k-Nearest Neighbour

SMOTE

Linear Regression
Logistic Regression

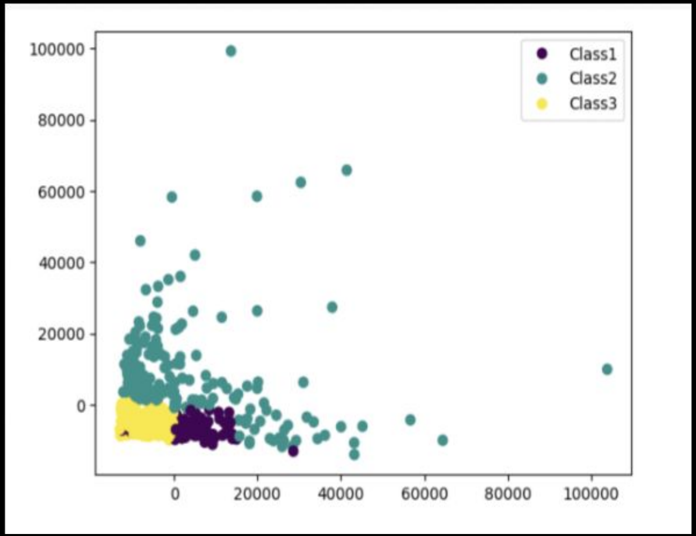
BUSINESS CASE - BLINK IT

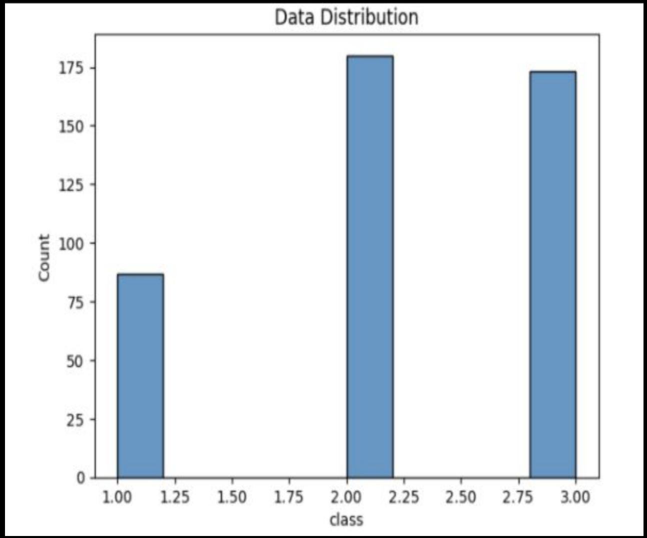
Blink it needs optimal number of delivery partners for each store.

Hence, classified stores into 3 based on outgoing deliveries.

- High Traffic
- Moderate Traffic
- Low Traffic

BLINK IT DATA





Multiclass Problem

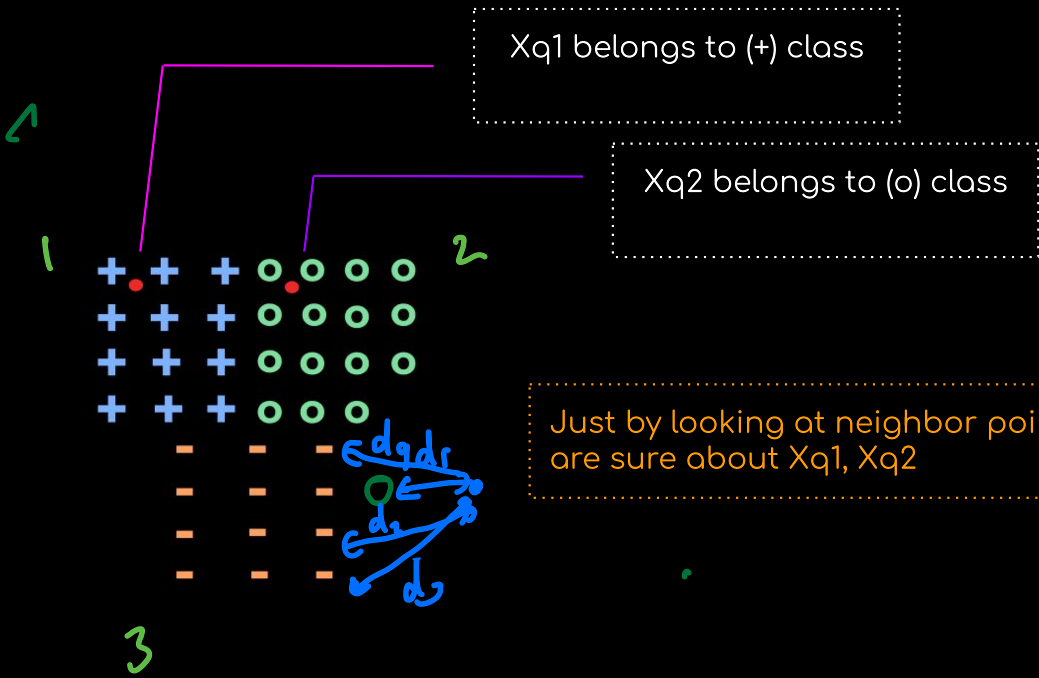
Non-Linear

Imbalance data

Will logistic regression work???

Logistic Regression requires extensive search for correctly polynomial feature

Need of a new algorithm with no features



$d_1 \to 2$
$d_2 \to 3$
$d_3 \to 3$

Just by looking at neighbor points, we are sure about Xq1, Xq2

K-nearest neighbour (kNN) model works on same intuition

$(x_1, y_1) \longleftrightarrow (x_2, y_2)$

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

↳ Euclidean distance

Class of datapoint (xq) depends on class of neighbouring points

How does kNN work?

If xq = [2,5] & data contains 6 data points:

Step 1 : Find euclidean distance:

(3, 6)   (2, 5)

	f^1	f^2	y
X^1	3	6	1
X^2	6	4	1
X^3	8	2	3
X^4	7	5	3
X^5	1	4	2
X^6	2	2	2

→

	f^1	f^2	y
X^1	3	6	1.41
X^2	6	4	3.00
X^3	8	2	6.48
X^4	7	5	5.00
X^5	1	4	1.41
X^6	2	2	2.00

Step 2 : Sort data based on distance:

	f^1	f^2	y
X^1	3	6	1.41
X^2	6	4	3.00
X^3	8	2	6.48
X^4	7	5	5.00
X^5	1	4	1.41
X^6	2	2	2.00

	f^1	f^2	y	y
X^1	3	6	1	1.41
X^5	1	4	2	1.41
X^6	2	2	2	2.00
X^2	6	4	1	3.00
X^4	7	5	3	5.00
X^3	8	2	3	6.48

Step 3 : Pick 3 data points having minimum distance:

	f^1	f^2	y	y
X^1	3	6	1	1.41
X^5	1	4	2	1.41
X^6	2	2	2	2.00
X^2	6	4	1	3.00
X^4	7	5	3	5.00
X^3	8	2	3	6.48

→ minimum distance from xq

Step 4 : Find majority class of these selected data points —>> class label for xq

	f^1	f^2	y	y
X^1	3	6	1	1.41
X^5	1	4	2	1.41
X^6	2	2	2	2.00

Majority class 2

Xq belongs to class 2

This selection of data points is decided by Hyperparameter “k”, hence the name kNN

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

$w_0, w_1, w_2 \rightarrow$ Parameter

$\times$

① s2, s3, s1, s4 → Assign class to xq

dist

sort & select

majority

What happens if k=4?

	f^1	f^2	y	y
X^1	3	6	1	1.41
X^5	1	4	2	1.41
X^6	2	2	2	2.00
X^2	6	4	1	3.00
X^4	7	5	3	5.00
X^3	8	2	3	6.48

Making predictions based on 4 nearest neighbours

2 data points >> class 1
2 data points >> class 2

Tie

Brain split problem

kNN cannot make predictions

Hence, it is advisable to keep k as odd value

What happens if k=5?

	f^1	f^2	y	y
X^1	3	6	1	1.41
X^5	1	4	2	1.41
X^6	2	2	2	2.00
X^2	6	4	1	3.00
X^4	7	5	3	5.00
X^3	8	2	3	6.48

Making predictions based on 5 nearest neighbours

Still a tie even if we keep k value as odd

Hack!

Randomly pick class labels of tied classes

Break:

8: 12 am

Here, kNN can pick class 1 or class 2 for xq

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

WORKING OF kNN:

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

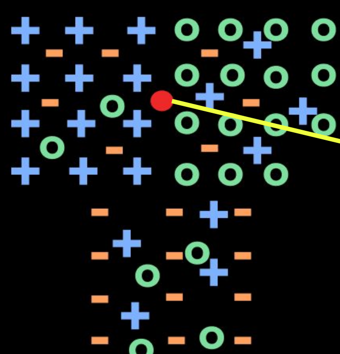
How does kNN has good performance on non linear multi class data?

Assume data contains:

(+)

(-) & k=5

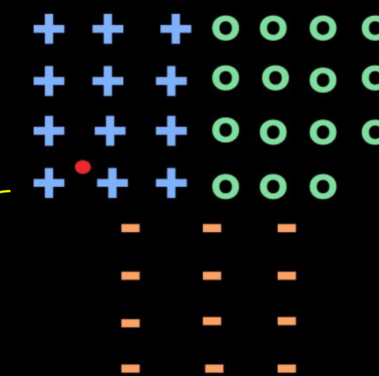
(o)



xq

X_sm = [2, 4, 8]

xq = [1, 6]



Class (+) = xq

kNN assumes neighbourhood as homogenous i.e, characteristics of nearest neighbour and xq will be same

kNN fails when data has a lot of noise/outliers

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

<u>WORKING OF kNN:</u>

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

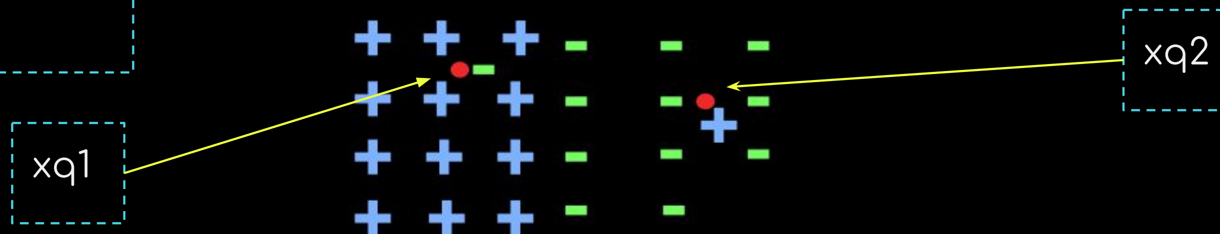
- kNN assumes homogeneous neighbourhood
- It is heavily impacted if outliers increases.

$wt \to 1 \quad \leftarrow 1 \to 1000 \quad \Rightarrow \frac{1000}{100} \approx 10$

$wt \approx 10 \quad \swarrow 0 \to 100$

Bias- variance tradeoff in kNN.. yes...or....no?

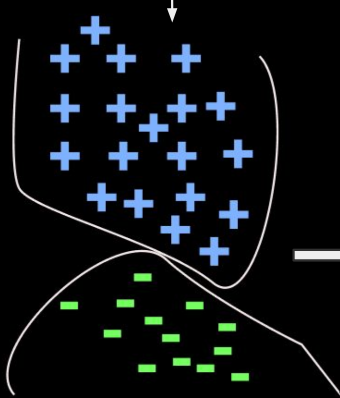
Suppose data contains 2 outliers:



For k=1

As xq1 closes to (-) outlier → Class prediction (-)

As xq1 closes to (+) outlier → Class prediction (+)



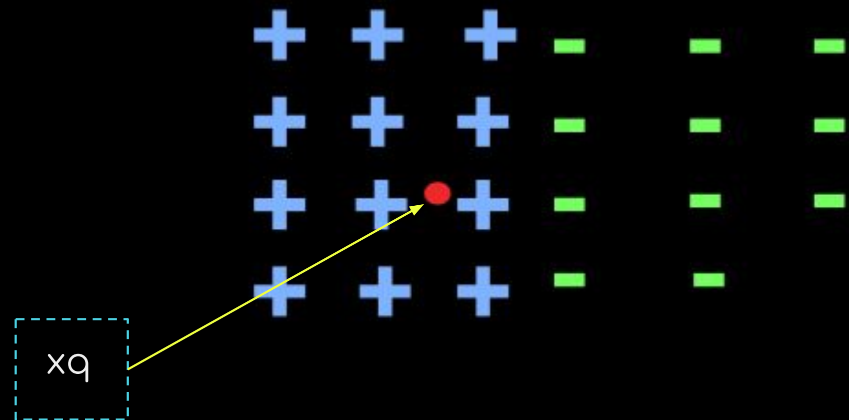
kNN trying to fit every data point

Rough Decision Boundary

Taking the same data, what will be class label for xq if k=72?

(+) = 31
(-)= 41

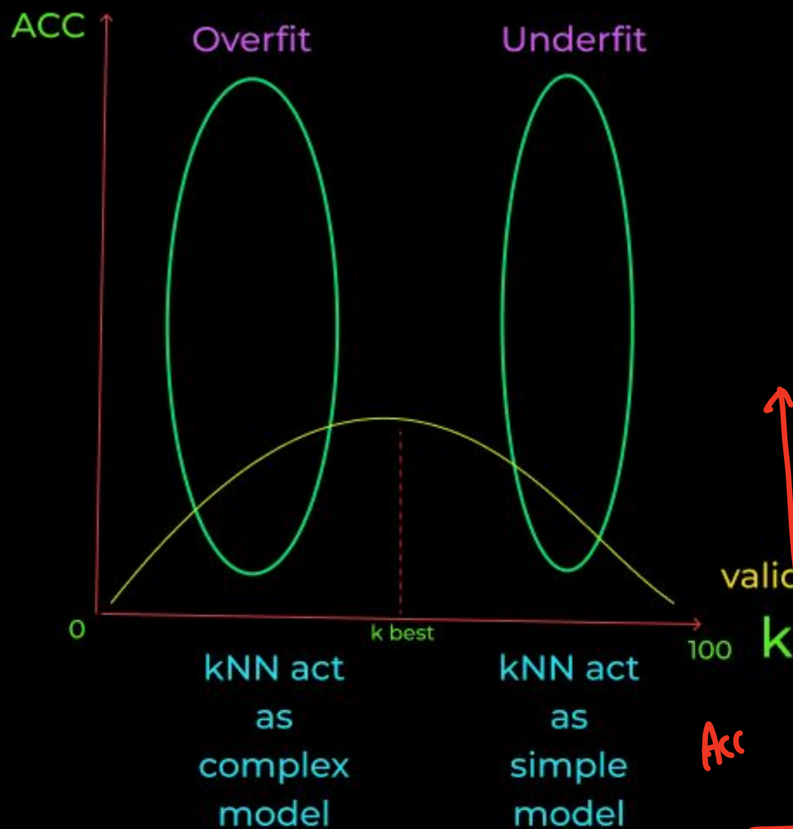
Xq-> (-) class as (-) >(+)



Even when xq is closer to (+), kNN does not fit training data

As k increases, kNN underfits

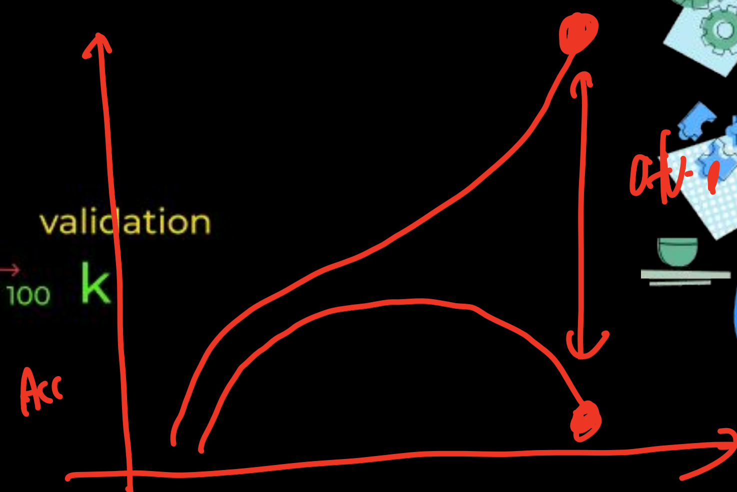
Summary!



Train : 100%
Test : 2%
⇓
overfit

k increases, kNN underfits

k decreases, kNN overfits



Training time complexity	Space complexity
O(1)	kNN stores entire training data.
No computation done by kNN	N x d
Stores the data only	Space complexity (O) N x d

Mobile Ph

Memory :

Test time complexity

· sort()

Step 1 : Find distance b/w training data and xq = O (n x d)

Step 2 : Sort data = O (nlogn) → Merge sort, Randomized quick sort

Step 3 : Pick nearest neighbour O(k)

Step 4 : Majority vote O(k)

As k << n & d, hence O(k) ignored

Time complexity = O (nd + nlogn)

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

<u>WORKING OF kNN:</u>

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

- kNN assumes homogeneous neighbourhood
- It is heavily impacted if outliers increases.

- If k increases, kNN underfits. (bias increase, variance decrease)
- If k decreases, kNN overfits. (bias decrease, variance increase)

POINTS TO REMEMBER

- Train time complexity —>> O(1)
- Test time complexity —>>> O(nd + nlogn)
- Space complexity —->> O (nd)

Diabetic Patient Example

Suppose we take

- 40 Diabetic (+) class
- 40 non diabetic (-) class

Features:
Gender (M,F)
Age
BP
Glucose Level
Blood group A+, B+. O+, AB+, AB-, O-, B-, A-

kNN **does not work** on categorical data as euclidean distance needs numeric data!

Convert categorical data into numerical data by ONE HOT ENCODING (OHE)

OHE to convert categorical data into numeric data

Gender	OHE 1	OHE 2
M	1	0
F	0	1
F	0	1
M	1	0
M	1	0
F	0	1
M	1	0
F	0	1

OHE of gender becomes : (n,2)
Similarly, OHE of Blood Group becomes : (n,8)

Total dimensions when One Hot Encoding

5 dimensional data

Feature	Dimension
Gender	2
Age	1
Blood Grp	8
Glucose	1
BP	1

8 dimensional data

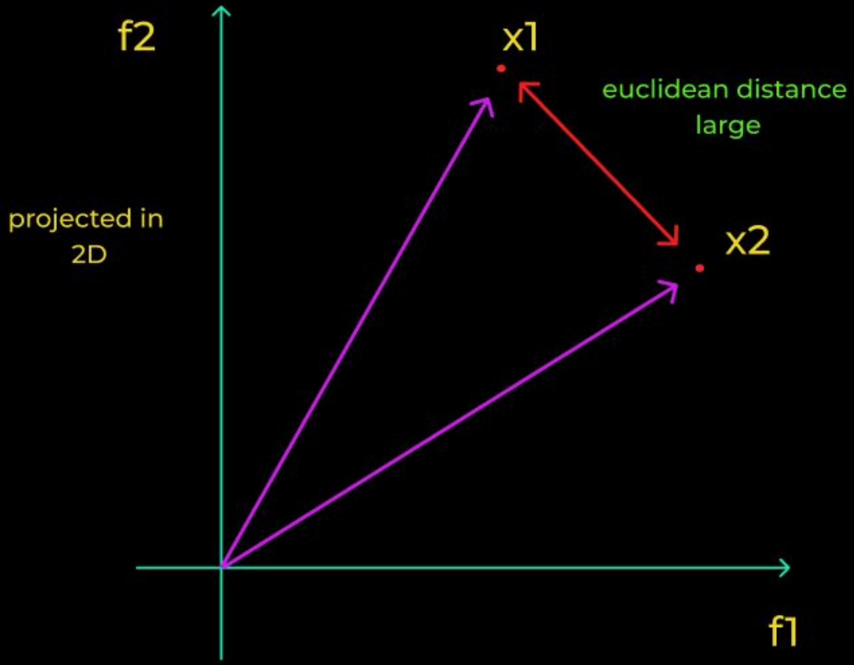
As OHE increases dimensions, it leads to curse of dimensionality

Hence, Target encoding shall be used

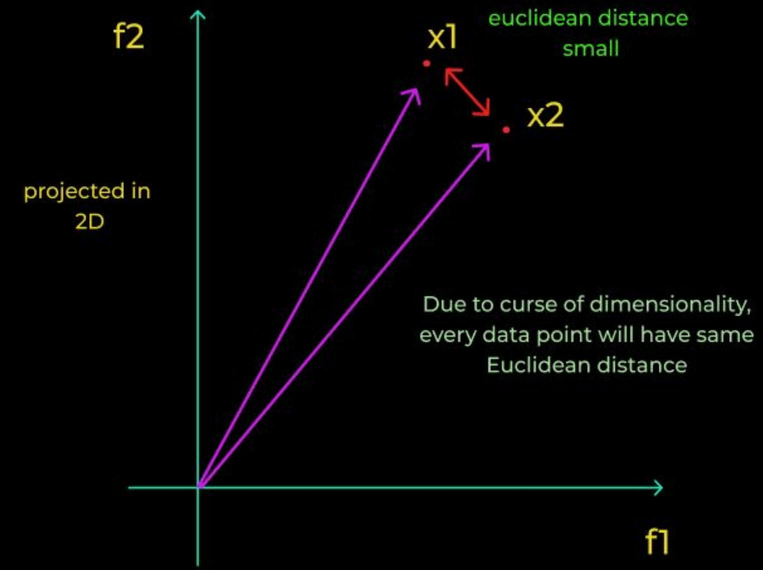
What is Curse of Dimensionality?

Suppose x1 and x2 have dimension = 4, then

Euclidean distance $= [\sum_{j=1}^{d}(x_j^{(1)} - x_j^{(2)})^2]^{\frac{1}{2}}$



Due to low dimension, Euclidean distance between x1 & x2 is very large



Euclidean distance cannot be used

Due to high dimension, Euclidean distance between x1 & x2 is very small

Conclusion : Euclidean distance fails when dimension is high

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

WORKING OF kNN:

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

- kNN assumes homogeneous neighbourhood
- It is heavily impacted if outliers increases.

- If k increases, kNN underfits. (bias increase, variance decrease)
- If k decreases, kNN overfits. (bias decrease, variance increase)

POINTS TO REMEMBER

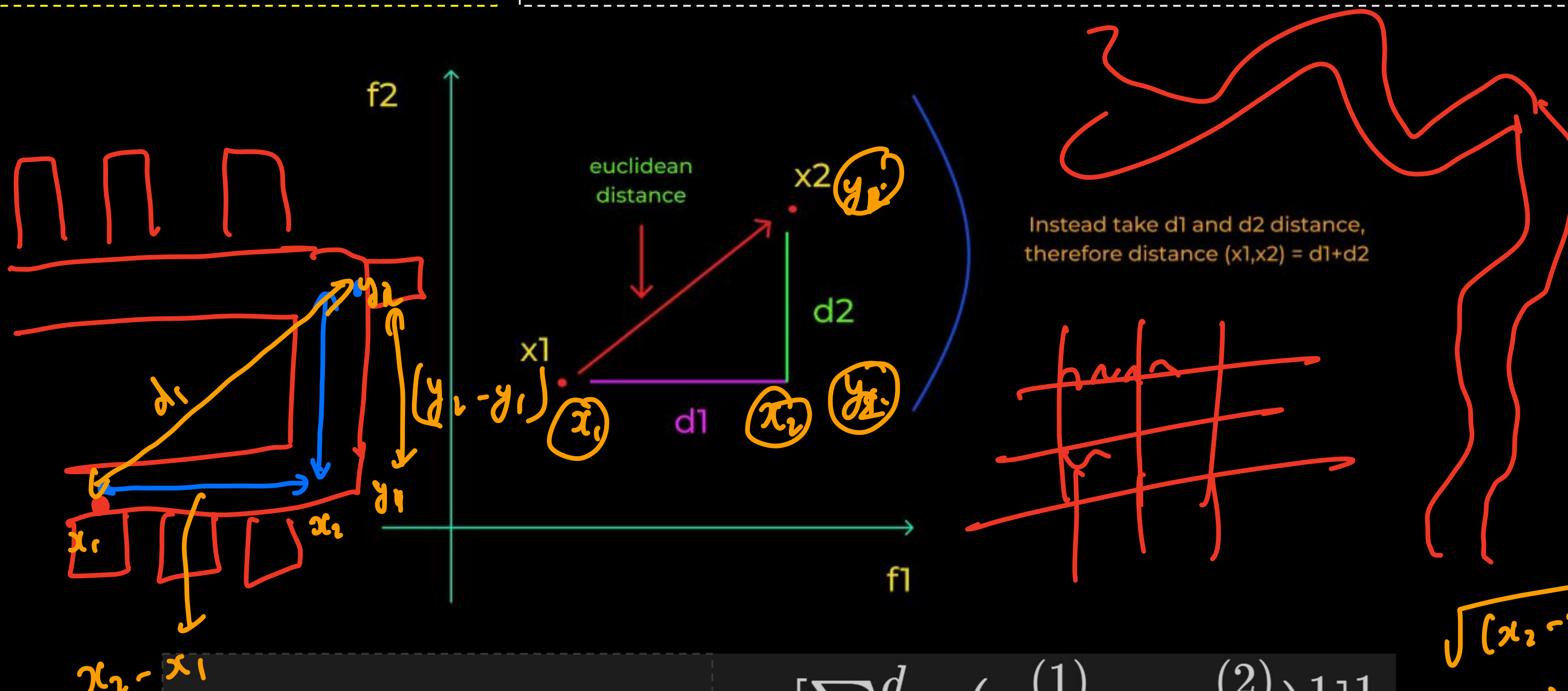
- Train time complexity —>> O(1)
- Test time complexity —>>> O(nd + nlogn)
- Space complexity —–>> O (nd)

- Euclidean distance fails when there is high dimension data

What other distance to use?

Manhattan Distance

Can be understood as distance measure as we walk on a path from x1 to x2



Manhattan distance $= [\sum_{j=1}^{d} (x_j^{(1)} - x_j^{(2)})^1]^1$

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

$(x_2 - x_1) + (y_2 - y_1)$

What other distance to use?

$\Sigma\left((x_2-x_1)^2\right)^{1/2}$

$\Sigma\left((x_2-x_1)^p\right)^{1/p}$

Minkowski Distance

$\Sigma\left((x_2-x_1)^1\right)^{1/1}$

Euclidean $= [\sum_{j=1}^{d}(x_j^{(1)} - x_j^{(2)})^2]^{\frac{1}{2}} \rightarrow x^2$ same as $|x|^2$

- similar to L2 Norm $\sum_{j=1}^{d} w_j^2$, hence called L2 distance

Manhattan $= [\sum_{j=1}^{d}(x_j^{(1)} - x_j^{(2)})^1]^1$

- similar to L1 Norm $\sum_{j=1}^{d} \boxed{|w_j|}$, hence called L1 distance

Generalized Equation $= [\sum_{j=1}^{d}(x_j^{(1)} - x_j^{(2)})^p]^{\frac{1}{p}}$ if $p=1 \rightarrow$ Manhattan, if $p=2 \rightarrow$ Euclidean

$\rightarrow 1 \rightarrow 1/\infty$

Manhattan distance & One Hot Encoding

Break: 8:15 AM

OHE creates a high dimensional sparse data

Manhattan Distance gives equal importance to all the features (even for irrelevant features)

Cosine Similarity for One Hot Encoding

Maths for ML

Since Cosine similarity focuses on direction of vectors, it easily ignores irrelevant features

$$\text{Cosine Similarity}(x^{(1)}, x^{(2)}) = \frac{x^{(1)} x^{(2)}}{||x^{(1)}||\ ||x^{(2)}||}$$

Ranges from (-1) to 1

Least similar

Most similar

High dim data

Distance metric used for kNN

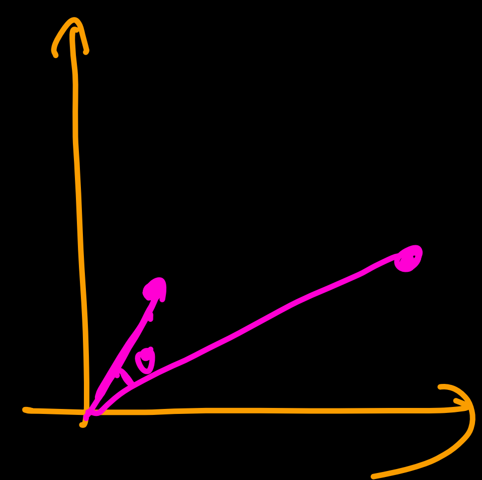
- Euclidean Distance – for low dimensional data
- Cosine similarity — for high dimensional data
- Manhattan – useful when data is like a map
- Minkowski – for using custom distance metric

NLP

cosine dist

ChatGPT

LLM



POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

<u>WORKING OF kNN:</u>

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

- kNN assumes homogeneous neighbourhood
- It is heavily impacted if outliers increases.

- If k increases, kNN underfits. (bias increase, variance decrease)
- If k decreases, kNN overfits. (bias decrease, variance increase)

POINTS TO REMEMBER

- Train time complexity —>> O(1)
- Test time complexity —>>> O(nd + nlogn)
- Space complexity —–>> O (nd)

- Euclidean distance fails when there is high dimension data

- Cosine similarity — for high dimensional data [-1,1]
- Manhattan – useful when data is like a map
- Minkowski – for using custom distance metric

Google images use kNN to provide famous monuments just by searching city.

kNN work so fast in Google searches?

By hashing algorithm —LSH (Locality Sensitive Hashing)

(1, 1, 3)
(1, 3, 5)

0, 2, 2

4

What is hashing?

Storing of data in key value pair (Analogous to Directory)

Key	Value
Delhi	Indiagate, Redfort Qutub Minar
Mumbai	Marine drive, Gateway of India

If query = Delhi

Returns Indiagate, Redfort, Qutub Minar

Quickly returns data

Time complexity O(1)

How does LSH work?

For hash table create randomised hash function (h(x))

	f^1	f^2	f^3
X^1	5	10	7
X^2	-1	0	-10
X^3	-20	20	20
X^4	100	20	30

Gives key for hash table

How does LSH work?

Suppose we take a random vector:

$$v_1 = [10,\ -20,\ 30]$$

v_{11} v_{12} v_{13}

And define:

$$h(x) = [\ 1 \text{ if } f_1 > v_{11}, \text{ else } 0,\ 1 \text{ if } f_2 > v_{12}, \text{ else } 0,\ 1 \text{ if } f_3 > v_{13}, \text{ else } 0]$$

	f^1	f^2	f^3	h(X)
X^1	5	10	7	[0,10]
X^2	-1	0	-10	[1,1,0]
X^3	-20	20	20	[0,1,1]
X^4	100	20	30	[0,1,0]

Hash table

Key	Value
[0,10]	X^1, X^4
[1,1,0]	X^2
[0,1,1]	X^3

Clubs them into one key

$P(h(X^1) == (h(X^4)) \uparrow$

Hence, LSH concludes (X^1) & (X^4) are near points

LSH's role in fastening kNN

LSH groups similar data points

Suppose for some xq, h(xq) = [0,1,0]

We run kNN only for data points having h(x) = [0,1,0], instead of whole data

This reduces testing time complexity as kNN is using a subset of data

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq] on the basis of neighbourhood.

<u>WORKING OF kNN:</u>

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

- kNN assumes homogeneous neighbourhood
- It is heavily impacted if outliers increases.

- If k increases, kNN underfits. (bias increase, variance decrease)
- If k decreases, kNN overfits. (bias decrease, variance increase)

POINTS TO REMEMBER

- Train time complexity —>> O(1)
- Test time complexity —>>> O(nd + nlogn)
- Space complexity —–>> O (nd)

- Euclidean distance fails when there is high dimension data

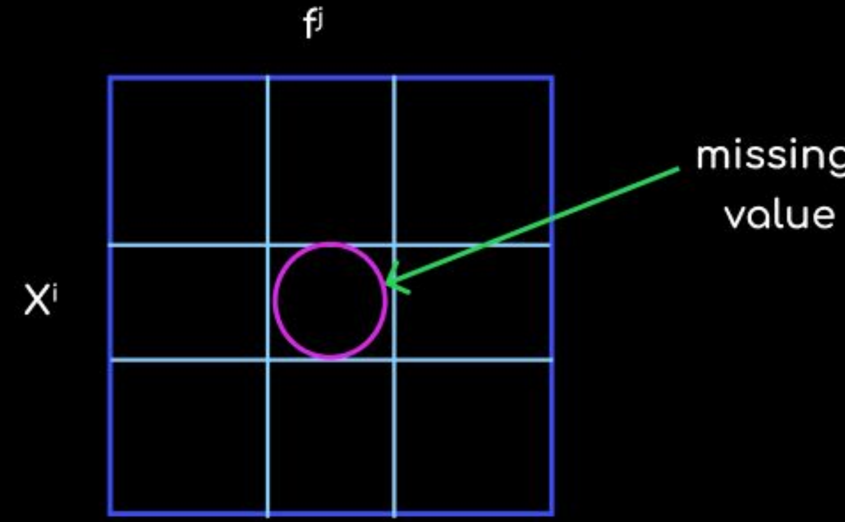
- Cosine similarity — for high dimensional data [-1,1]
- Manhattan – useful when data is like a map
- Minkowski – for using custom distance metric

POINTS TO REMEMBER

- LSH reduces testing time complexity by selecting a subset of data determined by h(x).

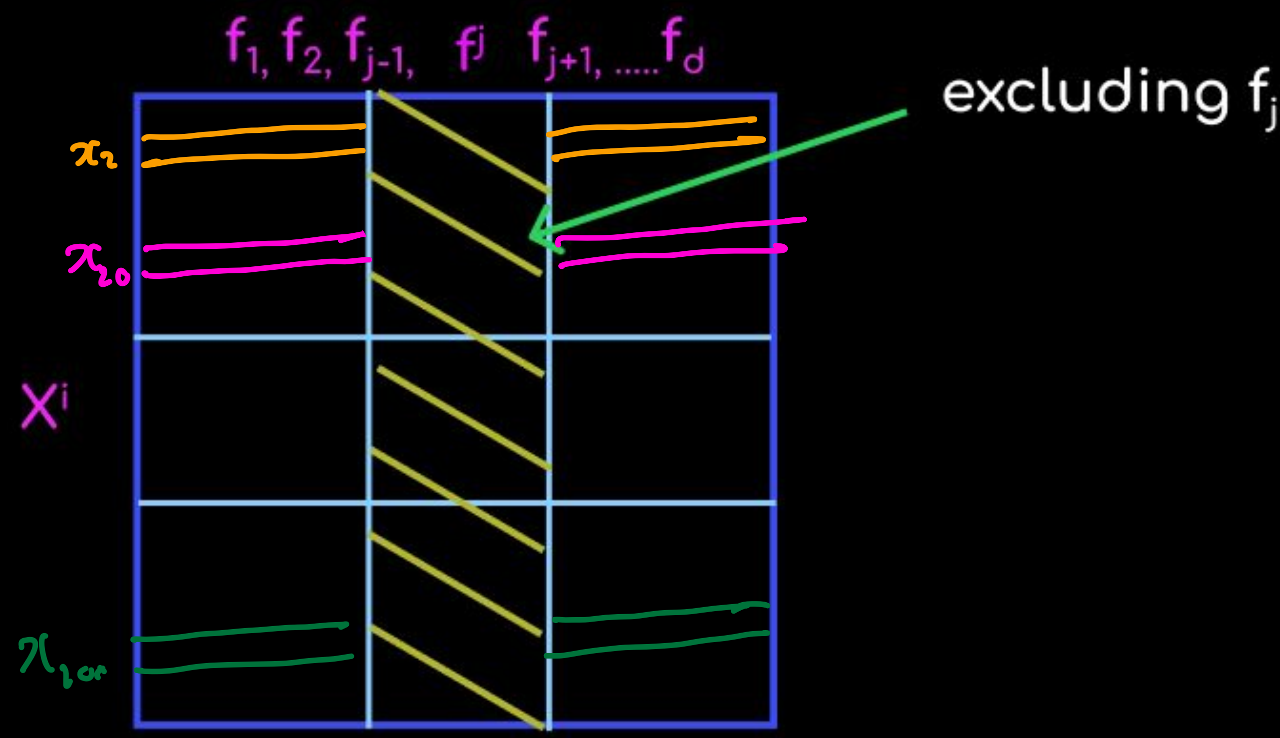
What are the techniques of imputing?

- Mean or median of Fj feature
- Analyzing data and manually impute value
- Mean and median of whole data



kNN for Imputation

Step 1 : Exclude fj from data



kNN for Imputation

Step 2: Find distance between xi and rest data — k nearest neighbour

Suppose x^1, x^{20}, x^{10}, x^{30} were nearest neighbours

if k=4

X^i

X^1
X^{20}
X^{30}
X^{10}

kNN for Imputation

Step 3: For these nearest neighbour, check value for fj value

$$X^{i}_{fj} = \text{Avg}\,(X^{1}_{fj}, X^{20}_{fj}, X^{30}_{fj}, X^{10}_{fj})$$

POINTS TO REMEMBER

- kNN is a non parametric algorithm.
- kNN predicts class of test data [xq on the basis of neighbourhood.

<u>WORKING OF kNN:</u>

- Find distance (xq and all training data)
- Sort distance
- Pick k nearest neighbors
- Majority rate of class prediction

- kNN assumes homogeneous neighbourhood
- It is heavily impacted if outliers increases.

- If k increases, kNN underfits. (bias increase, variance decrease)
- If k decreases, kNN overfits. (bias decrease, variance increase)

POINTS TO REMEMBER

- Train time complexity —>> O(1)
- Test time complexity —>>> O(nd + nlogn)
- Space complexity —–>> O (nd)

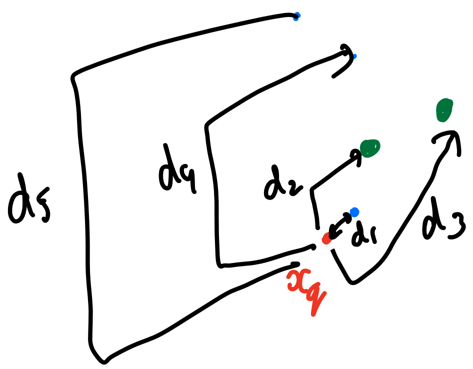
- Euclidean distance fails when there is high dimension data

- Cosine similarity — for high dimensional data [-1,1]
- Manhattan – useful when data is like a map
- Minkowski – for using custom distance metric

POINTS TO REMEMBER

- LSH reduces testing time complexity by selecting a subset of data determined by h(x).

- kNN can be used for imputation.

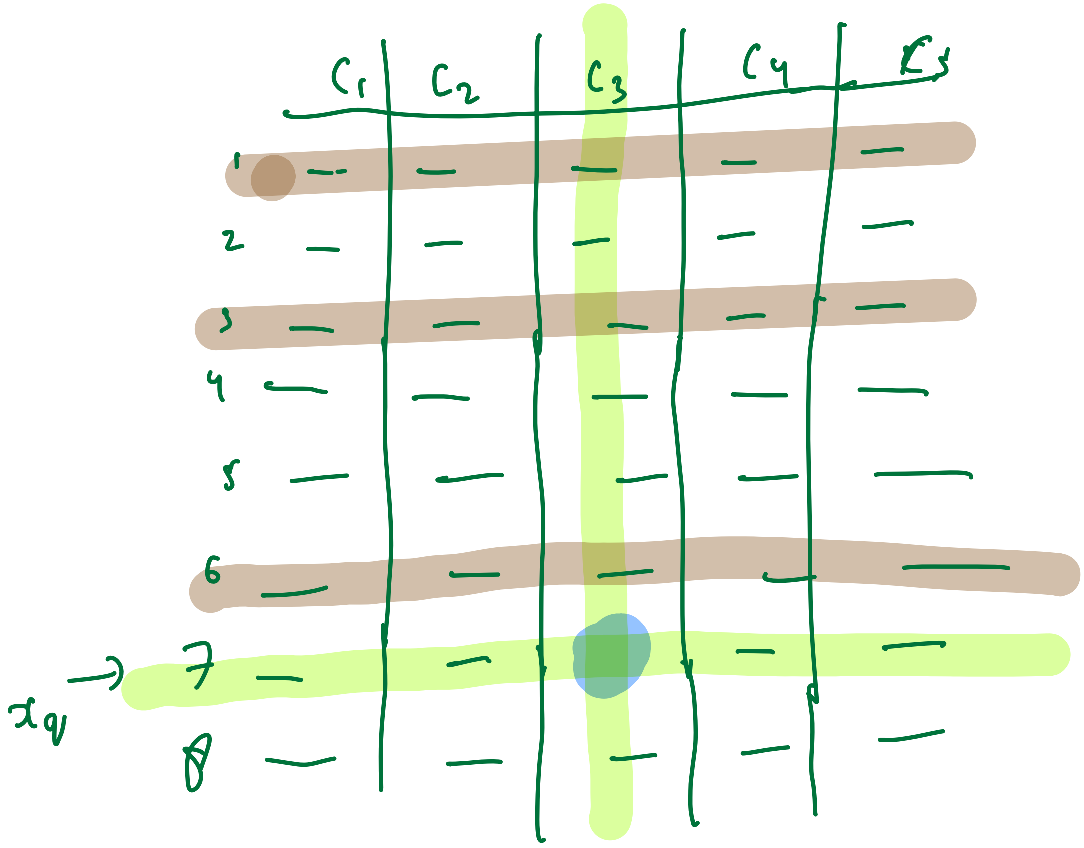


d_1 Blue
d_2 Green
d_3 Green
d_4 Blue
d_5 Blue

100

4

$1 \to 2, 3$



3 Neig