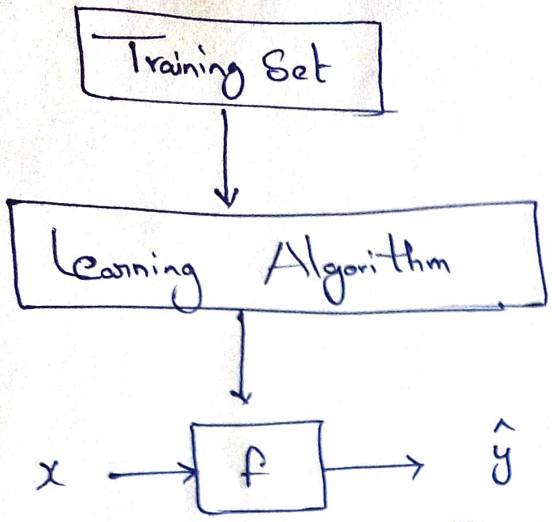
What is Machine Learning

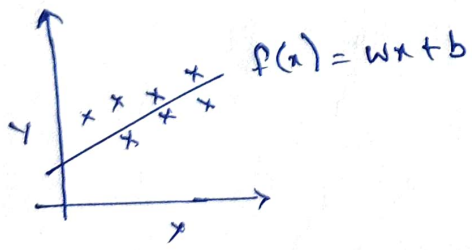
* Machine Learning is an approach to learn Complex patterns from existing data and use the patterns to make predictions on unseen data.



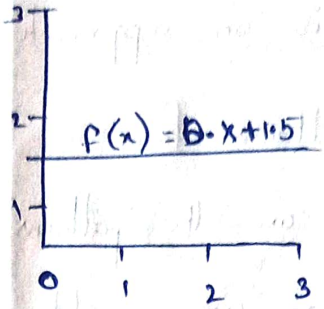
How to represent f ?

$f_{w,b}(x) = wx + b$ (or) $f(x) = wx + b$

↳ f takes the values of input 'x' & depending on values of w & b it will output some value which is the predicted value ($\hat{y}$)

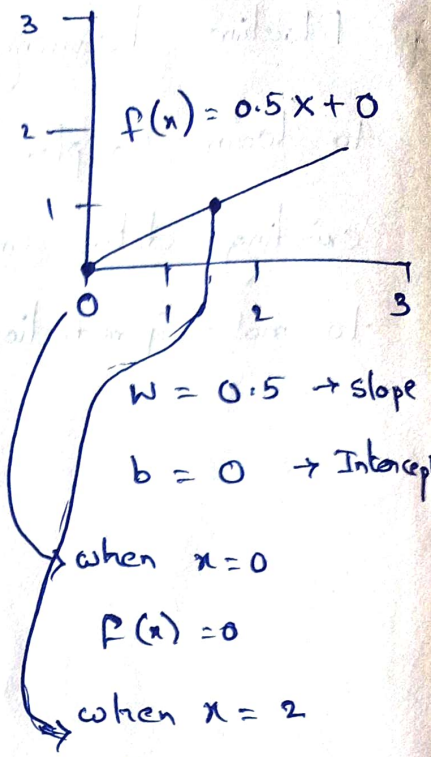


Cost fn ⇒ $f(x) = wx + b$



$w = 0$

$b = 1.5$



$w = 0.5$ → slope

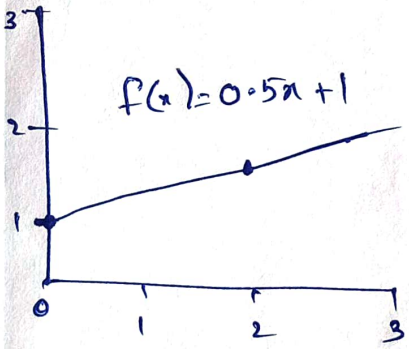
$b = 0$ → Intercept

when $x = 0$

$f(x) = 0$

when $x = 2$

$f(x) = 1$



$w = 0.5$

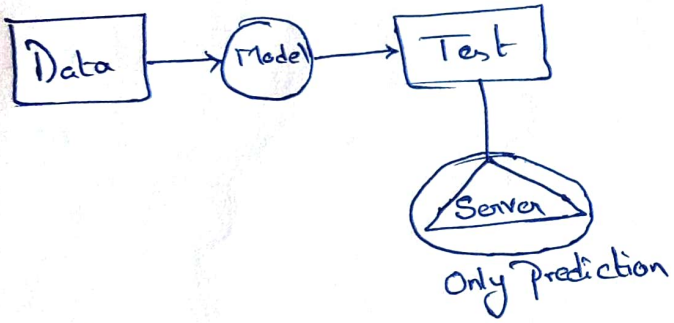
$b = 1$

Batch Vs Online Machine Learning

Batch

↳ We train the model in a single shot using entire data. We do this offline.

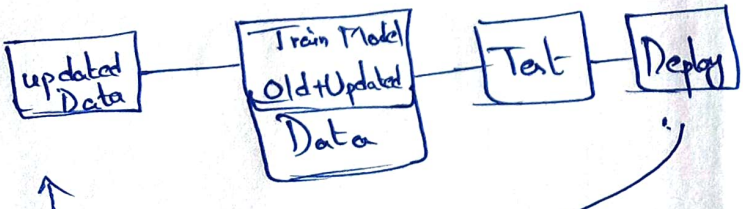
↳ When the model is trained then we deploy it on Server



Con's

① We train the model once & then deploy it. There is no retraining the model on the new data. As a result overtime the model doesn't perform very well on new data.

Solution



This cycle keeps on going (24hrs, Week, 1 month, ...)

② Lots of Data

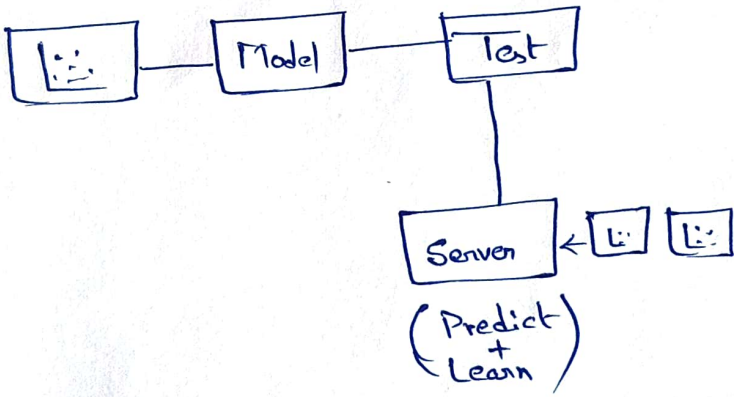
↳ Can't process with whole data

③ Hardware Limitation

④ Availability

Online → River Library

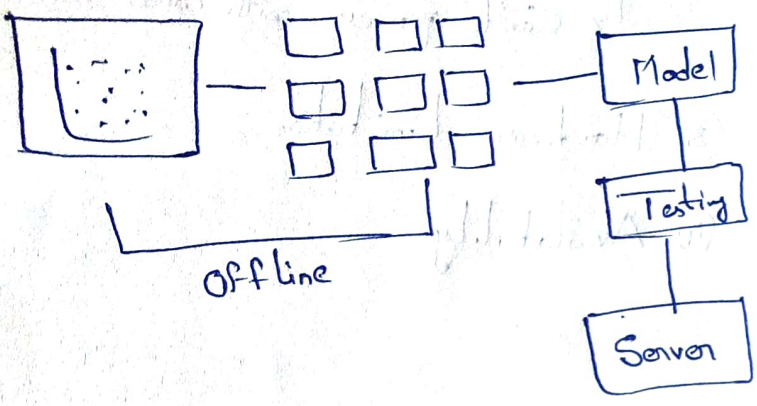
↳ It is done incrementally on the Server using mini batches



When to Use?

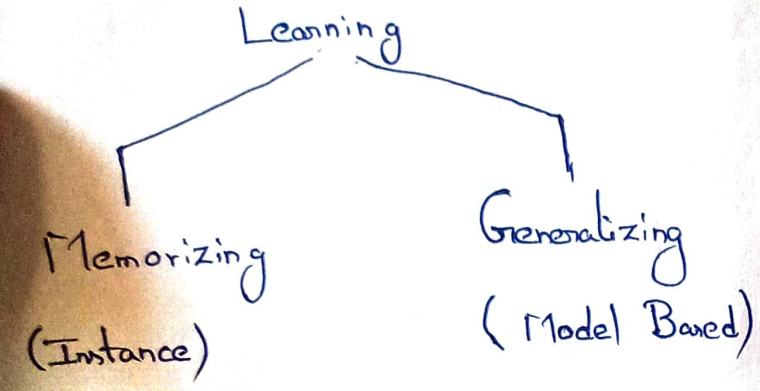
1. When there is fast Concept/context Change.
2. Cost Effective
3. Faster Solution

Out of Core Learning



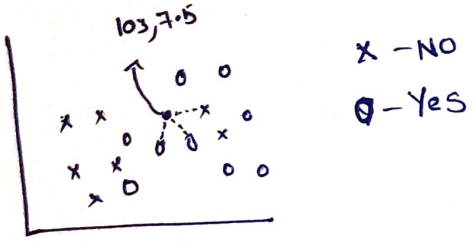
Here we are using the technique of online learning in offline. We are splitting the large data into small chunks & then we are training the model incrementally.

Instance Vs Model Based Learning



Instance Based

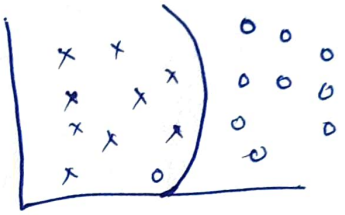
iq	CgPa	placement
80	8	Y
70	7	N



Here often even a new point comes we try to close it is to its three neighbors if two are yes and one neighbor is no. then the new point belongs to yes

Model Based

iq	CgPa	Placement
80	9	Y
70	7	N



It tries to understand the behaviour of the points & based on that it generalizes.

Tensors

↳ A Container to store numbers

2 3

Scalar → 0D-Tensor

[1, 2, 3, 4] → 1D Array / array

↳ Vector → 1D Tensor

No. of axis = Rank = dimension } of Tensor

↳ 1D Tensor

↳ Vector → 4D

Ex:- [1, 2]

↳ 1D Tensor

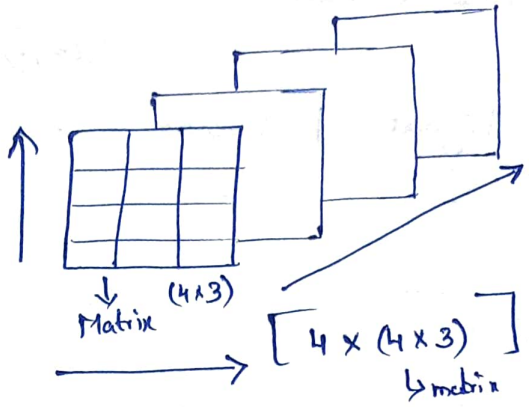
↳ Vector → 2D

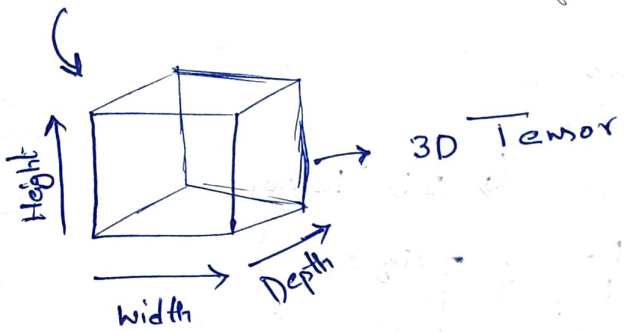
row →

$$\begin{bmatrix} [1,2,3] \\ [4,5,6] \\ [7,8,9] \end{bmatrix}$$

↳ Matrix → 2D Tensor

ND Tensors





Ex:-

NLP

	Hi	Rahul	Ankit	Bala
Hi Rahul	1	0	0	0
Hi Ankit	0	1	0	0
Hi Bala	0	0	1	0
	0	0	0	1

2D Tensor

Hi Rahul → [[1,0,0,0], [0,1,0,0]]

Hi Ankit → [[1,0,0,0], [0,0,1,0]]

Hi Bala → [[1,0,0,0], [0,0,0,1]]

3D Tensor

[[[1,0,0,0], [0,1,0,0]],
[[1,0,0,0], [0,0,1,0]],
[[1,0,0,0], [0,0,0,1]]]

Ex: 2

Time Series Data

Stock Market Data

(365, 2)

	Highest	Lowest
Day 1	—	—
Day 2	—	—
Day 3	—	—
⋮		
Day 365	—	—

↳ 2D Tensor

[[-, -] → Day 1
[-, -]
⋮
[-, -] → Day 365]

This is for One year.

For 10 years

⇒ (10, 365, 2)

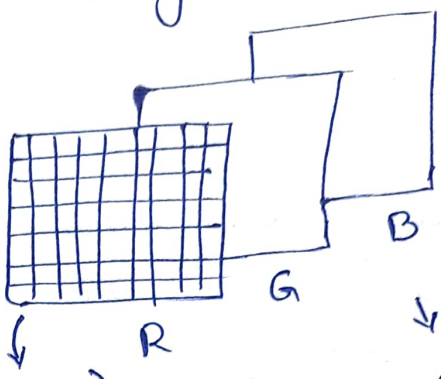
↳ 3D Tensor

[[[-, -] → One year
[-, -]]
⋮
[[-, -] → 10th year
[-, -]]]

3D Tensor (10, 365, 2) ↓ Time Axis

4D Tensors

↳ Images



(1200, 800)

↓ 1 Color Image

(3, 1200, 800)

↳ 3D Tensor

If we have 50 color Images

⇒ (50, 3, 1200, 800)

↳ 4D Tensor

5D Tensors

↳ ~~Videos~~ Videos

Videos are basically collection of frames.

frames are nothing but images.

60 sec Video → 4 Videos

↳ 30 FPS

↳ 480p → (480 × 720) → 3 channels

⇒ (1800, 480 × 720 × 3)

↳ 4D Tensor

⇒ (4, (1800, 480 × 720 × 3))

↳ 5D Tensor

Understanding Data

Asking Questions

1. How big is the data?
   - ↳ df.shape
2. How does the data look like?
   - ↳ df.head() / df.sample(5)
3. What is the data types of Cols?
   - ↳ df.info
4. Are there any missing values?
   - ↳ df.isnull().sum()
5. How does the data look mathematically?
   - ↳ df.describe()
6. Are there duplicate values?
   - ↳ df.duplicated.sum()
7. How is the Correlation b/w cols?
   - ↳ df.corr()
8. What is the distribution of data?

Exploratory Data Analysis

Univariate Analysis

- Categorical Data

(a) Countplot

↳ sns.countplot(df['Survived'])

↳ df['Survived'].value_counts

(b) PieChart

df['Survived'].value_counts.plot(kind='pie', autopct='%.2f')

Numerical Data

(a) Histogram

↳ import matplotlib.pyplot as plt

plt.hist(df['Age'])

(b) Distplot

sns.distplot(df['Age'])

↳ Probability Density Function

(c) Boxplot

↳ sns.boxplot(df['Fare'])

(d) Min, Max, Mean

Bi-Variate Analysis

(1) Scatterplot (Numerical - Numerical cols)

↳ sns.scatterplot(df['total'], df['tip'])

↳ Bivariate

↳ sns.scatterplot('', '', hue=df['sex'])

↳ Multivariate Analysis

(2) Bar Plot (Numerical - Categorical)

↳ sns.barplot

(3) Box Plot (Numerical - Categorical)

(4) Distplot (Numerical - Categorical)

(5) Heat Map (Categorical - Categorical)

(6) Cluster Map (Categorical - Categorical)

(7) Pairplot

(8) Lineplot (Numerical - Numerical)

Feature Engineering

① Feature Transformation

- Missing Values
- Handling Categorical Features → One-Hot Encoding
- Outlier Detection
- Feature Scaling

② Feature Construction

③ Feature Selection

④ Feature Extraction

Feature Construction

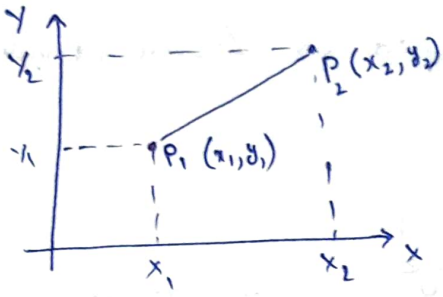
↳ You Create a Completely new Column from the existing features.

SibSp	ParCh

SibSp → Siblings col & Spouse col

ParCh → Parent col & Child col

Feature Scaling



Euclidean Distance b/w P_1 & P_2

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

	X	Y
	Age	Salary	Purchase
	50	83000	1
	27	48000	0

$(x_2-x_1)^2 = 529$

$(y_2-y_1)^2 = 1225000000$

↳ This effects the euclidean distance.

Feature Scaling

- Standardization
- Normalization
  - MinMax

Standardization

↳ Also called as Z-Score Normalization

Age	Transform → Age'
27	0.08
28	
29	
30	
31	
32	
⋮	

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

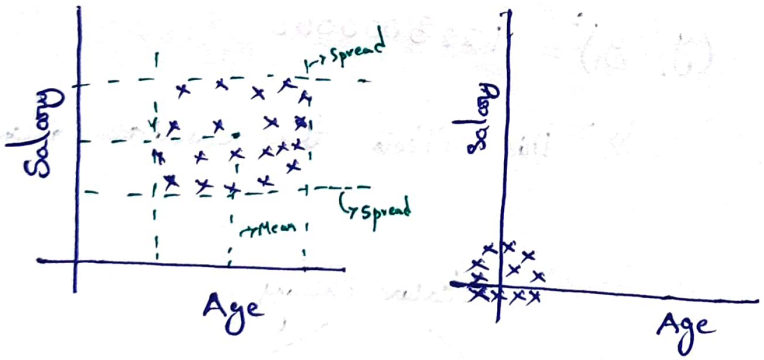
$\mu \rightarrow$ Mean

Standard deviation

$\sigma \Rightarrow$ ~~Sigmoid~~

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}}$$

↳ no. of data points



Mean centering

$\mu = 0$

$\sigma = 1$

Scaler.fit_transform

↳ It finds μ & σ

transform

↳ It transforms the data using μ & σ

we only fit once & then transform.

Ex:

~~X_train_Scaled = Scaler.fit~~

Scaler.fit(X_train)

X_trains = Scaler.transform(X_train)

X_tests = Scaler.transform(X_test)

When to Use Standardization

(i) K-Means Algorithm } Euclidian Distance

(ii) KNN Algorithm }

(iii) PCA

(iv) Artificial Neural Network

(v) Gradient Descent

↳ Linear Regression

↳ Logistic Regression

Feature Engineering

↳ Feature Engineering is the process of using domain knowledge to extract features from raw data.

These features can be used to improve the performance of ML algorithms.

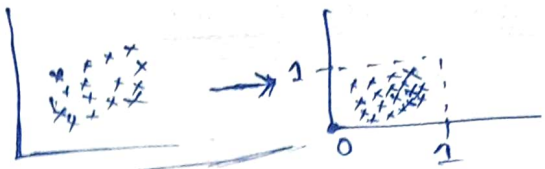
Normalization

↳ The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values (or) losing information.

- Min Max Scaling
- Mean Normalization
- Max AbsScaling
- Robust Scaling

Min - Max Scaling

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$



↳ Min - Max scaling

Max AbsScaling

↳ Used when we have Sparse data

Sparse data — Where 0's are more in data

Robust Scaling

$$x_i = \frac{x_i - X_{median}}{IQR}$$

↳ Robust to outliers

Normalization Vs Standardization

1. Is feature Scaling important ?
2. Most of the problems will be solved using Standardization
3.

Feature Transformation

Encoding Categorical Data

Nominal Categorical Data

↳ No relationship b/w ~~two~~ values.
(or) no heirarchy b/w data.

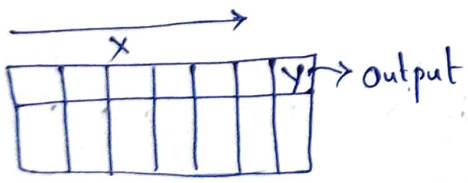
Ex:- States, Names, brand ...

Ordinal Categorical Data

↳ Heirarchy of the data

Ex:- Family Heirarch, Military Ranks,

Ordinal Encoding



- If the inputs are Ordinal Categorical data then we go for "Ordinal Encoding".
- If the Output Column is Categorical then we go for "Label Encoder".

↳ sklearn.preprocessing import OrdinalEncoder

↳ for input Columns

↳ sklearn.preprocessing import LabelEncoder

↳ for Output Column

One-Hot Encoding

↳ sklearn.preprocessing import OneHotEncoder

Column Transformer

↳ from sklearn.compose import ColumnTransformer

```
transformer = ColumnTransformer(transformers=[
    ("tnf1", SimpleImputer(), ['Fever']),
    ("tnf2", OrdinalEncoder(), ['cough']),
    ("tnf3", OneHotEncoder(), ['gender', 'city'])
], remainder="passthrough")

transformer.fit_transform(x_train)
transformer.transform(x_test)
```

Intro to Machine Learning

Machine Learning

- **Supervised** → If target Column is available
- **UnSupervised** → If target Column is not available
- **Semi Supervised** → If target Col is available for partial records

Supervised Learning

Classification	Regression
Target Column * Has descrete values Binary / Multiple values Ex:- 0/1 (or) 1,2,3,4	Target Column * Has Continuous Values Ex:- 1.2, 1.4, 1.6, 2.3, 2.5, 2.7 ....

Linear Regression

* It deals with regression based Problems

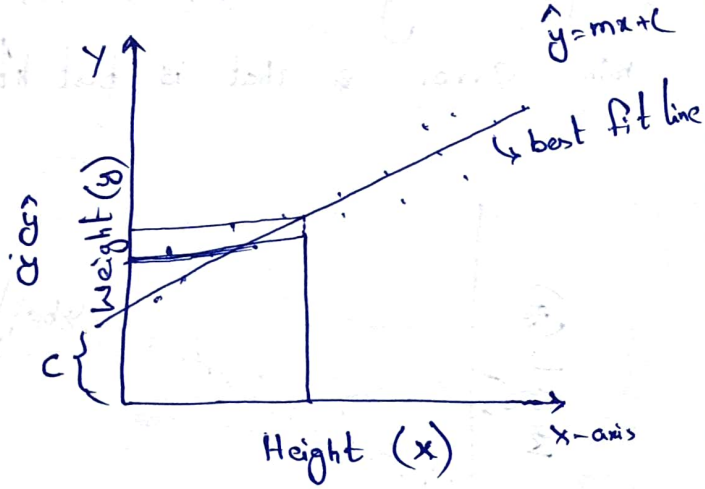
Let's Consider an Example & understand how it works.

feature 1 → Height (x)	feature 2 (target col) → Weight (y)
5.5	57
6	64
4.7	49
5.3	61
5.0	?

We are given Some heights & their Corresponding weights, we need to find the weight for a Particular height based on the given data.

How Linear Regression Model Works ?

First the model takes all the data points & points them on a Co-Ordinate System.



Then it finds a best fit line where error is minimum.

Gradient Descent Algorithm

$$\text{Error} = \sum_{i=1}^{n} |\hat{y} - y|$$

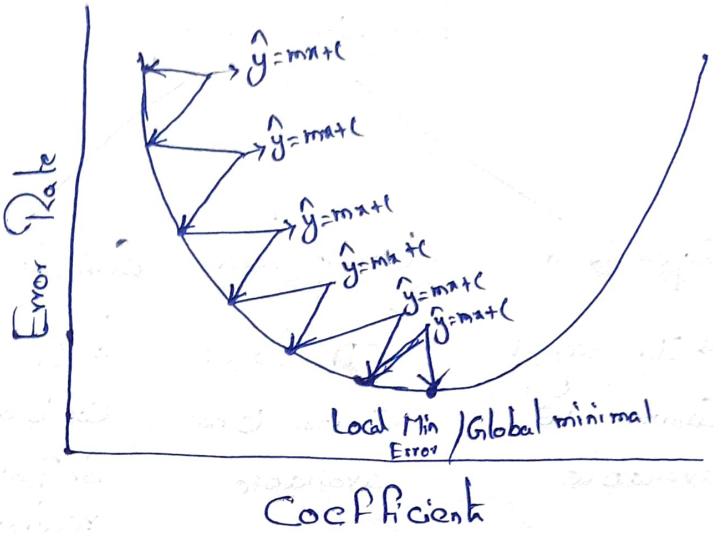
↳ Absolute Error

$\hat{y}$ → Predicted Value

y → Actual Value

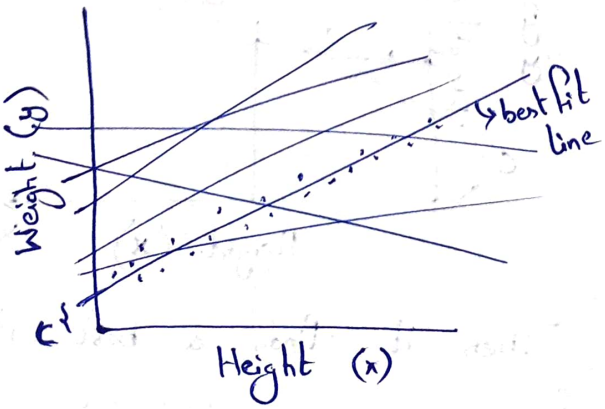
$\hat{y} = mx + c$ → Straight line eq

m ↓ Slope, c ↓ intercept



How is it going to find best fit line?

When we start training the model it randomly draws lines & checks if the error is close to error. Then it keeps doing it till it finds the min error & that is best fit line.



This algorithm is used to reduce the error.

At first a random initialization of m & c is done & the error is checked. If the error is not close to zero then m & c are updated and the Procen is repeated till the global minima is reached & the m & c values are noted.

(2)

Updates of m & C

↳ Convergence Algorithm

How is new m & C values evaluated for every iteration

$$m_{new} = m_{old} - \eta \times \frac{d\mathcal{E}}{dm}$$ ← rate of change of Error w.r.t change in slope

$$C_{new} = C_{old} - \eta \times \frac{\delta\mathcal{E}}{\delta C}$$ ← rate of change of Error w.r.t change in intercept

$\eta \rightarrow$ Learning rate

↳ rate at which m & C should change

$\frac{\delta\mathcal{E}}{\delta m}$, $\frac{\delta\mathcal{E}}{\delta C}$ → Extent to which ~~the~~ how much reduction happened

Let's Consider MSE

$$\text{Error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$

$$\mathcal{E} \Rightarrow \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx+c))^2$$

$$\mathcal{E} \Rightarrow \frac{1}{n}\sum_{i=1}^{n}\left((y_i)^2 + (mx+c)^2 - 2y_i(mx+c)\right)$$

$$\frac{\delta\mathcal{E}}{\delta m} \Rightarrow \frac{1}{n}\sum_{i=1}^{n} 2x(mx+c-y_i)$$

$$\Rightarrow \frac{1}{n}\sum_{i=1}^{n} 2x(\hat{y}-y_i)$$

$$\boxed{\frac{\delta\mathcal{E}}{\delta m} \Rightarrow \frac{2}{n}\sum_{i=1}^{n}(\hat{y}-y_i)\cdot x}$$

$$\boxed{\frac{\delta\mathcal{E}}{\delta C} \Rightarrow \frac{2}{n}\sum_{i=1}^{n}(\hat{y}-y_i)}$$

Error Term

- **MSE** — Mean Squared Error
  $$\mathcal{E} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}-y_i|^2$$
- **MAE** — Mean Absolute Error
  $$\mathcal{E} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}-y_i|$$

Through Hyper Parameter Tuning we can adjust the learning rate

Usually Learning Rate would be

η - 0.01 (or) 0.001 (or) 1

Simple Linear Regression

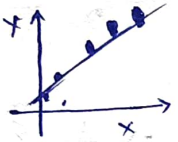
↳ $\hat{y} = mx + c$

Multiple Linear Regression

↳ $\hat{y} = m_1x_1 + m_2x_2 + m_3x_3 + \cdots + m_nx_n + c$

Assumptions of Linear Regression

1.) Linearity



$(x_1, x_2, x_3 \cdots \cdots x_n) \longrightarrow y$

↳ The inputs are in linear to the output

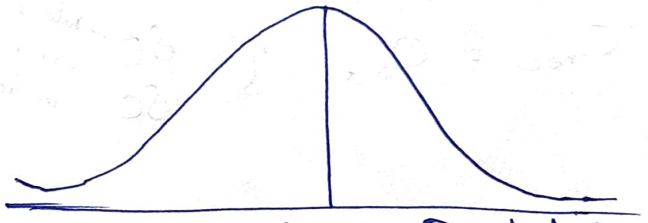
2.) Independence

$(x_1, x_2, x_3, x_4 \cdots \cdots x_n)$

∴ Each feature is independent of each other

3.) Normality

If we can plot all the ~~red~~ residuals / loss function / Error values on the graph then they follow normal / Gaussian Distribution



Normal / Gaussian Distribution

mean = median = mode

4. Homoscedalicity

The Variance of the residuals should be Constant.

Variance → spread / dispersion

↳ should be Constant

~~Normalization~~

Linear Regression – OLS

↳ Scikit Learn uses this by default.

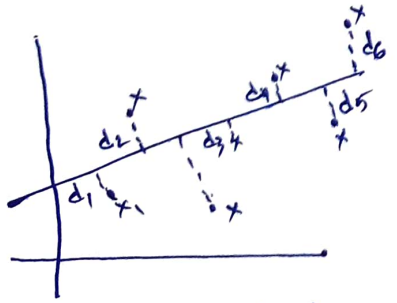
↳ Preferred when there are fewer dimensions in data

$$Y = mx + b$$

$$\star \boxed{b = \bar{y} - m\bar{x}}$$

$$\boxed{m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

$\bar{x} \rightarrow$
$\bar{y} \rightarrow$
} Mean



$$E = d_1 + d_2 + d_3 + d_4 + \cdots + d_n$$

↳ $E = d_1^2 + d_2^2 + d_3^2 + d_4^2 + \cdots + d_n^2$

$$\boxed{E = \sum_{i=1}^{n} d_i^2} \rightarrow \text{Error Function}$$

why are we squaring? why not mod?

Squaring

Reason 1 → Converts negative / distance to positive

Reason 2 → We are penalising Outliers

Reason 3 → It is differentiable.

Mod

↳ It is not differentiable

$$d_i = (y_i - \hat{y}_i)$$

$$\Rightarrow \boxed{E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

$$\hat{y}_i = mx + b$$

$$\boxed{E(m,b) = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$

$$\frac{\delta E}{\delta b} = \frac{\delta}{\delta b} \sum_{i=1}^{n} (y_i - mx_i - b)^2 = 0$$

$$= \Sigma \frac{\delta}{\delta b} (y_i - mx_i - b)^2 = 0$$

$$\Rightarrow \Sigma (y_i - mx_i - b) = 0$$

$$\Rightarrow \frac{\Sigma y_i}{n} - \frac{\Sigma mx_i}{n} - \frac{\Sigma b}{n} = \frac{0}{n}$$

$$\bar{y} - m\bar{x} - \frac{nb}{n} = 0$$

$$\Rightarrow \bar{y} - m\bar{x} - b = 0$$

$$\Rightarrow \boxed{b = \bar{y} - m\bar{x}}$$

$$E = \sum \left(y_i - mx_i - \bar{y} + m\bar{x}\right)^2 = 0$$

$$\frac{\delta E}{\delta m} = \sum \frac{\delta}{\delta m}\left(y_i - mx_i - \bar{y} + m\bar{x}\right)^2 = 0$$

$$\Rightarrow \sum +2\left(y_i - mx_i - \bar{y} + m\bar{x}\right)\left(-x_i + \bar{x}\right) = 0$$

$$\Rightarrow \sum -2\left(y_i - mx_i - \bar{y} + m\bar{x}\right)\left(x_i - \bar{x}\right) = 0$$

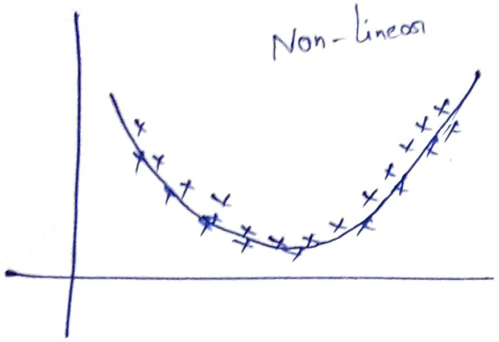
$$\Rightarrow \sum \left[\left(y_i - \bar{y}\right) - m\left(x_i - \bar{x}\right)\right]\left(x_i - \bar{x}\right) = 0$$

$$\Rightarrow \sum \left[\left(y_i - \bar{y}\right)\left(x_i - \bar{x}\right) - m\left(x_i - \bar{x}\right)^2\right] = 0$$

$$\Rightarrow \sum \left(y_i - \bar{y}\right)\left(x_i - \bar{x}\right) = \sum m\left(x_i - \bar{x}\right)^2$$

$$\Rightarrow \boxed{m = \frac{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)\left(x_i - \bar{x}\right)}{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}}$$

Polynomial Linear Regression



Simple PR

degree = 0

Polynomial degree : 0 $\quad \hat{y} = c \times x^0 \rightarrow$ Constant value

" = 1 $\Rightarrow \hat{y} = c \times x^0 + m x^1 \rightarrow$ Simple Linear Regression

" = 2 $\Rightarrow \hat{y} = c x^0 + m_1 x^1 + m_2 x^2$

" = n $\Rightarrow \hat{y} = c x^0 + m_1 x^1 + m_2 x^2 + \cdots m_n x^n$

Exploratory Data Analysis

Steps

(i) Load the dataset

Df = Pd.read_csv(file path)

Df.head() → Top 5 features

Df.shape → no.of rows & colmn

Df.info()

(ii) Df.isnull() Check for null values

Df.isnull().Sum() → Shows null value for each feature

null value → missing value / NAN

↳ replacing null values is called as Data Imputation

- Measures of Central Tendency
- KNN Algorithm

(iii) Check for Distribution

- Symmetric → Mean
- Skewed → Median

eda Df.describe()

↳ gives the descriptive stats of the data

(iv) Check if it is in SNF

SNF – Standard Normal Form

↳ $\mu = 0$ → Mean

$\sigma = 1$ → Standard Deviation

Why Convert/Scale to SNF ?

↳ To make sure the model is not biased to any one feature.

Range → -1 to 1

↳ when we scale all the features will be in this range

(v) ~~Finding Co-relations~~

* Convert Object → Categorical data type

(V) One Hot Encoding

↳ Convert Categorical data to numerical data (Ex: 0, 1, 2...)

Feature		Red	Green
Red	→	1	0
Green		0	1
Yellow		0	0
Green		0	1

dataframe = pd.get_dummies (dataDF)

dataframe.head (30)

dataframe = pd.get_dummies (DF, drop_first=True)

↳ Less no. of features

fit_transform

↳ 1.

↳ helps to get the value of μ & σ

(vi) Co-efficient Correlation

dataframe.corr()

(vii) Scaling of Numeric features

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

dataframe[ ] = dataframe.fit_transform()

SNF

f1 Age → $Z = \frac{x_i - \mu}{\sigma}$

21 →

25

27 - -

44

47

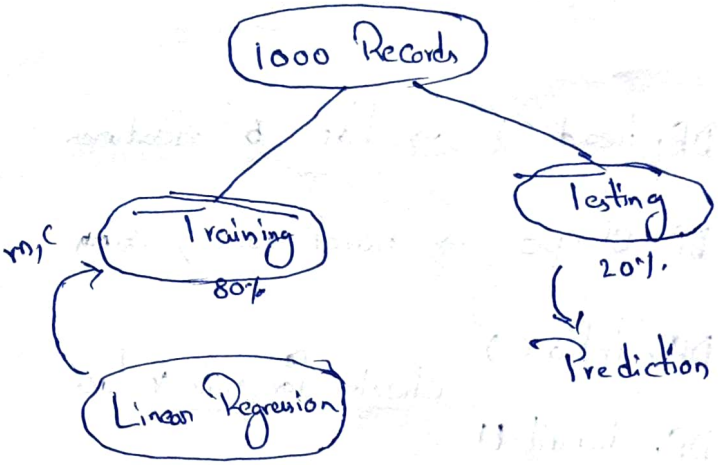
$\mu = \frac{21+25+27+44+47}{5}$

$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}}$

n ↳ no. of data points

=

(viii) Model Preperation (Training & Testing)



Model Splitting

from sklearn.model_selection import train_test_split

Model Training

from sklearn import LinearRegression

model = LinearRegression()

model.fit (x_train, y_train)

y_pred = model.predict (x_test)

Model Evaluation

↳ (i) MSE — $\frac{1}{n}(y-\hat{y})^2$

(ii) RMSE — $\sqrt{MSE}$ → ↓↓

(iii) R^2 — closer to 1

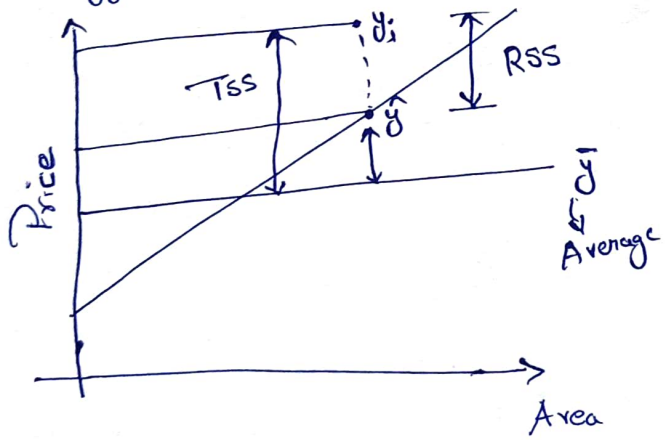
(iv) Adjusted R^2

Performance Metrics

R^2

* $1 - \frac{RSS}{TSS}$ → Residual Sum of Square / Total Sum of Square

1 − Smaller number / Bigger number



$$RSS = \sum (y_i - \hat{y})^2$$

$$TSS = \sum (y_i - \bar{y})^2$$

* If R^2 is close to 1 then it is a good accuracy.

* As the no.of features ~~are~~ increase, R^2 value also keeps increasing irrespective of the feature.

So, to overcome this drawback we have adjusted R^2.

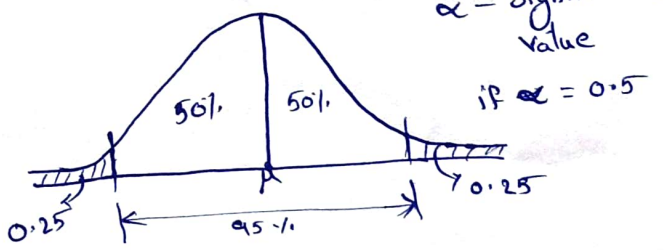
Adjusted R^2

↳ $1 - \frac{(1-R^2)(n-1)}{n-p-1}$

n - no.of records
p - no.of features

* As the no.of features increase, adjusted R^2 stays constant (or) decreases based on the feature.

P-Value → Statistics



f_1 - area → 0.0000 (p-value)

f_2 - bedrooms → 0.0000

f_3 - sex → 0.18000 → 100 − 18 = 82 → Passing only 18 times
↳ times it's to failing

If 'p' value is higher then the feature is irrelevant and it should be removed.

Real Time Use Cases

- HR (Employee Performance Evaluation)
- Finance

→ Z-Score

↳ When there are not too many outliers

IQR

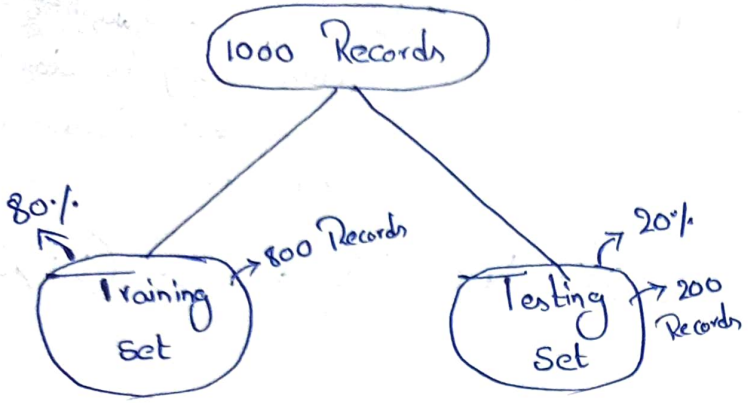
↳ When the outliers are too much more than 50% then we go for this method

Ex:-

ECommerce Data

Real Estate Data

Overfitting & Underfitting



{ Accuracy ↑↑ Error ↓↓ — Accuracy ↓↓ Error ↑↑

↳ Overfitting

↳ This is the main problem in ML

(Low Bias & High Variance)

↓ Training ↓ Testing

Bias & Variance ⇒ Error

↓ Training ↓ Testing

Q) why two Variables are used for error?

Data Distribution

(i) During training we might have data with same distribution but during testing the data might not have same distribution.

So to represent the errors during training & testing differently we have 2 Errors.

(ii) Complex / noisy data points

During training the error might be low because of the noisy data points (or) Outliers, this effects the testing phase.

So '2' Variables to estimate error during training & testing.

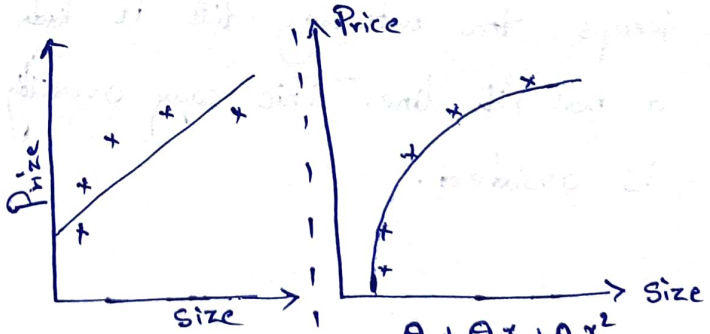
↳ Generalization happens because of the Outliers.

Ex: Generalising about the drivers of uber/ola.

Regularization Regression

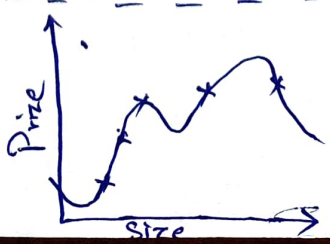
↳ To avoid / handle overfitting problem we want

↳ Low Bias & Low Variance



$\theta_0 + \theta_1 x$ — High Bias (underfitting)

$\theta_0 + \theta_1 x + \theta_2 x^2$ — Low Bias, Low Variance (Good Fitting)



Over Fitting

$(\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4)$

High Variance

Regression

↳ L1 Regularization (LASSO)

↳ L2 Regularization (RIDGE)

↳ L3 Regularization (L1 + L2)

LASSO → Very widely Used

$$\text{MSE} + \lambda \sum_{i=1}^{K} |m_i|$$

- K: no. of features
- $|m_i|$: slope/weightage
- λ: Hyper Parameter
- MSE: Error Function

↳ This helps in Overcoming the problem of Overfitting.

↳ when MSE/Error becomes Zero (or) close to zero, we are adding extra value, so that the model keeps ~~tres~~ retraining till it finds a best fit line. This way overfitting is avoided.

Ridge → Also used for Feature Selection

$$\text{MSE} + \lambda \sum_{i=1}^{K} (m_i)^2$$

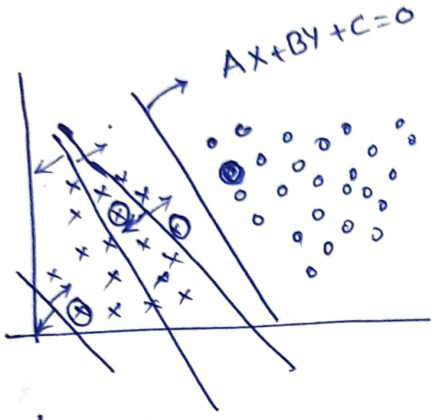
when slope/weightage is close to zero, that feature is removed.

Elastic Net Regularization (L1 & L2)

L1 + L2

Logistic Regression

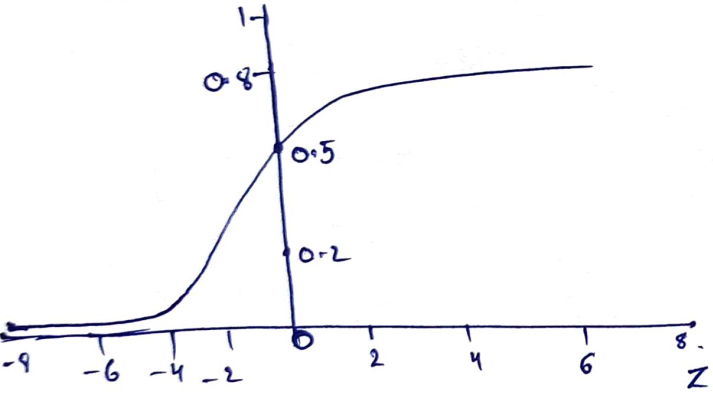
Perceptron Trick



loop = 1000

↳ random student

Sigmoid Function



$-\infty < z < \infty$

$0 < Y < 1$

This fn output will always be in the range of 0 to 1.

what we used to do

$\hat{y} = w_1 x_{i1} + w_2 x_{i2} + w_0 = z$ → step function $\begin{cases} z \geq 0 \to 1 \\ z < 0 \to 0 \end{cases}$

In Logistic Regression we will replace step function with sigmoid

$\hat{y} = w_1 x_{i1} + w_2 x_{i2} + w_0 = z$ → Sigmoid

$z \Rightarrow \sum w_i x_i = \sigma(z)$ ↳ sigmoid

~~if $z > 0$~~

if z is positive $\Rightarrow \geq 0.5$

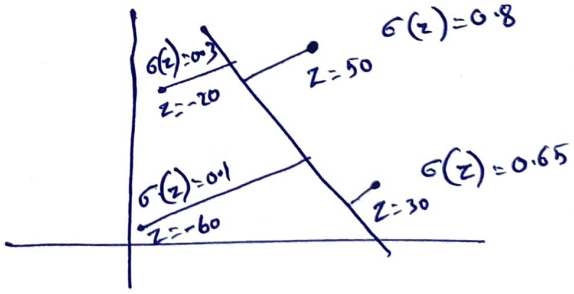
z is negative $\Rightarrow < 0.5$

Impact of Sigmoid

$w_n = w_0 + \eta(y_i - \hat{y}_i) x_i$

$y_i = \sigma(z)$

where $z = \sum w_i x_i$

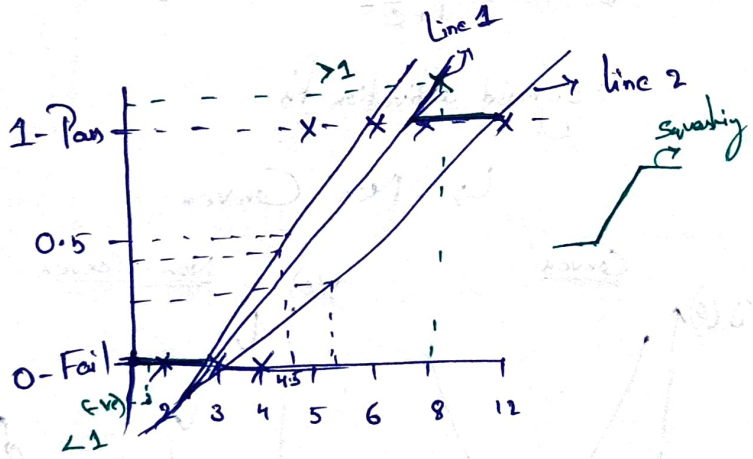


y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
1	0.8	0.2
0	0.65	-0.65
1	0.3	0.7
0	0.15	-0.15

Logistic Regression

↳ Deals with Binary Classification

Study Hours	Pass/Fail
2	Fail
3	"
4	"
5	Pass
6	"
8	"
12	" → Outlier



Case 1:- No Outlier using Linear Regression

* Here Linear Regression tries to find the best fit line.

Let say we need to predict for 4.5 hours,

Based on the best fit line, if we try to project the value on the best fit line and if the projection is below 0.5 then it belongs to 0 & if its above 0.5 then it is 1.

$\leq 0.5 \rightarrow 0$
$> 0.5 \rightarrow 1$
} Assumptions

when we have an outlier then the best fit line changes according to the Outlier. Now if we plot the same data on this line it gives a different value.

Ex:- ① 4.5 → Fail → For line 1
5 → Pass → "
6 } Pass → "
8

② 5 → Fail → For line 2
6 } Fail → "
8

Also we can see from the graph that for some points the value is >1 or <0.

But for binary classification the value should be b/w 0 & 1.

Why we can't use Linear Regression for Logistic Regression ?

1.) Best fit line (outliers)
↳ changes

2.) >1 & <0 { squash line }
↳ Linear Regression Can't do it.

$$h_\theta(x) = \frac{1}{1+e^{-z}}$$ → Sigmoid fn

Linear Regression Cost function

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

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

Logistic Regression Cost function

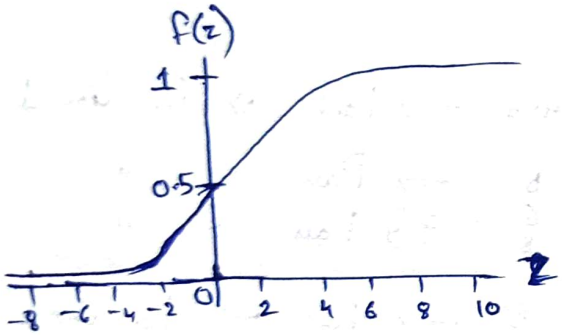
$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)^i - y^i\right)^2$$

$$h_\theta(x) = \frac{1}{1+e^{-z}} \qquad z = \theta_0 + \theta_1 x$$

↳ Sigmoid activation fn

↳ Non Convex

Sigmoid Function



if $z > 0$

$f(z) \geq 0.5$

else

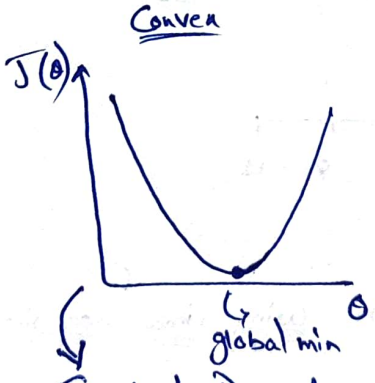
$f(z) < 0.5$

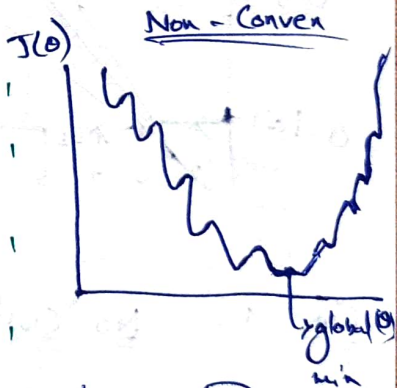
$h_\theta(x) = \theta_1 x + \theta_0$ → Linear Regression

$h_\theta(x) = \sigma(\theta_1 x + \theta_0) \qquad z = \theta_1 x + \theta_0$

↳ Activation Fn (sigmoid)

$= \sigma(z)$





Gradient Descent
↳ Linear Regression

Logistic Regression

→ In Order to replace this non-Convex Curve with Convex Curve we are using Sigmoid/Logloss Cost function.

Logistic Regression

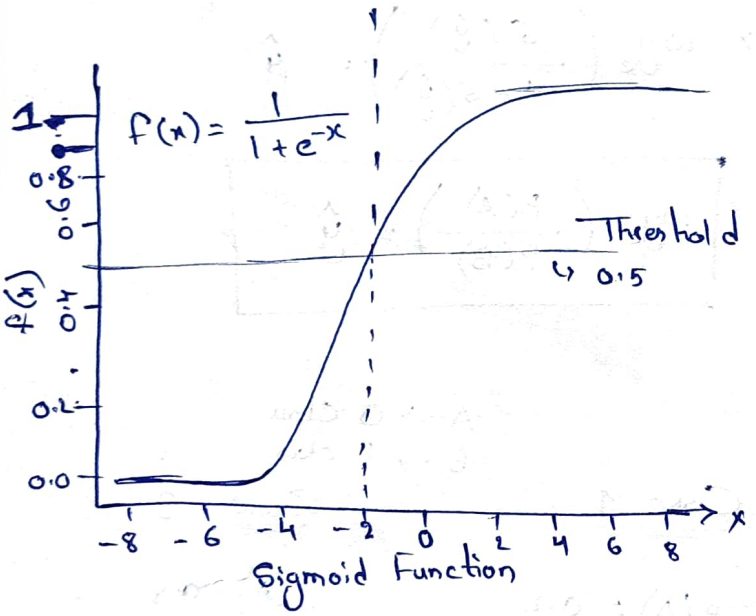
↳ Mostly used for Binary Classification

↳ It gives us the probability

Sigmoid → Probability { 0 → 0.6, 1 → 0.4

↓

Activation function



* Sigmoid Function gives us the probability based on the threshold value

If the threshold > 0.5 → 1

threshold < 0.5 → 0

In Linear Regression we want to find a straight line / best fit line

$$\hat{y} = mx + c$$

but in logistic Regression we want to find the probability of it being a straight line / best fit line

$$\therefore \sigma(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$$

So we apply sigmoid fn on the line

$$\sigma(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$$

$$= \frac{1}{1+e^{-(mx+c)}}$$

$$= \frac{1}{1+\frac{1}{e^{mx+c}}}$$

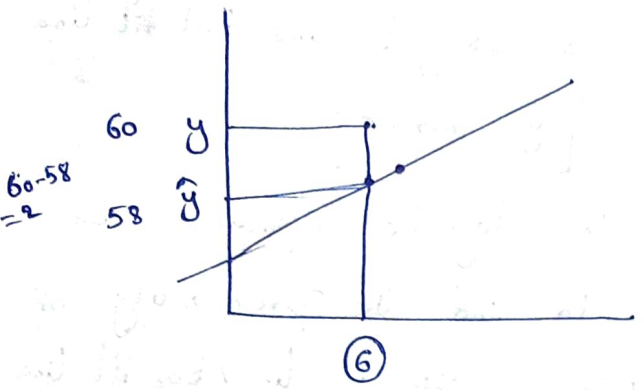
$$= \frac{1}{\frac{e^{mx+c}+1}{e^{mx+c}}}$$

$$\boxed{\sigma(\hat{y}) = \frac{e^{mx+c}}{e^{mx+c}+1}}$$

$P(A) + P(B) = 1$

$P(A) = 0.6$

$P(B) = 0.4$



⑥

In linear Regression we try to find the difference b/w the Predicted value & actual value. But in Logistic Regression we try to find what is the probability of ~~height~~ weight being 58 for the given height 6.

$$P(A) = \sigma(\hat{y})$$

$$P(B) = 1 - \sigma(\hat{y})$$

$$= 1 - \frac{e^{mx+c}}{e^{mx+c}+1}$$

$$\boxed{P(B) = \frac{1}{e^{mx+c}+1}}$$

$$\frac{\sigma(\hat{y})}{1-\sigma(\hat{y})} = e^{mx+c}$$

$$\log_e\left(\frac{\sigma(\hat{y})}{1-\sigma(\hat{y})}\right) = \log_e e^{mx+c}$$

$$\Rightarrow \log_e\left(\frac{\sigma(\hat{y})}{1-\sigma(\hat{y})}\right) = mx+c$$

$$\Rightarrow \log_e\left(\frac{\sigma(\hat{y})}{1-\sigma(\hat{y})}\right) = \hat{y}$$

$$\boxed{\log_e\left(\frac{P(A)}{P(B)}\right) = \hat{y}}$$

A → 0 class
B → 1 class

Case 1

$P(A) = 0.2$

$P(B) = 0.8$

Sigmoid fn

$\hat{y} = \log_e\left(\frac{0.2}{0.8}\right)$

$= -1.38$

↙

0 class

Case 2

$P(A) = 0.8$

$P(B) = 0.2$

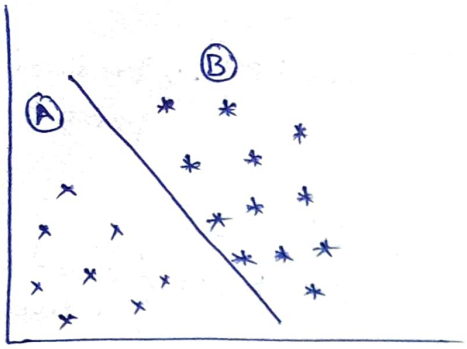
Sigmoid fn

$\hat{y} = \log_e\left(\frac{0.8}{0.2}\right)$

$= 1.38$

↙

1 class



Cost Function

Linear Regression

↳ Cost function $= \frac{1}{n} \sum_{i=1}^{n} (\hat{y} - y)^2$

Logistic Regression

↳ Cost function $= \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{1+e^{-(mx+c)}} - y \right)^2$

To get a Convex Curve

$$\text{Cost}(\hat{y}, y) = \begin{cases} -\log_e \hat{y} & ; y = 1 \\ -\log_e (1-\hat{y}) & ; y = 0 \end{cases}$$

($\hat{y}$: Predicted, y: Actual value)

↳ Log loss

Misclassification → case 1

$\hat{y} = 0$ (Predicted), $y = 1$ (Actual)

$= -\log_e \hat{y}$

$= -\log_e 0$

$=$ undefined $/ \infty$

$\Rightarrow \infty$

Proper Classification

$\hat{y} = 1$, $y = 1$

$= -\log_e 1$

$= 0$

$\Rightarrow 0$

Mis Classification

$\hat{y} = 1$, $y = 0$

$= -\log_e (1-\hat{y})$

$= -\log_e (1-1)$

$= \infty$

↳ Error is very high

Proper Classification

$\hat{y} = 0$, $y = 0$

$= -\log_e (1-0)$

$= -\log_e (1)$

$= 0$

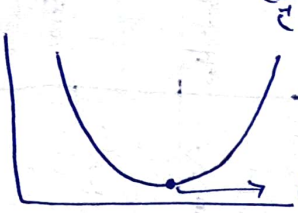
↳ Error is zero

* When there is a misclassification the error is very high & the training keeps going till the error is min.

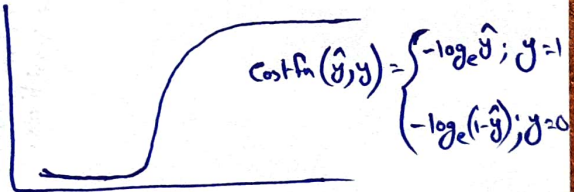
Why different Cost function ?

In Linear Regression we deal with Convex Curve, so the Cost fn is MSE.

Cost fn $\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y} - y)^2$



But in Logistic Regression we don't deal with Convex Curve. so we need a different Cost fn. (sigmoid fn)



$$\text{Cost fn} = \sum_{i=1}^{n} -y_i \log(\hat{y}) - (1-y_i)\log(1-\hat{y})$$

↳ Another way of writing cost fn

$$\text{Precision} = \frac{TP}{(TP+FP)} \rightarrow \text{Predicted Positive}$$

$$\text{Recall} = \frac{TP}{(TP+FN)} \rightarrow \text{Actual Positive}$$

↳ Very Serious problem → ↓ ↓

f_1 - Score

↳ $\dfrac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

Confusion Matrix

Actual Class (y) \ Predicted Class (ŷ)	Positive (Building will collapse)	Negative (Building will not collapse)	
Positive	True Positive (TP)	False Negative (Type II Error)	Sensitivity (or) Recall $\frac{TP}{(TP+FN)}$ → Out of total Actual Positive values how many are true positives.
Negative	False Positive (Type I Error)	True Negative	Specificity $\frac{TN}{TN+FP}$
	Precision $\frac{TP}{TP+FP}$ ← Out of total Predicted Positive values, how many True Positives	Negative Predictive value $\frac{TN}{TN+FN}$ ← Out of total Predicted Negative values, how many are True Negatives	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$

Hypothesis ↳ Building will collapse Predicted Class (ŷ)

↳ Negating hypothesis ↳ Building will not collapse

Q When Precision is important & when Recall is important &

↳

Recall should be used when we want to reduce False Negative.

Precision should be used when we want to reduce False Positive

Accuracy

↳ It is not a good measure to consider when there is imbalance in the data.

0 → 700 records → 70%
1 → 300 records → 30%

↳ Biased towards zero data.

F_Beta Score

↳ $(1+\beta^2)\frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$

$\beta > 1$: Recall is more important than Precision (e.g., 2)

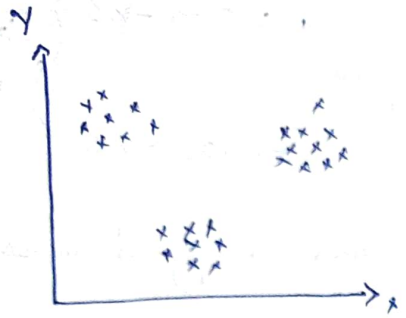
$\beta < 1$: Precision is more important than Recall (e.g., 0.5)

$\beta = 1$: If both Precision & Recall are important (F1 Score)

$$F_1 \text{ Score} = (1+1)^2 \frac{\text{Precision} \cdot \text{Recall}}{(1)^2 \text{Precision} + \text{Recall}}$$

$$= 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Logistic Regression → Multiclass



→ One Hot Encoding

f_1	f_2	f_3	o/p	O_1	O_2	O_3
–	–	–	O_1	1	0	0
–	–	–	O_2	0	1	0
–	–	–	O_3	0	0	1
–	–	–	O_1	1	0	0
–	–	–	O_3	0	0	1
–	–	–	O_2	0	1	0

One Versus Rest

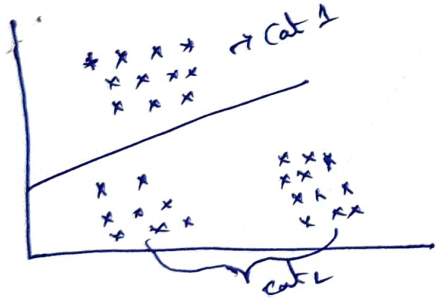
f_1	f_2	o/p
–	–	O_1
–	–	O_2
–	–	O_3
–	–	O_2
–	–	O_3

M_1 → I/P {f_1, f_2, f_3}, O/P {O_1}

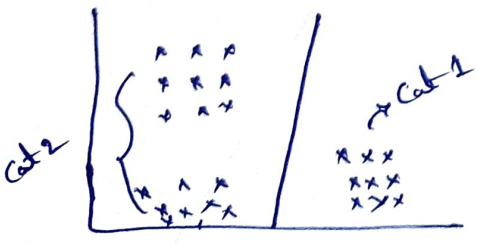
M_2 → I/P {f_1, f_2, f_3}, O/P {O_2}

M_3 → I/P {f_1, f_2, f_3}, O/P {O_3}

M_1 → Binary class.



M_2 → Binary Class



M_3 → Binary Class

New Test Data → M_1 → 0.25 → Probability

↳ M_2 → 0.20

↳ M_3 → 0.55

[0.25, 0.20, 0.55]

↑ ↑ ↑

M_1 M_2 M_3

∴ The output for the new test data is O_3, since M_3 used O_3 as output.

Decision Trees

↳ Can be used for both Regression & Classification based problems

Impurity Measure:-

* Model tries to predict which feature has lesser impurity

Lesser impurity (or) Higher Information Gain

① Gini Index

② Entropy

Gini Index

↳ Using this we can decide which node will be a root node/Parent node

	f_1	f_2	
	Gender	Class	Stay in Hostel
→ 1	M	9	Yes
→ 2	F	10	NO
→ 3	F	8	Yes
→ 4	F	8	NO
→ 5	M	9	Yes
→ 6	M	10	NO
→ 7	F	11	Yes
→ 8	M	11	Yes
→ 9	F	8	Yes
→ 10	M	9	NO
→ 11	M	11	NO
→ 12	M	11	Yes
→ 13	F	10	NO
→ 14	M	10	Yes

$$\text{Gini} \Rightarrow 1 - \sum_{i=1}^{c} P_i^2$$

Gini (class)

Class	Stays in Hostel		P(Y)	P(N)
8	Y = 2	N = 1	2/3	1/3
9	Y = 2	N = 1	2/3	1/3
10	Y = 1	N = 3	1/4	3/4
11	Y = 3	N = 1	3/4	1/4

↑ Probability

$\text{Gini(class} = 8) = 1 - (2/3)^2 - (1/3)^2 = 4/9$

$\text{Gini(class} = 9) = 1 - (2/3)^2 - (1/3)^2 = 4/9$

$\text{Gini(class} = 10) = 1 - (1/4)^2 - (3/4)^2 = 3/8$

$\text{Gini(class} = 11) = 1 - (3/4)^2 - (1/4)^2 = 3/8$

Weighted Average for Gini (class)

$$= \underbrace{3/14}_{\text{No. of records}} * \underbrace{4/9}_{\text{Gini of class}} + 3/14 * 4/9 + 4/14 * 3/8$$

$$+ 4/14 * 3/8$$

(Total ← no. of records)

$$= 0.404$$

Gender	Stay in Hostel		P(Y)	P(N)
M	Y = 5	N = 3	5/8	3/4
F	Y = 3	N = 3	3/6	3/6

$\text{Gini}(M) = 1 - (5/8)^2 - (3/8)^2 = 15/32$

$\text{Gini}(F) = 1 - (3/6)^2 - (3/6)^2 = 6/12 = 0.5$

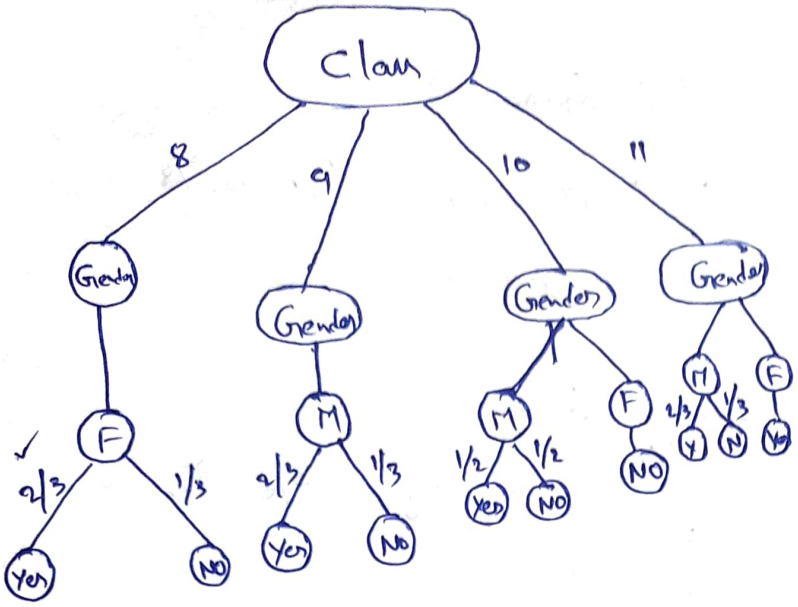
Weighted Average for Gini (Gender)

$$\text{Gini(Gender)} = \frac{8}{14} * \frac{15}{32} + \frac{6}{14} * 0.5$$

$$= 0.482$$

Gini (class) < Gini (Gender)

∴ Gini (class) will be the parent node.

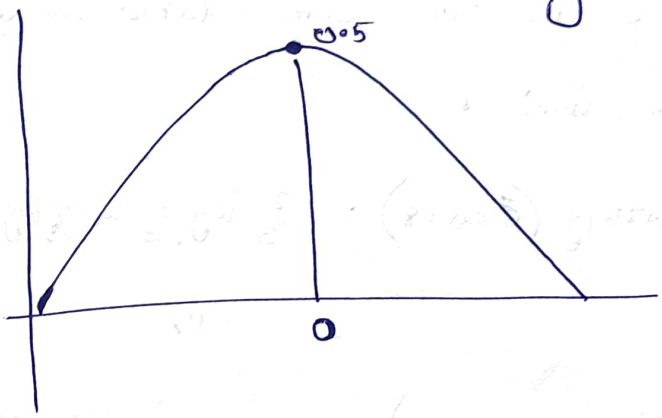


Impure Sample

$$\begin{cases} Y = 2 \\ N = 2 \end{cases} \Rightarrow 1 - (2/4)^2 - (2/4)^2 = 0.5$$

Gini Value → {0 to 0.5}

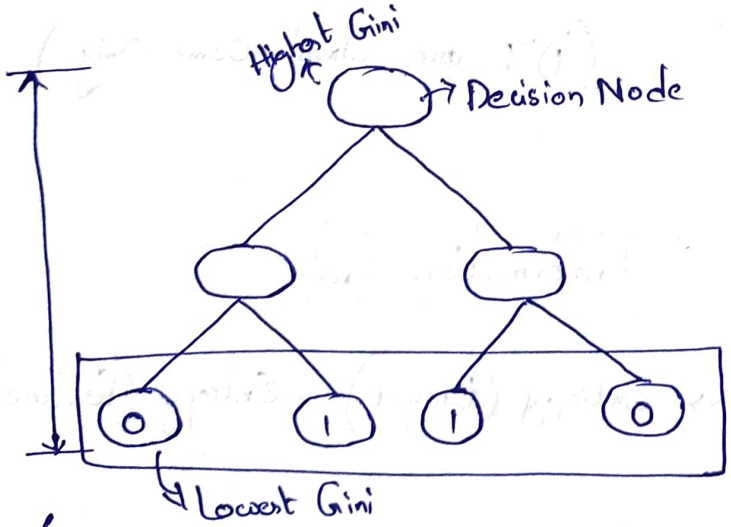
↓ Lowest ↓ Highest



Pure Sample

$$\begin{cases} Y = 3 \\ N = 0 \end{cases} \quad 1 - \sum_{i=1}^{c} p_i^2 \Rightarrow 1 - (3/3)^2 = 0$$

$$\begin{cases} Y = 0 \\ N = 3 \end{cases} \Rightarrow 1 - (3/3)^2 = 0$$



↳ As we are approaching the lower levels the Gini Value decreased.

Entropy

↳ It is a measure of Randomness

$$\text{Entropy} \Rightarrow -\sum_{i=1}^{c} P_i \log_2 P_i$$

↓

Computation is very expensive

we use the same table we used for Gini.

$$\text{Entropy (Class = 8)} = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}$$

$$= 0.918$$

$$\text{Entropy (class = 9)} = -2/3\log_2 2/3 - 1/3\log_2 1/3$$

$$= 0.918$$

$$\text{Entropy (class = 10)} = -1/4\log_2 1/4 - 3/4\log_2 3/4$$

$$= 0.811$$

$$\text{Entropy (class = 11)} = -3/4\log_2 3/4 - 1/4\log_2 1/4$$

$$= 0.811$$

Entropy (class)

$$\Rightarrow \frac{3}{14} * 0.914 + 3/14 * 0.918 + 4/4 * 0811 + 4/14 * 0.811$$

$$\Rightarrow 0.864$$

$$\text{Entropy (Gender = M)} = -5/8\log_2\frac{5}{8} - 3/8\log_2 3/8$$

$$= 0.954$$

$$\text{Entropy (Gender = F)} = -3/6\log_2 3/6 - 3/6\log_2 3/6$$

$$= 1$$

Entropy (Gender)

$$\Rightarrow \frac{8}{14} * 0.454 + \frac{6}{14} * 1$$

$$\Rightarrow 0.974$$

Gini / Entropy

↳ Result are always similar

(IT are almost same only)

Information Gain

↳ Entropy (target) − Entropy (feature)

$$\text{Entropy (target)} = -P_y\log_2 P_y - P_N\log_2 P_N$$

$$= -8/14\log_2 8/14 - 6/14\log_2 6/14$$

$$= 0.98522$$

Information Gain of Class $\Rightarrow 0.9852 - 0.856$

$= 0.1278$

Information Gain of Gender $\Rightarrow 0.9852 - 0.974$

$= 0.011$

$\therefore$ Class is providing more information in Comparision to gender.

Naive Bayes

↳ This algorithm works on the Principle of Baye's theorem

↓

Conditional Probability

↳ Probability of A given B

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \qquad P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(B|A) * P(A)$$

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	NO
OverCast	Hot	High	False	Yes
Rainy	Hot	High	True	No
Sunny	Mild	High	False	Yes
Sunny	Cold	Normal	False	Yes
"	"	"	True	No ~~Yes~~
OverCast	"	"	"	Yes
Rainy	"	"	False	Yes
Rainy	Mild	High	False	No
Sunny	"	Normal	"	Yes
Rainy	"	"	True	Yes
OverCast	"	High	True	Yes
"	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

~~P(Y | outlook~~

P(Y | Sunny, Hot, Normal, False)

=

P(N | Sunny, Hot, Normal, False)

=

$$P(B|A) * P(A)$$

(B: features, A: Yes)

P(Sunny | Yes) * P(Hot | Yes) * P(Normal | Yes)

* P(False | Yes) * P(Yes)

P(Sunny) * P(Hot) * P(Normal)

* P(False)

↳ P(B)

P(Sunny | No) * P(Hot | No) * P(Normal | No)

* P(False | No) * P(No)

P(Sunny) * P(Hot) * P(Normal)

* P(False)

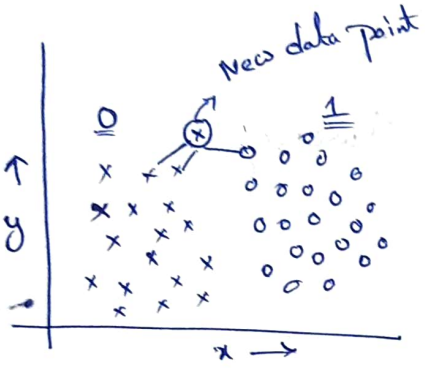
∴ P(Y | Sunny, Hot, Normal, False) = 0.6

P(N | Sunny, Hot, Normal, False) = 0.4

∴ Target = Yes

KNN Algorithm

↳ Works for both Regression & classifications



Final prediction = 0 class

↳ Classification Problem

Regression → avg $\left(\frac{1cr + 2cr + 3cr}{3}\right)$

= 2cr

Distance Metric

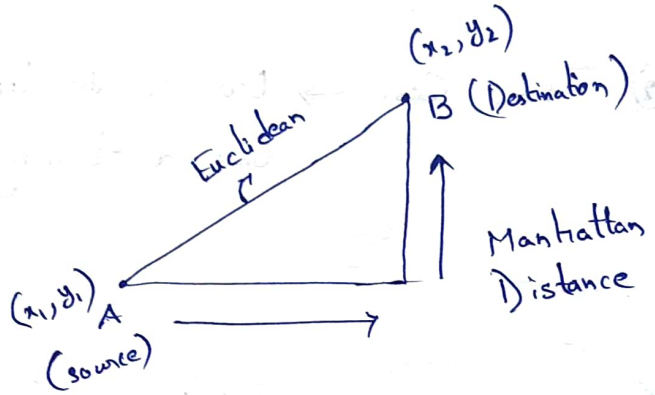
① Euclidean Distance → P=2 (Default)

↳ $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

② Manhattan Distance

Numeric Data

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



Categorical (or) Binary → Hamming Distance

"1 0 0" / "a b c" ⇒ 1 (Distance)

"d b c"

↳ Mismatch

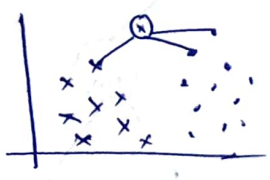
Binary →

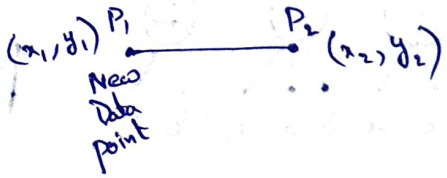
	1	0	0	1	0	0	1
	0	1	0	1	0	1	0
	1	1	0	0	0	1	1

Distance = 4

K → Odd number

↳ To avoid bias towards one class





Pros & Cons of KNN

Pros

① KNN doesn't make any assumptions about the distribution of data.

↳ It works for any distribution of data

② Time Cost → Less at the ~~the~~ time of ~~training~~

Cons

↳ It does all the calculations during prediction.

① It is a Lazy Learner

↳ At time of prediction it will take too much time.

② Finding optimal value of 'k'

* When we have huge amount of data, KNN should be avoided.

Applications

↳ Smaller data Point

↳ Data Imputation → null values

↳ SMOTE → Oversampling

↳ uses KNN Algorithm

Optimization

↳ Reduce Time & Space Complexity

↳ KD Tree

↳ Binary Search Tree

↳ Ball Tree

↳ Clustering