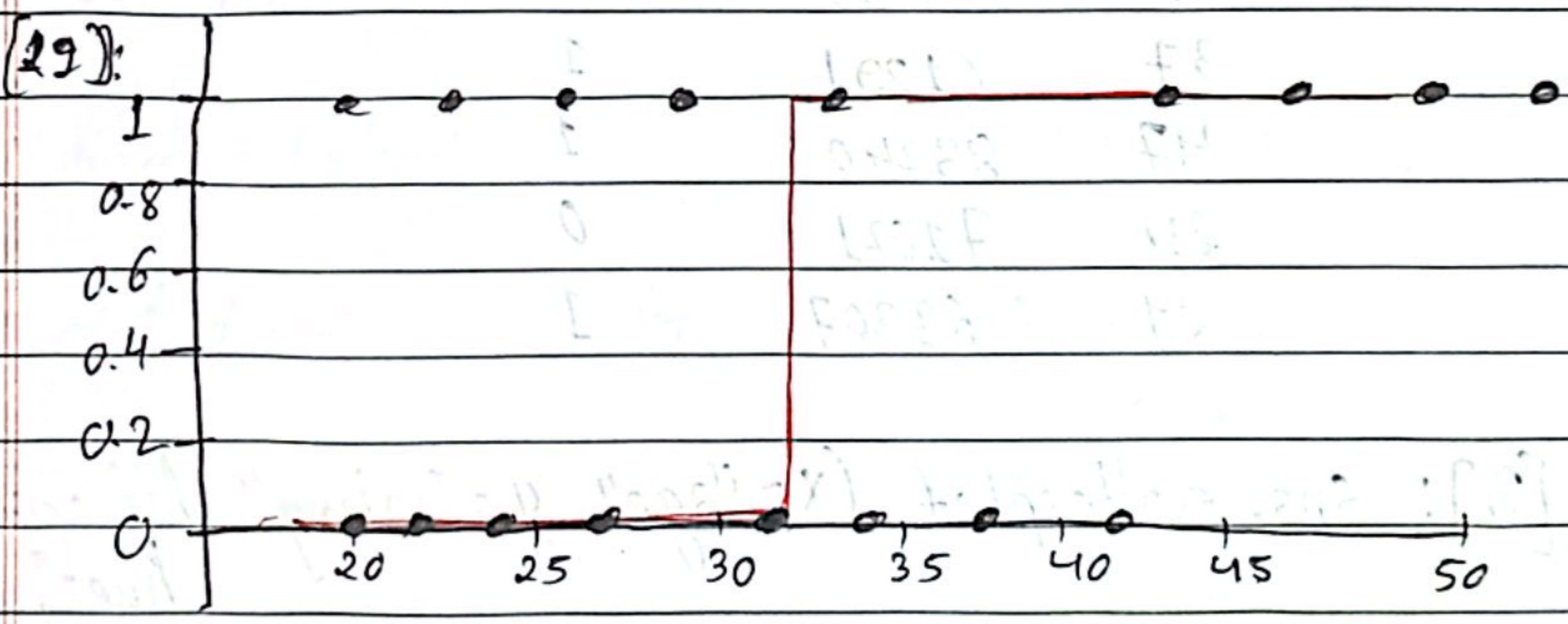
Date.
Page.

[9]: lr.score (x_train, y_train) * 100

[10]: lr.predict ([[30]])

[11]: sns.scatterplot (x="age", y="purchase", data=dataset)
sns.lineplot (x="age", y=lr.predict(x), data=dataset, color='red')
plt.show()

out[11]:



Teacher's Signature.....

Date
Page

* Logistic Binomial- multiple inputs:

[1]:
```
import pandas as pd
import seaborn as sns.
import matplotlib.pyplot as plt.
```

[2]:
```
dataset = pd. read_excel ("c:/ML Cource/Datasets/logistic.
dataset. head (5)
```

out[2]:

age	salary	purchase
20	107580	1
37	61291	1
47	83240	1
24	71021	0
21	63307	1

[3]:
```
sns. scatterplot (x="age", y="salary", data=dataset,
                  hue="purchase")
plt. show()
```

out[3]: *Graph*

[4]:
```
x = dataset. iloc[:, :-1]
y = dataset ["purchase"]
```

[5]:
```
from sklearn. model_selection import train_test_
                                     split
```

[6]:
```
x_train, x_test, y_train, y_test = train_test_split (x, y,
                  test_size = 0.3, random state= 50)
```

Teacher's Signature.....

Scanned with CamScanner

Date
Page

[7]: from sklearn.linear_model import LogisticRegression

[8]: lr = LogisticRegression()

[9]: lr.fit(x_train, y_train)

[10]: lr.score(x_test, y_test) * 100

[11] lr.predict([[20, 107580]])

[12]: from mlxtend.plotting import plot_decision_regions

[13] plot_decision_regions(x.to_numpy(), y.to_numpy(),
clf=lr)
plt.show()

out[13]: *Graph*

Teacher's Signature.....

Scanned with CamScanner

Date
Page

* Logistic Regression (Polynomial Input)

→ This model is used if data is not linear.
→ First Polynomial features is used on inputs then only model training is done.

[1]:
```
import pandas as pd.
import seaborn as sns
import matplotlib.pyplot as plt.
```

[2]:
```
dataset = pd.read_excel("C:/ML course/Datasets/logistic.xls
dataset.head(5)
```

out [2]:

age	salary	purchase
20	107580	1
37	61291	1
45	83240	0
24	71021	0
21	63307	1

[3]:
```
sns.scatterplot (x="age", y="salary", data=dataset,
                 hue="purchase")
plt.show()
```

Teacher's Signature.....

Scanned with CamScanner

Date
Page

[4]: x = dataset.iloc[:, :-1]
y = dataset["purchase"]

[5]: from sklearn.preprocessing import polynomialFeatures

[6]: pf = PolynomialFeatures(degree=3)
pf.fit(x)
x = pd.DataFrame(pf.transform(x))

[7]: from sklearn.model_selection import train_test_split

[8]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=46)

[9]: from sklearn.linear_model import LogisticRegression

[10]: lr = LogisticRegression()

[11]: lr.fit(x_train, y_train)

[12]: lr.score(x_train, y_train)
0.6

Teacher's Signature.....

Scanned with CamScanner

Confusion Matrix :

Date _____
Page _____

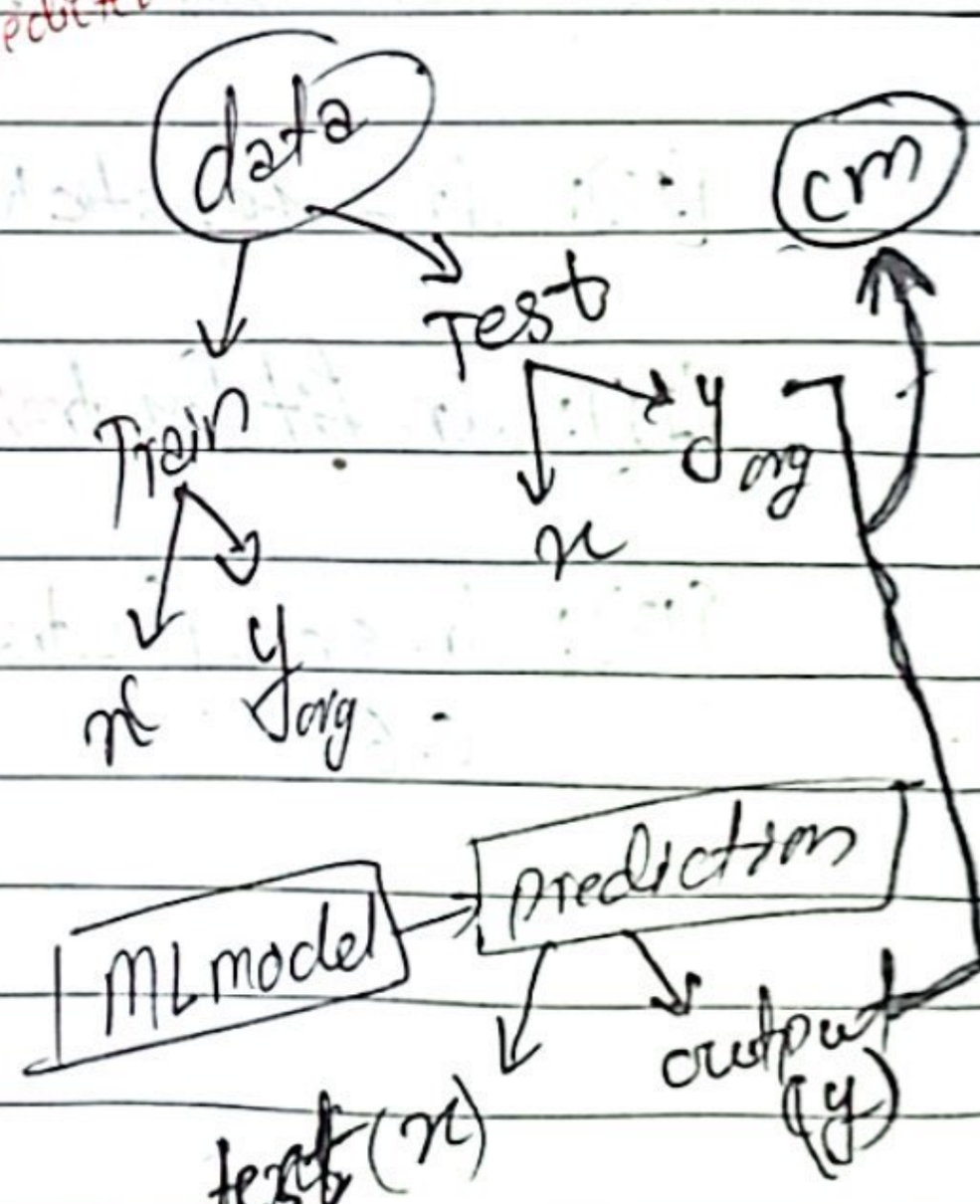
→ A confusion matrix is a simple & useful tool for understanding the performance of a classification model, like one used in machine learning or statistics.

→ It helps you evaluate how well your model is doing in categorizing things correctly.

→ It is also known as the error matrix.

→ The matrix consists of predictions result in a summarized form, which has a number of correct predictions and incorrect predictions.

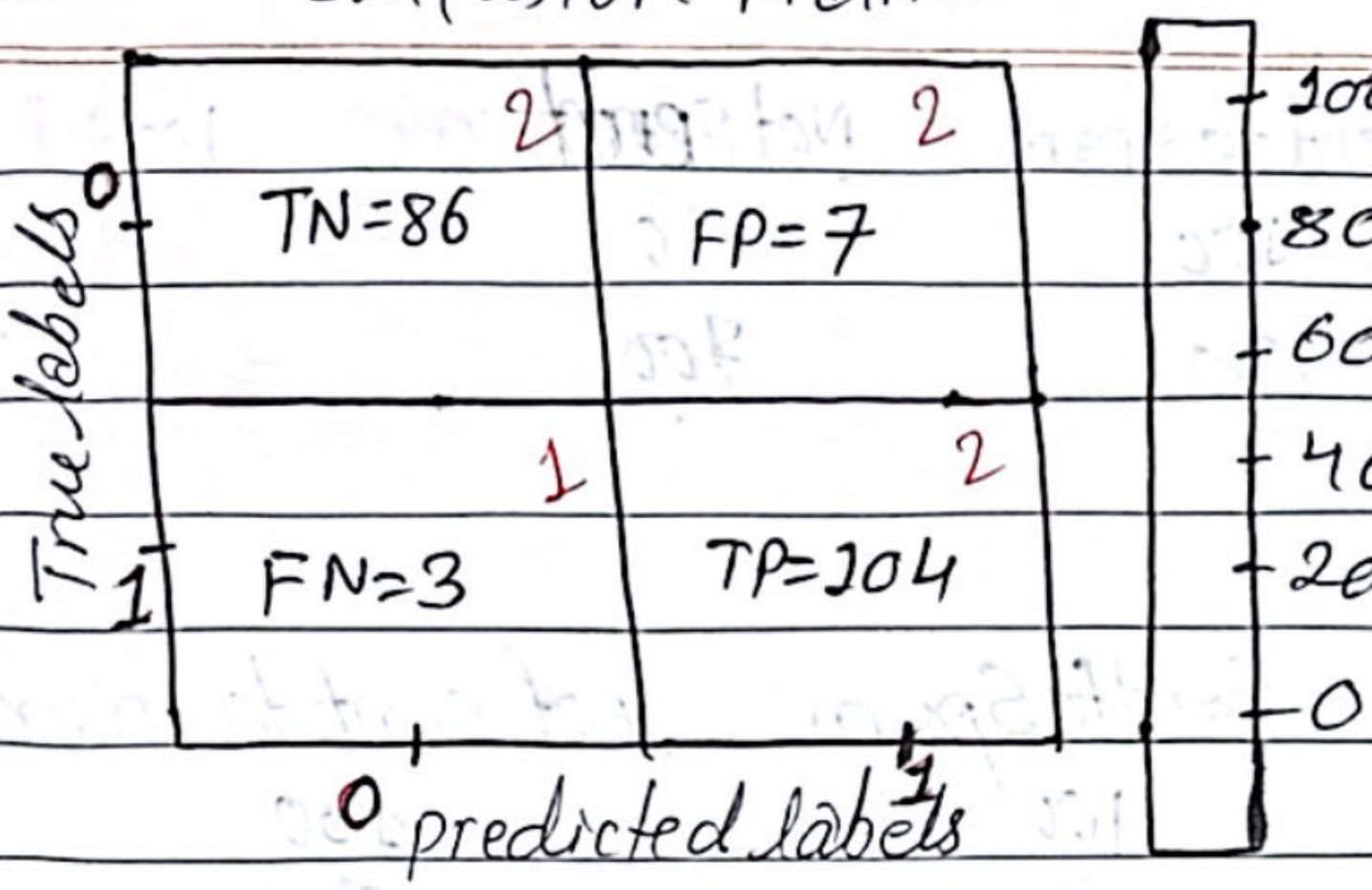
Sr.No.	Actual → y_{org}	Predicted → $y_{predicted}$
01	1	1
2	0	0
3	1	0
4	0	1
5	0	0
6	0	1
7	1	1



Teacher's Signature.....

Date
Page

confusion matrix



TN → True Negative

Dangerous

FN → False Negative → Type-II error

FP → False Positive → Type-I error

TP → True positive

$$\boxed{\text{Accuracy} = \frac{TN + TP}{TN + TP + FP + FN}}$$

$$\boxed{\text{Error} = \frac{FN + FP}{TN + TP + FP + FN}}$$

Teacher's Signature.....

Scanned with CamScanner

Date
Page

(A)

	Sent to spam	Not sent to spam
Spam	100	170
Not Spam	30	700

$\Rightarrow \frac{100+700}{100+170+30+700} = 80\%$

(B)

	Sent to Spam	not sent to spam
spam	100	190
Not spam	10	700

$\Rightarrow \frac{100+700}{100+10+190+700} = 80\%$

$\Rightarrow$ Here, (B) is selected due to circled value.

* **Precision:**

→ $\frac{TP}{TP+FP}$; It should be higher to reduce false Positive value

* **Recall:**

→ It should be higher to reduce False Negative value

→ $\frac{TP}{TP+FN}$

Teacher's Signature.....

Scanned with CamScanner

Date
Page

→ It helps us to measure how many positive samples were correctly classified by the ML model.

(A)

	Detected cancer	Not Detected	
Has cancer	1000	200	= 90%
No Cancer	800	8000	

(B)

	Detected Cancer	Not Detected	
Has Cancer	1000	500	= 90%
No Cancer	500	8000	

Model (A) is selected, due to lower value of Falsepositive value.

Scanned with CamScanner

Date
Page

→ If we want to reduce both FN & FP, we should make high F1 Score.

* F1 Score:

→ It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Example of confusion matrix:

```
[1]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt.

[2]: dataset = pd.read_excel("C:/ML Course/Datasets/
                                logistic.xlsx")
     dataset.head(5)

[3]: x = dataset.iloc[:, :-1]
     y = dataset["purchase"]
```

Teacher's Signature.....

Date
Page

[4]: from sklearn.model_selection import train_test_split

[5]: x_train, x_test, y_train, y_test = train_test_split (x, y,
test_size = 0.2, random_state = 41)

[6]: from sklearn.linear_model import LogisticRegression

[7]: lr = LogisticRegression ()
lr.fit (x_train, y_train)

[8]: lr.Score (x_train, y_train)

[9]: from sklearn.metrices import confusion_matrix,
precison_score, recall_score,
f1_score

[10]: cm = confusion_matrix (y_test, lr.predict(x_test))
cm

[11]: sns.heatmap (cm, annot = True)
plt.show()

[12]: precision_score (y_test, lr.predict (x_test)) * 100

[13]: recall_score (y_test, lr.predict (x_test)) * 100
[14]: f1score (y_test, lr.predict (x_test)) * 100

Teacher's Signature.....

Scanned with CamScanner

Imbalanced Dataset:

→ If training data has more number of value of one kind and less number of value of another, then model will be bias to more number data

age	salary	purchase
47	40,000	0
72	1,07,000	0
30	79000	1
39	85000	1
- -	- -	- -

⇒ dataset["purchase"].value_counts()

→ 1 58
0 42

⇒ No. of 1 is more, so model will be bias to
→ In this case, even if output should be 0, the model can 1.

→ To resample the data, we use 'imblearn' module as follow.

Teacher's Signature.....

Date
Page

Example:

[1]: import pandas as pd

[2]: dataset = pd.read_excel("logistic.xlsx")
dataset.head(5)

out[2]:

age	salary	purchase
47	40000	0
72	50000	0
30	79000	1
39	130000	1
22	107000	0

[3] dataset["purchase"].value_counts()

out [3]

purchase	
1	58
0	42

[4]: x = dataset.iloc[:, :-1]
y = dataset["purchase"]
# Random Under sampling

[5] from imblearn.under_sampling import RandomUnderSampler

[6] ru = RandomUnderSampler()
ru_x, ru_y = ru.fit_resample(x, y)

Teacher's Signature.....

Date
Page

[7] ru_y.value_counts()

out[7]: purchase
0 42
1 42

→ Here both 0 & 1 are 42, so model is not bias to one data.
→ This ru_x & ru_y is used to train & test model.

Random over sampling:

[8] from imblearn.over_sampling import RandomOverSampler

[9]: ro = RandomOverSampler()
ro_x, ro_y = ro.fit_resampler()

[10] ro_y.value_counts()
out[10] purchase
1 58
0 58

Teacher's Signature.....

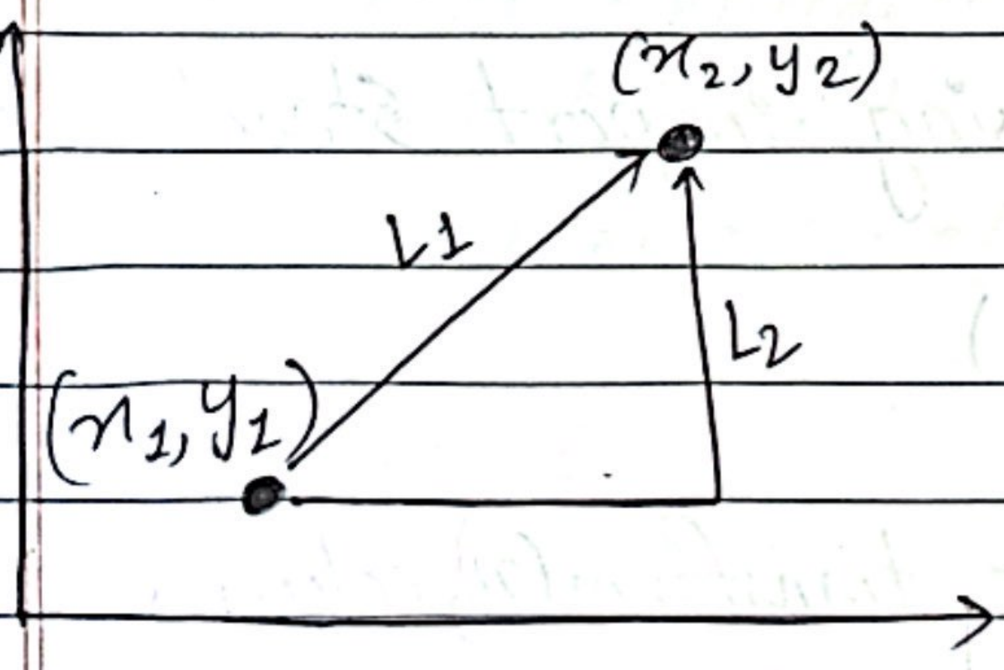
K-Nearest Neighbours (classification):

⇒ K-NN algorithm can be used for Regression as well as classification but mostly it is used for the classification problem.

→ KNN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

→ It is also called a lazy learner algorithm

→ It is based on the idea of finding the closest training samples (neighbours) to the input data point & making predictions based on these neighbours.



Man hattan distance (L_1)
Euclidean distance (L_2)

$$L_1 = |x_2 - x_1| + |y_2 - y_1|$$

→ Which distance to choose, it depends on project & database.

$$L_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Teacher's Signature.....

Date
Page

* practical:

```
import pandas as pd
import matplotlib.pylot as plt
import seaborn as sns

dataset = pd.read_excel("location/logistic.xlsx")

dataset.isnull().sum()
age      0
salary   0
purchase 0

x = dataset.iloc[:, :-1]
y = dataset["purchase"]

sns.scatterplot(x="age", y="salary", data=dataset, hue=
plt.show()                                  "purchase")
```

Graph

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
sc.fit(x)
x = pd.DataFrame(sc.transform(x), columns=x.columns)
```

Teacher's Signature.....

Date
Page

```
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=42)

from sklearn.neighbours import KNeighborsClassifier

Knc = KNeighboursClassifier(n_neighbors = 6)
Knc.fit(x_train, y_train)

Knc.score(x_test, y_test) * 100
90.0

Knc.score(x_train, y_train) * 100
90.0

for i in range (1, 30):
    Knc1 = KNeighborsClassifier(n_neighbors = i)
    Knc1.fit (x_train, y_train)
    print(i, Knc1.score(x_test, y_test), Knc1.score(x_train, y_tra
1    100.0    80.0
2    75.0    88.75
3    90.0    90.0
4.   90.0    91.25
5.   91.25    90.0
6.   90.0    90.0
```

Teacher's Signature.....

Date
Page

```
knc.predict([[0.414975, -0.782114]])
```

array([0])

~~plot_d~~

```
from mlxtend.plotting import plot_decision_regions

plot_decision_regions(x.to_numpy(), y.to_numpy(),
                      clf=knc)

plt.show()
```

* Graph *

Date
Page

K-Nearest Neighbors (Regression):

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns.

dataset = pd.read_csv("filelocation/generateddata.csv")

dataset.head(5)
```

Age	Experience	Salary
46	30	155827
38	30	177578
41	10	196323
61	15	195848
44	21	136037
- - -	- - -	- - -

```
sns.scatterplot(x="Age", y="Experience", data=dataset,
                hue="Salary")
plt.show()
```

Graph

```
x = dataset.iloc[:, :-1]
y = dataset["Salary"]
```

Teacher's Signature.....

Date
Page

```
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=42)

from sklearn.neighbors import KNeighboursRegressor

Knr = KNeighbors Regressor (n_neighbors = 6)
Knr.fit (x_train, y_train)

Knr.score (x_test, y_test) * 100
70.826

Knr.score (x_train, y_train) * 100
70.395
```

→ To increase the accuracy, we can change 'random_state' value, 'n_neighbors' value, and remove outliers, and we can also do scaling.

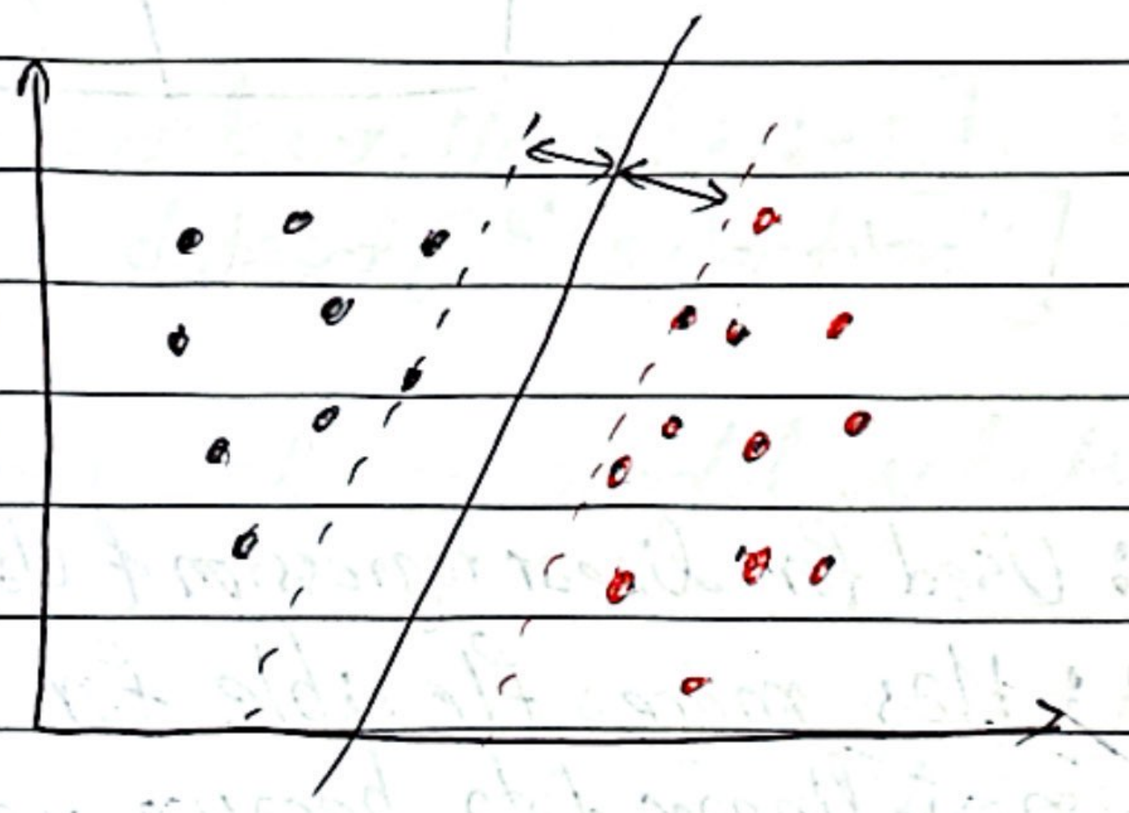
Teacher's Signature.....

Support Vector Machine (SVM):

→ SVM is one of the most popular Superwised Learning algorithms, which is used for Classification as well as Regression Problems.

x_1	x_2	y
–	–	1
–	–	0
–	–	1
–	–	0

0 - blue
1 - red



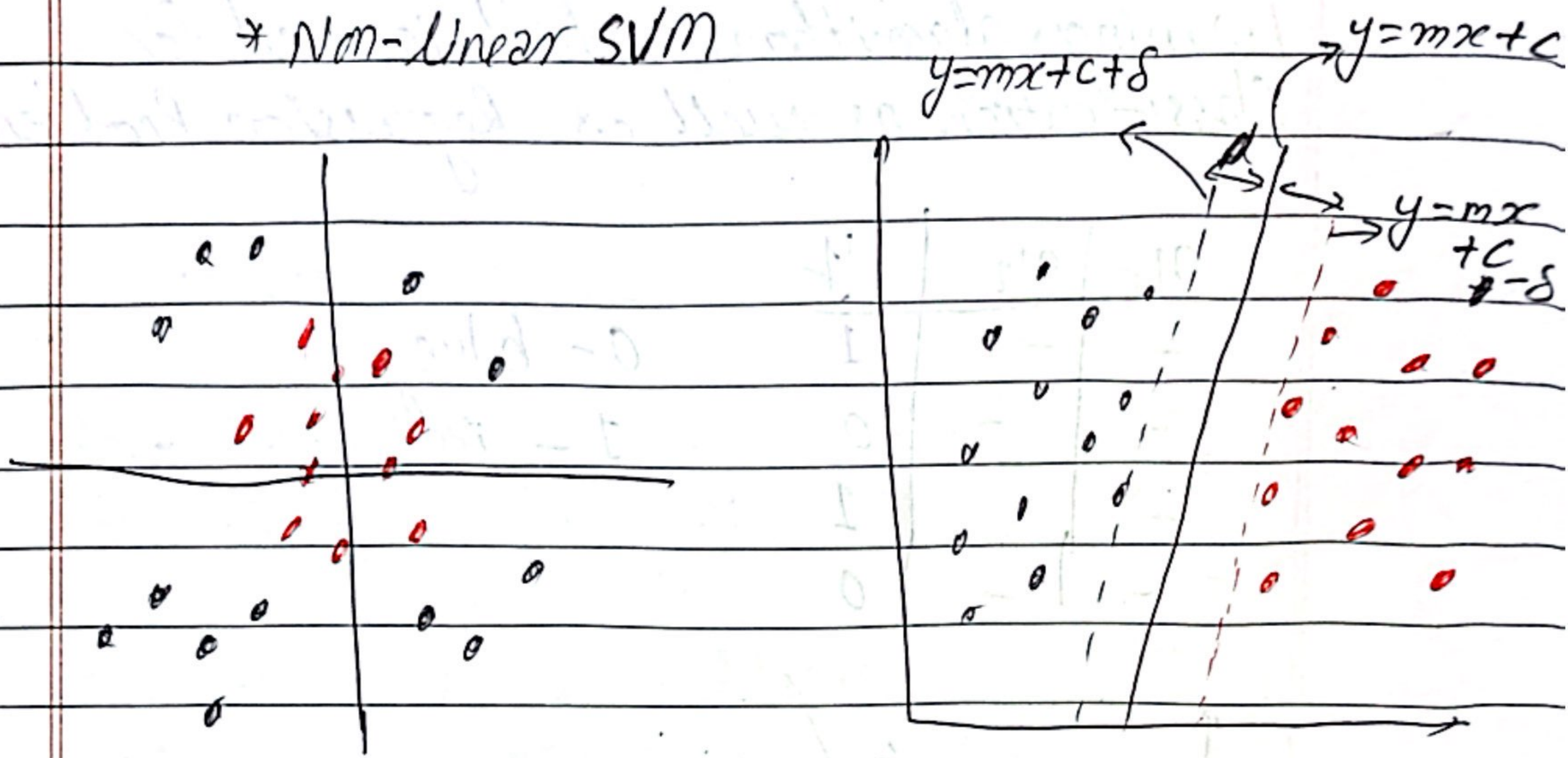
→ Hard Margin: The algorithm aims to find a hyperplane that perfectly separates the data into two classes without any misclassification.

→ Soft Margin: The algorithm allows for some misclassification to find a hyperplane that generalizes better to unseen data and is more robust to outliers.

Teacher's Signature.....

Date
Page

→ Types!
* Linear SVM
* Non-Linear SVM



→ Types:-
* Simple SVM : Used for linear regression & classification pattern
* Kernel SVM : Has more flexible for non-linear data because you can add more features to fit a hyperplane instead of a two-dimensional space

Linear : $K(w,b) = w^T x + b$
Polynomial : $K(w,x) = (\gamma w^T x + b)^N$
Gaussian RBF : $K(w,x) = \exp(-\gamma \|x_i - x_j\|^n$
Sigmoid : $K(x_i, x_j) = \tanh(\alpha x_i^T x_j +$

Teacher's Signature.....

Date
Page

* practical (classification):

```
import pandas as pd.
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv("placement_data".csv")

sns.scatterplot(x="cgpa", y="score", data = dataset,
                                        hue = "selection")
plt.show()

x = dataset.iloc[:, :-1]
y = dataset["selection"]

from sklearn.model_selection import train_test_split

x_train, x_text, y_train, y_test = train_test_split (x, y,
                                        test_size =0.2, random_state = 42)

from sklearn.svm import SVC

sv = SVC (Kernel = "poly")
sv.fit (x_train , y_train)

sv.score (x_test, y_test) * 100
100.0
```

Teacher's Signature.....

Date
Page

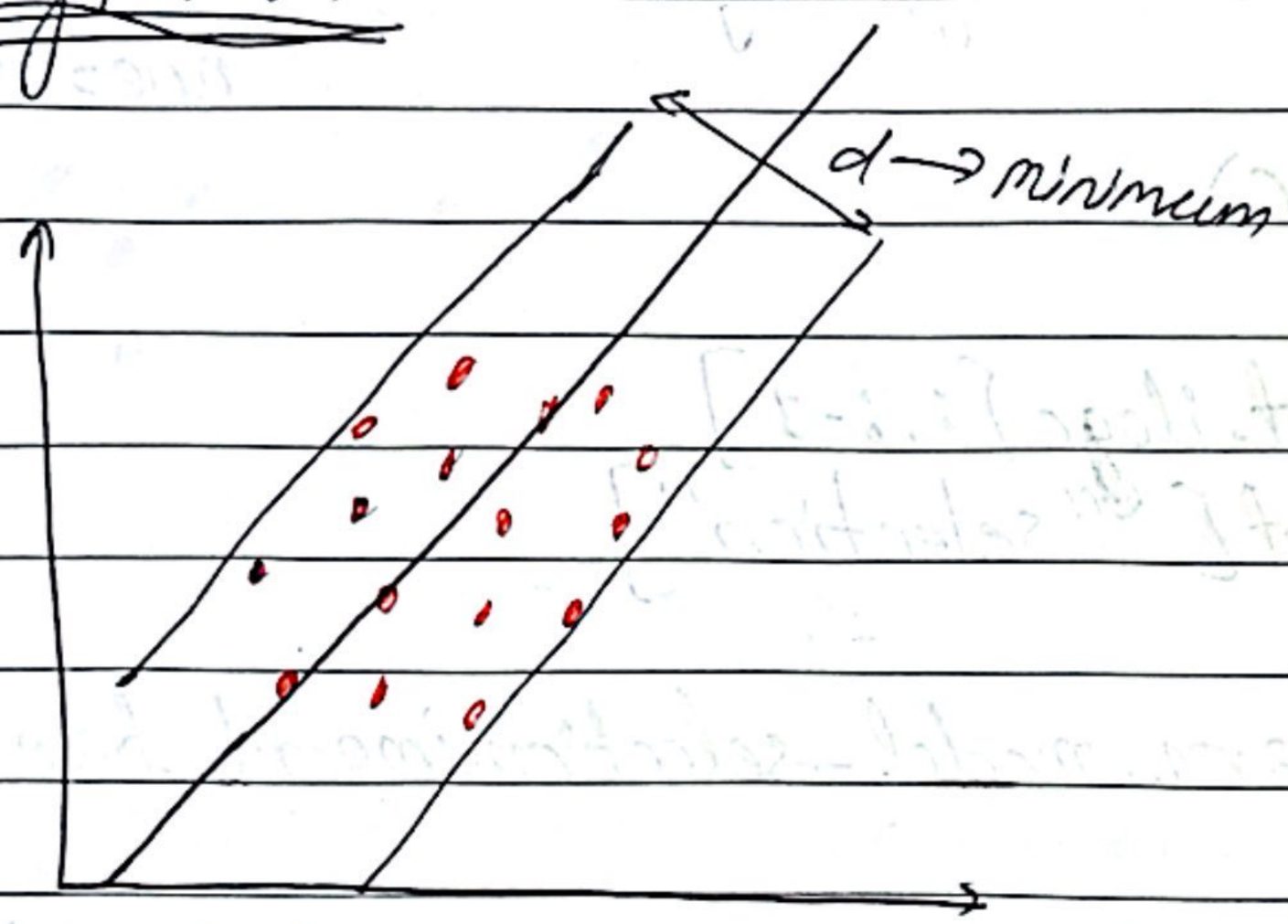
from mlxtend.plotting import plot_decision_regions

plot_decision_regions(x.to_numpy(), y.to_numpy(), clf=svc)
plt.show()

* Graph *

* Regression :

⇒

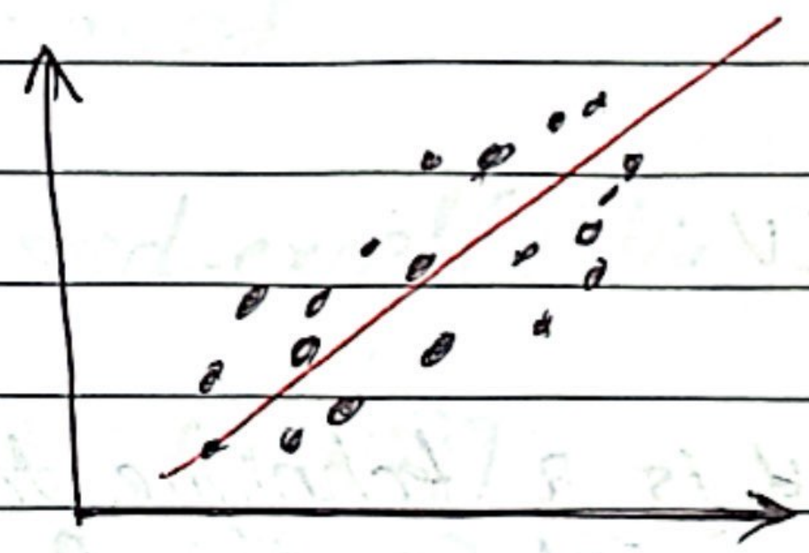


→ from sklearn.svm import SVR
svr = SVR() / svr = SVR(kernel="linear")

Hyper Parameter :-

(*) Model Parameters:

→ Model Parameters are configuration variables that are internal to the model, and a model learns them on it's own.



$$y = mx + c$$

(*) Hyper Parameter (mode):

→ Hyper parameters are those parameters that are explicitly defined by the user to control the learning process.
The best value can be determine either by the rule of thumb or by trail and error.

Date
Page

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameters tuning are:

- GridSearchCV
- RandomizedSearchCV

(*) GridSearch CV:

→ GridSearchCV is a technique to search through the best parameter values from the given set of the grid of parameters.

dt(A:a B:2)

	1	2	3
a	62%	72%	45%
b	55%	90%	85%

(*) ~~Randomized~~

(*) Randomized Search CV:

→ It goes through only a fixed number of

Teacher's Signature.....

Date
Page

hyper parameter settings.

→ It moves within the grid in a random fashion to find the best set of hyperparameters.

(*) practically in jupyter notbook:

```
import pandas as pd

dataset = pd.read_excel(r"location\level_salary.xlsx")

x = dataset[["level"]]
y = dataset["Salary"]

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
                                =42)

from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor()

dt.fit(x_train, y_train)
```

Teacher's Signature.....

Date
Page

```
dt.score(x_test, y_test
0.9995
dt.score(x_train, y_train)
1.0

from sklearn.model_selection import GridSearchCV,
                                    RandomSearchCV

df={"criterion": ["squared_error", "friedman_mse",
                  "absolute_error", "poisson"],
    "splitter": ["best", "random"],
    "max_depth": [i for i in range(2,20)]}
```

GridSearchCV

```
gd = GridSearchCV(DecisionTreeRegressor(),
                  param_grid =df)

gd.fit(x_train, y_train)

gd.best_params_
{'criterion': 'squared_error', 'max_depth': 8, 'splitter': 'best'}
```

→ If we put these params, then we can get best accuracy.

```
gd.best_score_
0.9999
```

Teacher's Signature.....

Random Search CV

Date
Page

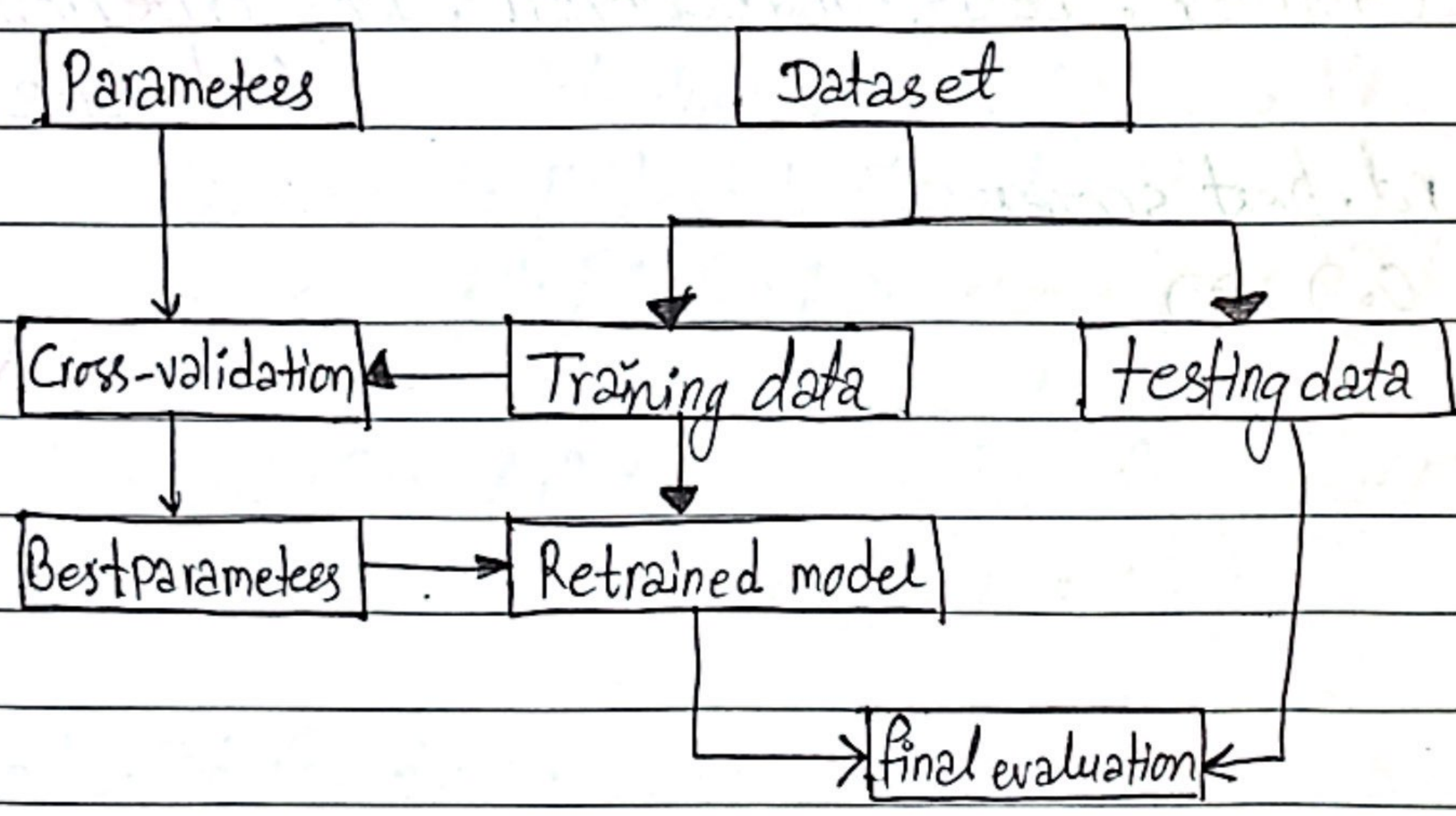
```
rd = RandomizedSearchCV(DecisionTreeRegressor(),
                        params_distributions = df,
                        n_iter = 20)

rd.fit(x_train, y_train)

rd.best_params_
{'splitter': 'best', 'max_depth': 17, 'criterion':
                                  'absolute_error'}

rd.best_scores_
0.9999
```

Teacher's Signature.....

Cross-Validation:

→ Methods to split data.

→ Cross-Validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data.



→ If cross-validation is 5, then whole dataset will be split into 5 different ways, and model train 5 different dataset, so we can see different 5 accuracy.

Teacher's Signature.....

Date _____
Page _____

→ Method used for cross-validation:

- leave p out cross-validation
- leave one out cross-validation
- holdout cross-validation
- Repeated random subsampling validation
- K-fold cross-validation
- Stratified K-fold cross-validation
- Time series cross-validation.
- Nested cross-validation.

(*) K-Fold Cross-Validation:

→ The original dataset is equally partitioned into K subparts or folds. Out of the K-folds or groups, for each iteration, one group is selected as validation data, and the remaining (K-1) groups are selected as training data.

→ Not suitable for an imbalance dataset.

dataset — 10/2 = 5

Train	Validation
5	5
5	5

validation training

Teacher's Signature.....

Date
Page

(*) Stratified K-Fold CV:

→ Used in classification type data
→ It works on imbalanced data.

(*) Leave one-out cross-validation (CV)

→ It is an exhaustive cross validation technique. It is a category of Lp OCV with the case of $p=1$.

→ Every data is ~~the~~ selected for ~~traing~~ training & testing data.

→ It takes more time for big/huge data.

(*) Leave-p out CV

→ It takes two or more data for training & ~~remaing~~ remaining data for testing. Similarly this is apply for all data in dataset.

Teacher's Signature.....

Date
Page

* Cross Validation (Practical)

```
import pandas as pd

dataset = pd.read_excel(r"location\package_prediction.xlsx")

dataset.head(5).
```

	CGPA	PACKAGES
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57

```
x = dataset[["CGPA"]]
y = dataset["PACKAGES"]

new_data = dataset.head(10)

x_new = new_data.iloc[:,:-1]
y_new = new_data["PACKAGES"]

from sklearn.model_selection import LeaveOneOut,
                                    LeavePOut, KFold, StratifiedKFold
```

Teacher's Signature.....

Date
Page

```
lo = LeaveOneOut()

for train, test in lo.split(x_new, y_new):
    print(train, test)

lp = LeavePOut(p=2)

for train, test in lp.split(x_new, y_new):
    print(train, test)

Kf = KFold(n_splits=5)

for train, tests in Kf.split(x_new, y_new):
    print(train, test)

skf = StratifiedKFold(n_splits=5)  # only work with classification data

# to check, how much accuracy can our model give to our data.
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import cross_val_score

q = cross_val_score(LinearRegression(), x, y, cv=5)
q.sort()
q
array([0.6648, 0.8275, 0.8589, 0.8709, 0.9194])
```

Teacher's Signature.....

Date
Page

```
p = cross_val_score (LinearRegression (), x, y,
                     cv = KFold (n_splits = 10))
p.sort()
p
```

```
array([0.5234, 0.5363013, 0.7150, 0.7554, 0.7567,
       0.7705, 0.8321, 0.8776, 0.8805, 0.9270])
```

Teacher's Signature.....

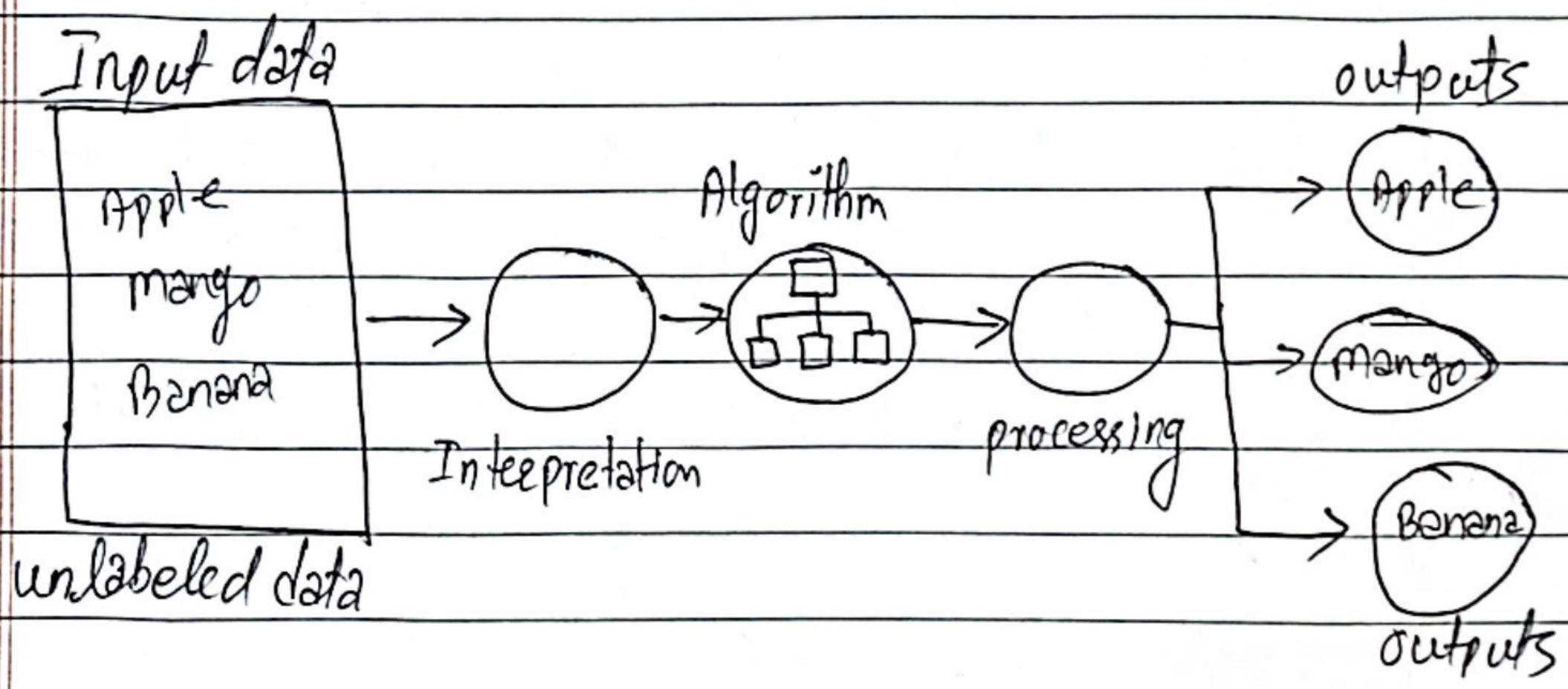
Unsupervised Learning

Date _____
Page _____

→ Unsupervised learning is a type of machine learning that learns from unlabeled data.

→ This means that the data does not have any pre-existing labels or categories.

→ The goal of unsupervised learning is to discover patterns & relationships in the data without any explicit guidance.



→ Types:
* Clustering → divided into separate groups
* Association → Arrangement in systematic ~~Arrang~~ way.

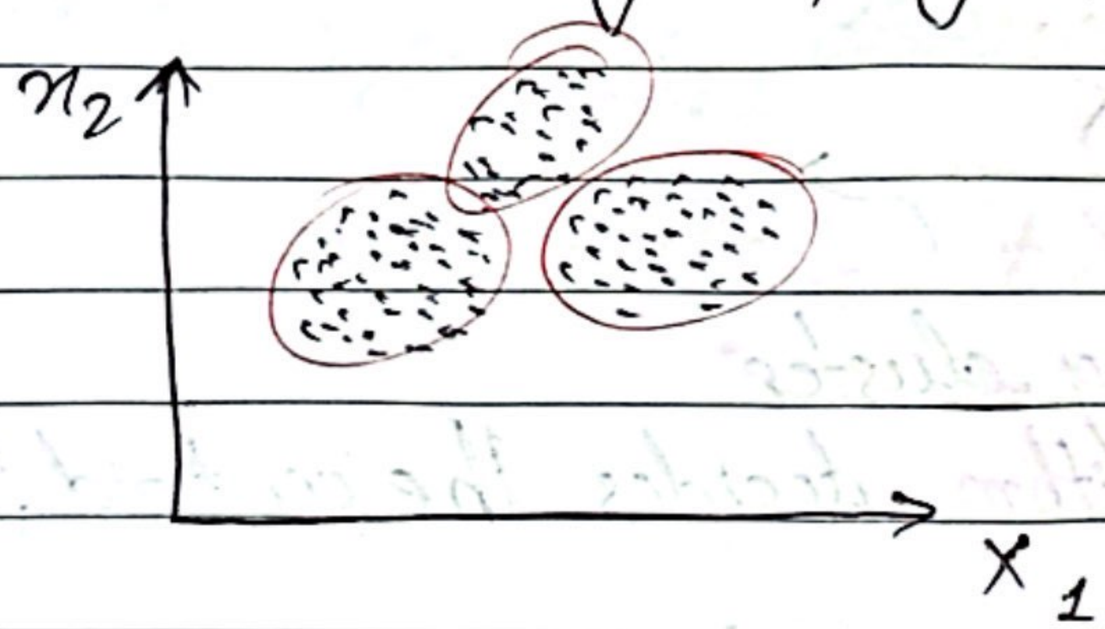
Teacher's Signature.....

→ List of some popular unsuperwised learning algorithms:

- K-means clustering
- Hierarchal clustering
- DBSCAN Clustering
- Apriori algorithm
- Principle Component Analysis.

→ Cluster:

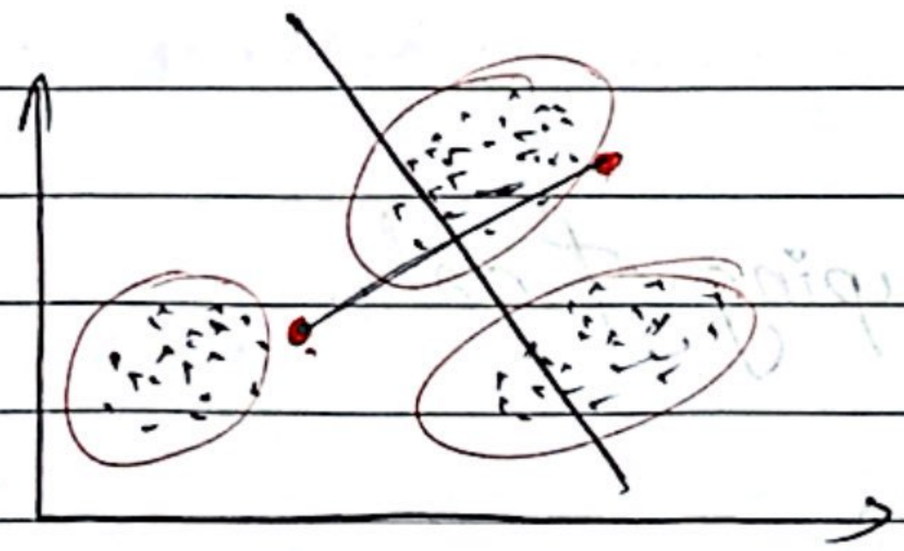
It is a grouping of data.



K-Means Clustering :

→ K-Means Clustering is an Unsuperwised Learning algorithm, which groups the unlabeled dataset into different clusters.

→ K defines the number of pre-defined clusters that need to be created in the process.



→ Process of making cluster
i) At first algorithm decides the centroid (centre point)
ii) A line is passed from the centre of line joining the centroid.
iii) Finds a mean

→ It is used for linear separable data.
→ There may be no. of groups.

→ To find the best cluster we use elbow method.

Teacher's Signature.....

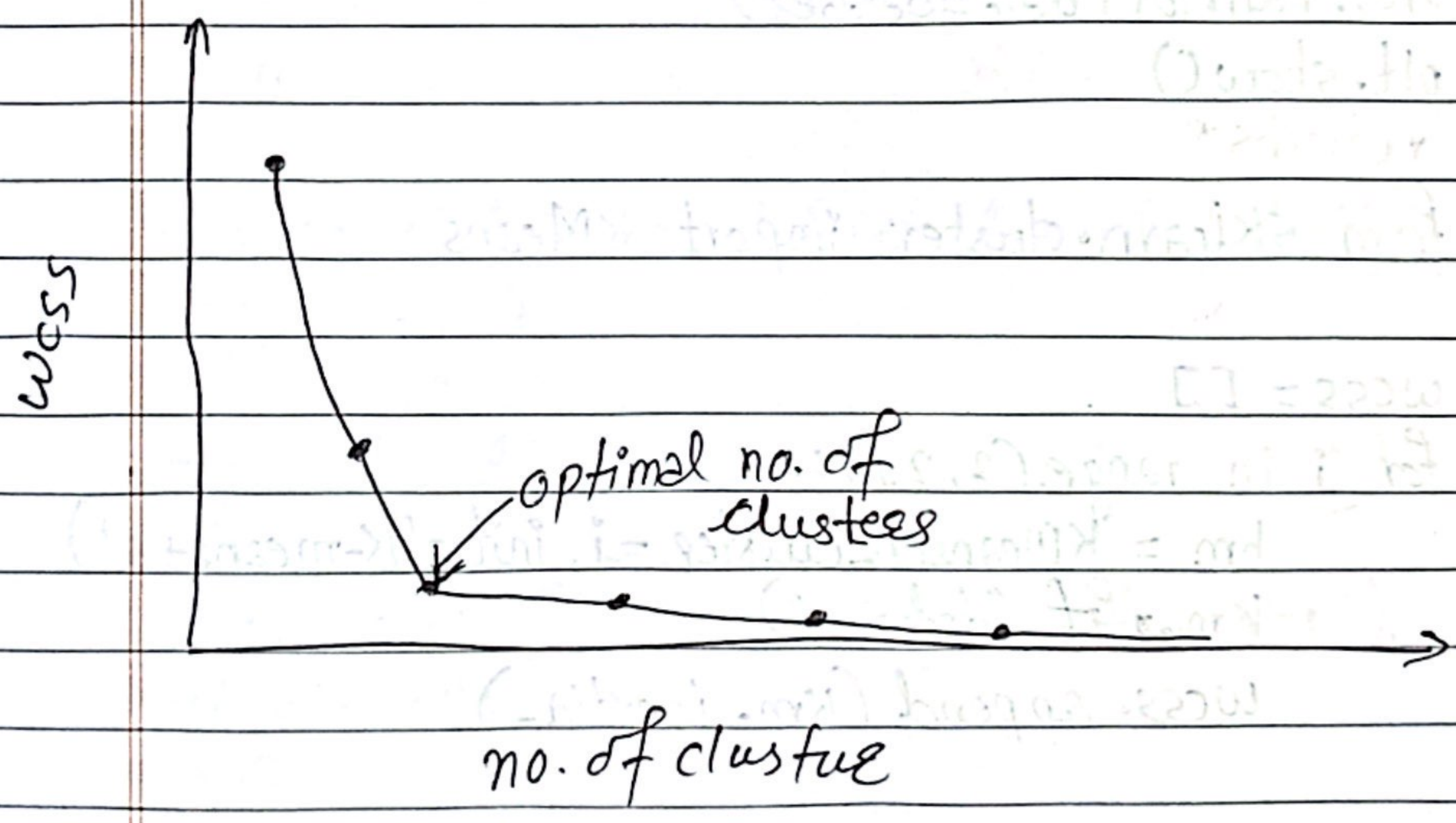
Date
Page

* Elbow method:

→ The elbow method is one of the most popular ways to find the optimal number of clusters.

→ This method uses the concept of WCSS value. WCSS stands for within cluster sum of squares, which defines the total variations within a cluster.

$$WCSS = \sum_{pi \text{ in cluster } 1} distance(Pi\,C_1)^2 + \sum_{pi \text{ in cluster } 2} distance(Pi\,C_2)^2 + \sum_{pi \text{ in cluster } 3} distance(Pi\,C_3)^2$$



Teacher's Signature. .....

Date
Page

* practical:

```
import pandas as pd.
import matplotlib.pyplot as plt
import seaborn as sns.

dataset = pd.read_csv('Iries_flower.csv')

dataset.head(4).
```

Sepal_length	sepal_width	pedal_length	petal_width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2

```
sns.pairplot(data=dataset)
plt.show()
```

Graphs

```
from sklearn.cluster import KMeans

wcss = []
for i in range(2,21):
    km = KMeans(n_cluster=i, init='K-means++')
    km.fit(dataset)
    wcss.append(km.inertia_)
```

Teacher's Signature.....

```
plt.figure(figsize=(10,5))
plt.plot([i for i in range(2,21)], wcss,marker="o")
plt.xlabel("no of cluster")
plt.xticks([i for i in range(2,21)])
plt.ylabel("wcss")
plt.grid(axis="x")
plt.show()

Kmn = KMeans(n_cluster = 3)
dataset["predict"]=Kmn.fit_predict(dataset)

dataset.head(4)
```

Sepal_length	sepal_width	petal_length	petal_width	predict
5.1	3.5	1.4	0.2	1
4.9	3.0	1.4	0.2	1
4.7	3.2	1.3	0.2	1
4.6	3.1	1.5	0.2	1

```
sns.pairplot(data=dataset, hue="predict")
plt.show()
```

Graphs

```
org_dataset= pd.read_csv("IRIS.CSV")

sns.pairplot(data=dataset, hue="species")
plt.show()
```

Graphs

Teacher's Signature. .....

Hierarchical Clusteeing

Date
Page

→ It is used in linearly separatable data.

→ It is used to group the unlabled datasets into a cluster and also known as hierarchical cluster analysis. or HCA

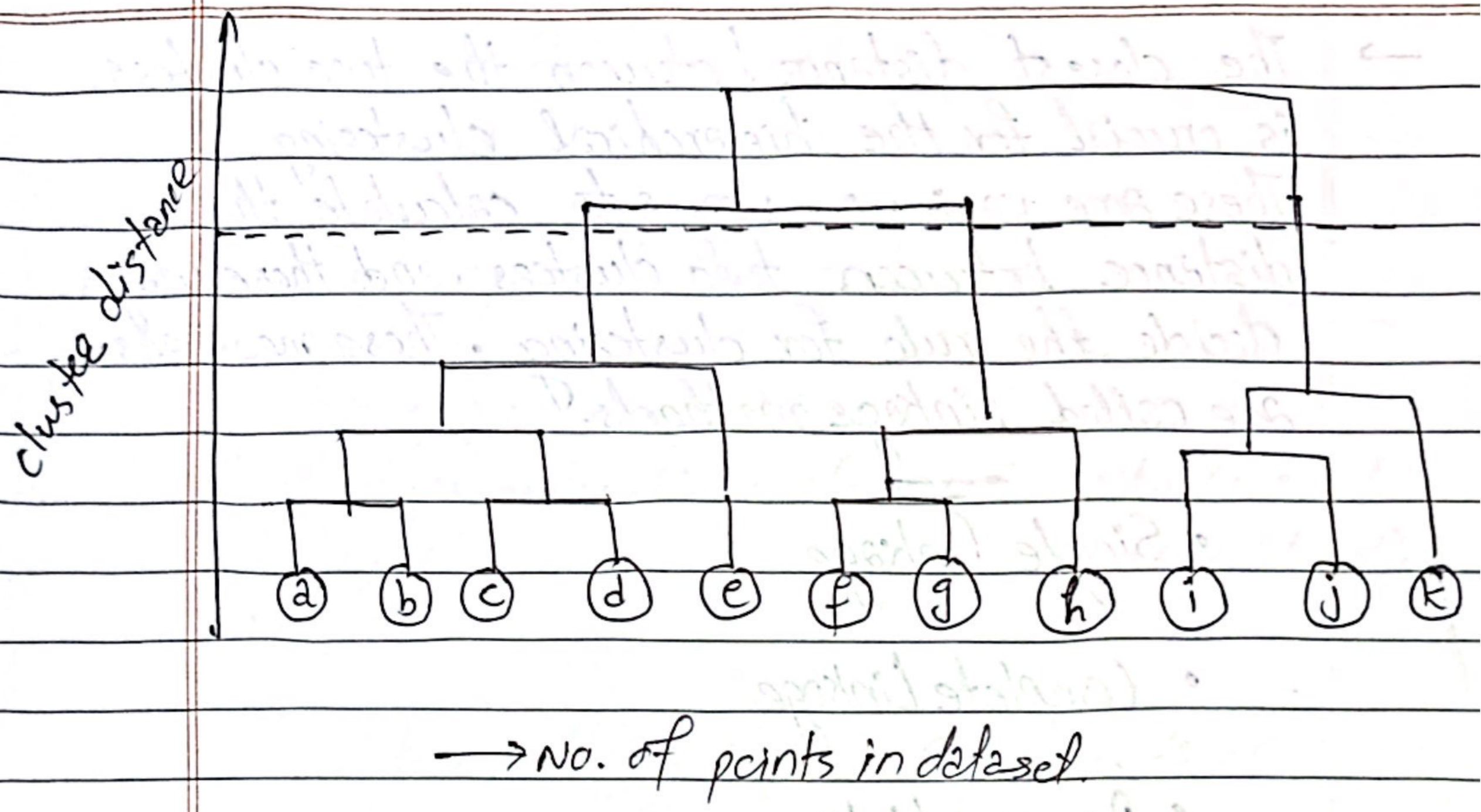
→ In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendogram.

→ The dendogram is a tree like structure that is mainly used to store each step as a memory that the HC algorithm performs.

→ The dendrogram plot, the Y axis shows the Euclidean distances between the data points. and the x-axis shows all the data points of the given dataset.

Teacher's Signature.....

Scanned with CamScanner



→ Hierarchical clustering technique has two approaches:

- Agglomerative: Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

- Divisive: Divisive algorithm is the reverse of the agglomerative algorithm top-down approach.

Date
Page

→ The closest distance between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called Linkage methods.

- Single linkage
- Complete Linkage
- Average Linkage
- Centroid Linkage.

*practical

```
import pandas as pd
import matplotlib.pyplot as plt.
import seaborn as sns.

dataset = pd.read_csv("Iris-flower.csv")
dataset.head(5).
```

sepal_length	sepal_width	petal_length	petal_width
5.1	3.5	1.4	0.2

Teacher's Signature.....

Date
Page

```
sns.pairplot(data=dataset)
plt.show()
```

Graph

```
import scipy.cluster.hierarchy as sc.

plt.figure(figsize=(20,20))
sc.dendrogram(sc.linkage(dataset, method='single',
plt.show()                        metric='euclidean'))
```

Graph

```
from sklearn.cluster import AgglomerativeClustering

ac = AgglomerativeClustering(n_clusters= 2, linkage =
                                        'single')
dataset["predict"] = ac.fit_predict(dataset)

dataset.head(5)
```

sepal_length	sepal_width	petal_length	petal_width	predict
5.1	3.5	1.4	0.2	1
4.9	3.0	1.4	0.2	1
4.7	3.2	1.3	0.2	1
4.6	3.1	1.5	0.2	1
5.0	3.6	1.4	0.2	1

```
sns.pairplot(data=dataset, hue='predict')
plt.show()
```

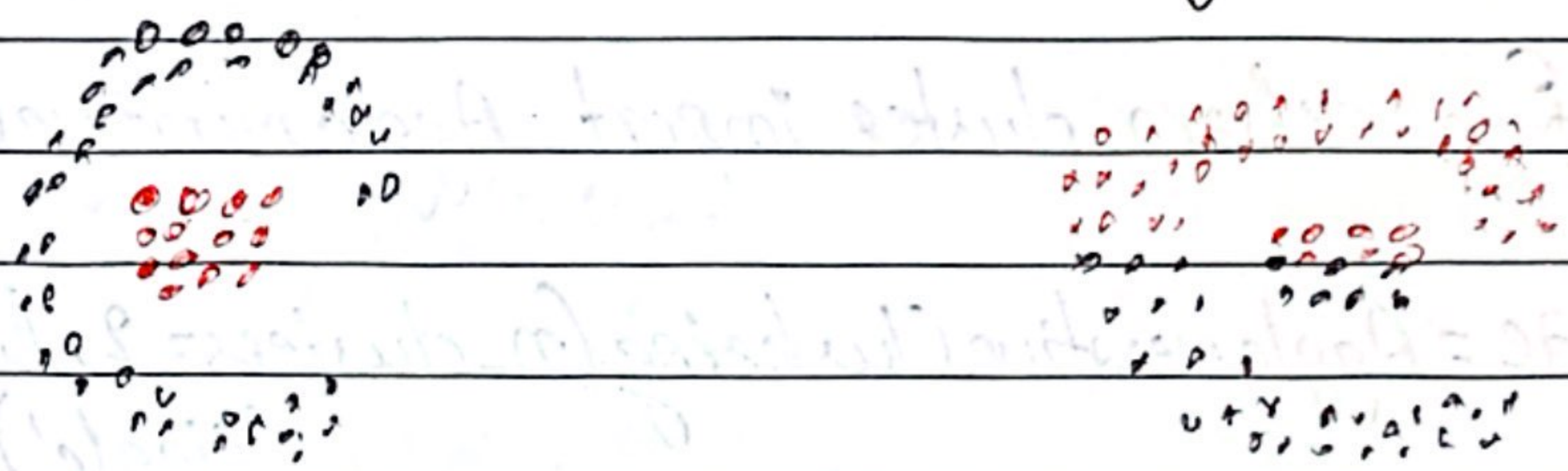
Teacher's Signature. .....

DBSCAN Clusteering:

⇒ Density-Based Spatial Clustering of Application with Noise.

→ The clusters found by DBSCAN can be any shape, which can deal with some special cases that other methods cannot.

→ It makes cluster according to density.



DBSCAN K-Means

→ It is used for non-linear separable dataset.

→ To make cluster:

i) Min point

ii) Epn (r)

a) core point
b) Boundary point
c) Noise point.

Date
Page

* practical

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns.

from sklearn.datasets import make_moons

a,b = make_moons (n_samples=250, noise=0.05)

df = {"data1": a[:,0], "data2": a[:,1]}

dataset = pd.DataFrame (df)

dataset.head(3)
```

data1	data2
-0.683742	0.808391
0.069786	0.280032
-0.611067	0.790674

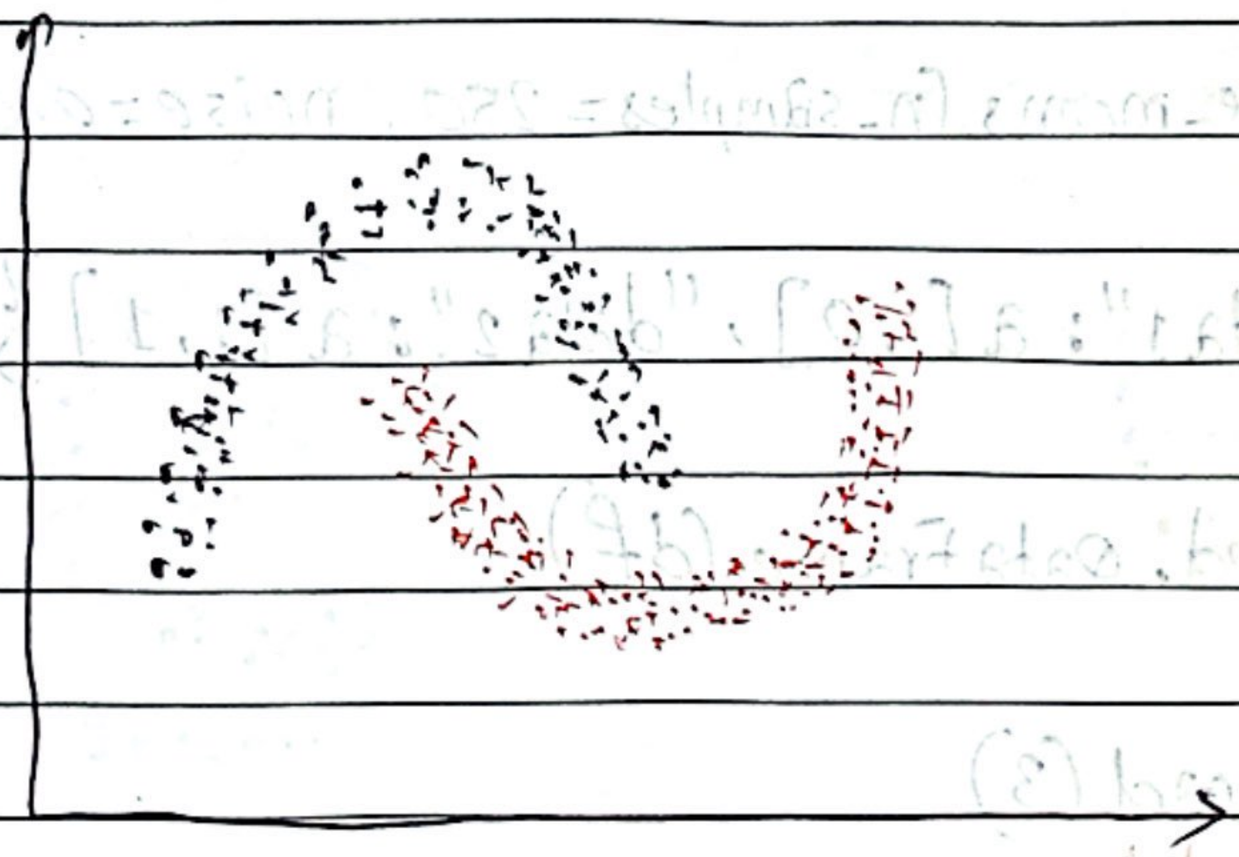
```
sns.scatteeplot (x="data1", y="data2", data=dataset)
plt.show()
```

Graph

Teacher's Signature. .....

Date
Page

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.2, min_samples=5)
dataset["predict"] = db.fit_predict(dataset)

sns.scatterplot(x="data1", y="data2", data=dataset,
plt.show()                                hue='predict')
```



Teacher's Signature.....

Date
Page

Silhouette Score:

→ Silhouette refers to a method of interpretation and validation of consistency within clusters of data.

→ Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique.

→ It's value ranges from -1 to 1.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$

and,

$$s(i) = 0, \text{ if } |C_I| = 1$$

which can be also written as:

$$s(i) = \begin{cases} -1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

From the above definition it is clear that,

$$-1 \le s(i) \le 1$$

(-1) → Bad clustering

(1) → Best clustering

Teacher's Signature. ....

Date
Page

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i,j)$$

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i,j)$$

Teacher's Signature.....

Date
Page

Association Rule Learning:

→ Association Rule Learning is a machine learning technique used to discover interesting relationships or patterns in large datasets.

→ It is often applied to transactional data, where items are bought or used together.

milk bread butter	milk, cereal bread, butter	milk bread butter
Customer 1	Customer 2	Customer 3

sugar eggs
Customer n

→ Association rule learning works on the concept of If and Else statements, such as if A then B.

If A → Then B

If element is called antecedent, and then statement is called Consequent.

Teacher's Signature.....

Date
Page

→ How does Association Rule Learning Work:
- Support
- Confidence
- Lift

	Item 1	Item 2	Item 3
Shopper-1	Eggs	Bacon	Soup
Shopper-2	Eggs	Bacon	Apple
Shopper-3	Eggs	Bacon	Apple
Shopper-4	Soup	Bacon	Banana
Shopper-5	Banana	Butter	–
Shopper-6	Butter	–	–

$Support(Bacon) = \frac{4}{6}$

Support :- Transactions (A) / Total Transaction
→ Find frequently bought items

Confidence
→ $conf(X \Rightarrow Y) = \frac{\text{no. of transaction containing } X \& Y}{\text{no. of transaction containing } X}$

→ Tell us how confident we can be that an item will be purchased.

Teacher's Signature. .....

Date
Page

lift

→ strong association.

→ $lift(X \Rightarrow Y) = \dfrac{Supp(X \cap Y)}{Supp(X) * Supp(Y)}$

lift > 1 : Both products buy together
lift < 1 : Both products are highest selling but not combinely.
lift = 1 : no associations.

→ Types of Association Rule learning:
- Apriori
- Eclat
- F-P Growth Algorithm.

→ Applications of Association Rule learning.
- Market Basket Analysis
- Medical Diagnosis
- Protein Sequence

Teacher's Signature.....

Apriori Algorithm Learning :-

Transaction ID	Items
T_1	{E, K, M, M, O, Y}
T_2	{D, E, K, N, O, Y}
T_3	{C, K, M, U, Y}
T_4	{C, E, I, K, O}
T_5	{A, E, K, M}

A, B, C, D, E, F, G, H, I, J, K, L
M, N, O, P, Q, R, S, T, U, V, W, X
Y, Z

Min support = 50%
Min Confident = 50%

$A = \frac{1}{5} \times 100 = 20\%$ (x)

$C = 2/5 \times 100 = 40\%$ (x)

$D = 20\%$ (x)

$E = 4/5 \times 100 = 80\%$ (✓)

$I = 20\%$ (x)

$K = 5/5 \times 100 = 100\%$ (✓)

Ny for all alphabet

KE = 4

KM = 3

KO = 3

Teacher's Signature. .....

Date
Page

Em = 2 (X)

EO = 3

Ey = 2 (X

mO = 1 (X)

my = 2 (X)

Oy = 2 (X)

Final product:

KE = 4
KM = 3
KO = 3
Ky = 3
EO = 3

KEM = 2
KEO = 3 (✓)
KEY = 2

(We will put KEO product together)

Teacher's Signature.....

Date
Page

$KE = \frac{4}{4} \times 100 = 100$

* Practical

```
import pandas as pd

dataset=pd.read_csv(r"C:\groceries-groceries.csv")

dataset.columns


market=[]
for i in range(0, dataset.shape[0]):
    customer=[]
    for j in dataset.columns:
        if type(dataset[j][i])=str:
            customer.append(dataset[j][i])
    market.append(customer)


l=[]
for i in market
    for j in i:
        l.append(j)

import collections
```

Teacher's Signature. .....

Date
Page

```
p = collections.Counter(l)

d = {"Item Name": p.keys(), "values": p.values()}

pd.DataFrame(d).sort_values(by=["values"], ascending
                                    =False)

from mlxtend.preprocessing.transactionencoder import
                                    TransactionEncoder

tr = TransactionEncoder()
tr.fit(market)

tr.transform(market)

df=pd.DataFrame(tr.transform(marked), columns =
                                    tr.columns)
df

from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.05, use_colnames=True,
                    max_len=3).sort_values(by=["support"])
```

Teacher's Signature.....

Frequent Pattern Growth Algorithm:-

Date
Page

'Fp patteen growth algorithm'

It is also a Association Rule Learning.

Steps in FP Growth:

Frequent patteen set
Ordeeed-item set
Ordeeed-item set & Conditional Frequent Patteen Tree is is built
Frequent patteen rules.

Dataset:

Transaction ID	Items	
T1	{E, K, M, N, O, Y}	{KEMOY}
T2	{D, E, K, N, O, Y}	{KEOY}
T3	{C, K, M, U, Y}	{KMY}
T4	{C, E, I, K, O}	{EKO}
T5	{A, E, K, M}	{EKM}

Frequenty buyed product/Item,
A = 1
C = 2
D = 1
E = 4

Teacher's Signature.....

Scanned with CamScanner

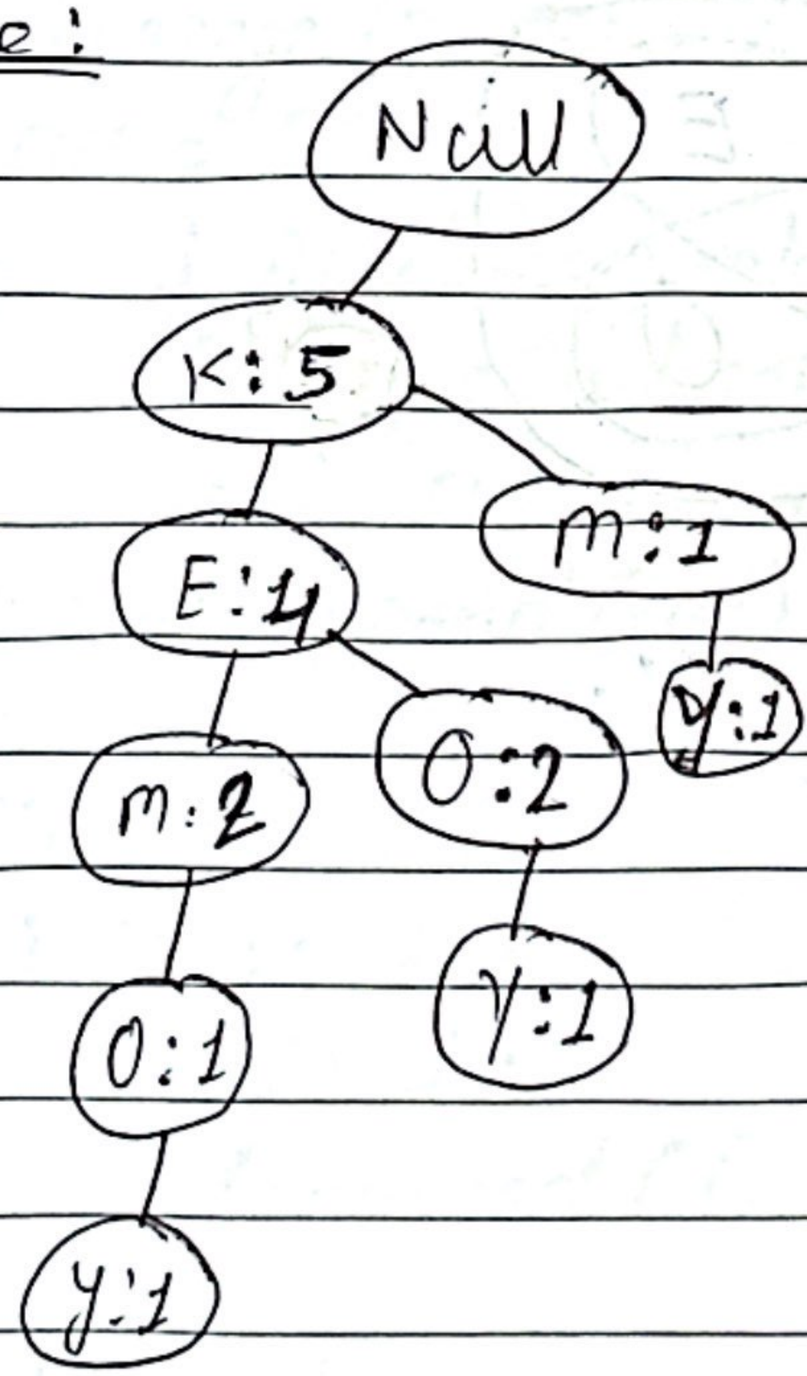
Date _____
Page _____

I = 1
K = 5
M = 3
N = 2
O = 3
U = 1
Y = 3

Min Support = 3

So, our item is, ∴ {E, K, M, O, Y} → ordered item set.

Tree:



Teacher's Signature. .....

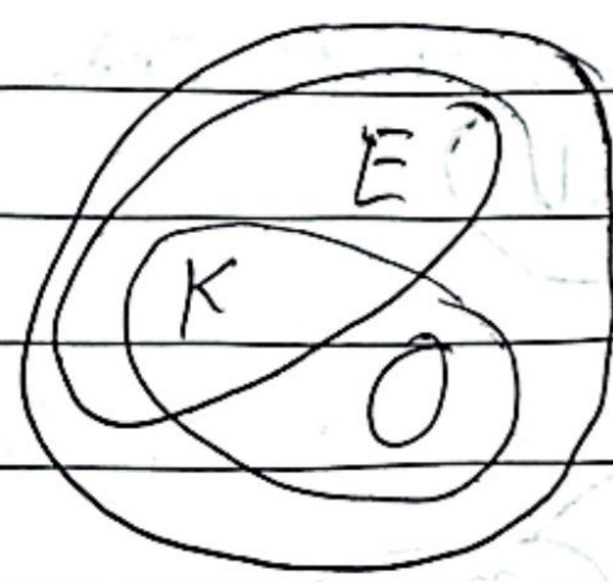
Date
Page

y	{O M E K} {O E K} {M K}
O	{M, E, K} {E K}
M	{E K} {K}
E	{K}

Frequent pattern rules:-

$\boxed{y \leftarrow K}$ $\boxed{O \leftarrow KE}$ $\boxed{M \leftarrow K}$

KE
KO
KEO



Teacher's Signature.....

Date
Page

* practical :

```
import pandas as pd

datasets = pd.read_csv(r"C:\ML Course\Dataset\"groceries.csv")

dataset.head(5)

dataset.shape
(9835, 33)

market = []
for i in range (0, dataset.shape[0]):
    cus = []
    for j in dataset.columns:
        if type(dataset[j][i])== str :
            cus.append(dataset[j][i])
    market.append(cus)

l = []
for i in market:
    for j in i :
        l.append(j)

import collections

P = collections.Counter(l)
```

Teacher's Signature. .....

Date
Page

```
d={"Item Name": p.keys(), "values": p.values()}

pd.DataFrame(d).sort_values(by=["values"],ascending
                                =False)

from mlxtend.preprocessing.transactionencoder import
                                TransactionEncoder.

tr = TransactionEncoder()
tr.fit(market)

df= pd.DataFrame(tr.transform(market), columns =
                                tr.columns_)

from mlxtend.frequent_patterns import fpgrowth

fpgrowth(df, min_support=0.05, use_colnames=True,
                max_len=3).sort_values(by=["support"])
```

Teacher's Signature.....

ENSEMBLE LEARNING:

→ Here, we train multiple models in a same model or dataset on same time, and after combining the answers of different models we give a single answer.

→ The Ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.
There are two main types of ensemble learning:
1) Bagging (Bootstrap Aggregating)

2) Boosting

* Basic Ensemble Techniques:
- Max Voting
- Averaging
- Weighted Average

* Advanced Ensemble Techniques
- Stacking
- Blending
- Bagging
- Boosting

Teacher's Signature.....

Date
Page

* Algorithms based on Bagging and Boosting

1) Bagging algorithms:

- Bagging meta-estimator
- Random forest.

2) Boosting Algorithms:

- AdaBoost
- GBM
- XGBM
- Light GBM
- CatBoost

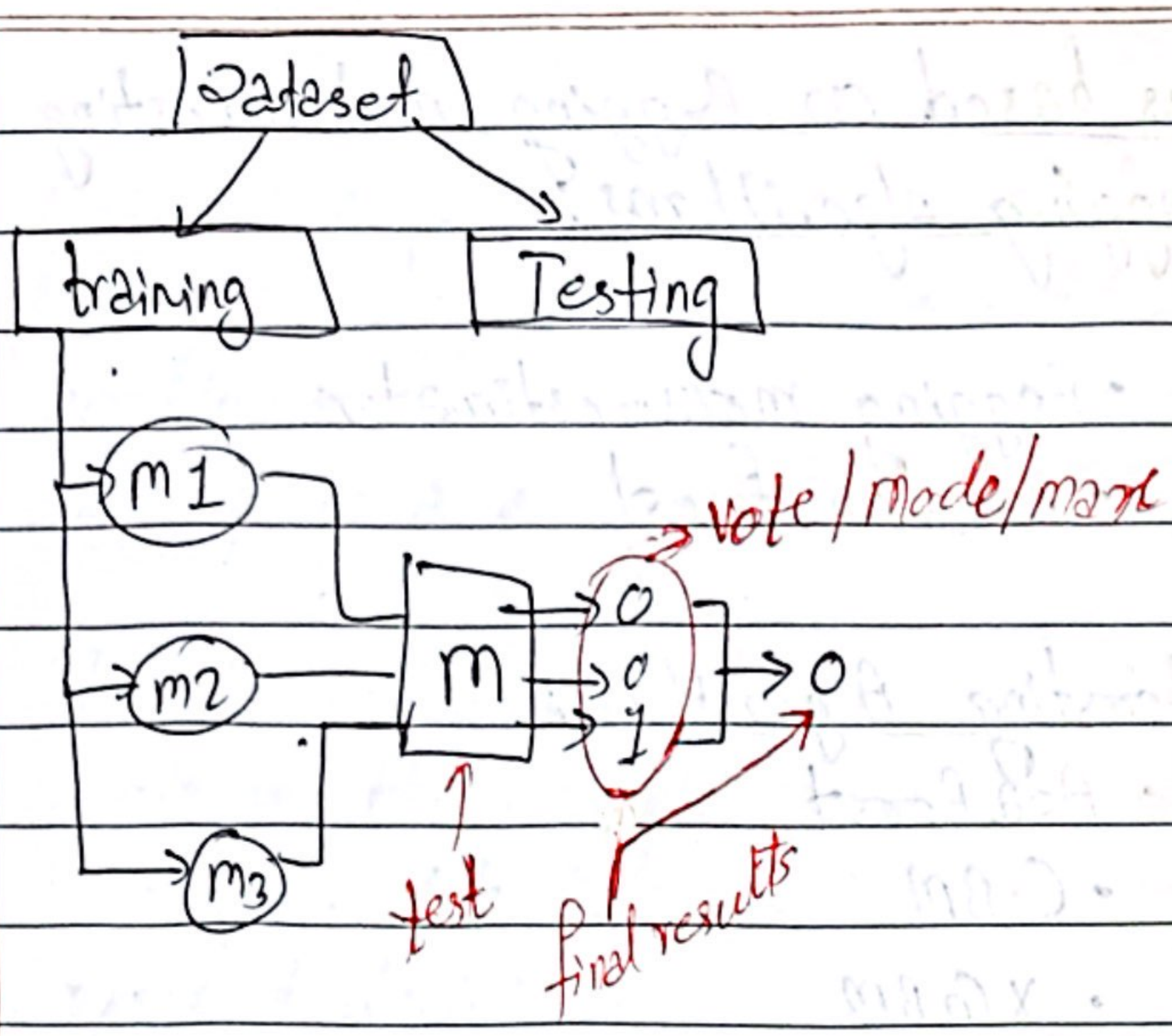
* Max Voting:

→ The max voting method is generally use for classification problems. In this technique, multiple models are used to make predictions for each data point.

→ The predictions by each model are considered as a 'vote'. The predictions which we get from the majority of the models are used as the final prediction.

Teacher's Signature. .....

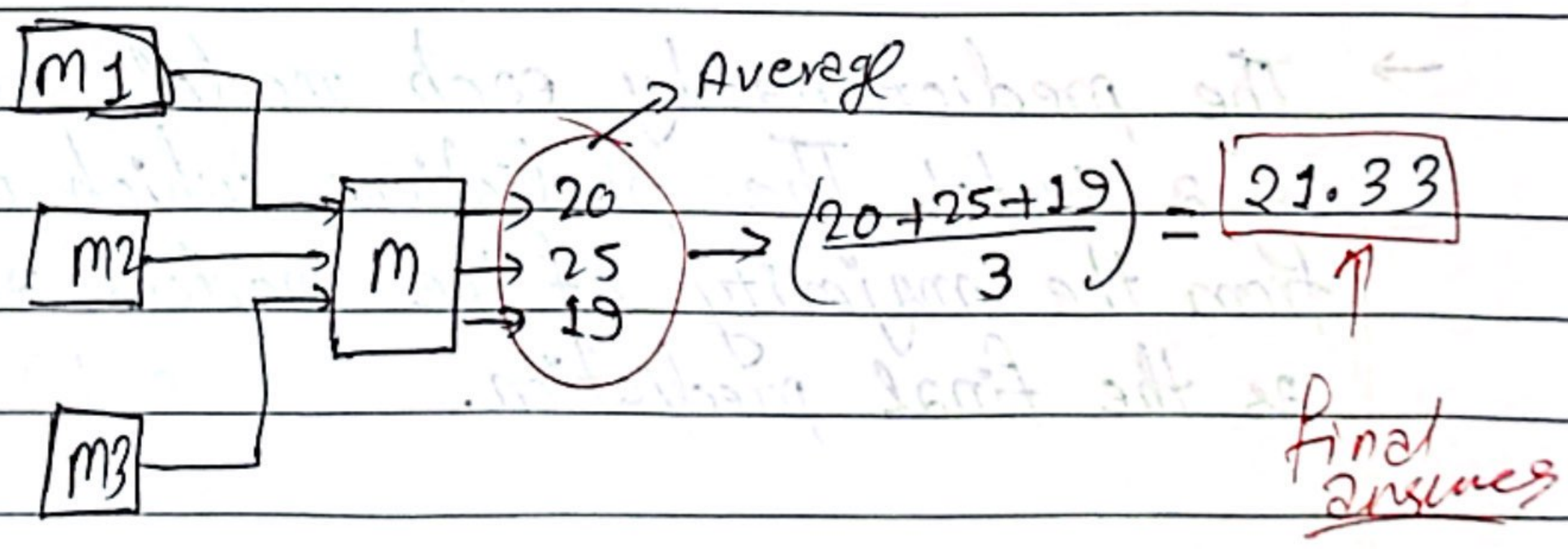
Scanned with CamScanner



* Averaging & Weighted Average Voting:

→ Take an average of predictions from all the models and use it to make the final prediction.

→ Averaging can be used for making predictions in reggression problems or while calculating probabilities for classification problems.



Date
Page

*practical (classification)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns.
from sklearn.datasets import make_moons

x, y = make_moons(n_samples = 1000, noise = 0.2)

df = {"x1": x[:,0], "x2": x[:,1], "y": y}

dataset = pd.DataFrame(df)

sns.scatterplot(x="x1", y="x2", data=dataset, hue="y")
plt.show()

x_a = dataset.iloc[:, :-1]
y_a = dataset["y"]

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 = 42)
```

Teacher's Signature. .....

Date
Page

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
dt.score(x_train, y_train)*100, dt.score(x_test, y_test)*100
```

(100.0, 94.0)

```
sv = SVC()
sv.fit(x_train, y_train)
sv.score(x_train, y_train)*100, sv.score(x_test, y_test)*100
```

(95.875, 96.5)

```
gnb = GaussianNB()
gnb.fit(x_train, y_train)
gnb.score(x_train, y_train)*100, gnb.score(x_test, y_test)*100.
```

(86.0, 88.5)

```
from sklearn.ensemble import VotingClassifier

li = [("dt1", DecisionTreeClassifier()), ("sv1", SVC()),
      ("gnb1", GaussianNB())]
```

Teacher's Signature.....

Date
Page

```
vc = VotingClassifier(li, weights=[5,4,7])
vc.fit(x_train, y_train)
```

```
vc.score(x_train, y_train)*100, vc.score(x_test, y_test)*100
```

(97.125, 96.5)

~~prd = DataFrame~~

```
prd = {"dt": dt.predict(x_test),
       "svm": svm.predict(x_test),
       "gnb": gnb.predict(x_test),
       "vc": vc.predict(x_test)}
```

```
pd.DataFrame(prd).head(5)
```

dt	svm	gnb	vc
1	1	1	1
1	1	1	1
1	1	1	1
0	0	0	0
1	1	1	1

* practical (Regression)

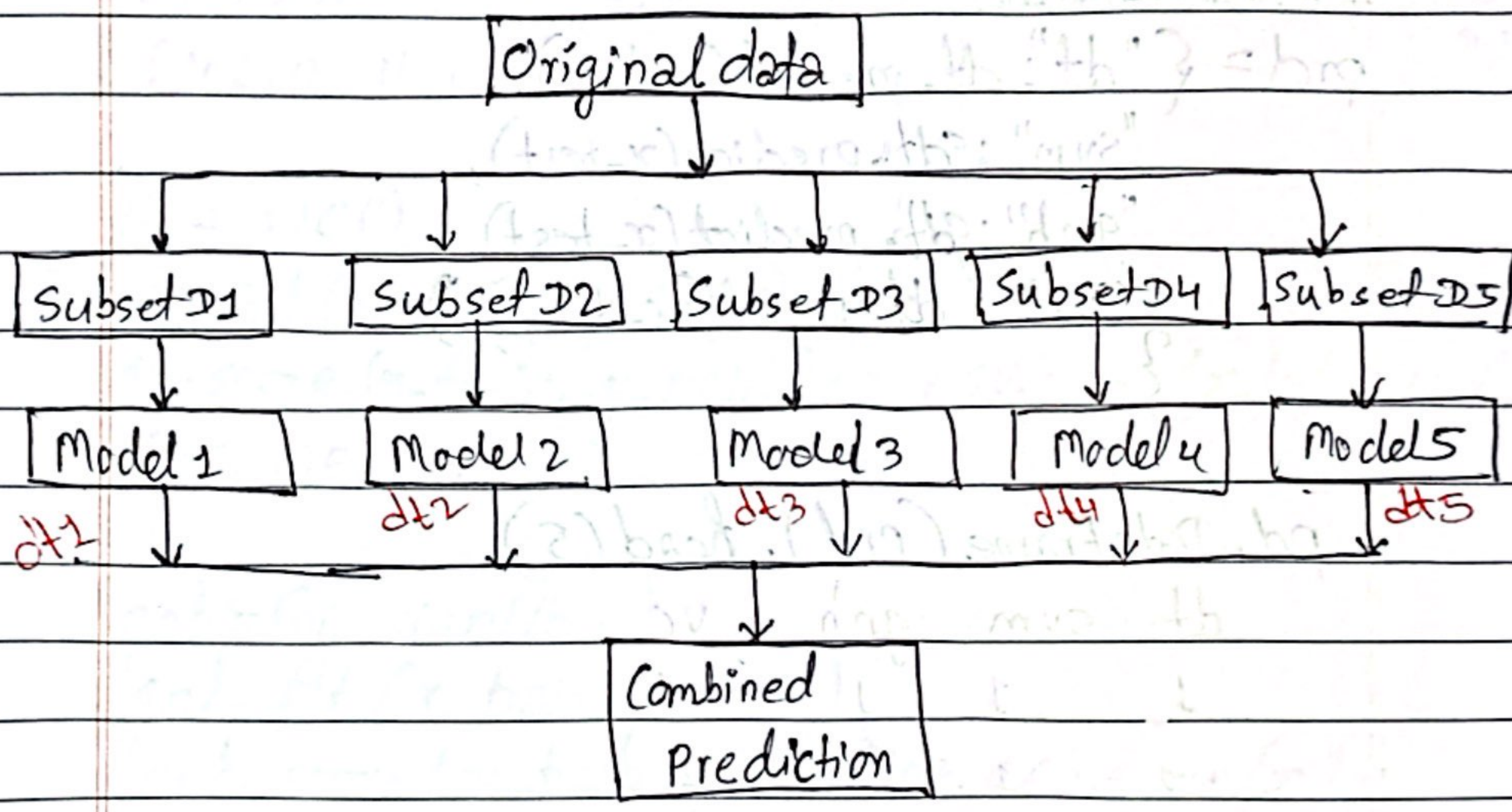
→ [ON LAPTOP]

→ It is similar to the classification example, process is same, but models are different and no. of models are also same (i.e. 3)

Teacher's Signature. .....

Bagging :-

⟹ Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for bagging may be less than the original set.



* Bagging Algorithms:

1) Bagging meta-estimator.

2) Random forest.

Teacher's Signature.....

Date
Page

1) Bagging meta-estimator:

⇒ It is an ensembling algorithm that can be used for both classification (Bagging Classifier) and regression (Bagging Regressor) problems.

→ It follows the typical bagging technique to make predictions.

2) Random Forest

→ Random Forest is another ensemble machine learning algorithm that follows the bagging technique. It is an extension of the bagging estimator algorithm.

→ The base estimators in random forest are decision trees.

* Practical (Bagging) (classification)

⇒
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
import seaborn as sns
```

Teacher's Signature. .....

Date
Page

```
x, y = make_moons(n_samples = 1000, noise=0.02)
df = {"x1": x[:,0], "x2": x[:,1], "y": y}
dataset = pd.DataFrame(df)

sns.scatterplot(x="x1", y="x2", data=dataset, hue="y")
plt.show()

x_a = dataset.iloc[:, :-1]
y_a = dataset["y"]

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
                                   x_a, y_a, test_size=0.2,
                                   random_state=42)

from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

bg = BaggingClassifier(estimator= SVC(), n_estimator
                                   =80)
bg.fit(x_train, y_train)
```

Teacher's Signature.....

Date
Page

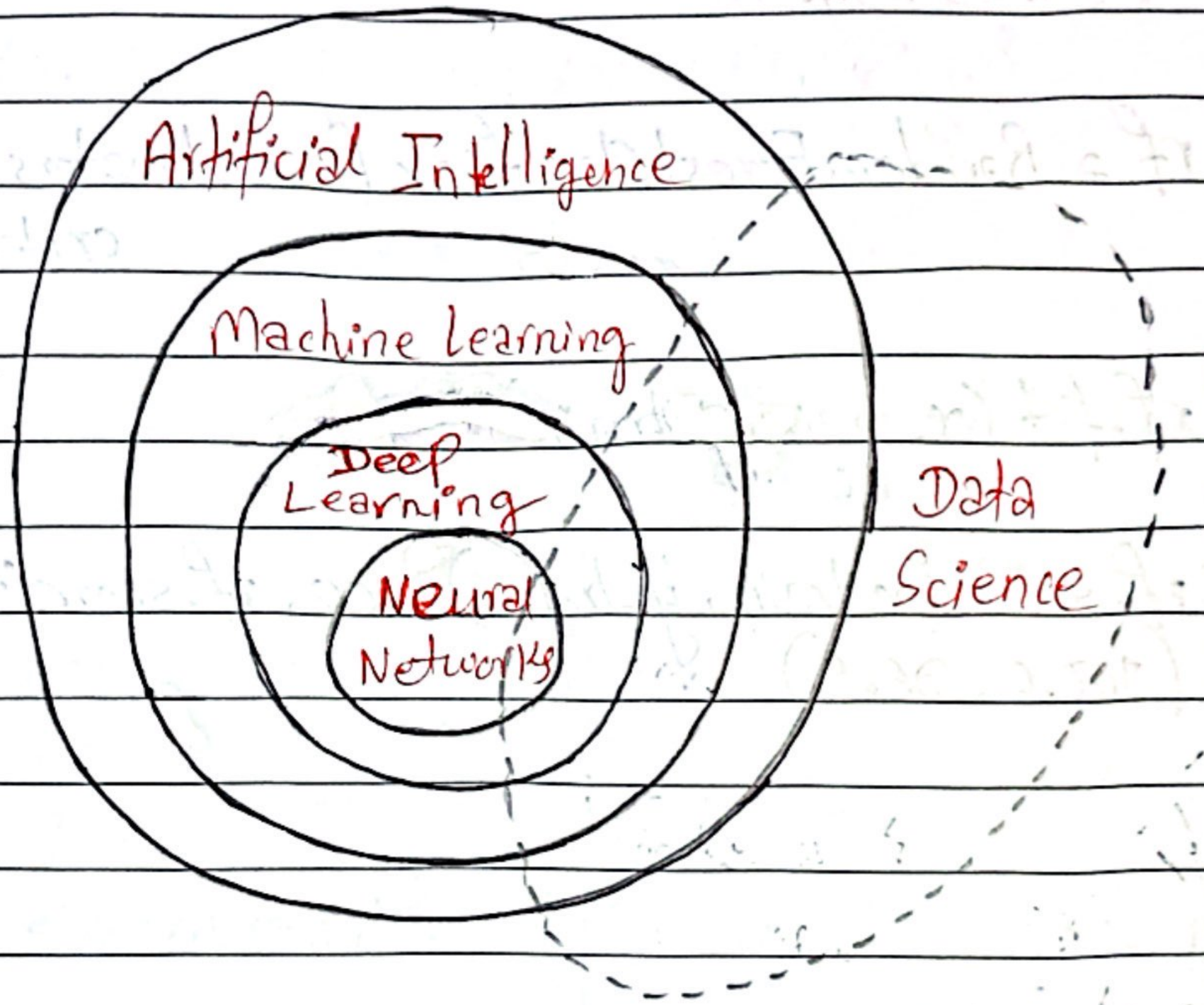
bg.score(x_train, y_train)*100, bg.score(x_test, y_test)*100
(95.75, 97.5)

rf = RandomForestClassifier(n_estimators=50, criterion='gini')

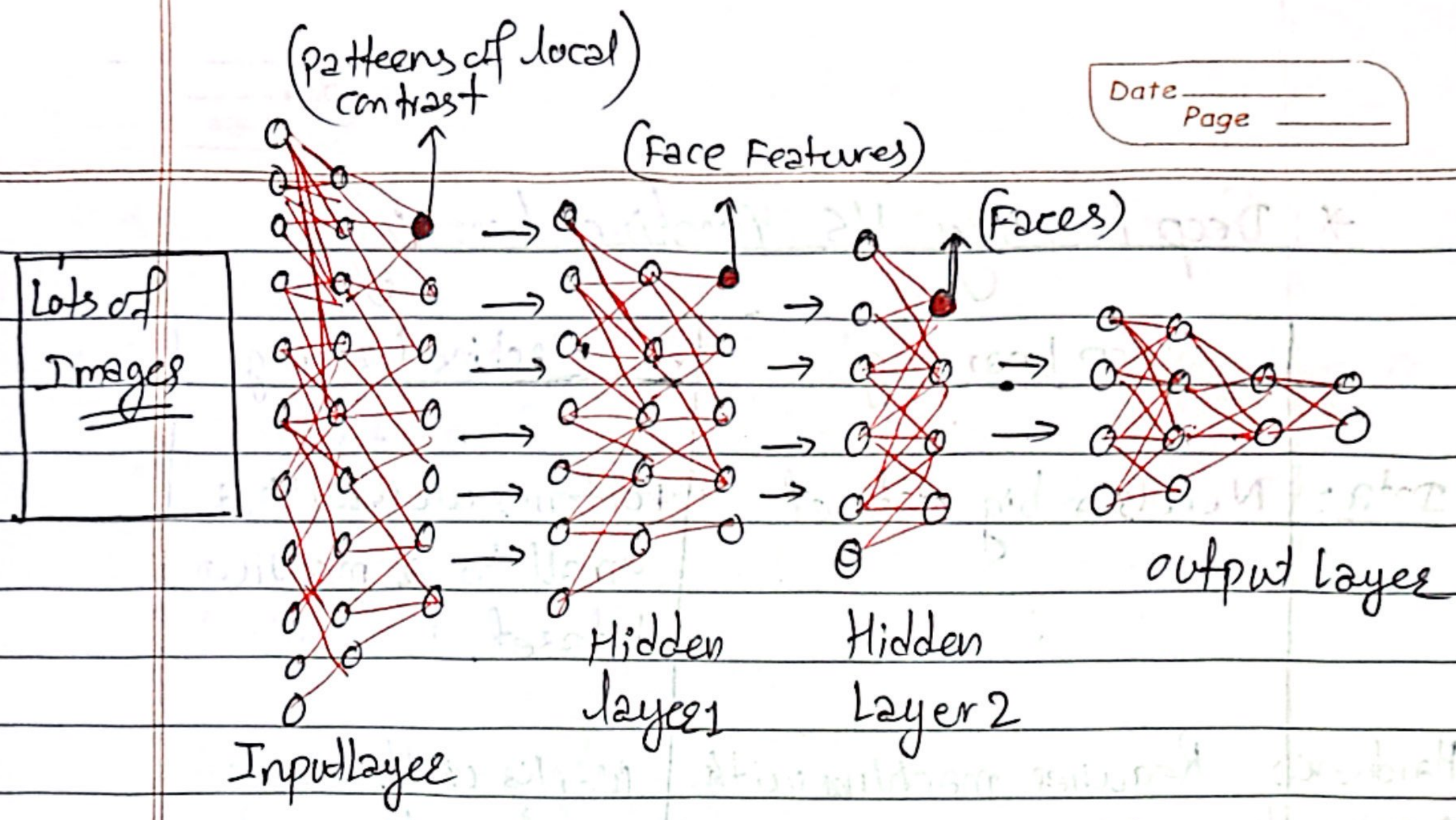
rf.fit(x_train, y_train)

rf.score(x_train, y_train)*100, rf.score(x_test, y_test)*100
(100.0, 96.5)

Deep Learning :



→ Deep learning is a collection of statistical techniques of machine learning for learning features hierarchies that are actually based on artificial neural networks.



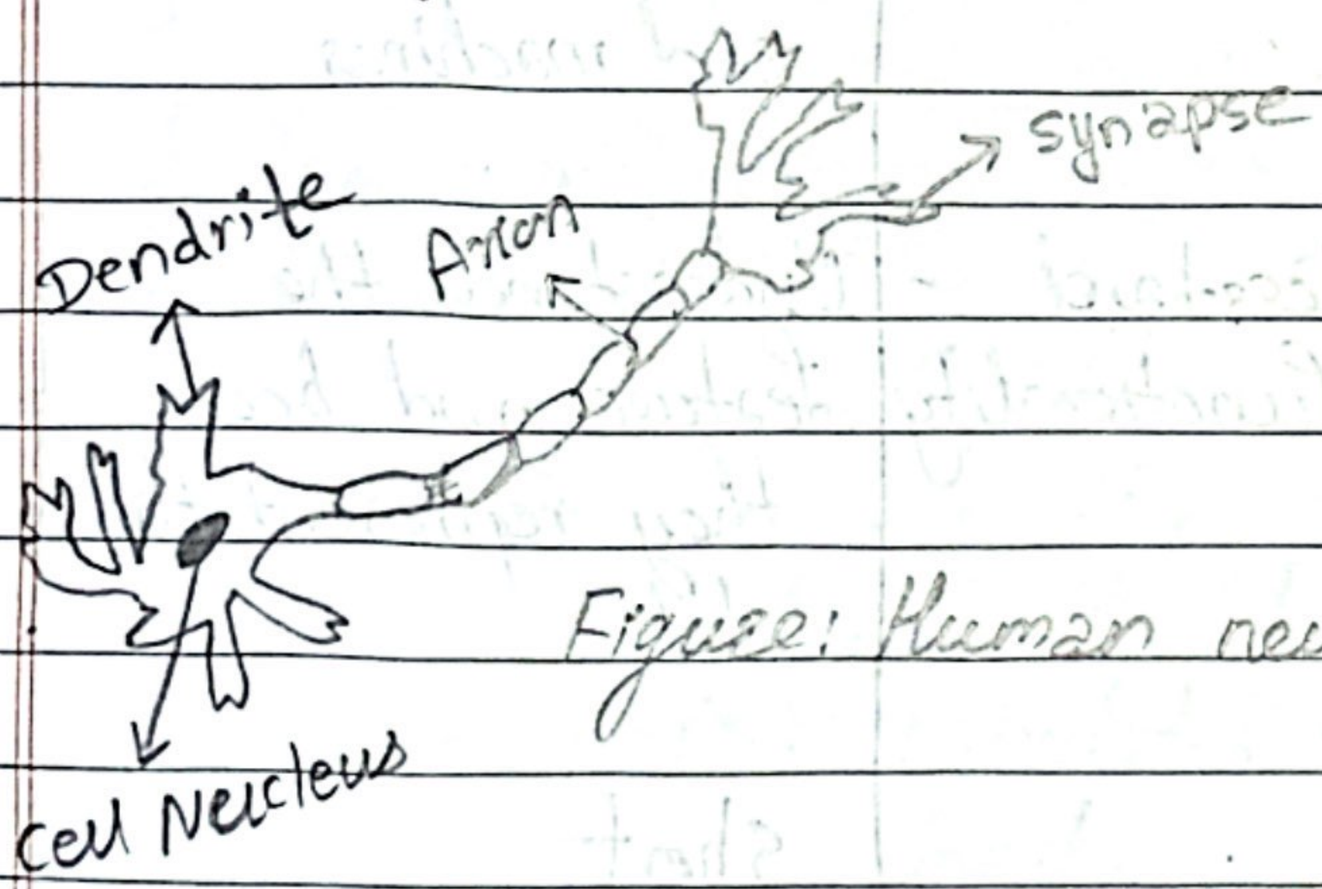
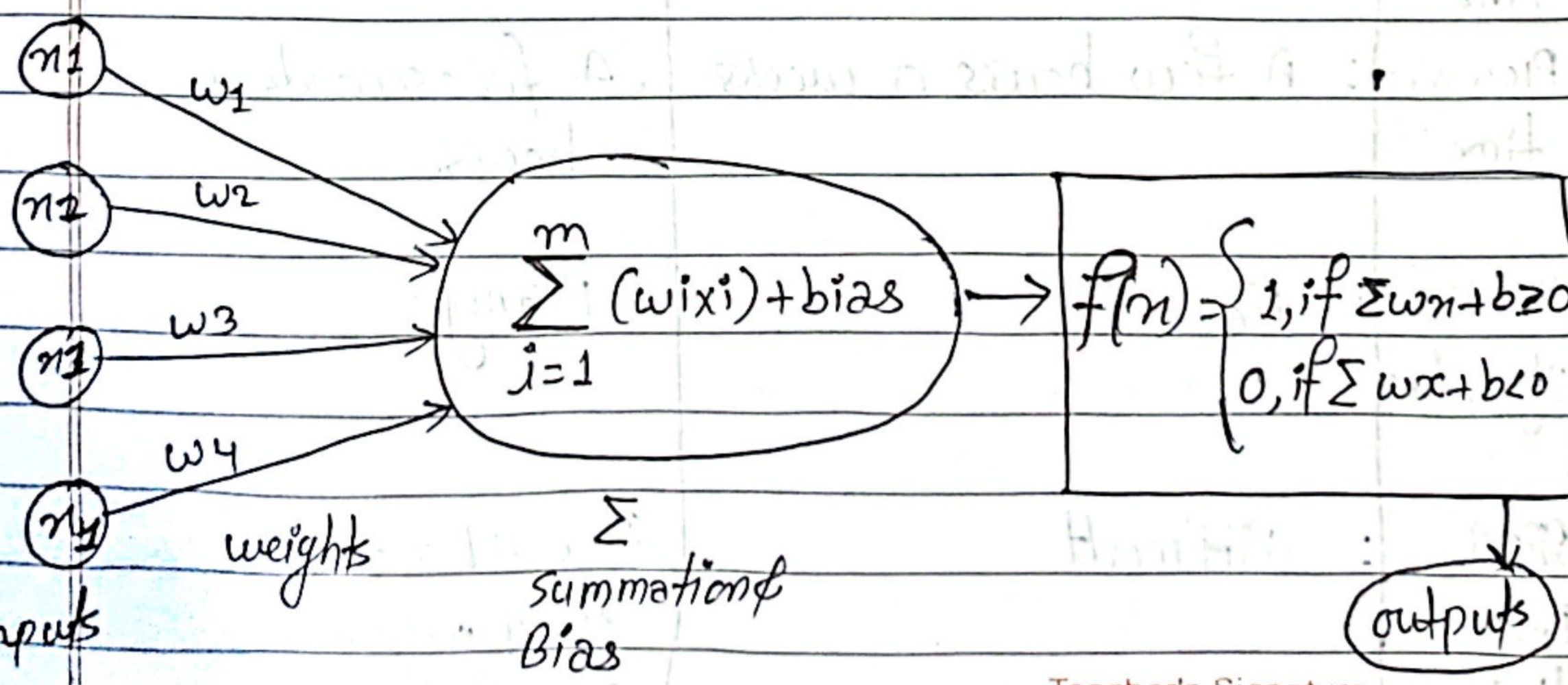


Figure: Human neuron



Date
Page

* Deep Learning VS Machine Learning :

	Deep Learning	Machine Learning
Data :	Needs a big dataset	Performs well with a small to a medium dataset.
Hardware requirements :	Requires machine with GPU	Works with low end machines
Engineering peculiarities :	Needs to understand the basic functionality of the data.	Understands the features and how they represent the data
Training Time :	Long	Short
Processing time :	A few hours or weeks	A few seconds or hours.
No. of algorithm :	Few	Many
Data Interpretation :	Difficult	Some ML algorithm are easy where as some are difficult.

Teacher's Signature.....

Date
Page

* What is Neuron and Neural Networks ?

→ ~~Neuro~~ Neuron, often refered to as a perceptron in the context of artificial neural network, is the fundamental unit of a neural network. It mimics the behaviour of biological neurons in the brain

Key components of Neuron:

i) Inputs (x_i) : Represents the input data or features.
ii) Weights (w_i) : Determine the importance of each input.
iii) Bias (b) : Allows the model to shift the activation function, improving flexibility
iv) Summation (Σ) : Calculates the weighted sum of inputs.

$$z = \sum_{i=1}^{n} w_i . x_i + b$$

v) Activation Function ($f(z)$) : Applies a non-linear transformation to the weighted sum to introduce non-linearity.

$$y = f(z)$$

Teacher's Signature. .....

Date
Page

Common Activation Functions:

- Sigmoid:

$$f(z) = \frac{1}{1+e^{-z}}$$

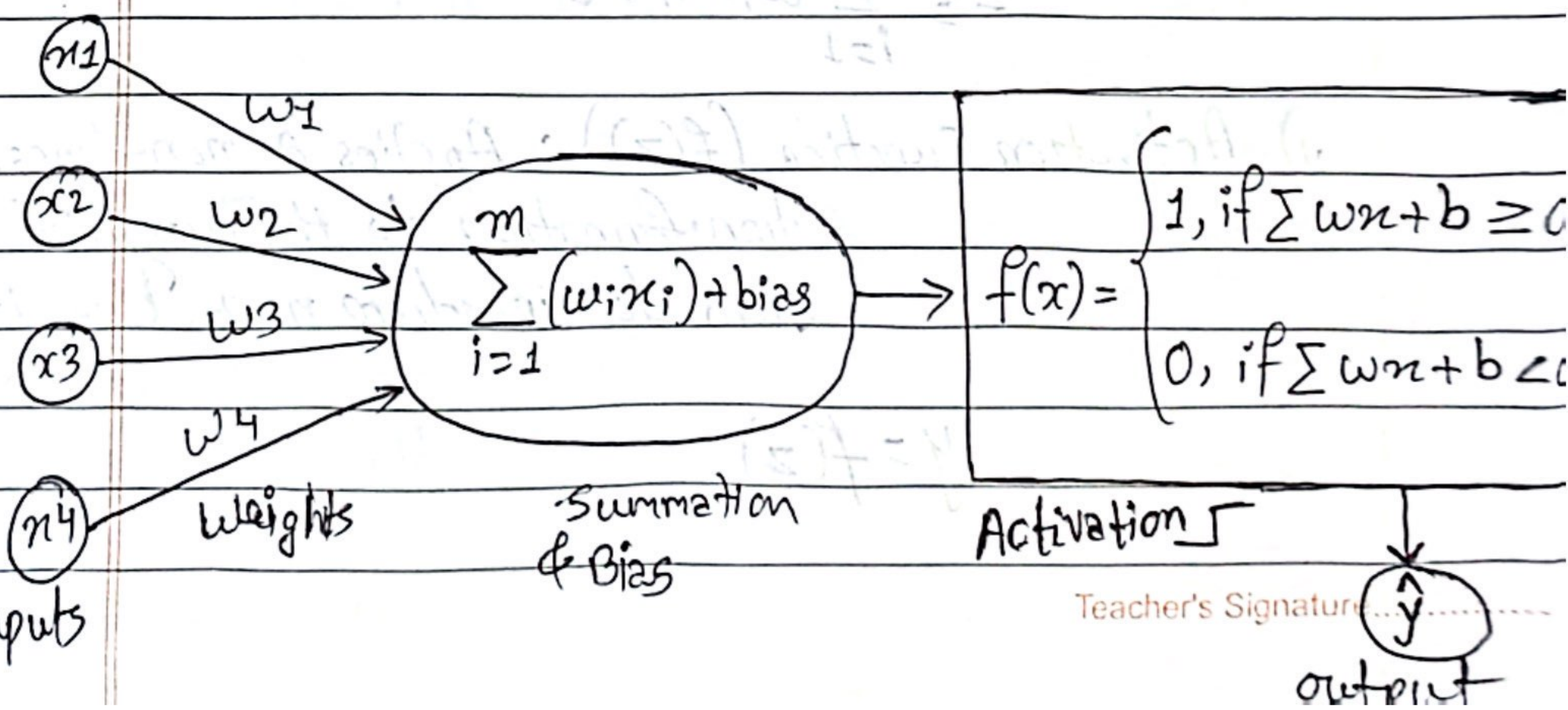
- ReLU (Rectified Linear Unit):

$$f(z) = \max(0, z)$$

- Tanh:

$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

→ Neural Networks is a collection of interconnected neurons organized into layers, designed to process and learn from data.



Teacher's Signature

Date
Page

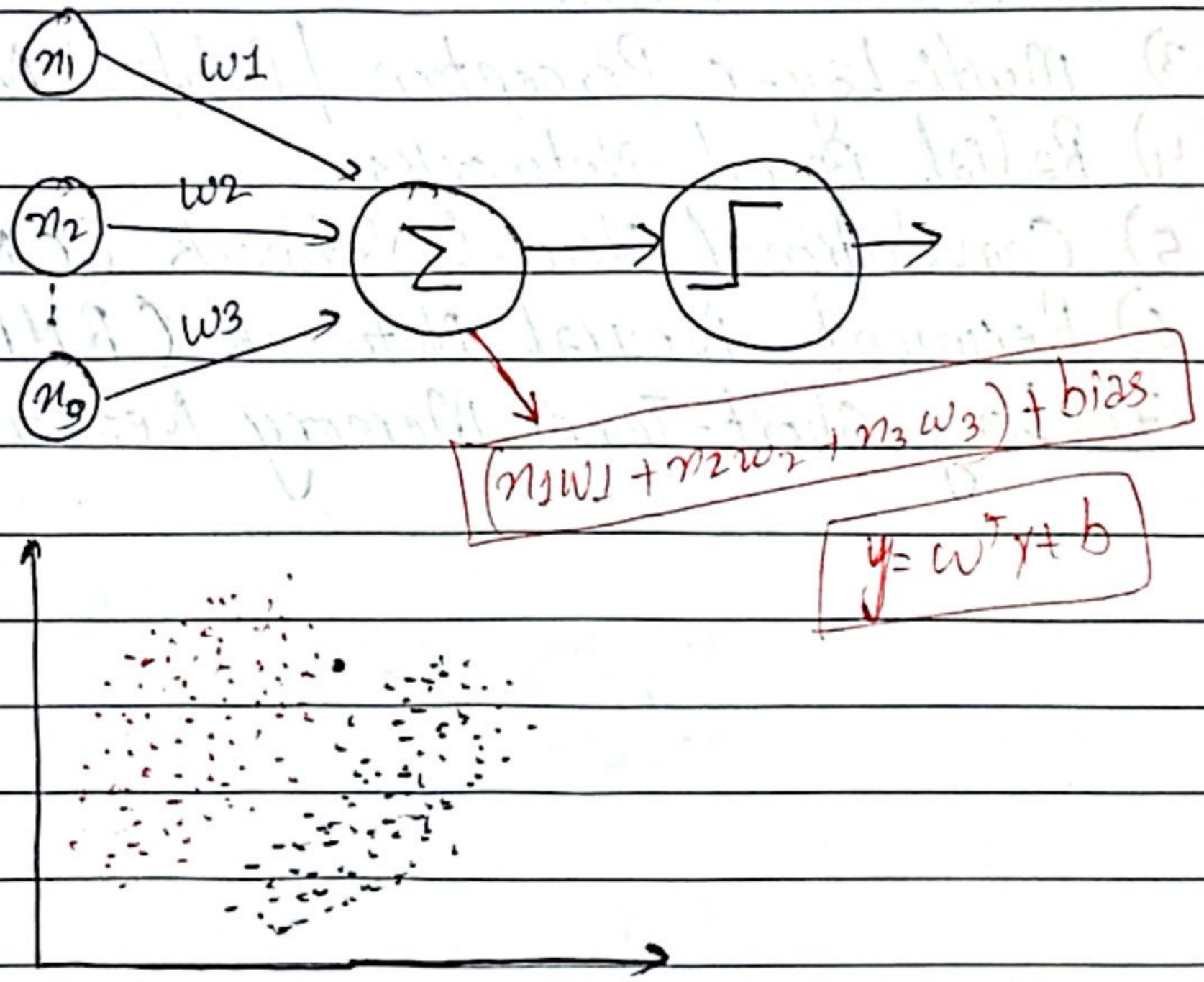
→ Types of Deep learning Networks :-

1) Perceptron
2) Feed Forward Network
3) Multi-Layer Perceptron / Artifical Neural Network
4) Radial Based Networks
5) Convolutional Neural Network (CNN)
6) Recurrent Neural Network (RNN)
7) Long Short-Term Memory Network

Teacher's Signature. .....

Perceptron:

→ It is Single layer perceptron.



* practical

```
import pandas as pd
import matplotlib.pyplot as plt.
import seaborn as sns
from mlxtend.plotting import plot_decision_regions

dataset = pd.read_csv(r"C:\MLcourse\placement_data.csv
```

Date
Page

```
sns.scatterplot(x="cgpa",y="score",data=dataset,
plt.show()                                hue="selection")

x= dataset.iloc[:,:-1]
y= dataset["selection"]

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=48)

from sklearn.linear_model import Perceptron

pr.fit(x_train, y_train)
pr.score(x_train,y_train)*100, pr.score(x_test,y_test)*10

plot_decision_regions(x.to_numpy(), y.to_numpy(),
                                df=pr)
plt.show()
```

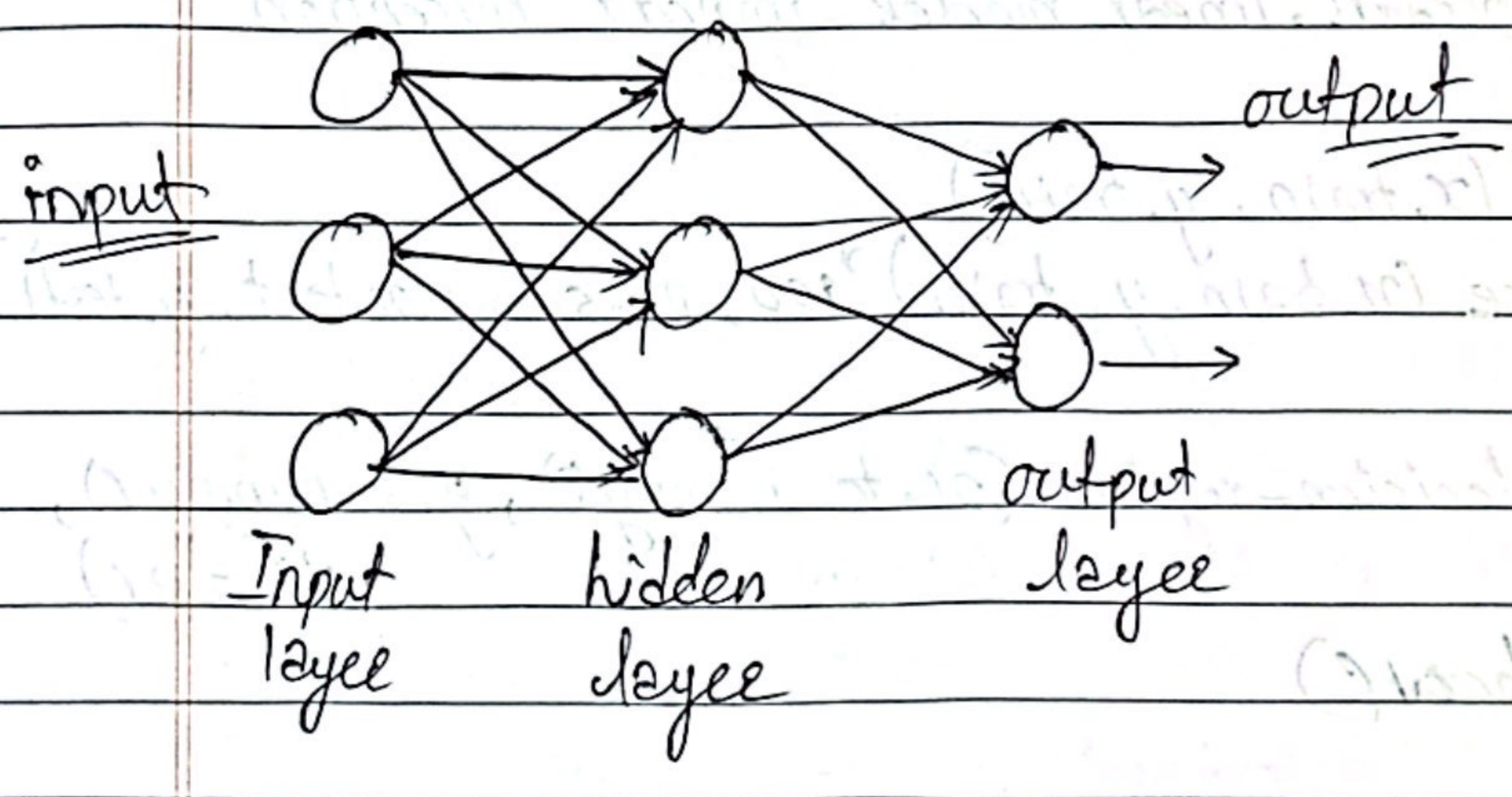
Teacher's Signature. .....

Multilayer Perceptron:

→ It is also called ANN (Artificial Neuron Network)

→ If we have non-linear separable data, single layer perceptron won't show correct prediction, so, we use multi-linear perceptron

→ In ANN, we add multiple single-layer to make multi-layer neural network so we can use it for prediction of non-linear separable data.

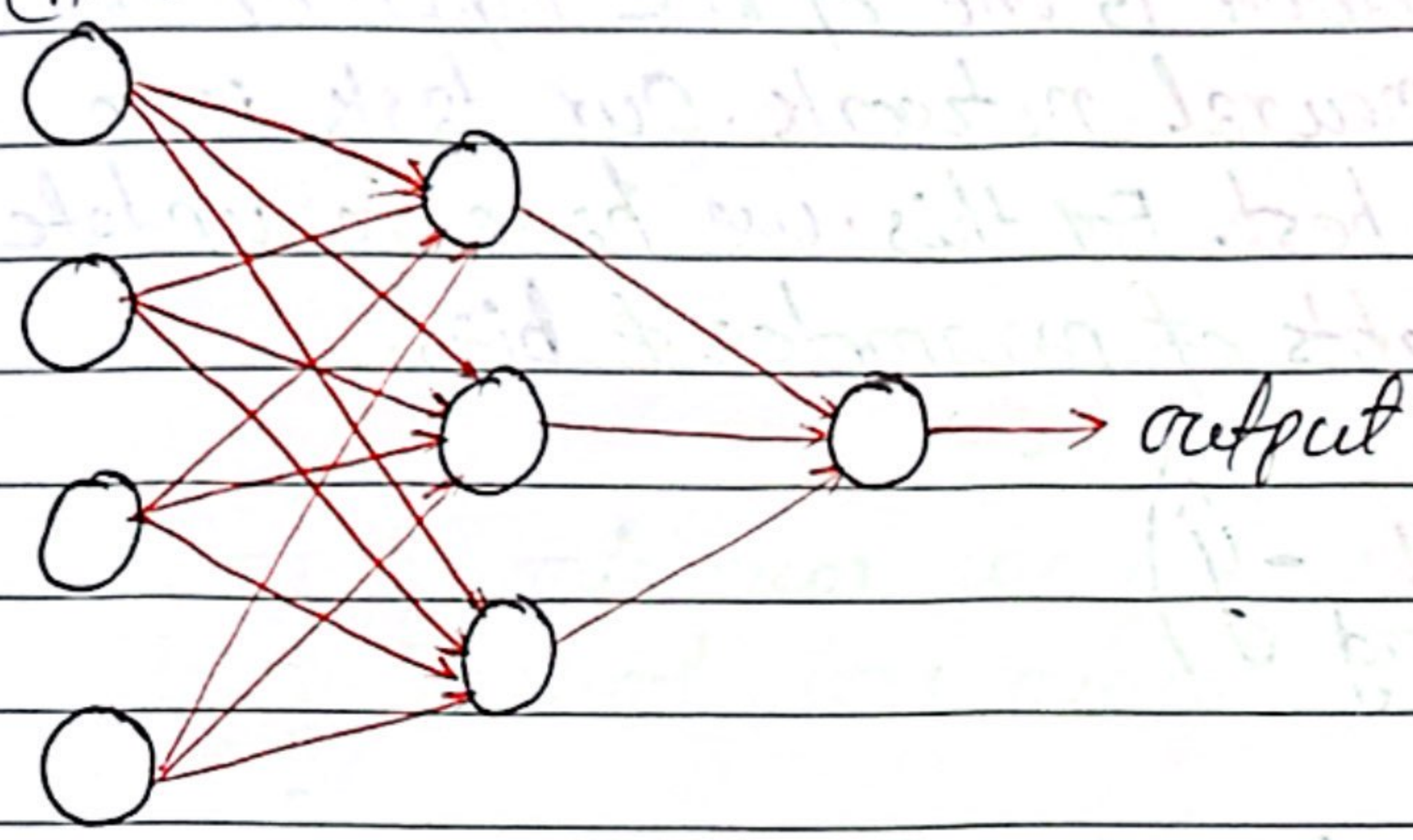


→ At first inputs are multiplied with weights & then product of inputs & weights are added with bias, then this summation is pass through a activation function, After this we get ~~out~~ our output.

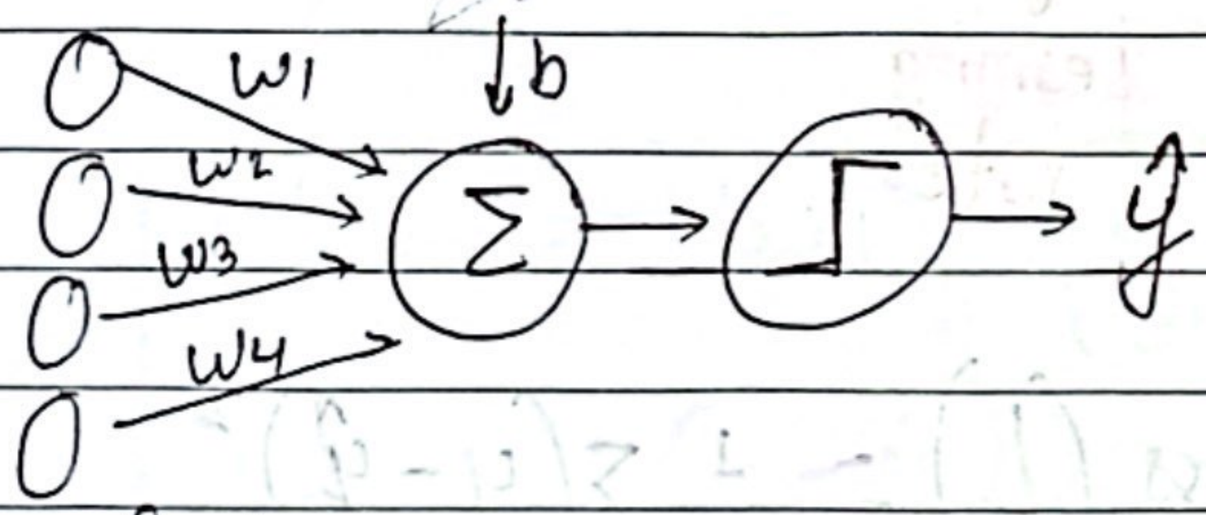
Date
Page

* Forward Propagation :-

(ANN)



$$\hat{y} = \sigma(w^T \cdot X + b)$$ → data are in matrix form



→ At first we assign weights' & bias' value. This is forward propagation.

→ If we need higher accuracy, we need to update ~~weigth~~ weights' & bias' value, this process is back propagation.

→ ~~To minimize the~~

→ To minimize the loss, use gradient decent.

Teacher's Signature. .....

Date
Page

* Back Propagation

→ Backpropagation is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter & bias.

→ $Loss = (y_{org} - \hat{y})$

→ gradient descent:

$$w_{new} = w_{old} - \lambda \left(\frac{dL}{dw}\right)$$

Learning rate

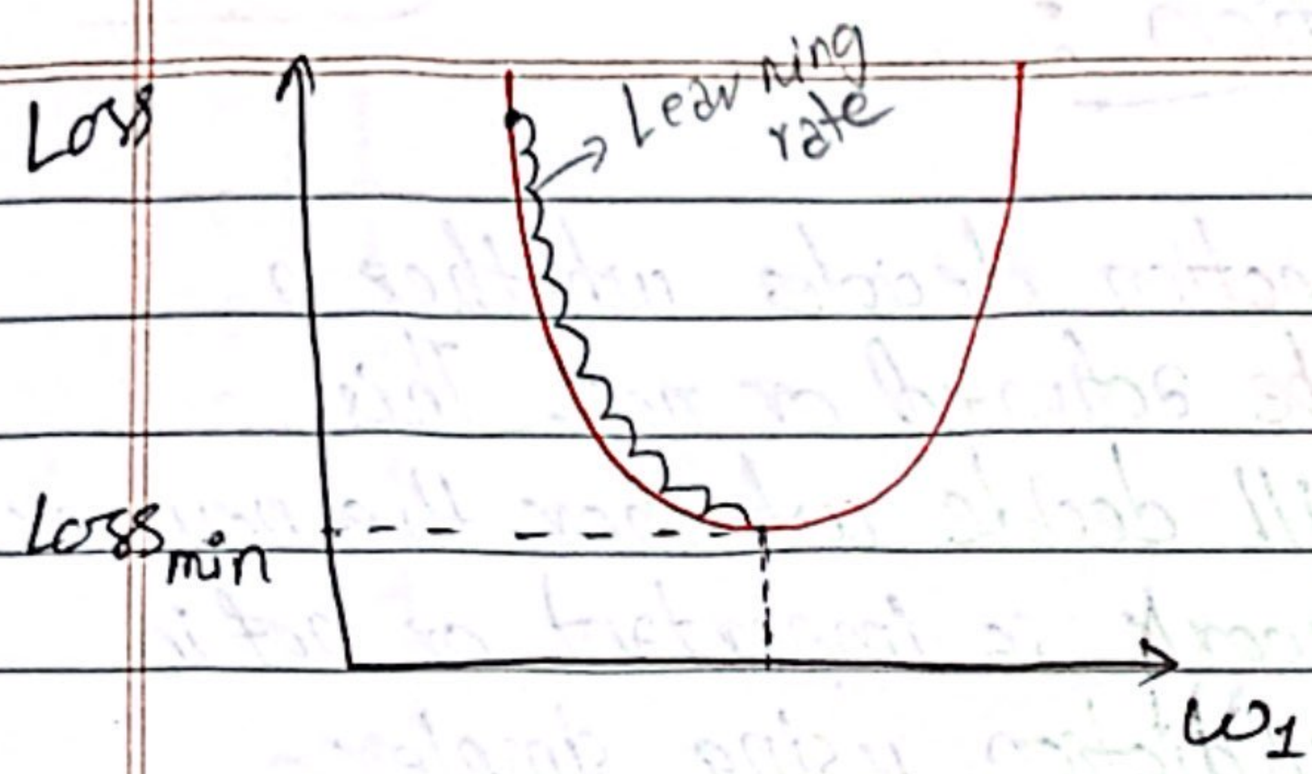
Mean Square loss $(L) = \frac{1}{n} \Sigma (y - \hat{y})^2$

where, $\hat{y} = w_1 x_1 + w_2 x_2 --- w_n x_n + b$

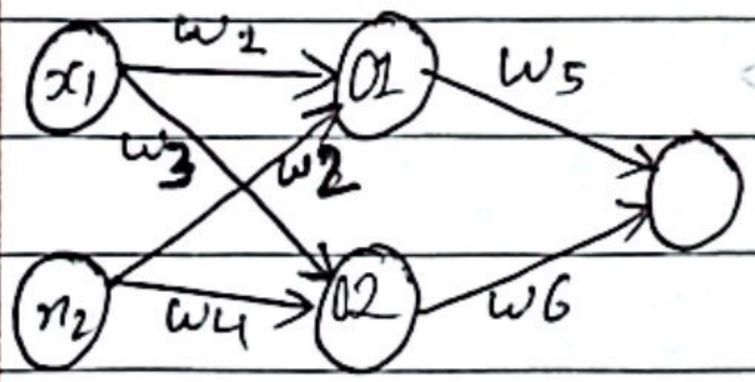
$$\left(\frac{dL}{dw_1}\right) = \frac{1}{n} \Sigma (y - w_1 x_1 - w_2 x_2 - b)^2 (-x_1)$$

Teacher's Signature.....

Scanned with CamScanner



→ If loss is minimum we get accurate value of w_1. To find loss minimum we use gradient.



$$\boxed{\begin{aligned} w_{5new} &= w_{5old} - \lambda\left(\frac{dL}{dw_5}\right) \\ L &= \frac{1}{n}\Sigma(y - \hat{y}) \end{aligned}}$$

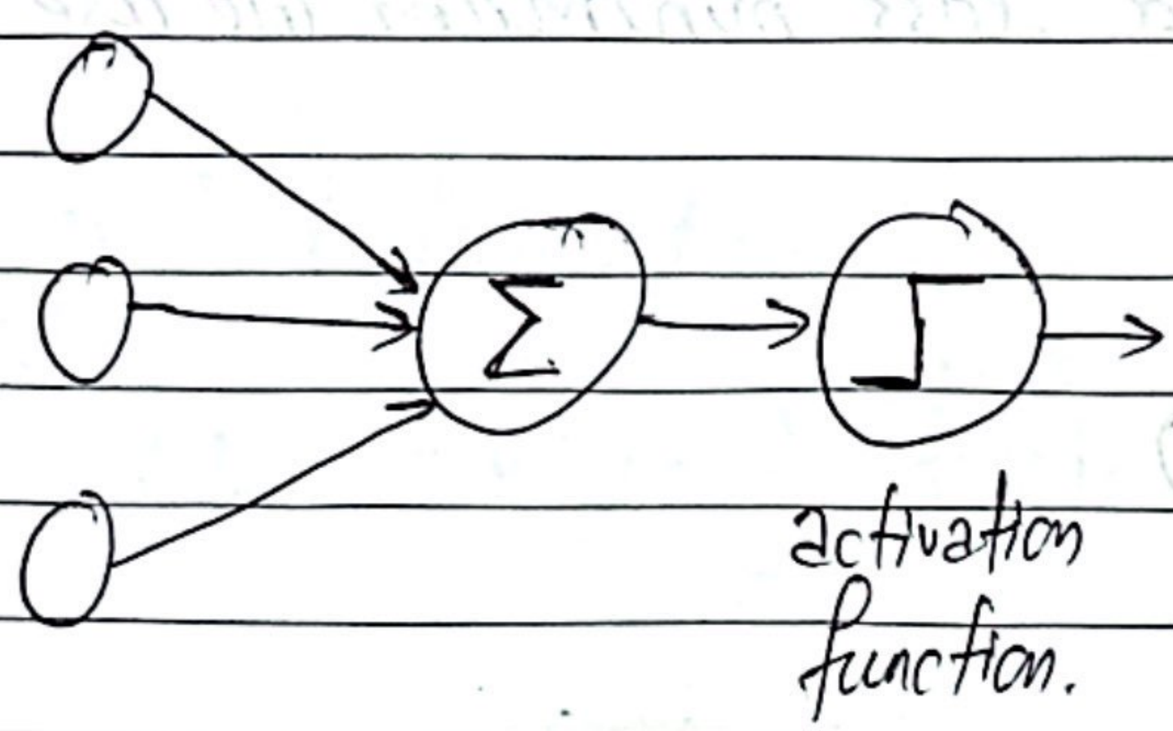
$$\hat{y} = (O_1 w_5 + O_2 w_6 + b_3)$$

$$\hat{y} = (x_1 w_1 + x_2 w_2 + b_1) w_5 + (x_1 w_3 + x_2 w_4 + b_2) w_6 + b$$

Date
Page

* Activation Function :

⇒ An activation function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.



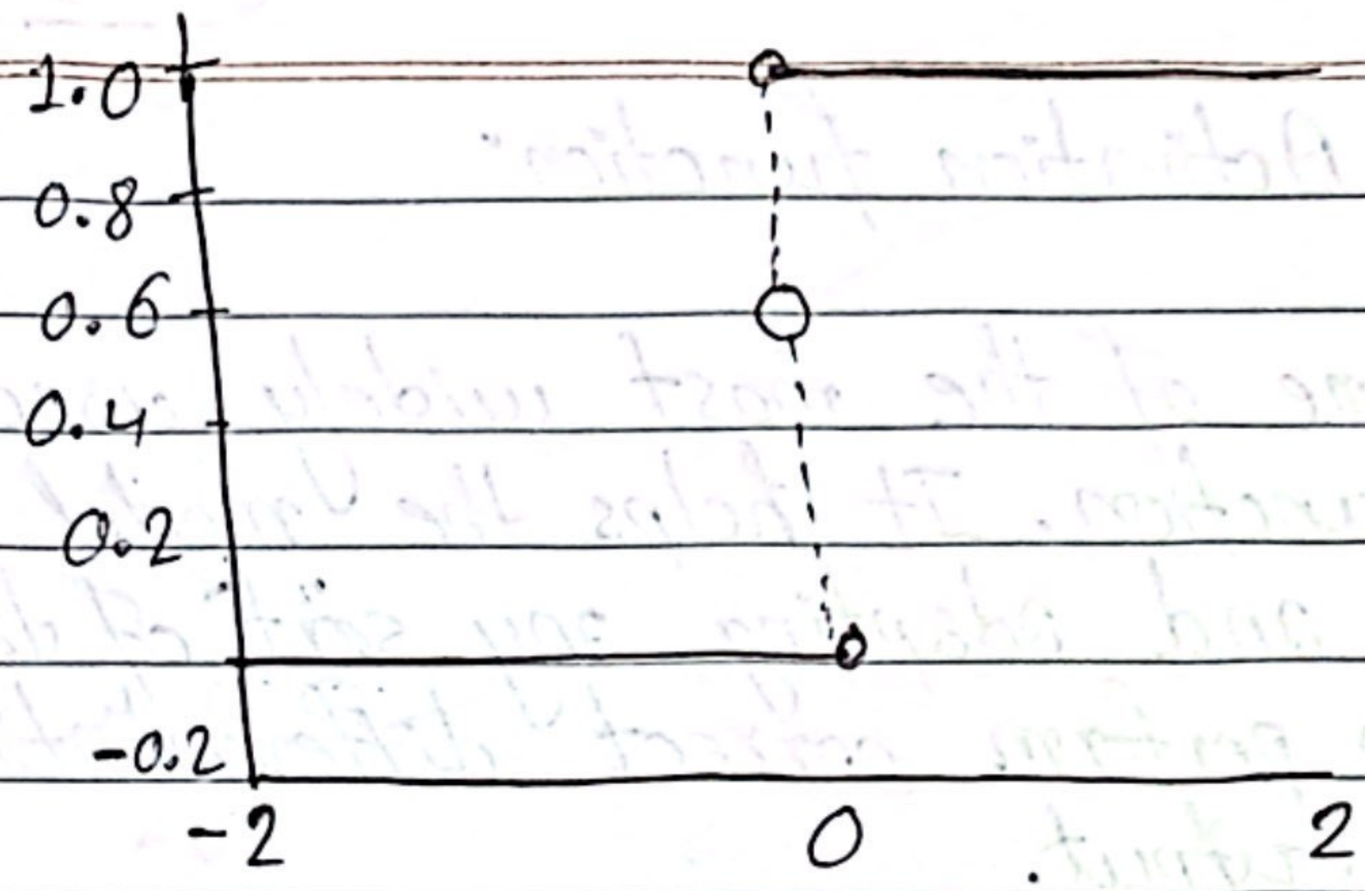
→ Types of Activation Function :-

- Binary Step Function
- Linear Activation Function.
- Non-linear Activation Function.

1) Binary step function :-

⇒ Binary step function depends on a ~~thousand~~ threshold value that decides whether a neuron should be activated or not.

Teacher's Signature.....



$$\hat{y} = x_1 w_1 + x_2 w_2 + b$$

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

2) Linear Activation Function:

→ It is used in regression output.

→ No changes in output, use Linear Activation Function.

→ The output of functions is not restricted in between any range. Its range is specified from -infinity to infinity.

Linear

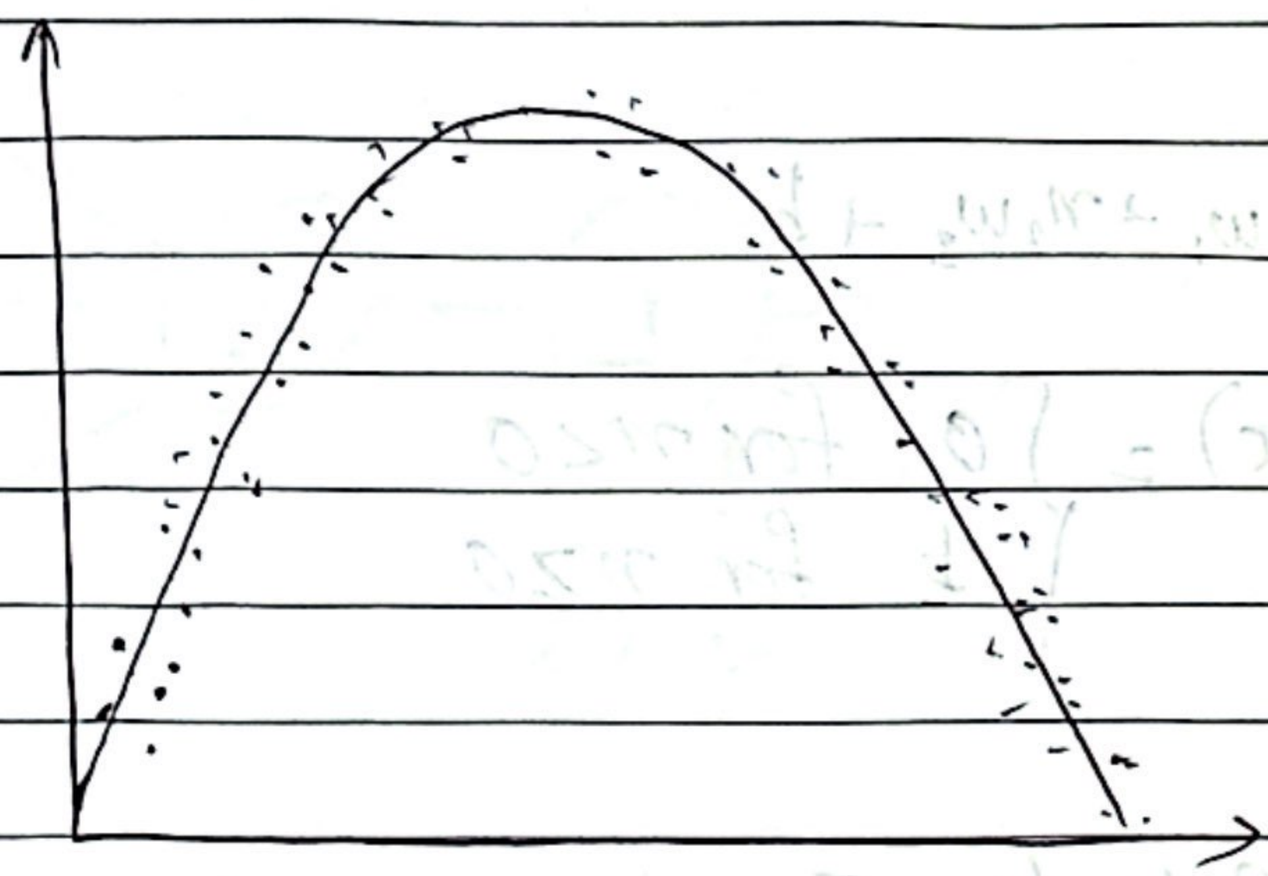
$$f(x) = x$$

Teacher's Signature. .....

Date
Page

3) Non-Linear Activation function:

⇒ These are one of the most widely used activation function. It helps the model in generalizing and adapting any sort of data in order to perform correct differentiation among the output.



→ Non-Linear Neural Networks Activation Functions.

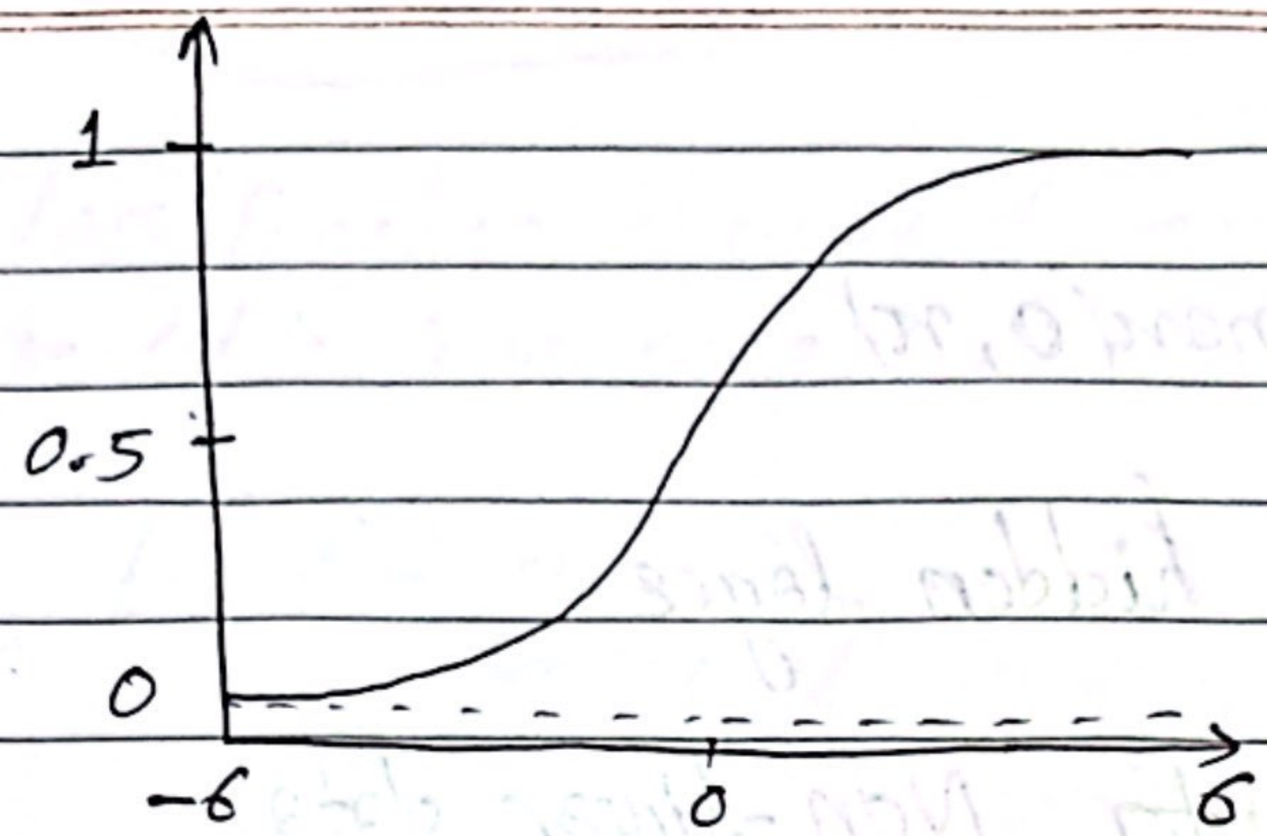
a) Sigmoid / Logistic Activation function
b) Tanh Function (Hyperbolic Tangent)
c) ReLU Function.
d) Softmax Function.

a) Sigmoid Function:-

$$f(x) = \frac{1}{1+e^{-x}}$$; desired output → 0, 1

Teacher's Signature.....

Date
Page



→ Can be used in RNN
→ It is differential activation function.
→ It is not zero centered.
→ It is used in multilable classification.

b) Tanh Activation function:

→ It is used in RNN.
→ output [-1 to 1]
→ It is differentiable and zero centeed as well.

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

c) Softmax Function:
→ It is used in multiclass classification.
→ It is used in categorical output

→ $$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Teacher's Signature. .....

Date
Page

d) ReLU Function:-

→ $f(x) = \max(0, x)$

→ It is used in hidden layer.

→ It is used for Non-linear data
→ It is differentiable.
→ It is used in CNN

Teacher's Signature.....

Date
Page

* Loss Function:

→ Loss Function is used to measure how our model is inaccurate.

$$Loss = y_{org} - \hat{y}_{prd}.$$

$$loss\ function = (y_{org} - \hat{y}_{prd})^2$$

→ The loss function is a method of evaluating how well your algorithm is modeling your dataset. It is mathematical function of the parameters of the machine learning algorithm

→ Loss Function:

1) Regression:
- MSE (Mean Square Error)
- MAE (Mean Absolute error)
- Hubber loss.

2) Classification:
- Binary cross-entropy.
- Categorical cross-entropy

Teacher's Signature. .....

Date
Page

3) Auto Encoder:
- KL Divergence.

4) GAN:
- Discriminator Loss
- Minmax GAN loss.

5) Object detection:
- Focal loss

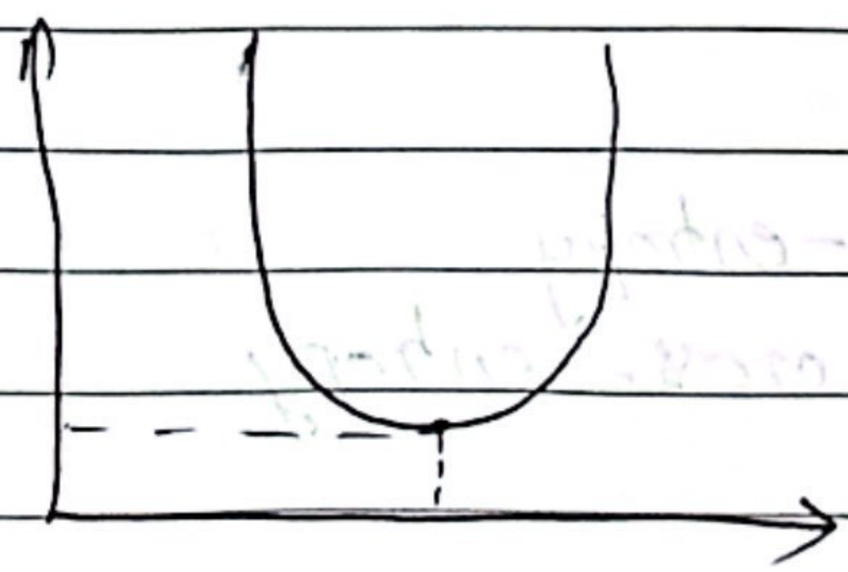
6) Word embeddings:
- Triplet loss

→ Explanation:

1) Regression Loss

a) Mean Squared Error / Squared loss / L2 loss:

→ $$MSE = \frac{1}{N} \sum_{i}^{N} (Y_i - \hat{Y}_i)^2$$



Teacher's Signature.....

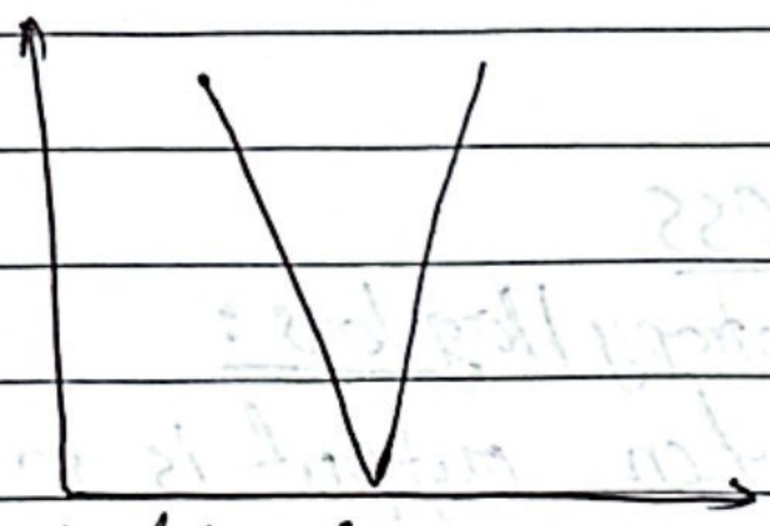
Date
Page

→ It shouldn't use when data has outliers

b) Mean Absolute error / L1 loss:

→ It is prefered to use MAE when outliers are present.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |Y_i - \hat{Y}_i|$$



→ Not differentiable loss or equation.

c) Huber loss:-

→ $$Huber = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(y_i - \hat{y}_i)^2 \qquad |y_i - \hat{y}_i| \leq \delta$$

$$Huber = \frac{1}{n} \sum_{i=1}^{n} \delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) \quad |y_i - \hat{y}_i| > \delta$$

Teacher's Signature. .....

Date
Page

n = the number of data points.

Y = the actual value of the data point. Also Known as true value

$\hat{Y}$ = the predicted value of the data point. This value is returned by the model

δ = defines the point where the Huber loss function transitions from a quadr

2) Classification Loss

(a) Binary Cross Entropy / log loss:

→ It is used when output is in binary format. i.e. output is in two form.

→ $$\text{log loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right)$$

(b) Categorical cross entropy:

⇒ When output is in categorical form.

$$\text{Loss} = -\sum_{j=1}^{K} y_i \log(\hat{y}_i)$$

Teacher's Signature.....

Date
Page

→ If the target column has numerical encoding to classes like 1, 2, 3, 4, - - - N then use sparse categorical cross entropy.

Teacher's Signature. .....

Date
Page

* Optimizer in Neuron Network :

⇒ Optimizer are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.

→ Different Optimizers :

a) Gradient Descent.
b) Stochastic Gradient Descent
c) Stochastic Gradient descent with momentum
d) Mini Batch Gradient Descent.
e) Adagrad.
f) RMSProp
g) AdaDelta
h) Adam

Teacher's Signature.....

Date
Page

* practical (ANN)

```
import pandas as pd
dataset = pd.readcsv("Churn_Modelling.csv")
dataset.isnull().sum()
```

no null value

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset["surname"]=le.fit_transform(dataset["surname"])
dataset["Geography"]=le.fit_transform(dataset["Geography"])
dataset["Gender"]=le.fit_transform(dataset["Gender"])

input_data = dataset.iloc[:, :-1]
output_data = dataset["Exited"]

from sklearn.preprocessing import StandardScaler
ss = StandardScalar()
input_data = pd.DataFrame(ss.fit_transform(input_data),
                          columns = input_data.columns)
inputdata.shape
```

(10000, 13)

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(inputdata,
                                output_data, test_size=0.2,
                                random_state= 42)

import tensorflow
```

Teacher's Signature. .....

Date
Page

```
from Keras.layers import Dense
from Keras.models import Sequential

ann = Sequential()

ann.add(Dense(11, input_dim=13, activation='relu'))
ann.add(Dense(9, activation='relu'))
ann.add(Dense(7, activation='relu'))
ann.add(Dense(5, activation='relu'))
ann.add(Dense(3, activation='relu'))
ann.add(Dense(1, activation='sigmoid'))

ann.compile(optimizer='adam', loss='binary_crossentropy',
            metrics=['accuracy'])

ann.fit(x_train, y_train, batch_size=100, epochs=50)

prd = ann.predict(x_test)

prd_data = []
for i in prd:
    if i[0] > 0.05:
        prd_data.append(1)
    else:
        prd_data.append(0)
```

Teacher's Signature.....

Date
Page

```
prd1 = ann.predict(x_train)

prd_data1 = []
for i in prd1:
    if i[0] > 0.5 :
        prd_data1.append(1)
    else :
        prd_data1.append(0)

from sklearn.metrics import accuracy_score

accuracy_score(y_test, prd_data) * 100
85.9
accuracy_score(y_train, prd_data1) * 100
86.4
```

Teacher's Signature. .....

Date
Page

Improve the performance of Neural Network:

→ i) Train Hyperparameters ⇒ hidden layer, node, activation function, loss function, optimizers, batch size, no. of epochs

ii) Vanishing / Exploding gradient

iii) Data
↳ Transfer learning

iv) Slow training

v) Overfitting

Identify Overfitting in DL:

⇒ Training. Score > Testing. Score ⇒ overfitting

Training. Score ≈ Testing. score ⇒ best fitting.

→ Overfitting is a common explanation for the poor performance of a predictive model.

→ Overfitting refers to an unwanted behaviour of a machine learning algorithm used for predictive modeling.

Teacher's Signature.....

Scanned with CamScanner

Date
Page

→ To Reduce overfitting:
- Cross-validation
- Train with more data
- Remove features
- Early Stopping
- Regularization
- Ensembling
- Hyperparameter

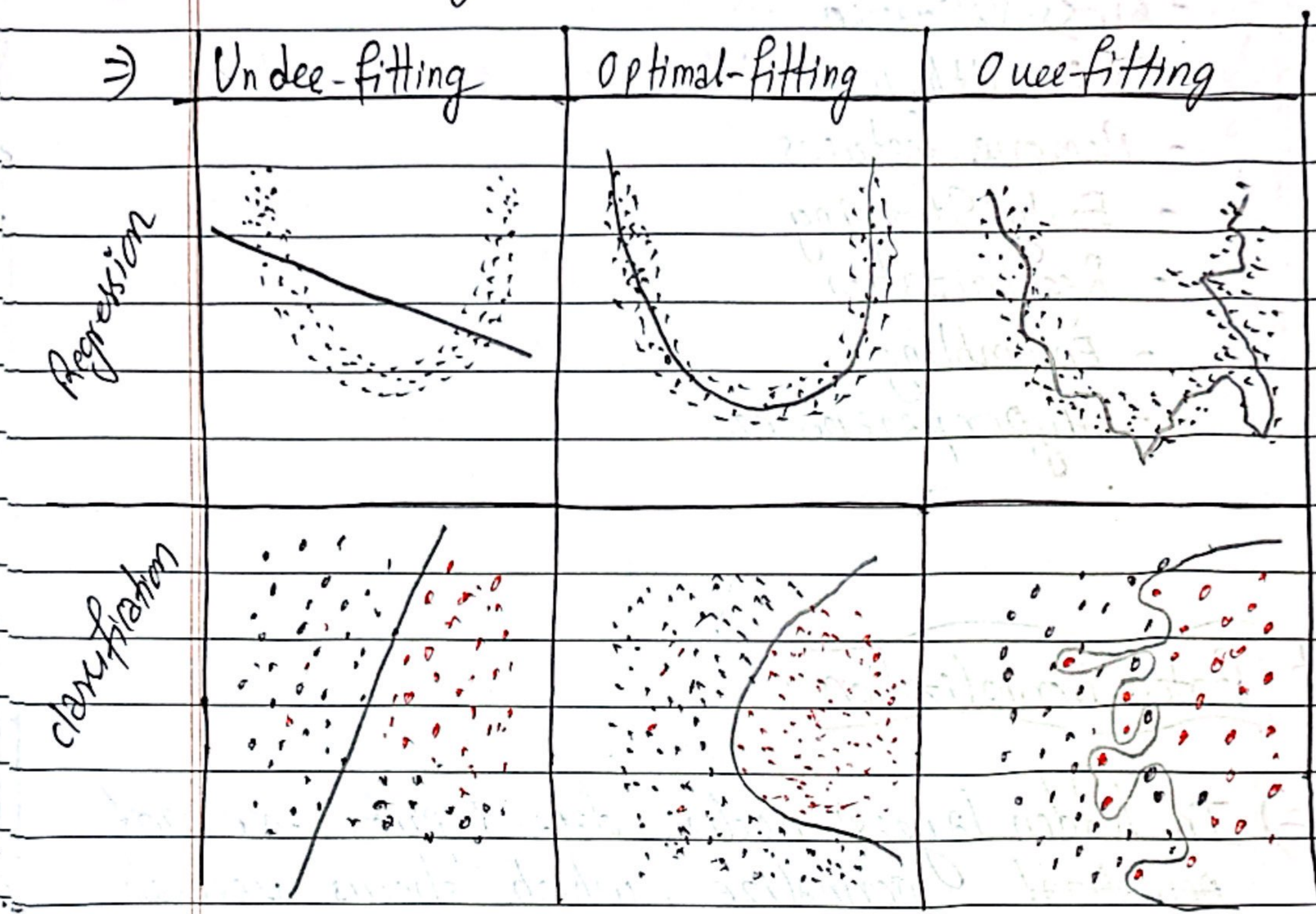
Batch Normalization:

⇒ In hidden layers' nodes, data inputs are not ~~normalized~~ normalize, which shows vanish gradient decent problem. To solve this problem we use Batch Normalization.

Teacher's Signature. .....

Date
Page

DropOut Layer:



→ All the forward & backwards connections with a dropped node are temporarily removed thus creating a new network architecture out of the parent network. The nodes are droped by a dropout probability of p.

Teacher's Signature.....

Date
Page

Vanishing Gradient Problem:

→ The vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods and backpropagation. In such methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight.

Solutions:

* Proper Weight Initialization
* Using Non-saturating Activation Functions
* Batch Normalization
* Gradient Clipping

Teacher's Signature. .....

Date
Page

Hyperparameter Tunning :

→ By changing the parameter's values, we can minimize overfitting.

Scanned with CamScanner

Convolutional Neural Network:

Date
Page

→ It is use for image classification.

→ CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep learning thus recognizes objects in an image by using a CNN.

→ At first image is converted into numbers in ANN. But accuracy is very low in ANN.

Input → convolutional layer → pooling layer → convolutional layer
Fully convolutional (FC) ← pooling layer ←

Color → (Red, Green, Blue)
1 pixel range → (0 - 255)

* How does it work?

→ Convolutional → Max pooling → Flattening → Full Convolutional

Teacher's Signature: .....

Convolutional, pooling, Flattening:

⇒ Image → Convolutional Layer → Pooling → Flattening

* Convolutional:

→ Now in mathematics convolutional is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

$$c = (f * g)(t) = \int_0^t f(T)\, g(t-T)\, dT$$

where,

c = convolutional output function
f = original input function.
g = function that is shifted over the input function.
t = variable representating range of shifting
T = shifting against t.

Teacher's Signature.....

Date _____
Page _____

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0
0	1	0	0	0

Input data

1	0	1
0	1	0
1	0	1

Kernel

4	3	4

Convoluted feature/ feature map

$1 * 1 = 1$

$0 * 0 = 0$

$0 * 1 = 0$

$1 * 0 = 0$

$1 * 1 = 1$

$0 * 0 = 0$

$1 * 1 = 1$

$1 * 0 = 0$

$+ \; 1 * 1 = 1$

4

* Pooling:

⇒ A pooling layer in another building block of CNN. It's function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network.

Teacher's Signature. .....

Date
Page

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

feature Map

$\xrightarrow[\text{Pooling}]{\text{Max}}$

pooled Feature Map

→ Max pooling
→ Avgeerage pooling.

* Flattening:

→ Flattening is converting the data into a 1-dimensional array for inputting it to the next layer.

→ Flatten output of the convolutional layers to create a single feature vector.

→ It is connected to the final classification model which is called a fully-connected layer.

Teacher's Signature.....

Date _____
Page _____

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y
1	1	0	4	2	1	0	2	1	cat

1	1	0
4	2	1
0	2	1

* Practical (CNN)

```
import tensorflow

from Keras.layers import Dense, Conv2D, Max Pooling2D,
from Keras. models import Sequential        Flatten

cnn = Sequential()

cnn. add (Conv2D(32, (3,3), input-shape =(64, 64, 3),
                                activation = 'relu'))
cnn. add (MaxPooling 2D (pool_size = (2, 2)))
cnn. add ((Conv2D (32, (3,3), activation = 'relu'))
cnn. add (MaxPooling 2D (pool_size = (2, 2)))
cnn. add (Flatten ())
```

Teacher's Signature. .....

Date
Page

```
cnn.add(Dense(64, activation='relu'))
cnn.add(Dense(32, activation='relu'))
cnn.add(Dense(16, activation='relu'))
cnn.add(Dense(8, activation='relu'))
cnn.add(Dense(4, activation='relu'))
cnn.add(Dense(1, activation='sigmoid'))

cnn.compile(loss='binary_crossentropy', optimizer=
                                        'adam')
from Keras.preprocessing.image import ImageDataGener

train_datagen = ImageDataGenerator(
                rescale=1./255,
                shear_range = 0.2,
                zoom_range = 0.2
                horizontal_flip = True)
test_datagen = ImagedataGenerator (rescale = 1./255)
train_generator = train_datagen.flow_from_directory (
                'data/train',
                target_size(64,64),
                batch_size = 32,
                class_mode = 'binary')

test_generator = test_datagen.flow_from_directory (
                'data/test',
                target_size = (64, 64)
                batch_size = 32
                class_mode = 'binary')
```

Teacher's Signature.....

Date
Page

```
model.fit_generator (
        train_generator,
          steps_per-epoch = 200,
           epoch = 5,
           validation-data = test-generator)


from Keras.preprocessing import image

img = image.load-img(r"image.jpg", target-size=(64,64)

img = image.img-to-array (img)

import numpy as np.

img = np.expand_dims (img, axis=0)

p = cnn.predict (img)

if p[0][0] < 0.5 :
      print ("dog")
else:
      print ("cat")
```

Dog.

Teacher's Signature. .....