M_1 : 95% of errors are below 0.1
M_2 : 80% of " " 0.1

$\therefore M_1$ is better than M_2

Classification Algorithm in Various Situation

31.1 Introduction

All the algorithms are used and applicable in various real world cases.

Lets say, I have a dataset with very few positive points and lot of negative points

D = +ve, -ve points
(+ve: very few; -ve points: lot of them)

} Happens a lot in Medical Applications

→ In 10k patients, 1k has cancer and rest 9k dont have cancer
→ 1M people visit website but only 1000 people purchasing the product.

31.2 Imbalanced Vs Balanced Dataset

2class classification :

$D_n \longrightarrow n_1$ +ve points
$\longrightarrow n_2$ -ve points

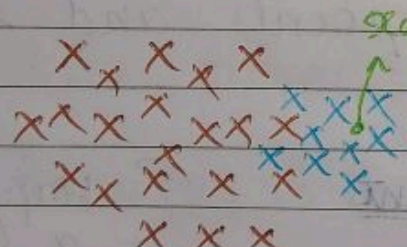
$n_1 + n_2 = n$.

if $n_1 \approx n_2 \rightarrow$ Balanced dataset
if $n_1 \ll n_2$ OR $n_2 \ll n_1$ $\rightarrow$ Imbalanced dataset

Severely Imbalanced :

$n_1 = 50$; $n_2 = 950$.

So, dataset would appear like this



X → -ve
X → +ve

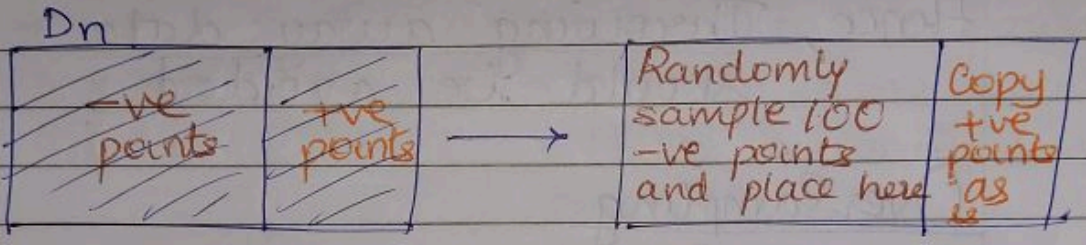
→ One class dominates over other
→ But given a point x_q, it lies in +ve cluster, hence we would classify it as positive point
→ By seeing the number of n_2, we may wonder that any query point may have high chance to be -ve. But thats not always the case

How to workaround Imbalanced Dataset problem?

⇒

Solution 1: Undersampling

$n_1 = 100$ (+ve)
$n_2 = 900$ (-ve) } > Dn



Undersampling states that given Dn, use a sampling trick to create a new dataset called Dn'

Now Dn' has 100 -ve and 100 +ve points which makes Dn' Balanced dataset. Hence in Dn'; we keep Minority points as is and sample Majority points.

Problem with Undersampling

Originally in Dn, we have 1000 points
900 -ve 100 +ve

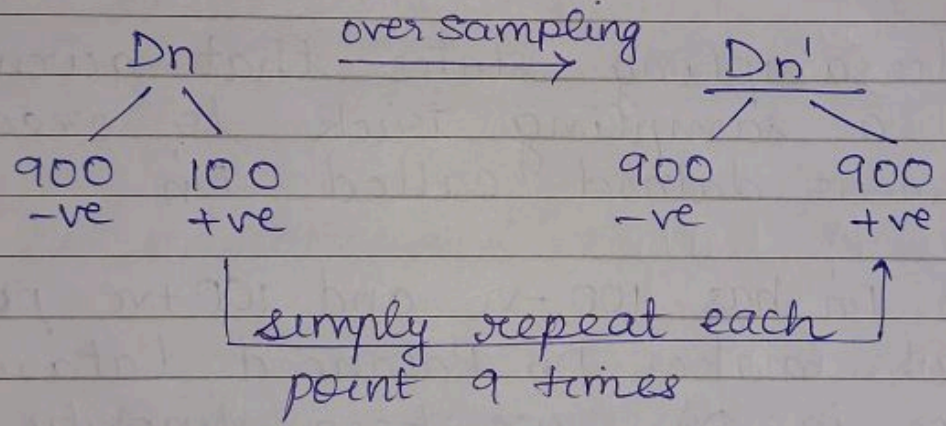
In Dn', we have only 200 points
100 +ve 100 -ve

So, we are throwing 800 points from Dn.

Since $|D_n'| \ll |D_n|$, the model may not be accurate since it doesnt work on all the dataset points.

Hence, Throwing away data should be avoided.

Over Sampling



Geometrically :

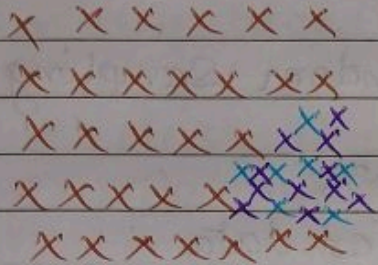
X : +ve
x : -ve
X : oversampled/ Repeated +ve points

placing the minority points on top of existing

∴ Oversampling is placing more points from the minority class in the dataset.

__/__/____

Is there other way of oversampling?
→ Instead of Repeating, take artificial or synthetic points within the minority region



x : -ve point
x : +ve point
x : extrapolating synthetic points in same region as minority (+ve)

One more variation of oversampling

Class Weight:

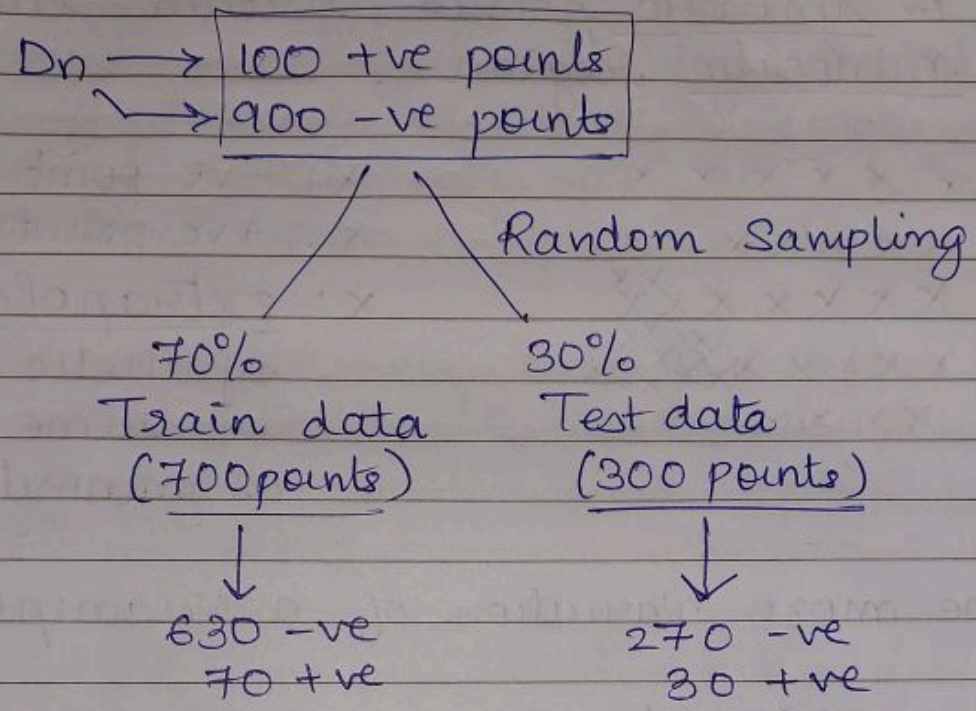
Dn → 100 +ve points
→ 900 -ve points

I would give weights to points:
Weight (+ve) = 9; More weight to Minority
Weight (-ve) = 1; Less ——— Majority

x x x x → count it as 1 point
x x x (x) → count it as 9 points
x x x

This trick is exactly same as Repeating technique of oversampling.

Other problem because of Imbalanced data :



We know Metric called Accuracy

Lets say my model f(x) says that every point is -ve

f(x): Every point is -ve {Dumb Model}

$$\text{Accuracy} = \frac{\text{Test data}}{} = \frac{270}{30} = 90\% \text{ (Very High)}$$

So, we would get very high Accuracy for DUMB ~~Accura~~ model.

Imbalanced dataset occur a lot of time in Real world.

31.3 Multi class Classification

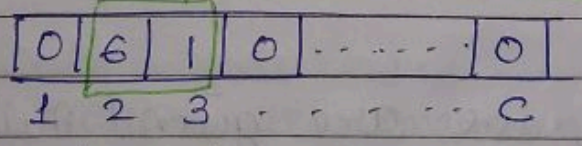
Binary classificaⁿ : $y_i \in \{0, 1\}$

Multiclass classificaⁿ $y_i \in \{0, 1, 2 \ldots\ldots 9\}$
Eg: MNIST.

K-NN :

$$D_n = \{(x_i, y_i) \mid x_i \in \mathbb{R}^d ;\ y_i \in \underbrace{\{1, 2, \ldots\ldots C\}}_{\text{C-class classificaⁿ}}\}$$

Lets say for K-NN ; there are total C classes ;
Given a query point x_q.



6 nearest neigh belong to class 2; Rest belong to class 1.

Assume its 7-NN ;
So, by majority vote,
$\hat{y}_q = 2$ (class 2)

Probability classifier

$$P(y_q = 2) = \frac{6}{7}$$

$$P(y_q = 3) = \frac{1}{7}$$

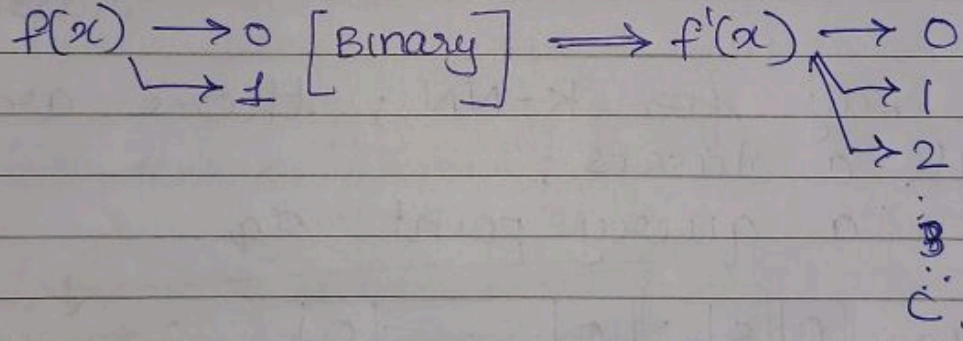
$$P(y_q = 1) = 0 = P(y_q = 4) = P(y_q = C)$$

Hence K-NN can easily be extended for multiclass classification.

There are classificaⁿ like
Logistic Regression → Binary classificaⁿ
└ cannot do Multiclass classificaⁿ easily.

Q] Given a multiclass classificaⁿ prob, can we convert it to Binary classificaⁿ problem?

Imagine we are given



Lets say, we are given Multiclass prob.
$y_i \in \{1, 2, 3 \ldots C\}$
We want to convert it to Binary prob.

① $D_n \longrightarrow \{(x_i, y_i) \mid y_i = 1\} \rightarrow +ve$
$\longrightarrow \{(x_i, y_i) \mid y_i \neq 1\} \rightarrow -ve$

② Similarly I can create Binary classifier for $y_i = 2$ & $y_i \neq 2$

③ Again, we will do for $y_i = 3$ and $y_i \neq 3$

So, if I have c classes, I will Build (c) Binary classificaⁿ, ~~I can build~~ Will train c models.

∴ Multi-class Problem. (c-classes) ⟶ c - Binary classificaⁿ Problems

- $f_1(x)$ → class 1 or not
- $f_2(x)$ → 2 or not
- ⋮
- $f_c(x)$ → c or not

Such a technique is called

1 Vs Rest

31.4 **K-NN; Given a distance or Similarity Matrix**

During classificaⁿ; $x_i \in \mathbb{R}^d$

↳ Vector is not easy to frame

Lets x_i : chemical compound (some paracetomal)

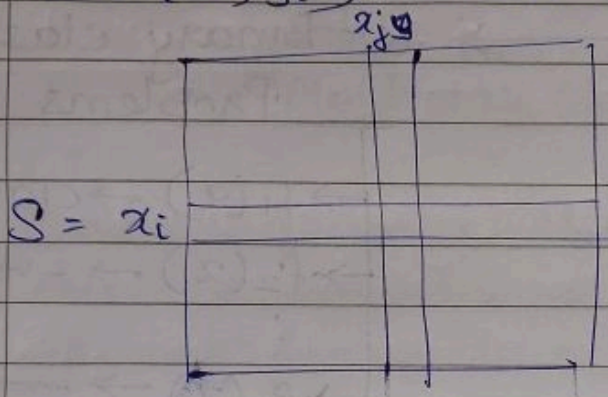
y_i : 1 → useful in curing fever.

It wont be easy to represent x_i in Vector form.

A Pharmacist would be able to tell similarity between two chemical compound.

~~But~~ Hence
Sometimes, we dont get x_i as Vector. We get Similarity matrices

~~Sim (x_i, y_i)~~ Sim (x_i, x_j)



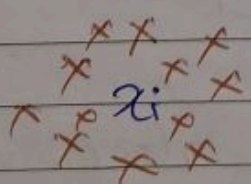
$S_{ij} = Sim(x_i, x_j)$

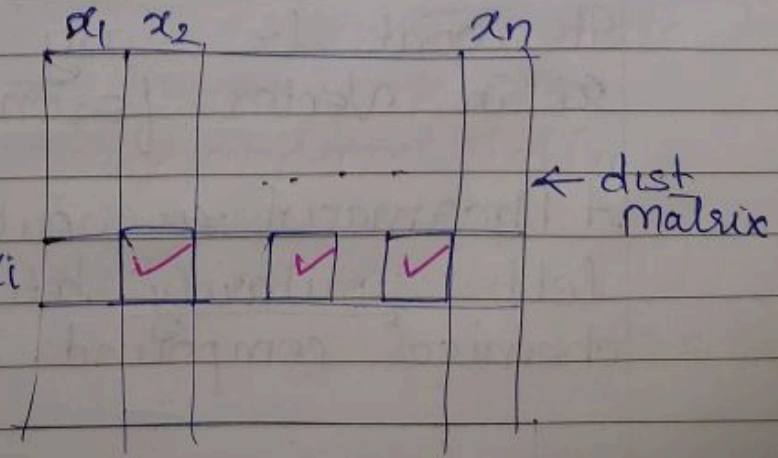
Once we receive Similarity Matrices, we can convert that into distance Matrices

$$\text{Dist}: d_{ij} = \frac{1}{S_{ij}}$$

Q Given the similarity matrix (S) OR distance Matrix (d) instead of $x_i \in \mathbb{R}^d$ explicitly. can K-NN work?

Ans: Yes.



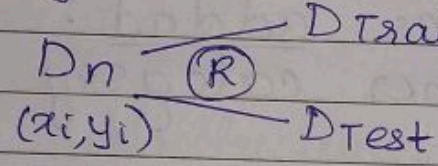


$d(x_i, x_1) =$
$d(x_i, x_2) =$
$d(x_i, x_3) =$
$\vdots$
$d(x_i, x_n) =$

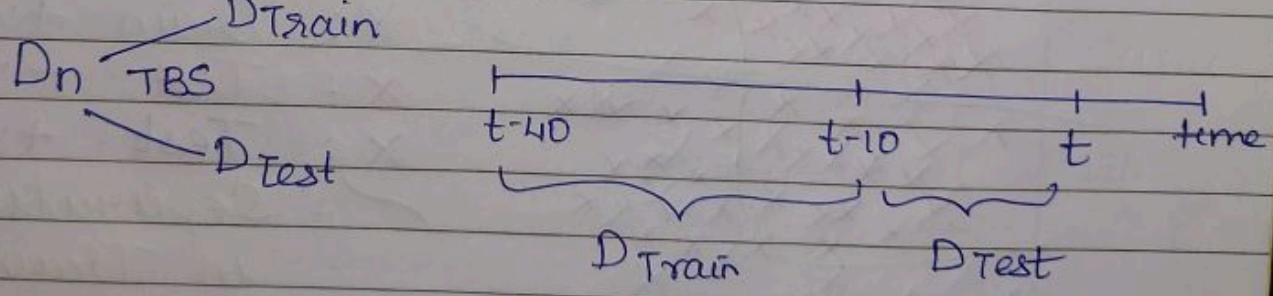
} KNN will pick up the Neighbours Nearest to x_i based on the distances

31.5 Train and Test Set Difference

Given D_n, we have seen random splitting



The problem happens when we have time based slicing (TBS)



Eg: Amazon Food Reviews

D_{Train} & D_{Test} could be very different.

Eg: In food reviews, data might have changed in past.

Food Reviews

~~t to t~~ → DTest

(t-10) to (t) :- new category of products

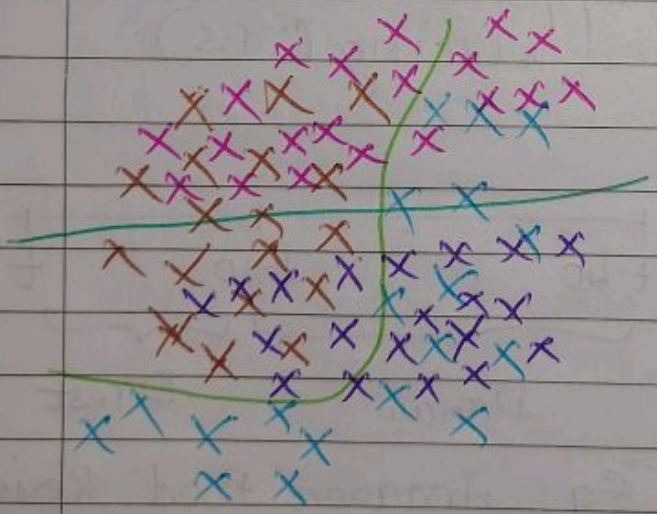
(t-40) to (t-10) : old category of products

DTrain

→ Data changes Over Time

Eg: In (t-10) to (t); new category of wine was added.
And some other category was Removed

Difference between Training & Test data



X : Train : -ve
X : Train +ve
x : Test -ve
x : Test +ve
~ : Separation in Train data
~ : Separation in Test data

DTrain & DTest are fundamentally different.

Distribution of -ve data has changed from D_{Train} and D_{Test}.

So if I use ~ line on Test data, it performs pathetically

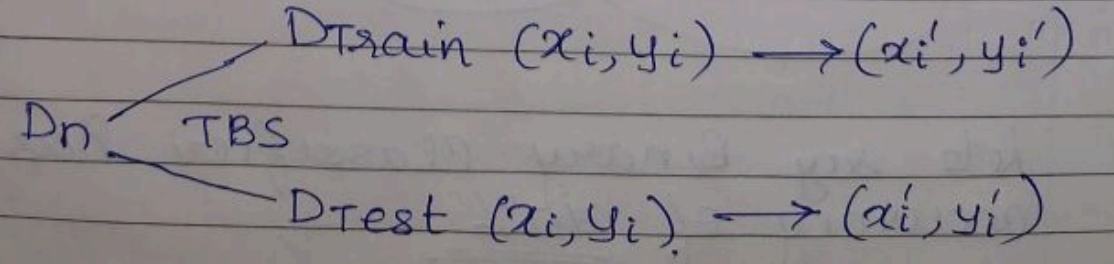
∴ C.V. Error = Low
But Test Error = high

One check we need to do:
Do D_{Train} and D_{Test} have <u>same distribution</u> ?

If the distributions are changing then we cannot use the same model.

Q] How to determine if data is changing over time?

i.e. How to test if D_{Train} and D_{Test} have diff. distribu^n ?



For each D_{Train} and D_{Test} point, we will create a new point $(x_i' y_i')$

I will create a new dataset D_n' such that

D_n' :- D_{Train} :- $y_i' = 1$; $x_i' = (x_i\ y_i)$ (concat) in D_{Train}

D_{Test} :- $y_i' = 0$; $x_i' = (x_i, y)$ concat in D_{Test}

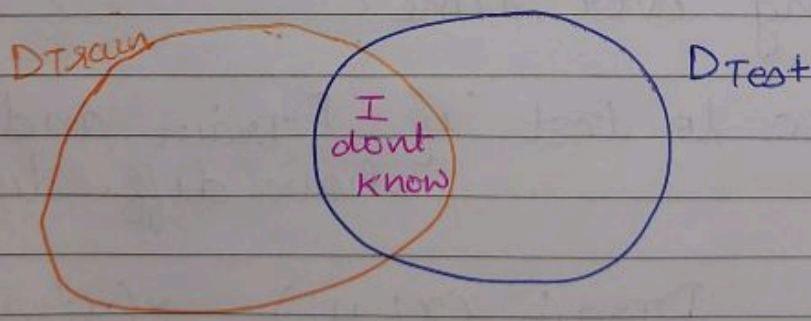
Now, build a binary classifier in D_n'

↓

we can use any Binary classifier

Eg : KNN

Binary classifier will be able to separate D_{Train} and D_{Test} to some extent

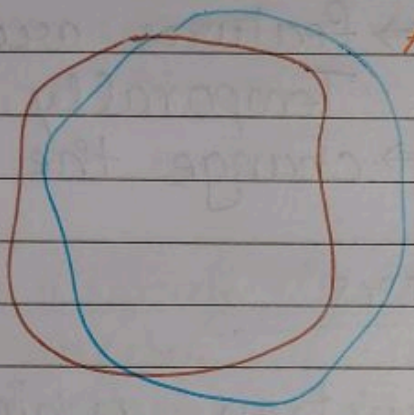


Lets say Binary classifier has accuracy of 70%

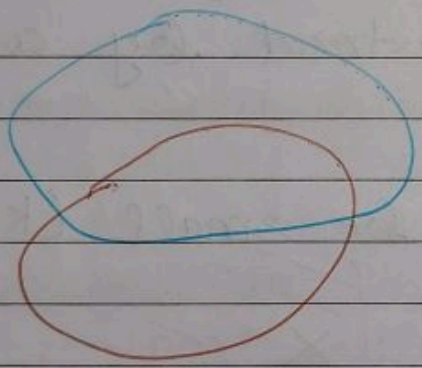
Means 30% of points are in "I dont know" region

70% Accuracy → Train and Test data can be separated partially

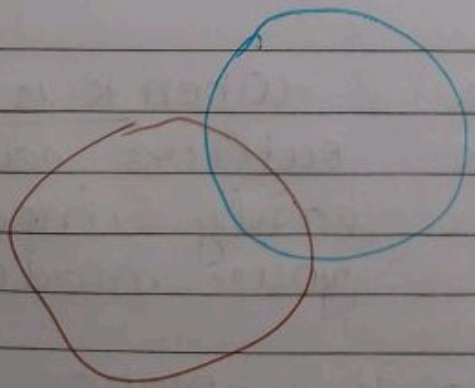
If we can separate them partially it means their distributions are different.



Almost Overlapping
We would get
LOW ACCURACY
It means the
DISTRIBUTIONS are
VERY SIMILAR



Less Overlap.
Binary classifier
accuracy is Medium
Distributions are
NOT VERY SIMILAR



Overlap is Very Low
Accuracy is HIGH
distributions are
VERY DIFFERENT

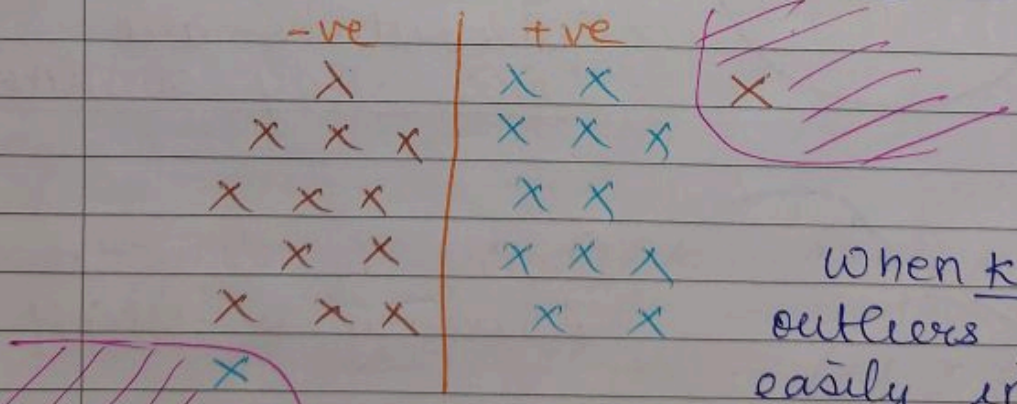
Hence

D_{Train} & D_{Test} ⟶ same distributions ✓
⟶ not same distributions
⇒ features are changing with time
⇒ features are not Temporally stable
⇒ change the features

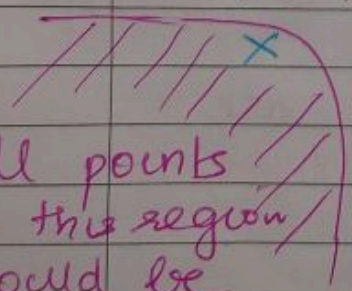
31.7 Impact of Outliers

A model is a function which can be easily understood by a decision surface.

For K-NN, let K be small (K=1)



When K is small, outliers can easily impact your model



all points in this region would be marked as +ve because of one Outlier

Less value of K is MORE Prone to Outliers

Lets say we are doing 10-fold C.V.

	Accuracy
K=1	97%
K=2	97%
K=3	97%
K=4	97%
K=5	97%
K=6	95%
K=7	92%

→ we should choose Largest value of k with Maximum Accuracy because lower values of K are more prone to outliers.

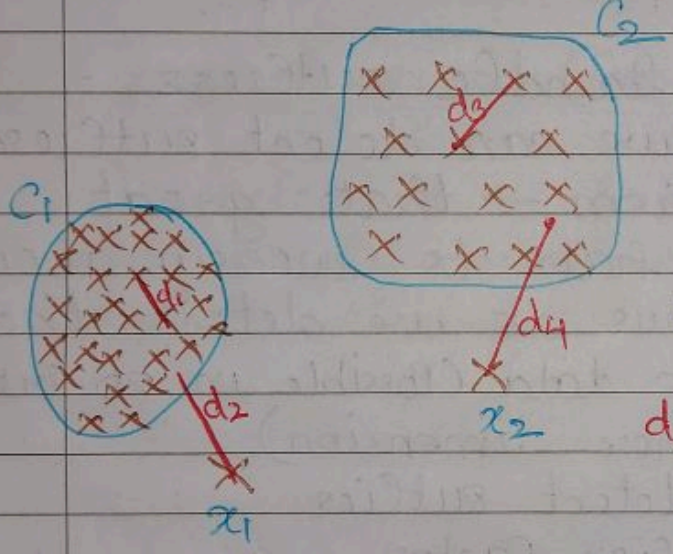
Other way to handle outliers:
Imagine if we can detect outliers and remove them → thats great. Then we dont have to worry about them. But how do we detect them?
→ Visualizing the data (Possible in 2D but not in higher dimension)
Technique to detect outlier
→ Local Outlier factor.

31.7 **Local Outlier Factor (LOF)** (Technique to detect outliers in data)

→ Inspired by ideas like K-NN.

Lets take a classificaⁿ setting
→ Take all -ve points. Find outliers.
→ —"— +ve —"—.
→ Remove them.

Consider group of -ve points and their cluster



C_1: very dense cluster
C_2: sparse cluster

x_1 and x_2 are outliers

d_i = avg dist of point considered from K-NN

Simple selection
→ Take each point x_i
→ Take its K-Nearest Neighbours
Lets K=5
→ Find avg distance to all 5 points
→ Sort all the points as per the distances.
→ d_4 is large. Hence its outlier.
Remove that point.
i.e. If Avg dist is high, then the point is outlier

Problem with this approach

Now x_2 will be declared as an outlier
But x_1 will NEVER be declared as outlier
Because $d_2 \approx d_3$; it is close to dense cluster.

If we have to declare x_2 as outlier, then all points with distance d_3 would be considered as outlier.

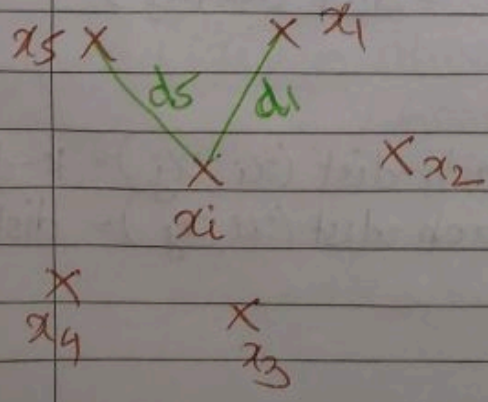
We have to compute Local density

The current method fails when we have more than one cluster.
One of the clusters is dense and other is sparse.

31.8 K-distance

K-distance (x_i) = distance of the K^{th} Nearest Neighbour of x_i from x_i

N(A) = Neighbour of A = set of all point belong to K-NN of x_i



d_5 = 5-distance.
d_1 = 1-distance

$$N(x_i)_{K=5} = \{x_1, x_2, x_3, x_4, x_5\}$$

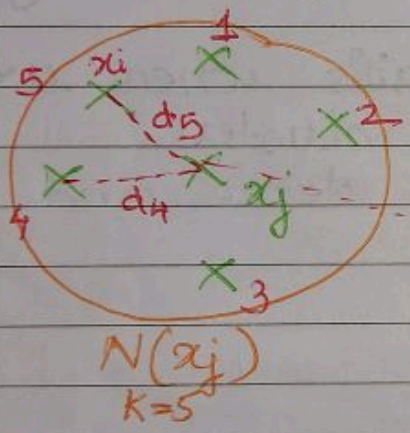
31.9 Reachability distance (A, B)

Reachealbility Distance (x_i, x_j) = Max [K-distance (x_j), dist (x_i, x_j)]

- K-distance (x_j): Distance of k^{th} NN of x_j from x_j
- dist (x_i, x_j): Actual distance between x_i and x_j

Geometrically,

Assume x_j and $K = 5$



① if $x_i \in N(x_j)$ then.

Reach-dist (x_i, x_j)
$= \text{Max}(d_5, d_4)$
$= d_5$ { Because d_5 is greater than d_4 }
$= \text{K-dist}(x_j)$

② else

{ $x_i \notin N(x_j) \Rightarrow x_i$ is not in Neighbourhood }

Reach-dist $(x_i, x_j) = \text{Max}(d_5, d)$
$= d$
$= \text{dist}(x_i, x_j)$

To sum it up:

if $x_i \in N(x_j)$ then Reach-dist $(x_i, x_j) = \text{K-dist}(x_j)$
else. " Reach-dist $(x_i, x_j) = \text{dist}(x_i, x_j)$

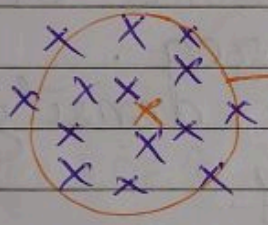
31.10 **Local Reachability Density : lrd(x_i)**

$$= \frac{1}{\sum_{x_j \in N(x_i)} \left\{ \frac{\text{reach-dist}(x_i, x_j)}{|N(x_i)|} \right\}}$$

All points belonging to $N(x_i)$

Number of elements in Neighbourhood (x_i)

This formula looks like Average.



$N(x_i)$ = Set of K-NN to x_i
$\therefore |N(x_i)| = K$
But Not always ; because 2 points may have same distance from x_i

So, we can think of LRD as
Average Reachability distance of x_i from its neighbourhood

lrd(x_i) = Inverse of Average Reachability distance of x_i from its Neighbourhood

We can also write LRD as

$$lrd(x_i) = \frac{|N(x_i)|}{\sum_{x_j \in N(x_i)} (\text{reach-dist}(x_i, x_j))}$$

$|N(x_i)| \rightarrow$ #points

$\sum_{x_j \in N(x_i)} (\text{reach-dist}(x_i, x_j))$ — some measure of distance

Intuitively, if we think about it, I am measuring density.
Density → number of points in some unit area.
Here we are computing density using local reachability

In denominator, I ~~can~~ am counting the sum of reach-dist in the locality of point x_i.

Hence we are calling it as

Local Reachability Density

Local → We are searching in Neighbourhood OR Locality

Reachability → We are counting Reachability distance

Density → Number of points by some measure of distance

31.11 **Local Outlier Factor (x_i)**

$$LOF(x_i) = \frac{\sum_{x_j \in N(x_i)} lrd(x_j)}{|N(x_i)|} * \frac{1}{lrd(x_i)}$$

Average lrd of points in Neighbourhood of x_i

If LOF(x_i) is large when.
Avg. component is ↑ OR $lrd(x_i)$ is ↓

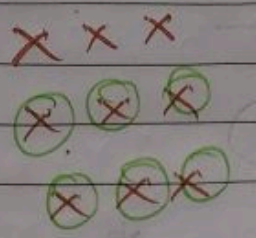
(Density of points in Neighbourhood is large) (My (x_i) own density is small)

LOF(x_i) is small when.
Avg Comp is ↓ OR $lrd(x_i)$ is ↑

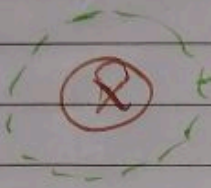
density around x_i is Large.

If LOF(x_i) is large. ⟹ Outlier
is small ⟹ Inlier

density (lrd) of x_i is small compared to its neighbours

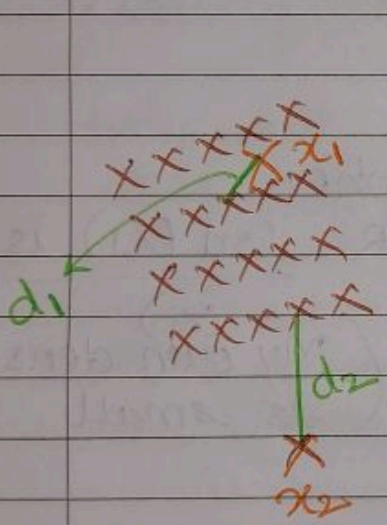


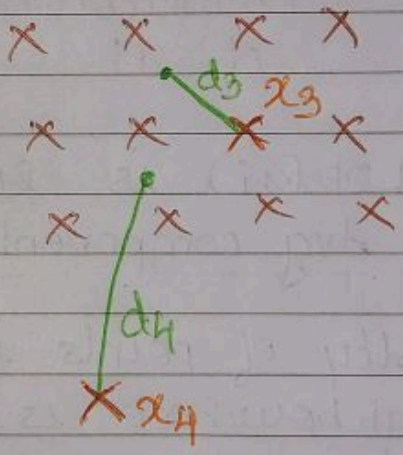
Large density of Neighbours



↦ small density compared to Neighbours

Other example:





x_1 : point in dense cluster
x_2 : point slightly away from dense cluster
x_3 : point in sparse cluster
x_4 : point far away from sparse cluster

Lets take x_1

All of its neighbours are very close to each other

Lets say $K = 5$.
Assume avg distance to 5-NN $= d_1$.
i.e. Average Reachability dist $= d_1$

Avg Reach-dist of $x_1 \rightarrow d_1$
$x_2 \rightarrow d_2$
$x_3 \rightarrow d_3$
$x_4 \rightarrow d_4$

Intuitively, we understand that
$d_1 < d_2 < d_3 < d_4$

We know $lrd = \frac{1}{\text{avg Reach-dist}}$ {Intuitively}

Since $d_1 < d_2 < d_3 < d_4$
$\therefore \frac{1}{d_1} > \frac{1}{d_2} > \frac{1}{d_3} > \frac{1}{d_4}$

$lrd(x_1)$ $lrd(x_2)$ $lrd(x_3)$ $lrd(x_4)$

Now $lrd(x_1)$ would be very similar to ~~lrd~~ lrd of all points close to x_1.

$$l.r.d(x_1) \approx l.r.d(x_i \in N(x_1))$$

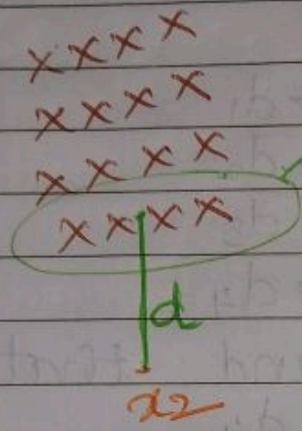
Density around x_1 would be roughly same as points ~~around~~ density around x_1.

$\therefore$ l.of (x_1) would be small
$\Rightarrow$ close to 1

__/__/____

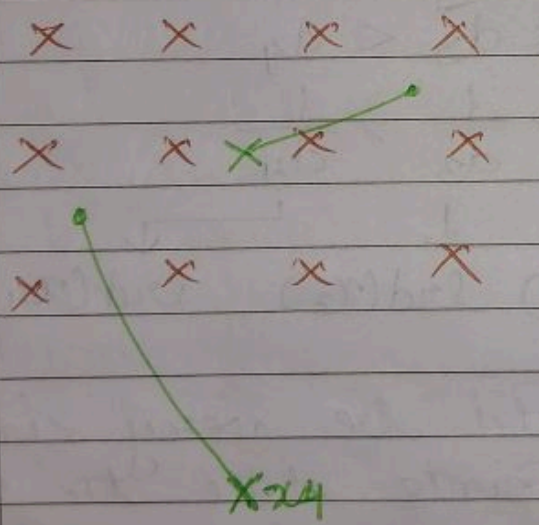
About x_2 :

$l.r.d(x_2) \ll l.r.d(x_i \in N(x_2))$



$\therefore l.o.f(x_2)$ would be large

About x_3 :



$l.r.d(x_3) \approx l.r.d(x \sim N(x_3))$

$\therefore l.o.f \approx 1$

About x_4

$l.r.d(x_4) << l.r.d(x \in N(x_4))$

$\therefore l.o.f(x_4)$ is large

$\therefore$ L·O·F $(x_1 \in C_1) \longrightarrow$ small
L·O·F $(x_2) \longrightarrow$ Large $\Longrightarrow$ Outlier
L·O·F $(x_3 \in C_2) \longrightarrow$ small
L·O·F $(x_4) \longrightarrow$ Large $\Longrightarrow$ Outlier

$\therefore$ Large L·O·F $\Longrightarrow$ Outlier

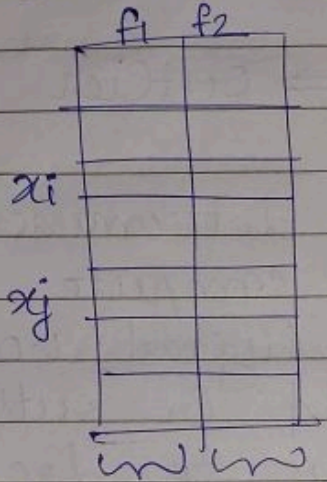
Hence, how do we determine Outliers ?
→ For each point x_i, compute L·O·F (x_i)
→ Pick points with highest L·O·F.
Most of them would be outliers
⇒ It is very subjective to decide the value of L·O·F to be considered as outlier (on where to set the threshold)
⇒ We would require Domain Knowledge.

Disadvantages of L·O·F

1. There is no interpretability of L·O·F value means, we know that if value is large, then its bad. But how bad it is
2. L·O·F value could lie anywhere from 0 to ∞. Imagine if we have a modified L·O·F which gives values from 0 to 1. This could act as a probability of a point which could be a outlier. This is called Local Outlier Probability

31.12 Impact of Scale and Column Standardization

Lets take a dataset with 2 features



Lets say values lie between 0 to 100

values lies between 0 to 1

This is called scale difference between f1 and f2.

Lets say $x_1 = [23, 0.2]$ } dist $(x_1, x_2) = 5$
$x_2 = [28, 0.2]$
$x_3 = [23, 1.0]$

dist $(x_1, x_2) = 5$
dist $(x_1, x_3) = 0.8$

Now here is a problem.
The distance varies because of scale difference between f1 and f2

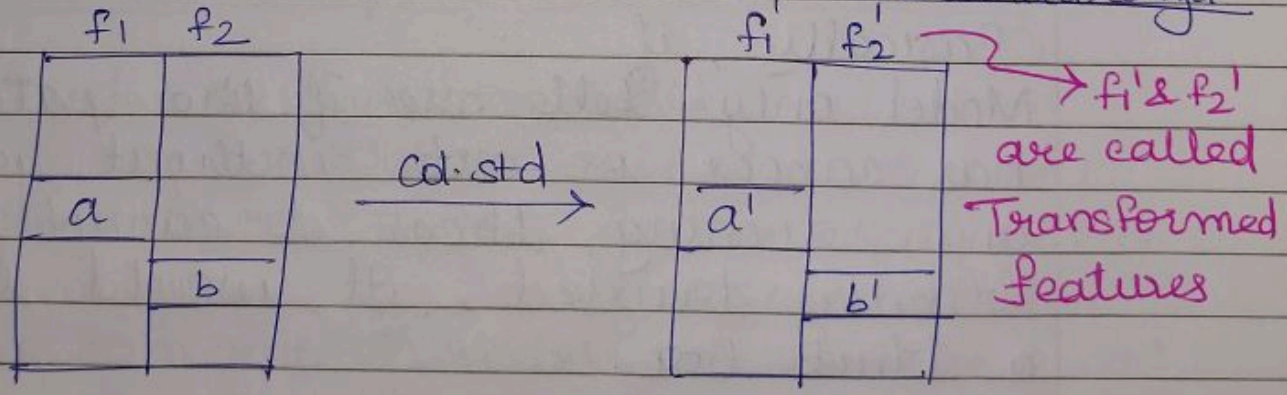
f_1 could have value from 0 to 100 so x_1 and x_2 are not very different from each other. Because out of the possible '100' differences, they have difference of only 5.

Between x_1 and x_3, max diff possible is 1. Because f_2 ranges from 0 to 1.
Now diff (x_1, x_3) = dist $(x_1, x_3) = 0.8$
so, x_1 and x_3 differ almost 80% of max allowable difference.

∴ Thinking logically:

$$d(x_1, x_3) > d(x_1, x_2)$$

Way to solve this issue ⇒ Column Standardizaⁿ



such that Mean(f_1') = 0 = Mean(f_2')
and Std-dev(f_1') = 1 = Std-dev(f_2')

$$a' = \frac{a - \mu_1}{\sigma_1} \quad \& \quad b' = \frac{b - \mu_2}{\sigma_2}$$

where $\mu_1, \sigma_1 \Rightarrow f_1$ } Mean & Std Dev
$\mu_2, \sigma_2 \Rightarrow f_2$ }

Since Euclidean dist can be impacted by differences in scale, and
Euclidean dist is used in K-NN then
[it is important to do column standardization before k-NN]

There are some techniques like Decision Trees which are scale independent.

31-13 Interpretability

Consider a medical application where the model tells if the patient has cancer or not.
Basically, if
Model only tells me if the patient has cancer or not without giving any reasoning, then it cannot be blindly trusted. It would be a black box

$$x_q \longrightarrow f \longrightarrow y_q$$

Doctor has to be very sure.
If model gives only class label, then it is insufficient ~~you~~ without reasoning

Model should give reasioning on why $y_q = 1/0$
It would be very useful to doctor.

Lets say x_q has 10 features and model tells that
f_5 coorresponds to test 5 and value is high
f_8 ⟶ test 8 ⟶ Low.
& hence, the patient has cancer ($y_q = 1$)
Doctor understands these tests and those are medical test.

A Model which gives an explanation are called Interpretable Model.
Model which gives no explanation are Black Box model.

Is K-NN interpretable model or not?

Depends on Domain Model.
Lets say K = 7
Now $x_q \longrightarrow y_q = 1$; Reasoning: all 7 NN of x_q are labelled as 1
We can give 7 Vectors of each NN

K-NN is interpretable when d is small and K is small.
This changes from domain to domain
→ True in Medical Domain

31.14 **Feature Importance & Forward Selection**

Lets say we have 10 features
$f_1, f_2 \dots f_{10}$

$D_n^{10} \rightarrow f \rightarrow$ if the model can tell which features are more useful.
(model)

Eg: If model f: predict height
and features are f_1: ~~height~~ weight of person
f_2: hair colour
useful feature — f_3: gender
f_4: country
f_5: skin colour

Feature Importance → sort features by important for classificaⁿ task in order of importance.
→ Useful in understanding model better
→ Interpretability increases

K-NN doesn't give us feature importance by default.

Forward Feature Selection

Given large dimension, people try to reduce dimension.

1000-dim → 100 dim

One way is PCA
But PCA ~~does~~ doesnt return the useful or important features.

One technique to get that is forward feature selection

$D = \{Training, Test\}$

(1) We have all features $f_1, f_2 \cdots f_d$.
(2) Assume our model is K-NN
(3) I will train model only using f_1

only f_1 → [Tr. | Test / Model] → Accuracy on Test data using only $f_1 = a_1$
only f_2 → [Tr. | Test / Model] → $= a_2$
⋮
only f_d — — $= a_d$

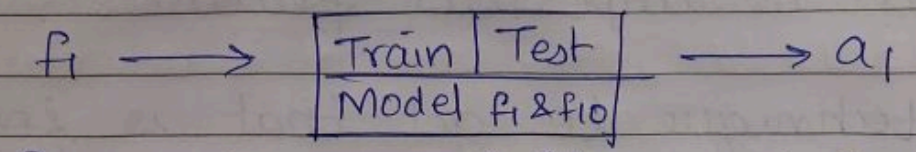
(4) Check which of them has highest accuracy.
Say f_{10} has highest Accuracy.

(5) Now my model becomes

$\{f_1, f_2, \cdots f_9, f_{11} \cdots f_d\}$ [Train | Test / Model | f_{10}] → successful feature

Rest of feature

⑥ Will train ~~f~~ Model using f1 and f10



Now, say a_5 is largest.
So, will finalize f_{10} and f_5 are best features.

⑦ At end of 2 Iterations;

$\{f_1, f_2 \cdots f_4, f_6 \cdots f_d\}$ — Remaining features.

$\{f_{10}, f_5\}$ — Selected features.

One thing which people argue is
→ In first iteration, we got the accuracies for features as
$f_{10} > f_6 > f_8 \cdots$
So why didnt we take f_6 as second best feature.

It is because if f_6 and f_{10} are exactly same/similar feature (in error) added then they would have same/similar accuracy

At each stage, we ask that given the best feature, which new feature adds the most value (highest accuracy).

Backward Selecⁿ:
It takes all features and then removes feature in every iteration which adds least value.

Considering Time Complexity:

At each iteration, we are training and testing 'd' models.

Iteration 1 ⇒ Traing & Testeng ⇒ d models
2 d-1
3 d-2

Hence a LOT of time complexity is there. We should use it only when we have time.

Forward/Backward Selection ~~ta~~ techniques are Model Agnostic ⇒ Means they dont caree which model it is.

Disadv: Lot of time

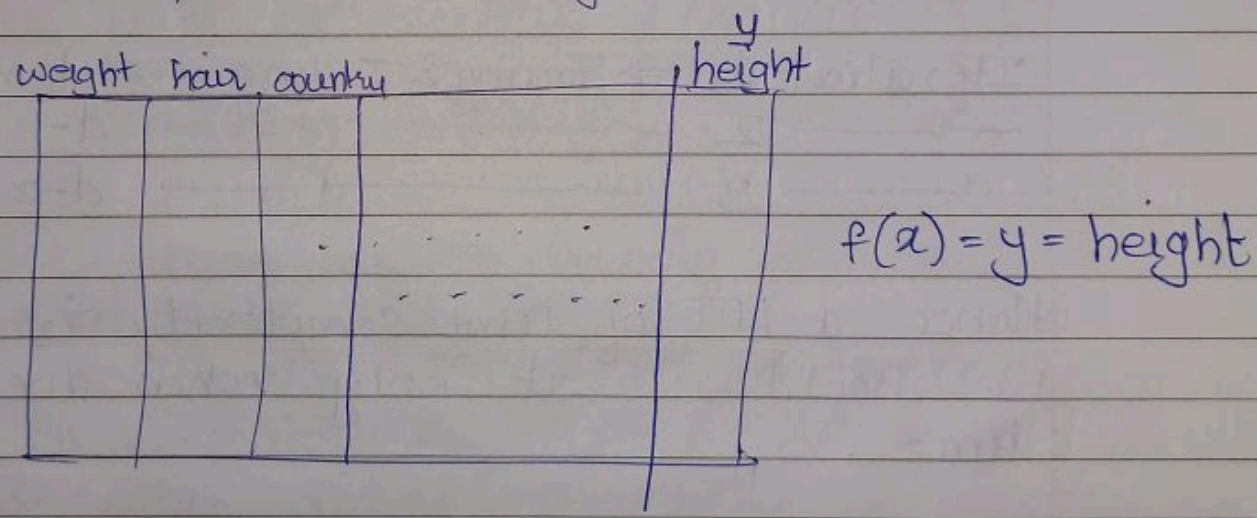
31.15 Categorical and Ordinal features

We remember Iris dataset which had 4 features : PL, PW, SL, SW
(Real valued Numbers)

and we are given datapoint $x_i \in \mathbb{R}^d$

But in real world, data is not always real valued.

Eg: we are given dataset and asked to predict height



(1) weight ⇒ numerical ⇒ we can take value as is

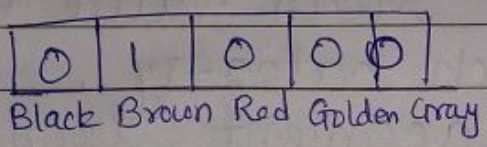
(2) Hair colour ⇒ non-numerical
⇒ {Black, Brown, Golden .... }

① Give a number to each colour
② something like
{Black, Brown, Red, Golden, Gray}
1 2 3 4 5

Numbers are ordinal ; means they can be compared
Something like 3 > 1 which would imply Red > Black ⇒ Absurd.

One solution to it : One hot encoding

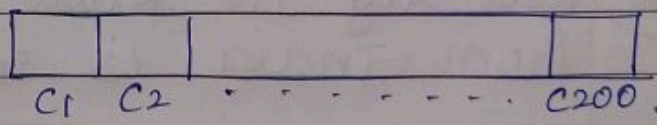
I would create an array for hair colour



if xi : Brown.

So one hot encoding is the binary vector of the size of Number of distinct elements

Problem with this approach :
Consider the feature country ; there are total 200 countries
∴ Vector would be of size 200



Also, the vector would be very sparse. As the person would belong to only 1 country.

If the number of distinct values for a categorical feature is large then One hot encoding can create sparse and large vector.

(3) One hack is to replace country vector with Avg height of people in the country because our objective is to create a model to predict the height of a person.

Replace the country with Avg height
We are taking Mean value of height for each category.
Its Mean Replacement technique

Another way to convert categorical feature to domain feature is through domain knowledge.

Eg: Country $\xrightarrow{\text{Replace}}$ distance from India
(categorical) (numeric).

(4) Lets say we know that distance from India is related to height.

Ordinal Feature

Lets say a feature is

f: How do you rate Indian food ?
{v.Good, Good, Avg, Bad, v.Bad}

⇓

Categorical feature

There is Logical ordering among feature

So, we can convert to Numeric
Like: V.Good = 5 ..... , V.Bad = 1

Problem with this approach

f_i	f_i'	f_i''
V.Good	5	10
Good	4	6
Avg	3	3
Bad	2	2
V.bad	1	1

} We can give any numeric values as long as they are Ordinal

The assignment of values depend on domain knowledge.

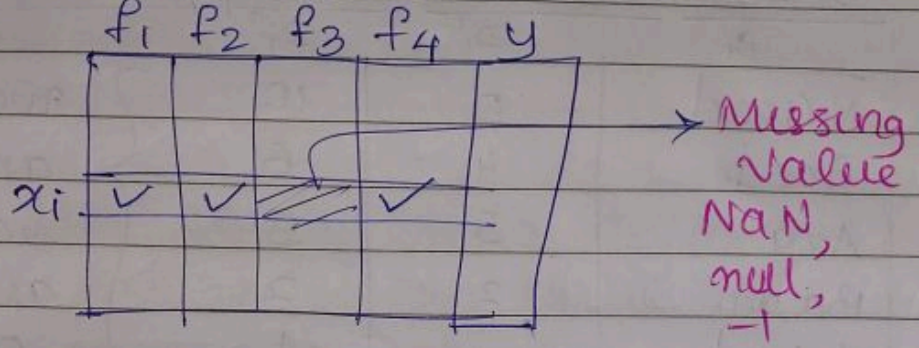
So we have following techniques

~~One ho~~ Catogorical
- One hot technique
- Mean Replacement
- Domain knowledge

} Problem dependent.

31.16 Missing Values Imputation

Lets assume I have 4 features



And value is not present for one feature.

It may be due to
→ Data Corruption
→ Collection error
} → Very Common

There are numerous ways to featureize it :

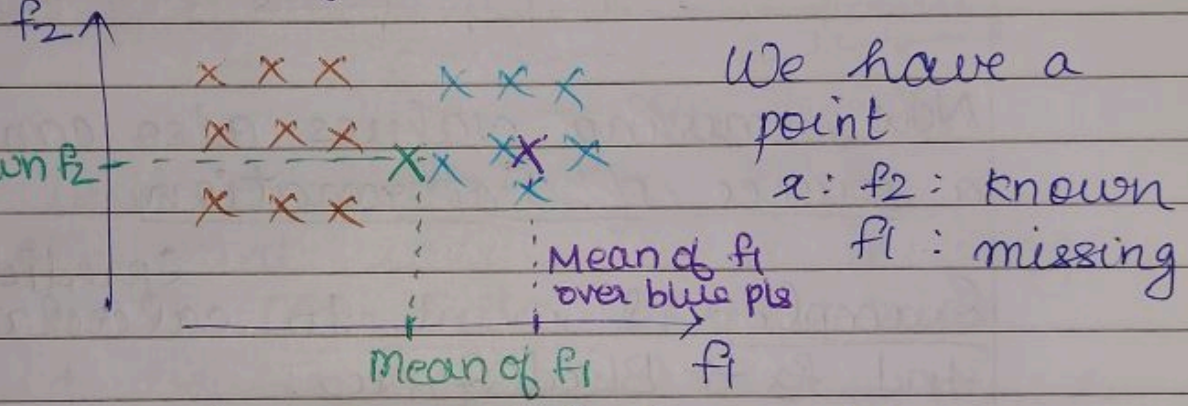
① Imputation: 3 ways to do it
→ Mean replacement
→ Median replacement
→ Most frequent value (mode)

Strategy : Take all non-missing value for the feature ; Take Mean, Median, mode.

Sklean. → Imputer Method

While doing a classification task where the output is 0,1
→ We impute based on class label.
→ $x_i \rightarrow f3$: Missing and $y_i = 1$
then instead of computing mean of all f3, compute the average of those f3 for which y_i is 1

Geometrically



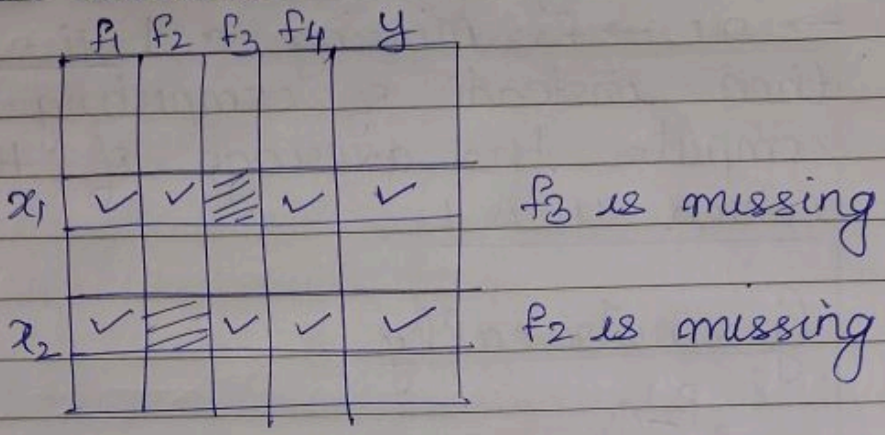
Lets say I calculated f_i as the mean of all f_i values, then my point x would lie over X as shown

Now, we know that $y = f(x) = 1$ (class)
So, we would calculate mean of only those points which has $y_i = 1$.
So, my point x would lie in (X) as shown

Logically X is correct placement given $y = 1$

③ New Missing Value feature

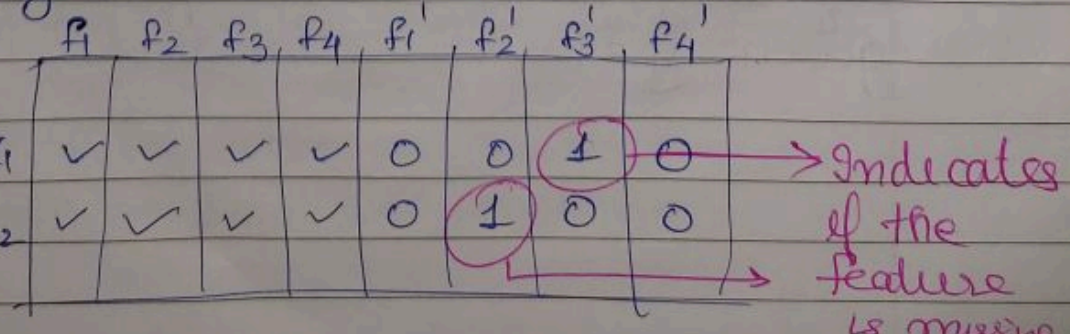
For given dataset



Now, missing values also can be a source of information

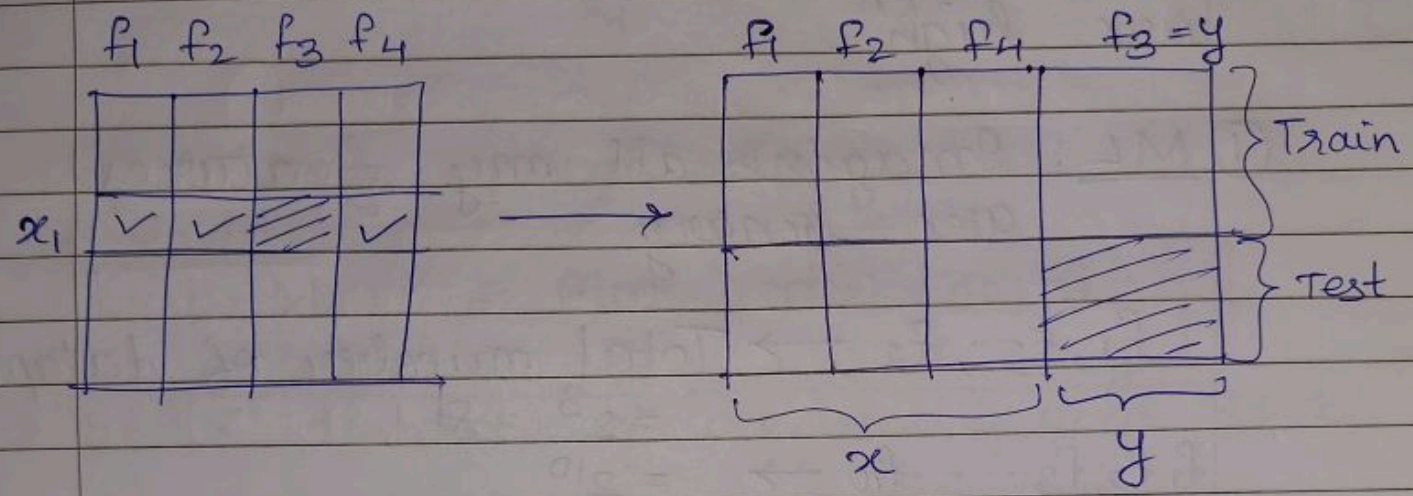
Example: I want to calculate (predict) height
And f_3 → Blood Group.
If my feature f_3 is missing for a particular region, it would mean the people are below poverty line and they dont have expenses money to conduct the blood test

What we can do is add another set of features (f_1', f_2', f_3' ----) which would indicate if the feature is missing



④ Model Based Imputation

We can take any model (say KNN)



We will take $f_3 = y$
All the points where f_3 is missing would be Test Data. Rest points would be Train Data.
We would calculate the $f_3 = y$ ^ for Test data based on Train data.

K-NN is used a lot of time for Model Based imputation
Because K-NN has a feature of Neighbourhood.

For given point x_1 ; f_3 is missing

So, if x_2, x_3, x_4 are in neighbourhood, their features are similar to x_1.
∴ We get idea of f_3 for x_1

3147 Curse of dimensionality

Explains the multiple phenomenons that happens when dimensions are high.

① ML: Imagine all my features are binary.

$f_1, f_3; f_2 \rightarrow$ Total number of datapoints $= 2^3 = 8$.

$f_1 f_2, f_3 \cdots f_{10} \rightarrow = 2^{10}$

As dimensionality ↑, the number of datapoints to perform good classification/regression for a good model increases exponentially

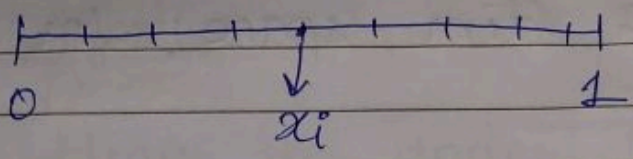
Hughes Phenomenon

If 'n' → size of dataset is fixed
[Performance decreases as Dimensionality increases].

② Distance functions (esp. Euclidean dist)

The intution we have about distances in 3D is not valid for high dimensional space.

Lets say in 1-D world : n-random points



If I take point x_i

$$\text{Dist-Min}(x_i) = \min_{x_j} \{ \text{dist}(x_i, x_j) \text{ where } x_i \neq x_j \}$$

It is distance of nearest point to x_i

$$\text{Dist-Max}(x_j) = \text{Max}_{x_i} \{ \text{dist}(x_j, x_i) \text{ where } x_j \neq x_i \}$$

Typically

$$\frac{\text{Dist-Max}(x_i) - \text{Dist-Min}(x_i)}{\text{Dist-Min}(x_i)} > 0 \quad \text{when } d = 1, 2, 3$$

Now, when d increases significantly, the above term reaches near to 0.

$$\therefore \lim_{d \to \infty} \left(\frac{\text{distMax}(x_i) - \text{distMin}(x_i)}{\text{distMin}(x_i)} \right) \longrightarrow 0$$

It means the since the above terms becomes 0,

$$\text{distMax}(x_i) \approx \text{distMin}(x_i)$$

This is an alarming thing.

In high dim space, for n-random points.

$$distMax(x_i) \approx distMin(x_i)$$

⇓

Every pair of points are equally distant from each other.

i.e. $dist(x_i, x_j) \approx dist(x_i, x_k)$

In this scenario,
K-NN does not make sense.
Hence K-NN doesnt work well in high dimensional space.

Solution: Instead of Euclidean dist, use other distances like cosine similarity.
Hence, people prefer cosine similarity in high dimensional space problems like Text classification

Twist/caveat:

High dim & Dense → Impact of dimensionality is high

High dim & Sparse → Impact of dimensionality is Lower

↓

non uniform/random spread of data

The entire concept is with assumption that points are distributed randomly and uniformly

③ Overfitting and Underfitting

As d↑, the overfitting also increases. We would learn this during Linear Regression

→ Solutions → ① Forward feature selection
Pick the most useful subset of features
② Use dimensionality reduc^{n}. like PCA, tSNe. They donot use class label and are not classifica^{n} oriented.
③ Some people also change distance measures → cosine similarity instead of Euclidean distance.
⇒ Sparse representa^{n} instead of dense representation.
BoW ←
W2V ←

31-8 # Bias Variance Tradeoff

We talked out underfitting & overfitting
(K=n) (K=1)

How do we mathematically analyze underfitting & overfitting?

→ Bias Variance Tradefff.
→ Most used idea in statistical ML.

We know generalization error as the error on future unseen data.

Generaliza^n Error = Bias^2 + Variance + Irreducible Error

↓

(future unseen data of M)

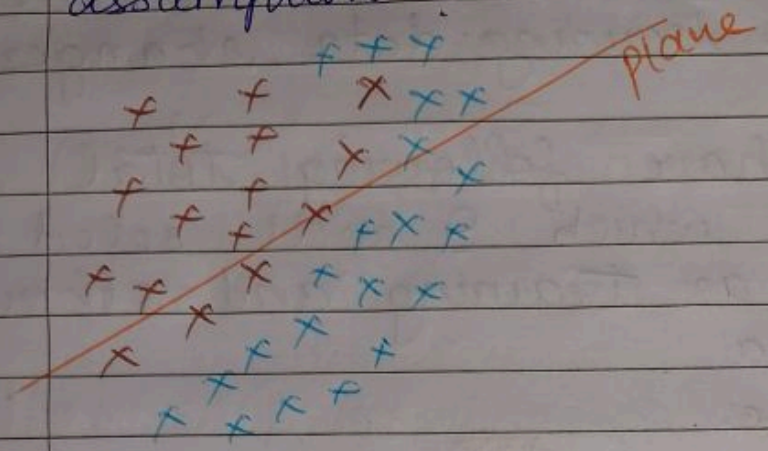
We would learn Math proof of same in linear Regression

Irreducible error → error cannot be reduced further

Bias error → Error due to simplifying assumptions.

High Bias ⇒ Underfitting

What is error due to simplifying assumption ?

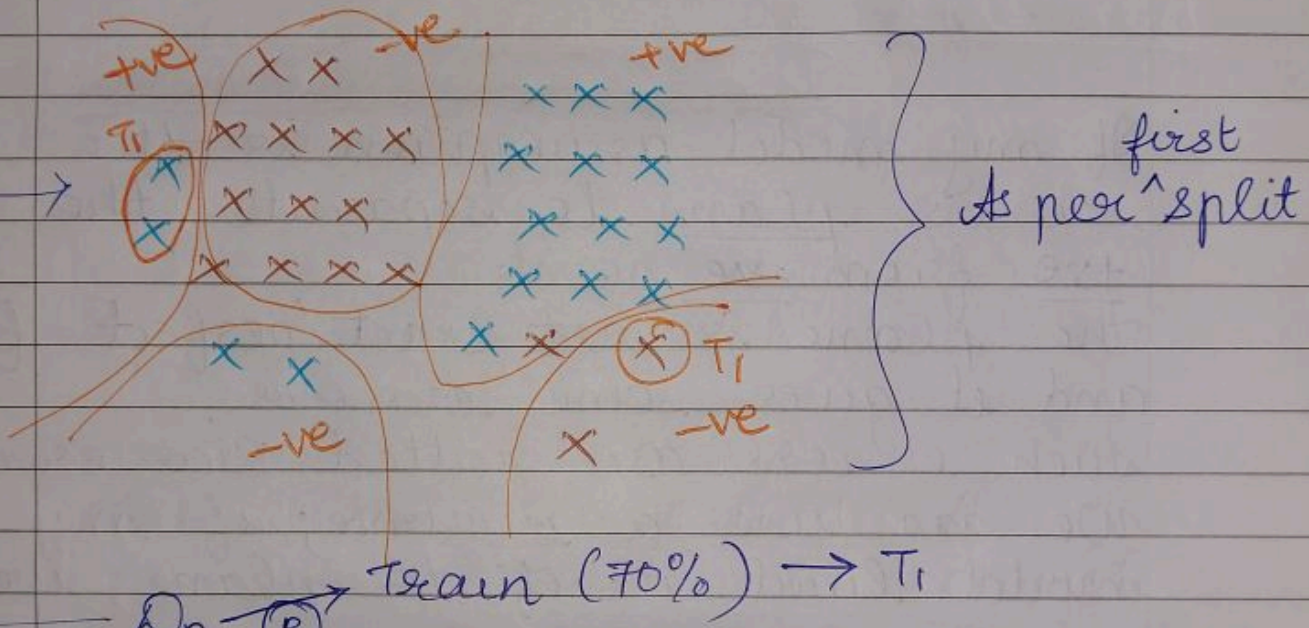


If my model assumption is the we use plane to separate the +ve from -ve points
The plane is not exact/perfect fit. and it gives some errors.
Such errors are called Bias error.
We can use a curve; if my model limit itself to plane, we get Bias Errors.

Lets say in K-NN, if $K = n$.
and 80% of points are -ve
20% of points are +ve
So, for every query point x_q, y_q = -ve
Hence, dominant class becomes the class label of x_q. Biass error would be high
Hence, high Bias would be Underfitting

High Variance: means the difference in model OR how much a model changes as training data changes

Imagine I have following total data out of which I will select some data as training and other as test data.



D_n (R) → Train (70%) → T_1

→ Test (30%)

I split Dn one more time

D_n → Train (70%) → T_2

→ Test (30%)

Model is nothing but decision surface

Now when one of the point from T_1 is replaced by T_2, the decision surface changes.

When K=1, small change in dataset results in very diff model (= decision surface).

Small change in dataset → Large change in model → High Variance Model.

↓

Overfit Model

as K↑ ; Variance ↓

So if we want to :

Generalize Error ↓ = Bias² ↓ + Var ↓ + Irreducible Error.

No underfit No overfit

K=1 ; Bias ↓ Var ↑.

As K↑ ; Bias ↑ (slightly) Var ↓ (drastically)

K=n ; Bias ↑ (Very High) Var ↓ (very Low)

Lets say

K=1 ; 10 + 100 + 3 = 113 overfit

K=5 ; 12 + 10 + 3 = 25 ✓

K=n ; 100 + 2 + 3 = 105 underfit

~~178~~
178

So, we want to design a model that balances both Bias & Variance

∴ Bias Variance Tradeoff ⇒ Theme of ML

↳ Overfit Vs Underfit

31.19 Intuitive Understand of Bias and Variance.

We have a dataset

D → D_{Train} → Model
D → D_{Test}

Case 1: High Bias OR Underfit
Using the "Model", I can compute two types of errors.

Train Error: difference between y_i and $\hat{y}_i$ on train data

$$\text{Train Error} = \text{diff}(y_i, \hat{y}_i)_{Train}$$

Test Error: $\text{diff}(y_i, \hat{y}_i)_{Test}$

If Train Error = ↑ then Bias ↑.

② High Variance OR Overfit

If Train Error ↓ and Test Error ↑
⇒ Overfit
⇒ Variance ↑.

If Train data changes slightly and there is high model change.
⇒ High Variance

31.21 Best and Worst case of Algo

For any ML algo, it is imp to understand Best and Worst Case.

① Regarding K-NN
① when dimensionality is small (<10)
⇒ K-NN is a good algo.
② when dimensionality is large
⇒ We get into curse of dimensionality
⇒ Interpretability reduces
⇒ Runtime complexity of LSH / K-D tree ↑

② Low Latency System:

⇒ System where we want fast search results
Eg: Google Search, Amazon products

K-NN should not be used in Low Latency Systems

(3) find right distance measure

→ then K-NN is good. (Eg: cosine similarity)

For text, cosine dist works well.

For Genome data, Euclidean dist works

If someone gives Similarity Matrix or dist Matrix → K-NN

We have to be careful while using K-NN.

32.1 NAIVE BAYES ALGO

Simplistic or unsophisticated algo using Baye's Theorem

Imagine I have a datapoint x with n features and k possible outcomes or classes.

$$x = \langle x_1, x_2, x_3 \ldots x_n \rangle \Rightarrow d = n$$
$$y_i \in \{c_1, c_2, \ldots c_k\}$$

$P(C_k|x)$ = Probability of a class label given data point x. (variable)

Using simple Bayes Theorem.

$$\underbrace{P(C_k|x)}_{\text{Posterior}} = \frac{\overbrace{P(C_k)}^{\text{Prior}} \cdot \overbrace{P(x|C_k)}^{\text{Likelihood}}}{\underbrace{P(x)}_{\text{evidence}}}$$

We know that
$$P(C_k) \cdot P(x|C_k) = P(x \cap C_k)$$

Now lets say I calculate the probability for each of the classes

$P(C_1|x)$
$P(C_2|x)$
$P(C_3|x)$ → say, its Largest; we will assign x to C_3
$\vdots$
$P(C_k|x)$

$P(C_1|x)$
$P(C_2|x)$
$P(C_3|x)$
⋮
$P(C_K|x)$

} All of them have same denominator, hence we need to consider the numerator.

Also the denominator doesn't contain the term C_K (class) which is the evaluation criteria. So, we can ignore denominator.

So we can say

$$P(C_K) \cdot P(x|C_K) = P(x \cap C_K) = \underbrace{P(C_K, x)}_{\text{another way of writing}}$$

Since denominator is same for all

$$P(C_K|x) \propto P(C_K, x)$$

This is called Joint Probability Model.

We know $x = \langle x_1, x_2 \cdots x_n \rangle$

$$\therefore P(C_K, x) = P(C_K, \langle x_1, x_2 \cdots x_n \rangle)$$
$$= P(\langle x_1, x_2 \cdots x_n \rangle, C_K)$$

We know that ~~$P(A|B) = P($~~

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

... as per definition of cond. probab.

∴ Similarly

$$P(\overbrace{x_1}^{A}, \overbrace{x_2, \cdots x_n, C_K}^{B}) = P(x_1 | x_2, x_3, \cdots x_n, C_K) * P(x_2, x_3, \cdots x_n, C_K)$$

Keeping the first term as is, lets break the second term

$$= P(x_1 | x_2, x_3 \cdots x_n, C_K) \rightarrow \text{FIRST TERM}$$
$$* \underbrace{P(x_2 | x_3 \cdots x_n, C_K) * P(x_3, x_4 \cdots x_n, C_K)}_{\text{SECOND Term.}}$$

So, if we keep on breaking further

$$= P(x_1 | x_2, x_3 \cdots x_n, C_K)$$
$$* P(x_2 | x_3 \cdots x_n, C_K)$$
$$* \cdots$$
$$* P(x_{n-1} | x_n, C_K)$$
$$* P(x_n | C_K)$$
$$* P(C_K)$$

$\rightarrow P(C_K, x_1, \cdots x_n)$

$$P(C_K | x) \propto P(C_K, x)$$

Computing the probability calculation is very complex.

Eg:

$P($weight = 180 | hairColor = Black, hairLength = 5cm, skinColour = Brown, eyeColour = Black$)$

It is difficult to compute probability with exact same features.

So, we go with Naive Assumption of conditional independence.

$P(A|B,C) = P(A|C)$
A is conditionally dependent on B.
hence, we can remove B.
OR
A is conditionally independent of B.

$P(x_i | x_{i+1}, \dots x_n, C_k) = P(x_i | C_k)$
x_i is conditionally independent of $x_{i+1}, x_{i+2}, \dots x_n$.

Hence, we are making Naive assumption that features are conditionally independent of each other.

$\therefore P(C_k | x_1, x_2 \dots x_n) \propto P(C_k)$
1st Term ← $* P(x_1 | C_k)$
2nd Term ← $* P(x_2 | C_k)$
$\vdots \quad \vdots$
n^{th} term ← $* P(x_n | C_k)$

$\therefore P(C_k | x_1, x_2 \dots x_n) \propto P(C_k) \cdot \prod_{i=1}^{n} P(x_i, | C_k)$

So, we arrive at a fundamental term of exact probability.

$$P(C_K | x_1, x_2 \cdots x_n) = \frac{1}{Z} P(C_K) \cdot \prod_{i=1}^{n} P(x_i, C_K)$$

So, we will compute
$P(C_1|X)$, $P(C_2|X)$, $P(C_3|X)$ ..... $P(C_K|X)$
and compute max value out of it

This is called Maximum a Posteriori Rule
→ These are Posterior Probability.

32.2 Toy Example

We have a dataset D with features like

Outlook $\in$ {Sunny, Overcast}
Temperature $\in$ {Hot, Mild, Cold}
Humidity $\in$ {High, Normal}
Wind $\in$ {Weak, Strong}
Response $\in$ {Yes, No}

Response indicates whether its good time to play.
Its a binary classification Task.
with 4 categorical features.

So, we have 2 stages
- Training (learning)
- Test

Refer Shatterline Blog for exact Dataset

For K-NN there is no training stage
There is only Test/Evaluation Stage.

Building K-d tree / LSH we require training stage.
Training phase → also called Learning phase.

We have to predict

$P(\text{Response} = \text{yes} \mid x_q)$
and
$P(\text{Response} = \text{No} \mid x_q)$

→ The higher probability will be y_q.

We saw in Naive Bayes that

$$P(C \mid f_1, f_2, f_3, f_4) \propto P(f_1 \mid C) * P(f_2 \mid C) * P(f_3 \mid C) * P(f_4 \mid C) * P(C)$$

Referring the dataset (Shatterline blog)

$$P(C = \text{yes}) = \frac{9}{14}$$

Now lets consider the individual probabilities

P(Outlook = O | class (Yes/No))

	Frequency		Probability in class	
Outlook	Yes	No	Yes	No
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	total = 9	total = 5		

Similarly count for rest of features.

In training phase, we compute all the likelihood probabilities

Time Complexity : Roughly O(ndc)

- n : num of data points
- d : features
- c : class labels

Considering the small # of d and C.
Time Complexity : O(n)

Space Complexity :

We need to save P(fi/c) → 'd' space
and P(c) → 'c' space

∴ Space Compl : O(d * c)

Overall Time Compl : $O(nd)$ } Training
Space Compl : $O(dc)$ } Phase.

Lets say x_q = (Outlook = Sunny,
Temperature = Cool,
Humidety = High,
Wind = Strong).

y_q = ?

$P(C = Yes \mid x_q) \propto P(Sunny \mid C = yes)$
$* P(Cool \mid C = yes)$
$* P(High \mid C = yes)$
$* P(Strong \mid C = yes)$

Calculated in Training Phase ← $\propto$ 2/9 * 3/9 * 3/9 * 9/14

$\propto$ 0.0053

$P(C = No \mid x_q) \propto \frac{0.0025}{0.0205}$

} As per Aposterior Rule; y_q = No

Test Time Complexity (Test data)

$= O(d \cdot c)$

Since we have to look up 'd' features for 'c' classes.

* Naive Bayes is much better as compared to K-NN in terms of space complexity.
* Naive Bayes is much more memory efficient at runtime

32.3 Naive Bayes on Text data

→ Very popular technique for Text data
→ First of techniques used for email Spam filters.

Spam Filter :- email → Spam
↳ Not Spam.

Review : +ve/-ve.

Task at hand:

	Class
← text 1 →	1
← text 2 →	0
	1
	⋮
	0

Task : $P(y_q = 1 | text_q)$ } compare which is greater
$P(y_q = 0 | text_q)$ }

Step we follow:

text → Stopwords
Stemming } Preprocessing
n-grams
↓
Binary BoW
→ we just check if word is present or not
→ dont count # occurence

Binary BoW

Now text $\longrightarrow$ $\{w_1, w_2, \ldots\ldots w_d\}$
represented

$$P(y=1|text) \propto P(y=1 | \{w_1, w_2 \ldots\ldots w_d\})$$

$$= P(y=1) * P(w_1|y=1) * P(w_2|y=1) * \ldots * P(w_d|y=1)$$

$$\propto P(y=1) * \prod_{i=1}^{d} P(w_i|y=1)$$

Illy $P(y=0|text) \propto P(y=0) * \prod_{i=1}^{d} P(w_i|y=0)$

Now

$$P(y=1) = \frac{\#\text{ training points with } y=1}{\text{Total \# of training points}}$$

$$P(y=0) = \frac{\#\text{ training points with } y=0}{\text{Total \# of training points}}$$

$$P(w_i|y=1) = \frac{\#\text{ datapoints (Train) which contain word } w_i}{\#\text{ datapoints (Train) with } y=1}$$

We have divided Train data in 2 parts

$x_i \rightarrow$	y_i	
$text_i \rightarrow$	1	} Train data with $y=1$
	⋮	
	0	} Train data with $y=0$
	⋮	
	0	

Hence for Text Classificaⁿ ;

Spam Detection
Polarity of a review
} ~~Not~~ Naive Bayes is a very good baseline
↓
Benchmark algo.

We usually compare the complexity of other algo with Naive Bayes.

Lets say :
Naive Bayes → 98% Accuracy
Logistics Regression →
G.B.D.T →
D.L →

Suppose.
Deep Learning gives 98.5% Accuracy
& Naive Bayes ~ 98 % ~
(Simplistic Algo)

So D.L. gives only small incremental accuracy over N.B.

32.4 Laplace Additive Smoothing

Coming to text processing

Training :- We have compiled

$P(y=1)$; $P(y=0)$

$P(w_1|y=1)$; $P(w_1|y=0)$
$P(w_2|y=1)$; $P(w_2|y=0)$
$P(w_3|y=1)$; $P(w_3|y=0)$
⋮
$P(w_m|y=1)$; $P(w_m|y=0)$

} likelihoods

Lets go to Test stage:
After all preprocessing
we get $text_q = (w_1, w_2, w_3', w')$

Lets say w' is not present in the set of words which we finalized after preprocessing

w' is a completely new word and hence not used in training data at all OR if we are using bigrams or trigrams, then the sequence is not present
~~in text~~

Now

$P(y=1 \mid text_q) = P(y=1 \mid w_1, w_2, w_3, w')$

$= P(y=1) *$
$P(w_1 \mid y=1) *$
$P(w_2 \mid y=1) *$
$P(w_3 \mid y=1) *$
$P(w' \mid y=1)$

We have already computed

We need to find this

Challenge: How to get $P(w' \mid y=1)$ & $P(w' \mid y=0)$

→ One may say that why worry about one word. We can completely ignore it.
But ignoring it would imply $P(w' \mid y=1) = 1$ & $P(w' \mid y=0) = 1$ which is incorrect.

Now

$$P(w' \mid y=1) = \frac{P(w', y=1)}{P(y=1)}$$

(Intersection)

$= \frac{\#\text{ points such that } w' \text{ occurs and } y=1}{\#\text{ points where } y=1} \rightarrow n_1$

$= \frac{0}{n_1} = 0 \Rightarrow$ We cannot use this as it will make entire term as 0

The workaround is
Laplace Smoothing (OR Additive smoothing)
(Not to be confused with Laplacian smoothing)

We have seen

$$P(w' \mid y=1) = \frac{0+\alpha}{n_1+\alpha K}$$

K = # distinct values w' can take

So, here w' can be present → 1
OR not present → 0
∴ K can take 2 values; $K = 2$

In a non text processing case,
I want to take value of feature f_i

$$P(f_i = a \mid y=1) = \frac{0+\alpha}{n_1+\alpha K}$$

↓
# of distinct values that f_i can take

α can be any numerical value.
Typically $\alpha = 1$; not always

Why is all this useful?

Let say $n_1 = 100$; and $K = 2$ {since w' can be 1 or 0}

$$P(w' \mid y=1) = \frac{0+\alpha}{100+2\alpha} = \frac{\alpha}{100+2\alpha}$$

Case 1: $\alpha = 1$; $P(w'|y=1) = \frac{1}{102} \neq 0$

Now, a question might arise is why to go through all such complex calculation.
Why not replace $P(w'|y=1)$ with $\epsilon = 0.001$ (very small value).
There is mathematical foundation behind this.

Case 2: α is large ; $\alpha = 10000$

$$P(w'|y=1) = \frac{0 + 10k}{100 + 20k} \rightarrow \frac{10k}{20100} \approx \frac{1}{2}$$

When α is large and we know that $w' = 0$ } Two possibilities.
$w' = 1$ }

$$P(w'|y=1) = P(w' \neq y=0) = \frac{1}{2}$$

We say that both possibilities are equally likely.

This is much better assumption where we say that w' is equally likely to appear in +ve class as in -ve class

Hence, with Laplace Smoothing

$$P(w_i \mid y=1) = \frac{(\#\text{datapoints with } w_i \text{ and } y=1) + \alpha}{(\#\text{datapoints with } y=1) + \alpha k}$$

for all words (~~Both~~ Irrespective of present in training data or not)

This is a Smoothing operation. Also called as Negative Smoothing.

Smoothing:

Consider we have a sparse dataset.

$$P(w_i \mid y=1) = \frac{2+\alpha}{50+k\alpha} \quad \text{where } k=2$$

$\alpha = 1 \Rightarrow P(w_i \mid y=1) = \frac{2+1}{50+2} = \frac{3}{52}$

$\alpha = 10 \Rightarrow$ ~~—~~ $= \frac{2+10}{50+20} = \frac{12}{70}$

$\alpha = 100 \Rightarrow$ ~~—~~ $= \frac{102}{250}$

$\alpha = 1000 \Rightarrow$ ~~—~~ $= \frac{1002}{2050} \approx \frac{1}{2}$

As $\alpha \uparrow$ the Probab also $\uparrow$

(*) As $\alpha \uparrow$, we are moving the likelihood probabilities to uniform distribution

(*) When n_i is small $\Rightarrow$ denominator is small
$\Rightarrow$ we have less confidence in ratio

And when we have less confidence in ratio, we use α and the probability tends to go to $\frac{1}{2}$ as $\alpha \uparrow$

↳ This is whole idea of Laplace smoothing.

It is called smoothing because we are smoothing the value towards uniform distribution.

Often times, we apply $\alpha = 1$
Also called 1 additive smoothing.
This avoids 0 probability problem.

α is related to Bias Variance TradeOff

32.5 Log Probabilities and Numerical Stability

We know

$P(y=1 \mid w_1, w_2, \ldots w_d)$

$= P(y=1)$

$* P(w_1 \mid y=1)$

$* P(w_2 \mid y=1)$

$\vdots$

$* P(w_d \mid y=1)$

} all probab lie 0 - 1

Lets say d is large and all prob lie between 0 and 1.

Imagine $d = 4$; & $P(y=1 \mid w_1, w_2, w_3, w_4)$

$= 0.2 \times 0.1 \times 0.2 \times 0.1$

$= 0.0004$

The value is so small when $d=4$; Imagine $d=100$, the value would be very minuscule.

This results in numerical stability issues

In Python, double precision has only 16 significant values.

So, with further small values, python starts rounding.

This issue is called Numerical underflow

Resolution: Instead of using these value, we can use log (probability value).

Why log?

$\log(0.0004) = -2.23979$

Instead of operating on small values, we can use log values

Such values are called as Log Probabilities

Also log is a monotonic function means as $x \uparrow$; $\log(x) \uparrow$.

Hence we can compare logs of both probabilities ($y=1$ & $y=0$) to assign a class to y_i

$$\log[P(y=1 \mid w_1, w_2 \ldots w_d)]$$

$$= \log(P(y=1)) + \sum_{i=1}^{d} \log[\log P(w_i \mid y=1)]$$

This is because $\log(ab) = \log a + \log b$

Hence the product term in the equation gets converted to summation.

Thats why log is used widely in Math. It converts product to summation.

Similarly $\log(a^b) = b \log a$.

Hence log converts
$\log(ab) = \log(a) + \log(b)$ ⇒ converts Mult → Addition
$\log(a^b) = b\log(a)$ ⇒ converts Exponential → Multiplu.

32.6 BIAS VARIANCE TRADE OFF

We know
High Bias ⟶ Underfitting
High Variance → Overfitting

In Naive Bayes, there is one parameter alpha (α) which determines underfitting or overfitting

Case 1: $\alpha = 0$;

$$P(w_i | y=1) = \frac{(\#\text{train points where } w_i \text{ occurs \& } y=1)}{(\#\text{train points where } y=1)}$$

Imagine total datapoints are 1000 and there is a word which occurs only twice (very rare)

$$P(w_i | y=1) = \frac{2}{1000}$$

This means we are giving a probability value (very small) to a very rarely occuring word.
⇒ Means we are considering that outlier in calculation
⇒ Means we are Overfitting

Overfitting is related to High Variance ⇒ small change in Training data results in Dramatic Model changes.

In our training data, we had 2 texts containing the word w_i.

Now if we change the dataset slightly the probability changes dramatically

Since w_i occurs only in 2 out of 1000 cases,
A small change in D_{Train}
i.e. Remove the 2 texts containing w_i

$P(w_i | y=1)$ changes from $\frac{2}{1000}$ to $\frac{0}{1000}$

Drastical change from $\frac{2}{1000}$ to 0

Hence, when $\alpha = 0$; small change in Dtrain implies large change in model
$\Rightarrow$ High Variance
$\Rightarrow$ Overfitting

Case 2: α is very large ; say $\alpha = 10k$

$$P(w_i|y=1) = \frac{2+10k}{1000+2(10k)} \approx \frac{1}{2}$$

$\downarrow$

$k = 2$

$$\therefore P(w_i|y=1) = P(w_i|y=0) \approx \frac{1}{2}$$

This happens for all words.
So, a model cannot distinguish if word belongs to class 1 or 0.
$\Rightarrow$ This implies Underfitting

$\therefore$ When α = V. Large.

$$P(y=1|w_1,w_2 \ldots w_d) = P(y=1) * \prod_{i=1}^{d} \underbrace{P(w_i|y=1)}_{\approx \frac{1}{2}}$$

$$P(y=0|w_1,w_2 \ldots w_d) = P(y=0) * \prod_{i=1}^{d} \underbrace{P(w_i|y=0)}_{\approx \frac{1}{2}}$$

** V. Imp

The class label is dependent on this term.

← Since val of both terms is $\frac{1}{2}$ →

Comparing this scenario to K-NN when $K = n$; & $n_1 \rightarrow +ve$, $n_2 \rightarrow +ve$ } $n_1 + n_2 = n$

→ If $n_1 > n_2$; Any x_q would be marked as +ve

Similarly since $\prod_{i=1}^{d} P(w_i | y=1) = \prod_{i=1}^{d} P(w_i | y=0)$

$\approx \frac{1}{2}$

The overall probability is dependent on $P(y=1)$ and $P(y=0)$.
Which implies number of points with classlabel 1 and 0.; whichever would be higher would have that label for any x_q]

similar scenario

Hence when α is very large
⇒ Underfitting
⇒ High Bias

Hence
Case 1 : $\alpha = 0$ ⇒ Overfitting
⇒ High Variance

Case 2 : $\alpha = 1$ ⇒ Underfitting
⇒ High Bias

So,

Q How to find Right α

In K-NN; we found right K using Simple cross validation OR 10-Fold ~~~~~.

Similarly he we find right α : using simple C.V. OR 10-Fold C.V.

So,

α in Naive Bayes → Hyper Parameters (since they control Bias Variance Trade off)

K in K-NN → Hyper Parameters

Hyper Parameters → found using C.V.

32.7 Feature Importance & Interpretability

In K-NN we saw that we can obtain feature importance using forward selection

In Naive Bayes, we have likelihood ratio

$\forall w_i$, $P(w_i|y=1)$ and $P(w_i|y=0)$

We have two imp. pieces of information

w_i	$P(w_i \mid y=1)$	$P(w_i \mid y=0)$
w_1	(Step 1).	
w_2	Sort w_i's	
w_3	based on	
:	$P(w_i \mid y=1)$	
w_n	descending	
	order	

(Step 1) Returns top words with highest probability
⇒ Very informative words or features in determining a data point belongs to +ve class.

Similarly

$P(w_i \mid y=0)$ → high; indicates that w_i is important word or feature in determining -ve class

Hence, feature importance is very trivial in Naive Bayes.

+ve class: Find words (w_i) with highest value of $P(w_i \mid y=1)$

-ve class: ~~~~~~ $P(w_i \mid y=0)$

It can be determined directly from model

This is unlike K-NN ~~the~~ where we used forward feature selection Here, in Naive Bayes, we get it directly from Model.

What about Interpretability:

Given a point:

$x_q \longrightarrow y_q = 1$

$\downarrow$

$\{w_1, w_2, w_3 \cdots w_{10}\}$

I am concluding $y_q = 1$ because x_q contains words (say w_3, w_6, w_{10}) which has high value of

$P(w_3|y=1)$;
$P(w_6|y=1)$;
$P(w_{10}|y=1)$

Similarly, I am also not concluding $y_q = 0$; because x_q contains words (say w_8, w_{15}, w_{20}) where

$P(w_8|y=0)$
$P(w_{15}|y=0)$
$P(w_{20}|y=0)$

} is very small

Lets take one example:
In Amazon reviews ; if we have a word named "phenomenal" or "great". the probability that review is +ve is very high.

Similarly; words like "terrible", "not good", occurs with $y_q = 0$ very often. Hence, we declare the review to be negative.

32.8 Imbalanced Data

What happens to Naive Bayes in terms of Imbalanced Data?

~~$P(y=1 | w_1, w_2, w_3 \cdots w_d)$~~

$$P(y=1 | w_1, w_2 \cdots w_d) = P(y=1) * \prod_{i=1}^{d} P(w_i | y=1)$$

$$P(y=0 | w_1, w_2 \cdots w_d) = P(y=0) * \prod_{i=1}^{d} P(w_i | y=0)$$

Lets say $n \rightarrow n_1$ (+ve)
$\quad\quad\quad\;\; \hookrightarrow n_2$ (-ve)

and assume $n_1 \gg n_2$

90% of points in training data → +ve.

$\therefore \frac{n_1}{n} = 0.9$ and $\frac{n_2}{n} = 0.1$

$\therefore P(y=1) = 0.9$ and $P(y=0) = 0.1$

Even if we assume that Product (π) terms have same value.

i.e. $\prod_{i=1}^{d} P(w_i|y=1) = \prod_{i=1}^{d} P(w_i|y=0)$

then the probabilities solely depend on $P(y=1)$ and $P(y=0)$

The dominating class (which, in our case is $P(y=1)$) gets the advantage which is incorrect

$P(y=1)$ is 9 times more than $P(y=0)$

Solution:

① Standard technique of upsampling and downsampling

i.e. $n_1 \approx n_2$; Hence $P(y=1) = P(y=0) = \frac{1}{2}$

② Drop $P(y=1)$ and $P(y=0)$

With upsampling and downsampling, we replace $P(y=1)$ and $P(y=0)$ with $\left(\frac{1}{2}\right)$

At the end we compare $P(y=1|w_1 \dots w_d)$ with $P(y=0|w_1 \dots w_d)$

So, replacing $P(y=1) = P(y=0)$ with $\frac{1}{2}$ doesnt matter. It is equivalent to dropping them.

(3) Some scientist and researchers have modified Naive Bayes to account for class Imbalance.
⇒ Not often used

Other Problem with Imbalanced Data:

$n_1 = 900$ (+ve) $\longrightarrow P(w_i|y=1)$

Majority class $= \frac{\quad}{900}$ → Numerator Ranges from 0 to 900

$n_2 = 100$ (-ve) $\longrightarrow P(w_i|y=0)$

Minority class $= \frac{\quad}{100}$ → Numerator Ranges from 0 to 100

→ Small value of Numerator

$P(w_i|y=1)$ Vs $P(w_i|y=0)$

Large value in Numerator

Small value in Numerator

When we do Laplace Smoothing, it impacts Ratio with small Numerator more than large Numerator

Laplace smoothing impacts Minority Ratio more

Eg:

	Minority Class (-ve)	Majority Class (+ve)
$P(w_i \mid y_i=1)$	$\frac{2}{100}$	$P(w_i \mid y_i=0)$ $\frac{18}{900}$
	$= 2\%$ ← same val →	$= 2\%$
	Assume $\alpha = 10$	
	$= \frac{2+10}{100+20} = \frac{12}{120}$	$= \frac{18+10}{900+20} = \frac{28}{920}$
	$\approx 10\%$	$\approx 3.04\%$
	Using same α; $2\% \rightarrow 10\%$ because Num & Denominator are same	Using same α $2\% \rightarrow 3\%$ because Numerator & denominator are large

Solution to this:

1. Using Std process of upsampling & downsampling
2. There are some hacks: I will have one α for +ve ~~ka~~ class & other α for -ve class.

32.9 Outliers.

How they are handled in Naive Bayes

What are Outliers

In Text classification.

$x_q = w_1\ w_2\ w_3\ (w')$

and

$w' \notin \{w_1, w_2, w_3 \ldots w_m\}$

these are set of words encountered in D_{Train}

Laplace smoothing takes care of this.

$$P(w'|y=1) = \frac{0+\alpha}{n_1 + 2\alpha}$$

Hence, if we get outlier at Test time, then laplace smoothing takes care of the same

What happens to Outlier at Train data

$\{w_1, w_2, w_3 \ldots w_m\}$ ← Set of words in D_{Train}

Suppose there is a word

w_8 → Occurs few times in +ve and -ve class.

→ outlier

Hack/solutions for Outliers in DTrain

① If a word (w_j) occurs ~~more~~ fewer than 10 (any number) times, then just ignore that word

② and Remove that word from training set.

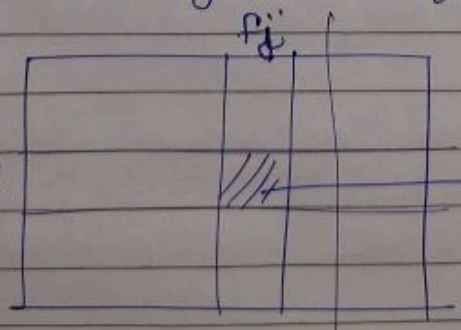
$w_j \notin \{w_1, w_2 \dots w_m\}$ (w_j Removed.)

② Using Laplace Smoothing with reasonable value of α.

32.10 Missing Values

Case1: Text Data:
There is no case of missing data as the word either occurs or doesnt.

Case2: Categorical features



Now $f_i \in \{a_1, a_2, a_3\}$ → all possible values of f_i in Rest of dataset

We assume NaN as one of the categorical values of fi
⇒ Imp. Hack.

∴ $f_i \in \{a_1, a_2, a_3, \text{NaN}\}$

Case 3 : Numerical features

We used Standard feature Impulations like Mean, Median, etc.

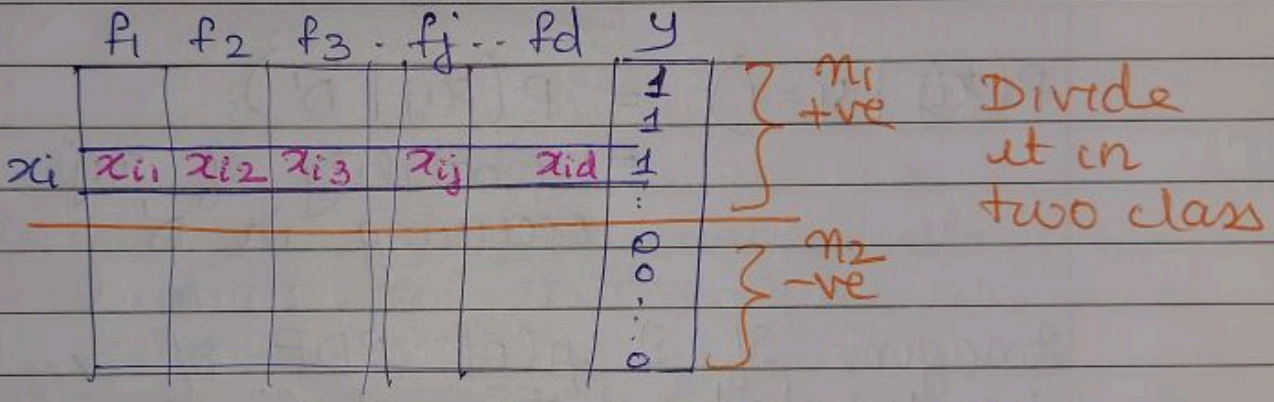
One such method is Gaussian Naive Bayes. We would discuss it shortly.

32.14 Naive Bayes with Numerical Features

NB: Binary features, Categorical — } Very straight forward.

NB: Numerical / Real valued.

Imagine I have dataset



$f_j \rightarrow$ real valued.

Fitting N.B in Real Valued scenario

$$P(y_i = 1 \mid x_{i1}\ x_{i2} \cdots x_{id}) \propto P(y_i = 1) * \prod_{j=1}^{d} P(x_{ij} \mid y_i = 1)$$

Equation Remains same

$P(y_i = 1)$ → Class Prior Probab. $= \frac{n_1}{n_1 + n_2}$

$P(x_{ij} \mid y_i = 1)$ → Probab of j^{th} feature given class label

Remember, we have to find

$P(x_{ij} | y=1)$

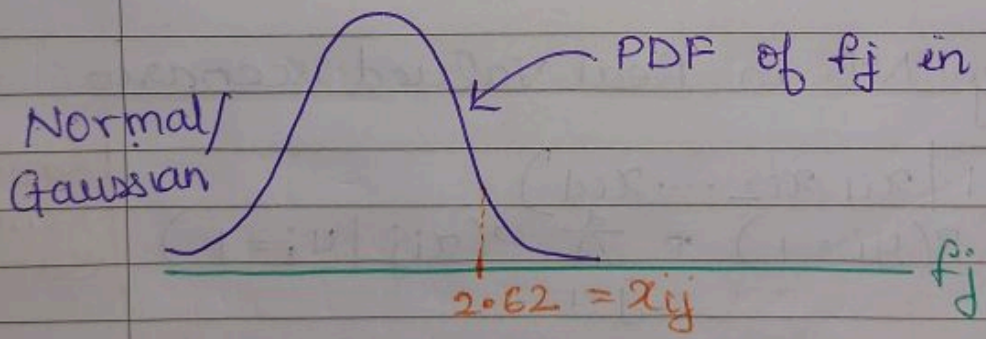
Since $y=1$; Lets concentrate on +ve data in D_{Train}

Let $D^{+ve}_{Train} = D'$

$P(x_{ij} | y=1) = P(x_{ij} | D')$

= Probability of x_{ij} occuring in D'

Imagine If I plot PDF of +ve Labeled data of f_j



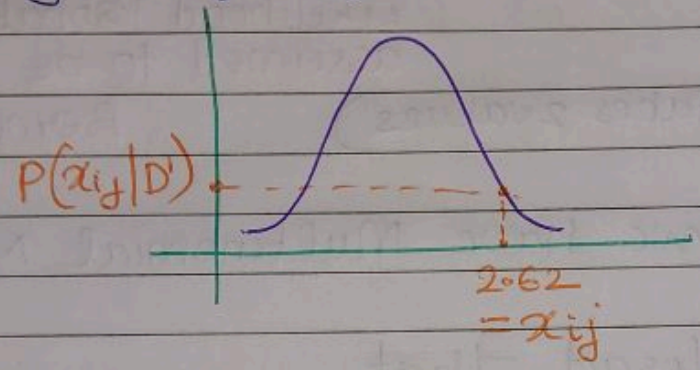
Lets assume, my curve is Gaussian

Lets say $x_{ij} = 2.62$ which lies in PDF

We can obtain $P(x_{ij} | D')$ from the PDF of f_j in D'

Lets assume the PDF follows Gaussian dist with $N(-1, 0.6)$ (μ, σ)

then I can get $P(x_{ij} | D')$ by the height of x_{ij} in PDF



Similarly we can concentrate on Dataset with $y=0$ and get ~~P(y~~ $P(x_{ij} | D'')$

Here, people assume f_j in D' follows Gaussian distribution with $N(\mu'_j, \sigma'_j)$

Also, they assume f_j in D'' (-ve dataset) follows Gaussian dist with $N(\mu''_j, \sigma''_j)$

When we make this assumption, such a model is called

Gaussian Naive Bayes

Likelihood Probabilities are ASSUMED to be Gaussian

Similarly, for words, we assume (text data)

$P(w_i / y = 1)$

0 1 (each word takes 2 values)

$\Rightarrow$ Bernoulli N.B.

Likelihood Probab are assumed to be Binary/ Bernoulli

Similarly we have Multinomial N.B.

Lets not forget that

Naive Bayes makes fundamental assumption of Conditional Independence
$\Rightarrow$ All features are independent of each other

32.15 Can Naive Bayes do Multi-class Classificaⁿ

$\Rightarrow$ Yes

Just like we do

$P(y_i = 1 \mid w_1\ w_2 \ldots w_d)$
$P(y_i = 0 \mid w_1\ w_2 \ldots w_d)$

We can also perform

$P(y_i = 2 \mid w_1\ w_2 \ldots w_d)$
$P(y_i = C-1 \mid w_1\ w_2 \ldots w_d)$

} compare all and pick largest as class label

Naive Bayes is designed for Multiclass classificaⁿ $\Rightarrow$ It can easily do it

32.13 Similarity of Dist Matrix

Naive Bayes cannot handle similarity Matrix if given instead of feature values
Because we have to compute prob. and we have product term. and we need exact feature value.

$$P(y_i=1 \mid f_1 f_2 \ldots) = P(y_i=1) \prod_{i=1}^{d} P(f_i \mid y_i=1)$$

f_i: exact feature value

& Naive Bayes doesnt used Distance
⇒ ——— is a probabilistic method where we need actual feature values

32.14 Large dimensionality

Naive Bayes used extensively in Text classification
Hence, it can handle large demensions

Only, we need to use log Probabilities

As dimensionality ↑, the Product term increases; hence use log Probabilities

$$P(y=1 \mid w_1 w_2 \ldots w_d) = P(y=1) \underbrace{\prod_{i=1}^{d} P(w_i \mid y=1)}$$

Lot of terms.