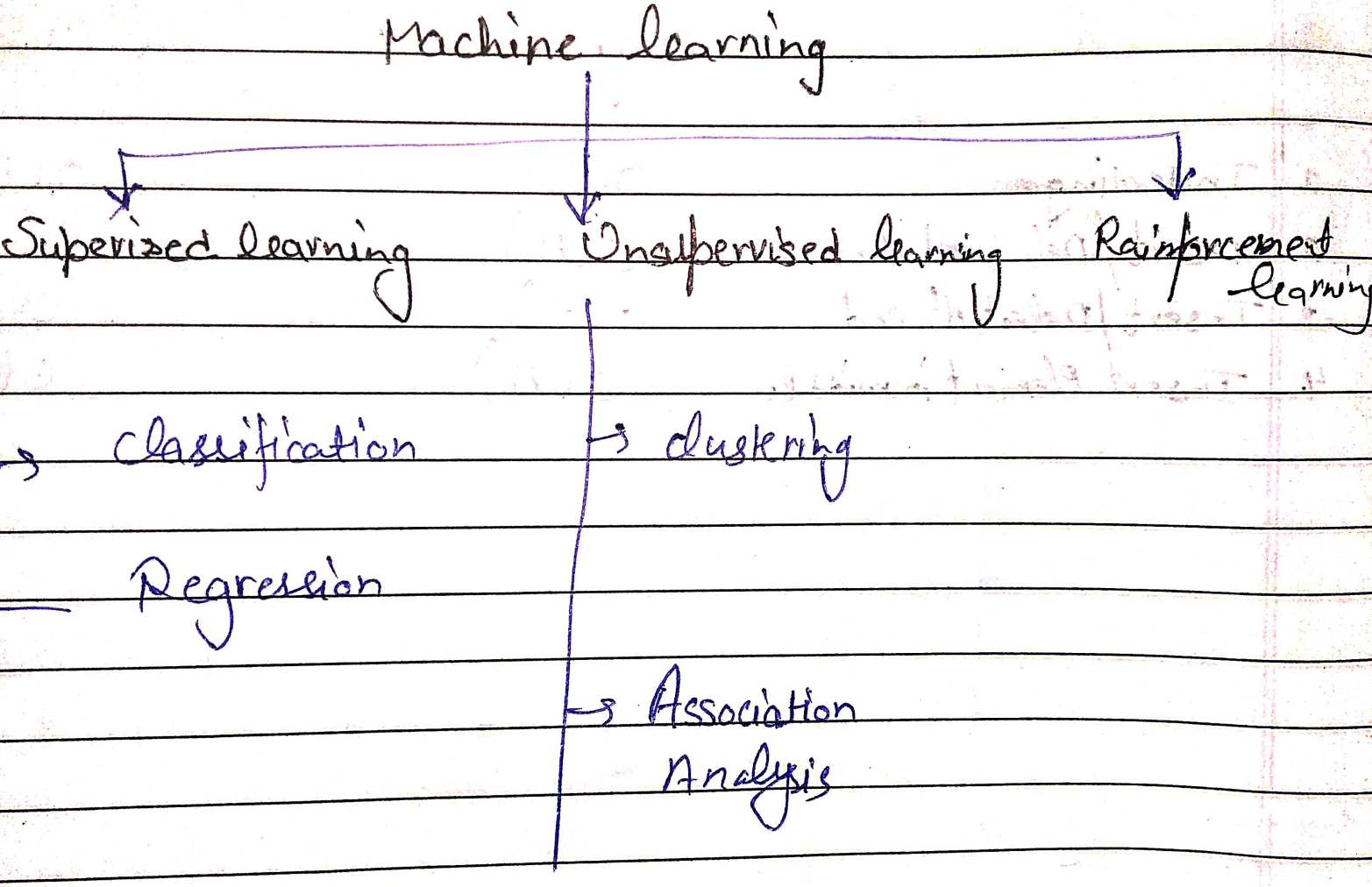
What is Machine learning?

The field of study that gives computer the ability to learn without being explicitly programmed

A computer program is said to learn from experience E with respect to task t and performance measure P


Machine learning
Supervised learning
Unsupervised learning
Reinforcement learning
Classification
Regression
Clustering
Association Analysis

Supervised

class or labels are available

labelled traning data is needed

- Naive Bayes
- K-nearest neighbour
- Decision tree
- Linear regression
- Logistic regression
- support vector machine etc

Unsupervised

there is no idea about the class and lables of a Particular data i.e the modle has find patterns in the data

the unlabelled data set is given to the model

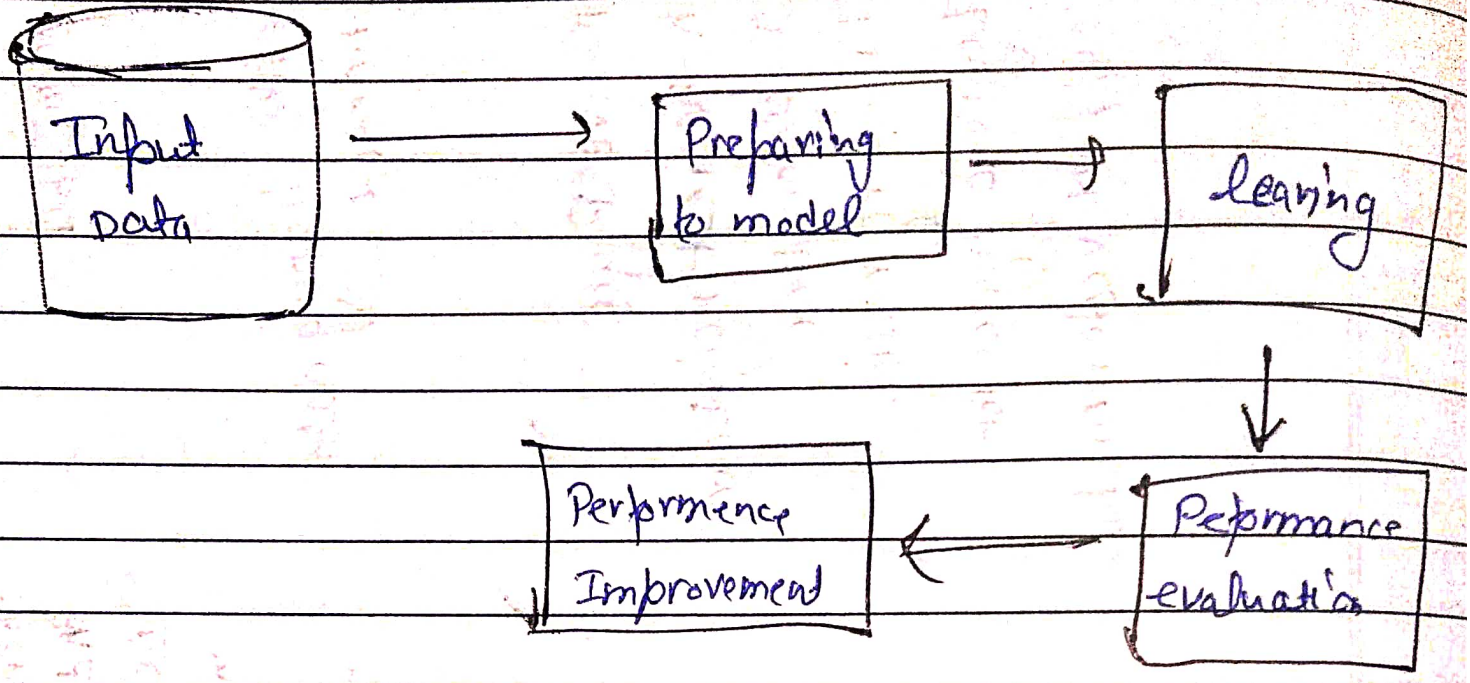
- K-mean
- PCA (Principal Component Analysis)
- SOM
- Apriori algorithum
- DBSCAN etc.

Reinforcement

there is also idea abat lables, if the modle output corred it get reward, othwise no

the model learn itself and update through reward / punishment.

- Q. learning.
- Sarsa



Step 1

(Preparing to Model)

Understand the type of Data,
Nature of Data quality
inter-feature relationship
Find potential issue in Data
Remediate data if needed
Apply pre-process steps :-

- ✓ Dimensionality reduction
- ✓ Feature subset selection

Step 2

(learning)

- Split Data into test-train
- Model Selection
- Cross-validation.

Step 3:-

(Performance evaluation)

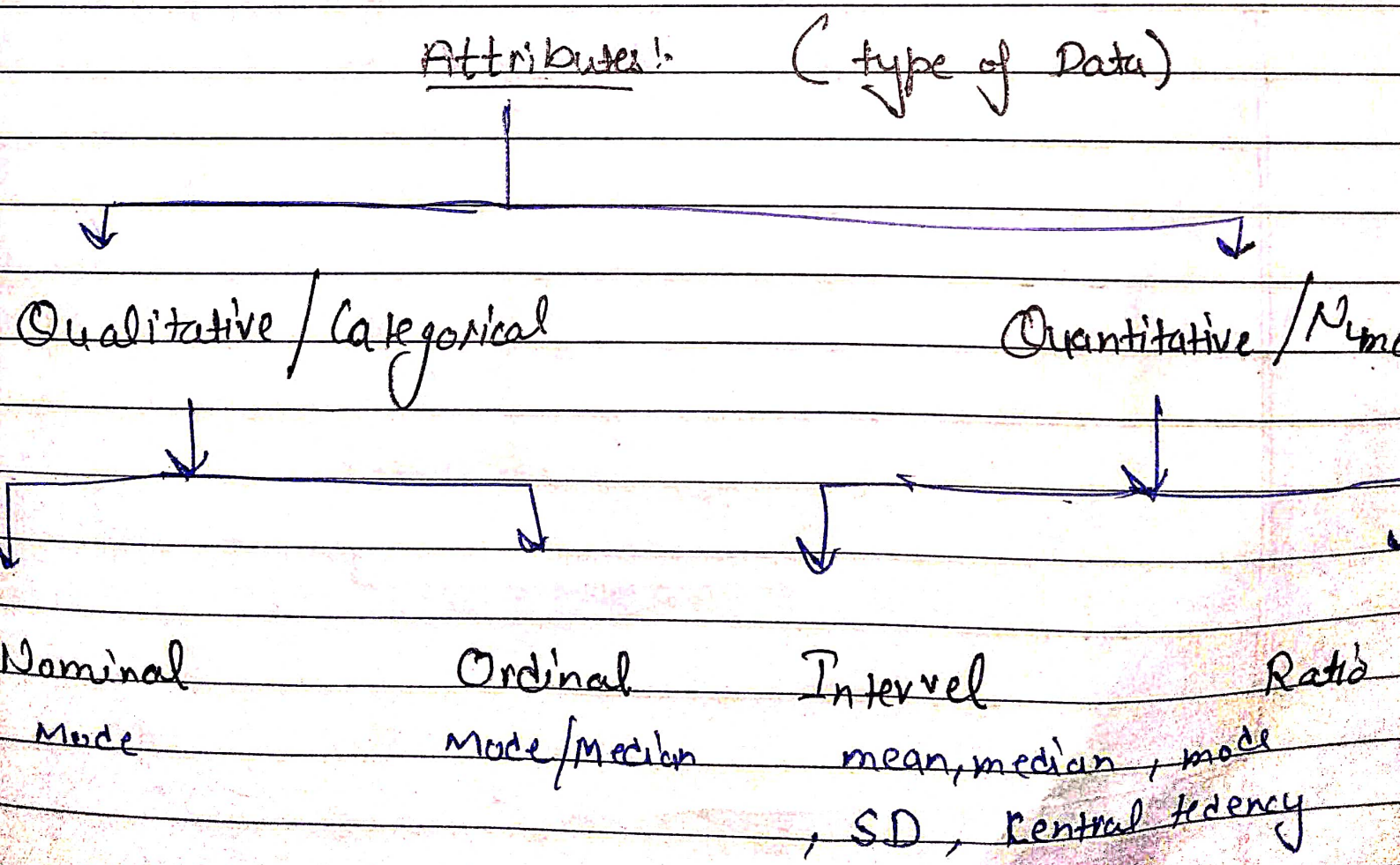
Examine the model performance.

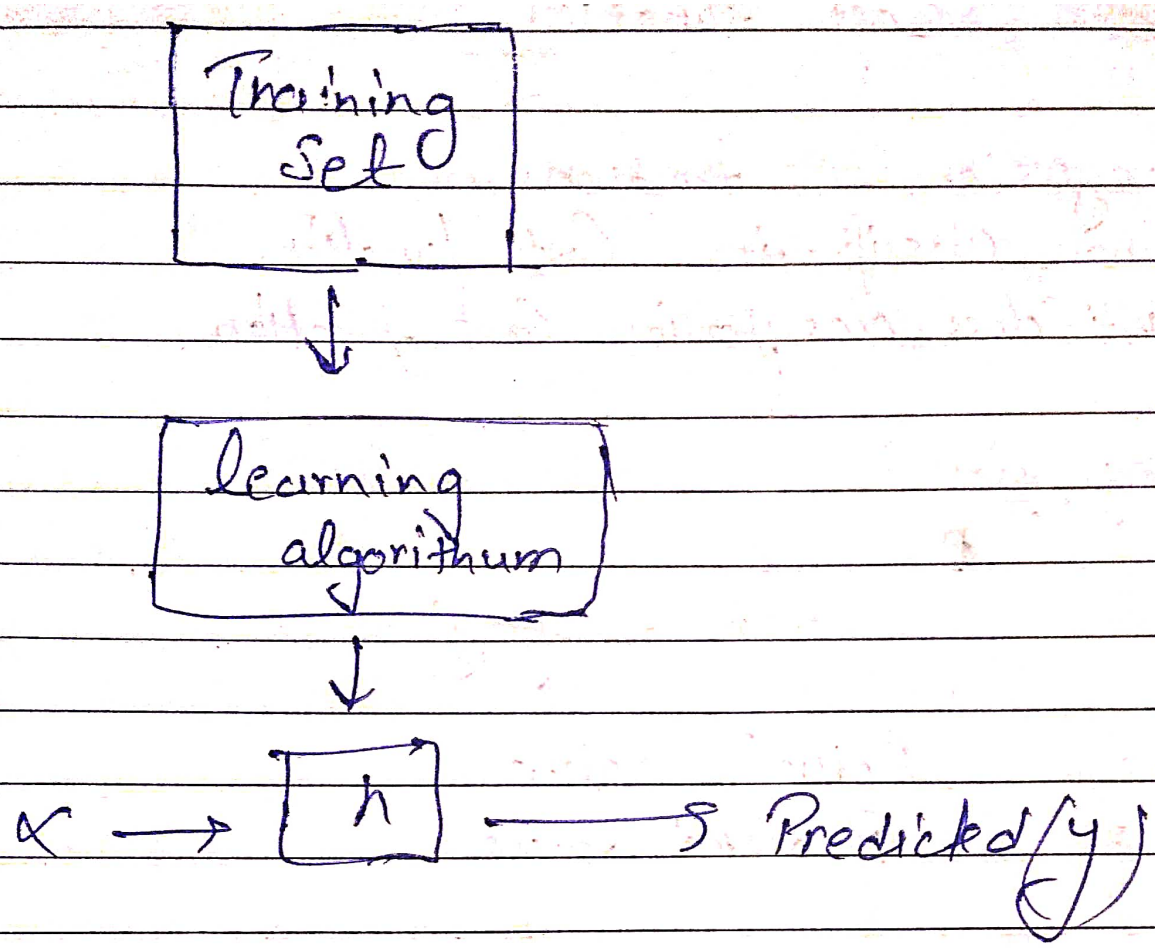
Visualize performance trade-offs using ROC curves.

Step 4:-

(Performance improvement)

- Tuning the model
- Ensembling
- Bagging
- Boosting





Cost function:

is a measure of how wrong the model is in terms of its ability to estimate the relationship b/w X and Y.

it help the learner to correct/change behaviour to minimize mistake.

Difference or Distance b/w the predicted value & the actual value.

(cost function refers to the loss of error)

Hypothesis Function:-

$$y = mx + c$$

$$h_\theta(x) = \theta_1 x + \theta_0$$

Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{i=m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

ML Gradient Descent:-

So, we have our hypothesis function and we have a way of measuring how well it fit into data. Now we need the estimate the prameter in hypothesis function, That's gradient descent came.

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} (J(\theta_0, \theta_1))$$

$$j = 0, 1$$

Gradient Descent for Linear Regression

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_i$$

Work !

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) x_i)$$

Linear Regression with Multiple Variable

$x_j^{(i)}$ = value of feature in j in the i^{th} training example

$x^{(i)}$ = the column vector of all feature input

m = no° of training examples

n = no° of feature $|x^{(i)}|$

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

θ_0 is the basic price of house.

θ_1 as the price per squre meter

θ_2 as the price per floor

$$h_\theta(x) = [\theta_0 + \theta_1 + \theta_2 + \cdots + \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

Note :- $x_0^{(i)} = 1 \quad (i = 1, \cdots m)$

Gradient Descent for Multivaribles

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_1^{(i)}$$

$$\theta_2 = \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x$$

other words :-

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_j^{(i)}$$

Feature Normalization :

We can speed up gradient descent by having each of our input value in roughly the same range. This is because θ is quickly decent on small range and slowly on large range

Two techniques that are feature scaling & mean normalization

In feature scaling divide the input values by the range

range :- (max value - min value)

mean normalization :- $\frac{x_i - \mu_i}{s_i}$

x_i :- input variable

μ_i :- average value.

s_i :- range (max - min) or standard deviation

Polynomial Regression:

We can combine multiple features into one

$x_3 = x_1 \cdot x_2$

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

Normal Equation:

Normal equation is a method of finding the optimum θ without iteration.

$$\theta = (X^T X)^{-1} X^T y$$

There is no need of feature scaling with normal equation

Gradient Desent	Normal equation
Need to choose alpha	• No need to choose alpha
Need many iteration	• No need to iterate
$O(kn^2)$	• $O(n^3)$ need to calculate $X^T X$
work well as n is large	• slow if n is large

$(X^T X)$ may be non-invertable.

Redundant feature :- When two features are very closely related.

Too many features (eg $m \leq n$) In this cause delete some feature or use (regularization)

m_x = mean of x

m_y = mean of y

$y = mx + c$

$(y - m_y) = b_{yx}(x - m_x)$

$b_{yx} = r \frac{\sigma_x}{\sigma_y}$

$$b_{yx} = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma y^2 - (\Sigma y)^2}$$

take n common

$$b_{yx} = \frac{\Sigma xy - n \cdot m_x \cdot m_y}{\Sigma y^2 - n \cdot m_y \cdot m_y}$$

Suppose : ss

$$b_{yx} = \frac{SS_{xy}}{SS_{xx}}$$

$$y - m_y = b_{yx} \cdot x - (b_{yx} * m_x)$$

$$y = b_{yx} \cdot x - b_{yx} * m_x + m_y$$

$$y = b_{yx} \cdot x + m_y - b_{yx} * m_x$$

b_0 (c) constant

$$y = b_1 * x + b_0$$
$$y = mx + c$$

r = co-relation

$$r = \frac{(n\Sigma(xy)) - (\Sigma(x))(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

$$\Sigma xy - n \cdot m_x \cdot m_y$$

Data Preparing

Data preparing make our dataset more appropriate for ML process.

After collecting raw data most important part are data pre-processing.

Data Pre-process techniques:-

Scaling:-

Most probably our dataset comprises of the attribute with varying scale, but we can't provide such data to ML algorithum hence it require re-scale
Make sure that attributes are at same scale.

```
from sklearn import preprocessing

df  # any data from
array = df.values

data_s1 = preprocessing.MinMaxScaler(feature_range(0,1))
data_s2 = data_s1.fit_transform(array)

Set print option (precision = 1)
```

it is generally used in K-NN & gradient descent

Normalization :-

This is used to rescale each row of data to have a length 1.
It is mainly used in spare dataset. Where we have a lot of zeros

L1 Normalization :-

That modifies that dataset values in a way that in each row the sum of the absolute values will always be up to 1.
also called least Absolute Deviations.

```
from sklearn import preprocessing :-

array = df.values

Data_Normalizer = preprocessing.Normalizer(norm='l1').fit(arr
Data_Normalized = Data_Normalizer.transform(array)

Set_printoption (precision = 2)
```

L_2 Normalization :-

that modifies the datasets values in a way that in each row the sum of the square will always be up to 1
also called least square

Same as l_1 - Normalization
difference is that :-
Data_Normalizer = Normalizer(norm = 'l2').fit(array)

Binarization :

We can make our data-binary
We use a binary threshold values
The values about the threshold values became 1
The values smaller the threshold values became 0

from sk-learn.Preprocessing import Binarizer

array = df.values

binarizer = binarizer (threshold = 0.5).fit (array)
Data_binaizer = binarizer.transform (array)

No requirement of precision

Standardigation :-

basically used to transform the data attributes with a Gaussian distribution

This technique is useful in like linear regression, logistic regression that assume a Gaussian distribution in input dataset and produce better result with rescaled data.

```
from sk-learn.preprocessing import standard scaler!

array = df.values

datascaler = standardScaler().fit(array)
datascalered = datascaler.transform(array)

Set.printoption(precision = 2)
```

Data labeling :-

Label Encoding :-

Most of the sklearn functions expect that the data with number labels rather than word labels, Hence we need to convert such labels into number labels

We use sklearn.Preprocessing.labelEncoder() !

```
lable_encoder = preprocessing.labelEncoder()
df["columnname"] = label_encoder.fit_transform(df[column_name])
df["c.name"].unique.
```