Machine Learning

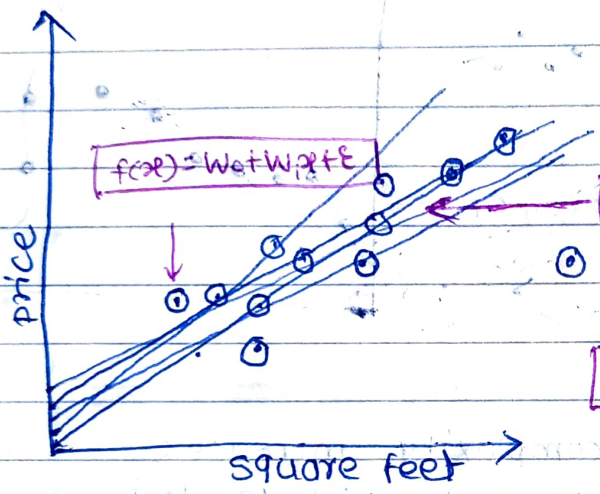
A Case study Approach

(Supervised) Regression

App

Predicting house price.

Linear Regression



which line is to be selected decided with RSS

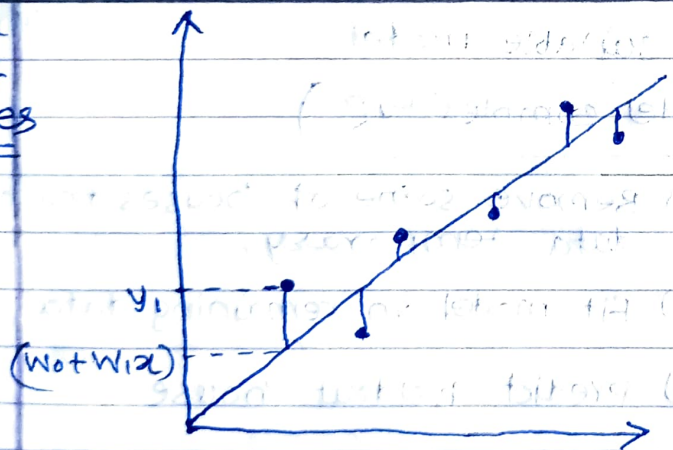
$f(x) = w_0 + w_1 x$

slope, value, Intercept

$y = f_w(x)$

(w_1, w_0) are Regression parameters

Residual Sum of squares



The line which gives min RSS should be selected.

$RSS = w_0$

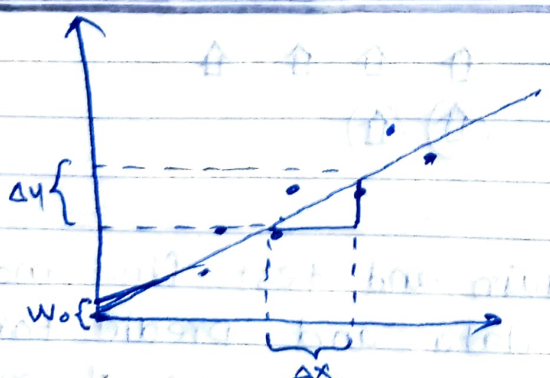
$(y_1 - [w_0 + w_1 x_1])^2 +$

$(y_2 - [w_0 + w_1 x_2])^2 +$

$(y_3 - [w_0 + w_1 x_3])^2 +$

--- all houses

Predicting House price



$$RSS(w_0, w_1) = \sum_{i=1}^{n} (y_i - [w_0 + w_1 x_i])^2$$

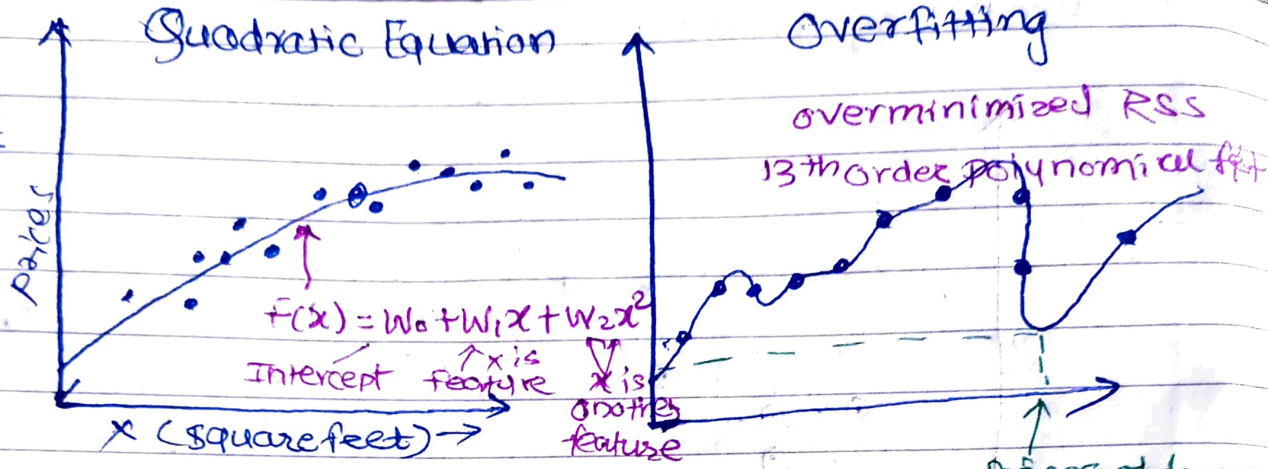
w_0 = Intercept

$w_1 = \frac{\Delta y}{\Delta x}$

$$y = w_0 + w_1 x + \varepsilon_0$$

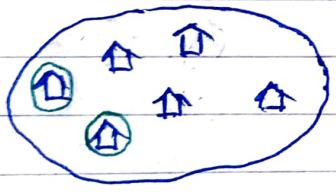
By putting value of x i.e sq feet we can get predicted house price.

Higher Order Regressions



Hence as model complexity increases i.e. Overfitting occur and which leads to problem and not suitable model

How to choose model complexity?



1) Remove some of houses from data temporary,
2) Fit model on remaining data
3) predict heldout house

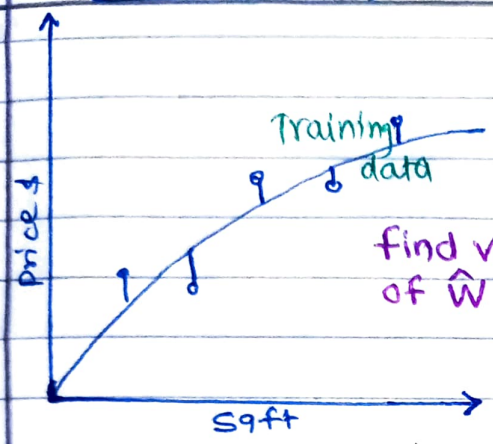
Spit data into -

Training data - ⇧ ⇧ ⇧ ⇧

Test data - ⇧ ⇧

we spit data to train and test find model that fits training data and predict for test data and found the performance of model

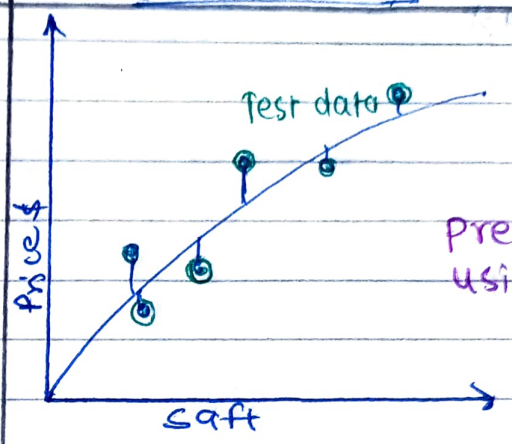
Training error



Training error(w) =

$$(\$_{train_1} - f_w(sq.ft_{train}))^2$$
$$+ (\$_{train_2} - f_w(sq.ft_{train}))^2$$
--- for all data

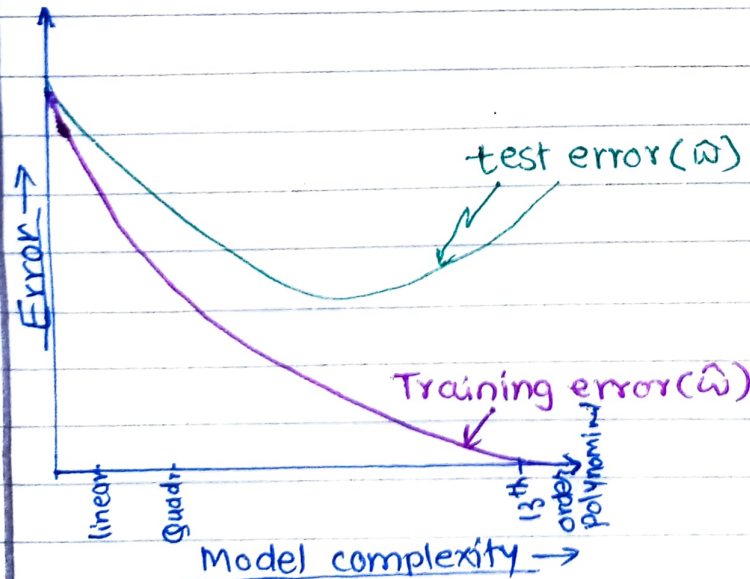
Test error



Test error($\hat{w}$) =

$$(\$_{test_1} - f_{\hat{w}}(sq.ft_{test1}))^2$$
$$+ (\$_{test_2} - f_{\hat{w}}(sq.ft_{test2}))^2$$
--- for all test data

Variation of Test/Training Errors with model complexity



* As we increase model complexity we fit the training data more precisely as seen in overfitting for 13th polynomial hence training error decreases.

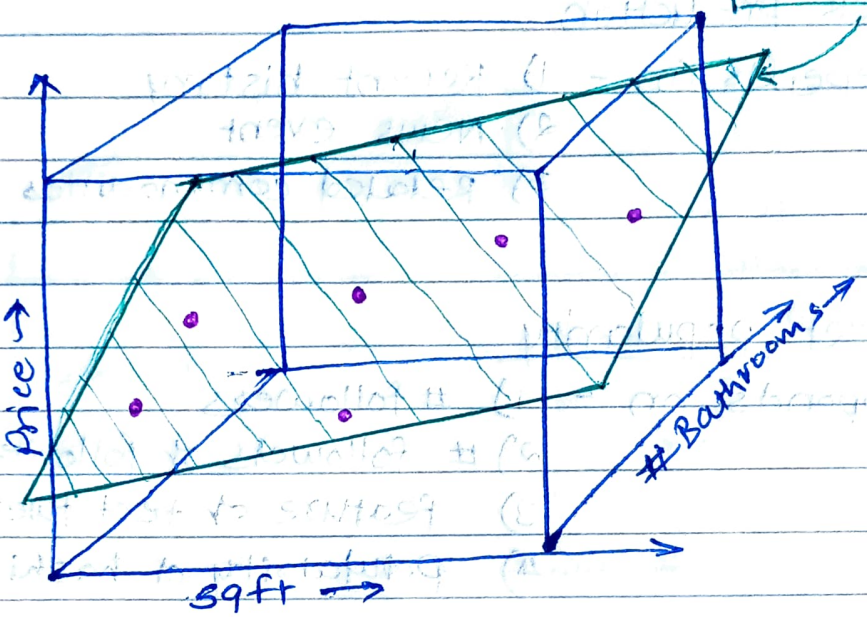
* But with increase in complexity test error decrease to optimal and then increases due to overfitting.

Adding other features

we can add more features to model

$$f_w(x) = w_0 + w_1(x) + w_2(x)$$

intercept, sqft, #Bathrooms



Here we added no. of Bathrooms which cause model to have two features and which is given by hyperplane

we also can add many more features to model like for house price.

↳ 1) Sqft
2) # Bathroom
3) # Bedrooms
4) Lot size
5) Year of built.

Other eg. where regression used -

1) Stock prediction

Depends on - 1) Recent history
2) News event
3) Related commodities

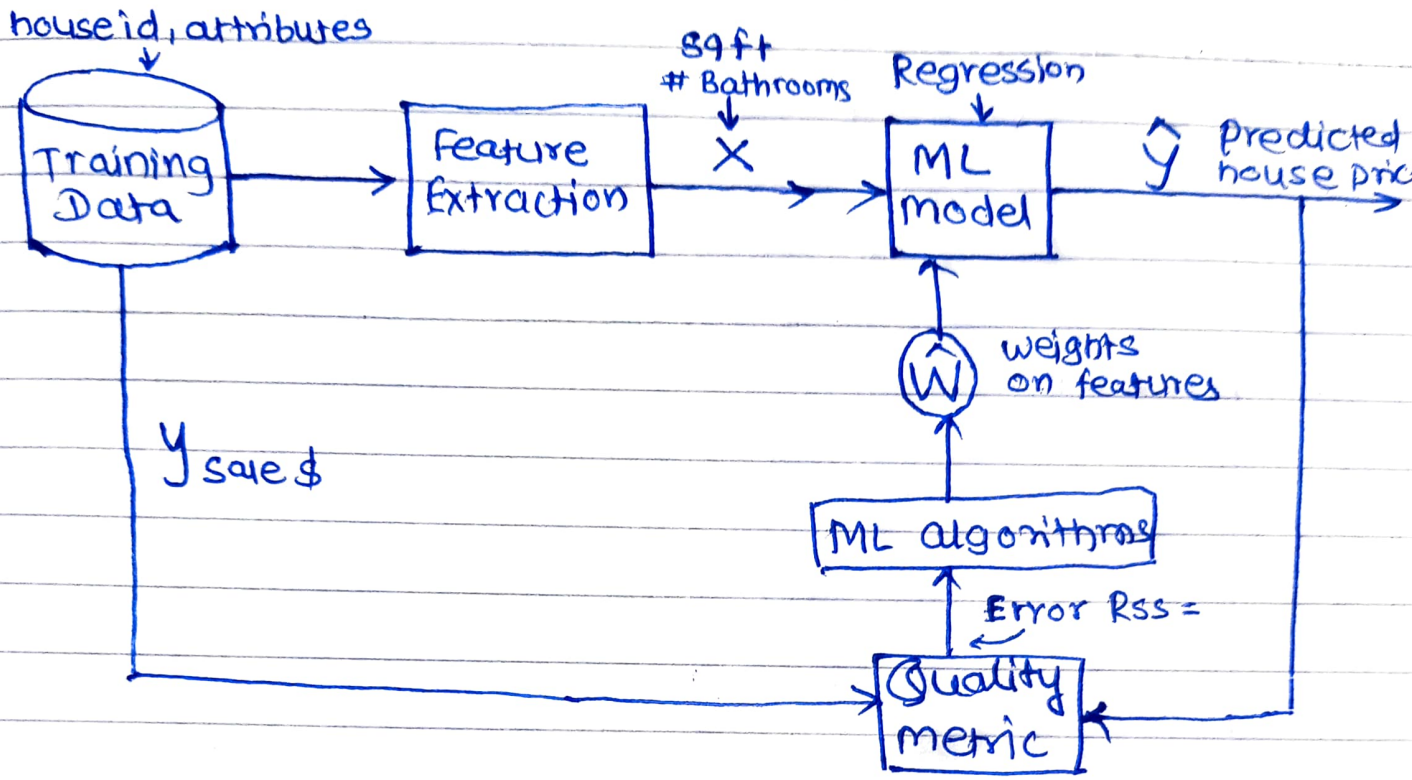
2) Tweet popularity

Depends on - 1) # followers
2) # followers of followers
3) feature of text tweeted
4) popularity of hashing.

3) Temp of particular space in smart house

Depends on - 1) Thermostat settings
2) Open/closed windows
3) Vents
4) Temp outside
5) Time of day

Regression ML Block Diagram

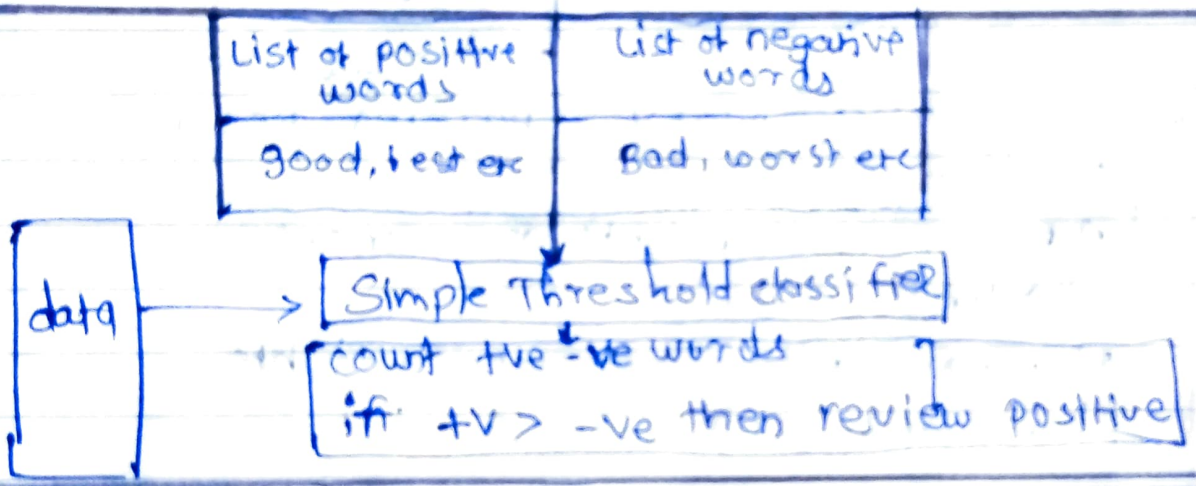


(Supervised)
Classification

IANT

plication — Spam filtering, suggesting the categories of article, video etc, Image classification., disease classification, Reading our mind.

near ssifier — To classify review as sentence is positive or negati we count all positive words negative words if #positive > # negative review is positive



Simple threshold have problems like -

1) How to get list of +ve or -ve
2) words have diff. degree of sentiment
   ↳ e.g. good < great
3) Single words aren't enough we need to address it by more features.
   ↳ e.g. good / not good

Hence we will use training data with adding weight to it as follow and examine the sentence as follow

good = 1.0
great = 1.5
awesome = 2.7
terrible = -2.1
Bad = -2

food was good; taste was awesom but service was terrible

score(x) = 1.2 + 1.7 - 2.1
= 0.8

As ; score > 0
Review is positive.

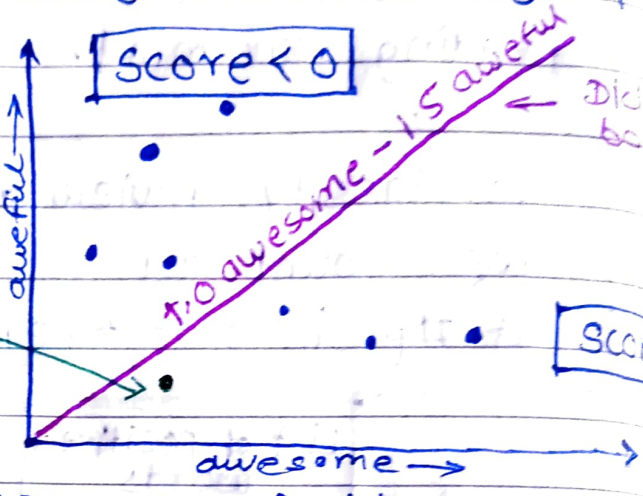
Decision Boundary

If only two words have non zero weight

words	weight
Awesome	1.0
Awetul	-1.5

Score = 1.0 # Awesome − 1.5 #awetul

* Sushi was awesome
  Food was awesome
  But service was awetul



* Decision Boundary separates positive or negative predictions.

for linear

1) ~~when~~ two weights are non zero
   ↳ line (2D)

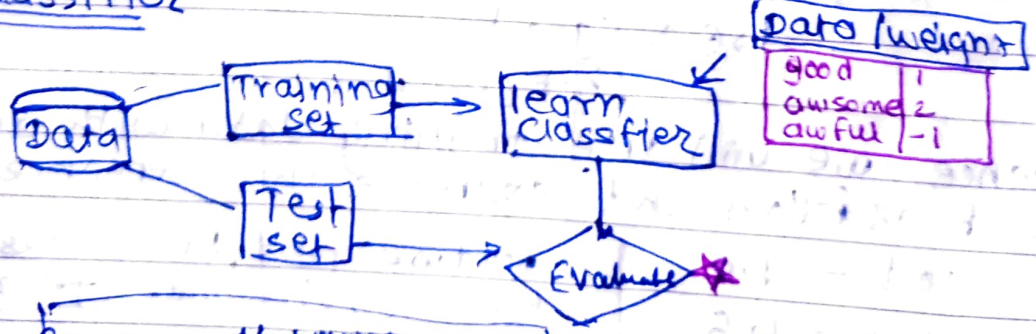
2) when three non zero weights
   ↳ plane (3D)

3) when many non zero weights
   ↳ Hyperplane

for general

complecated shapes.

Training classifier

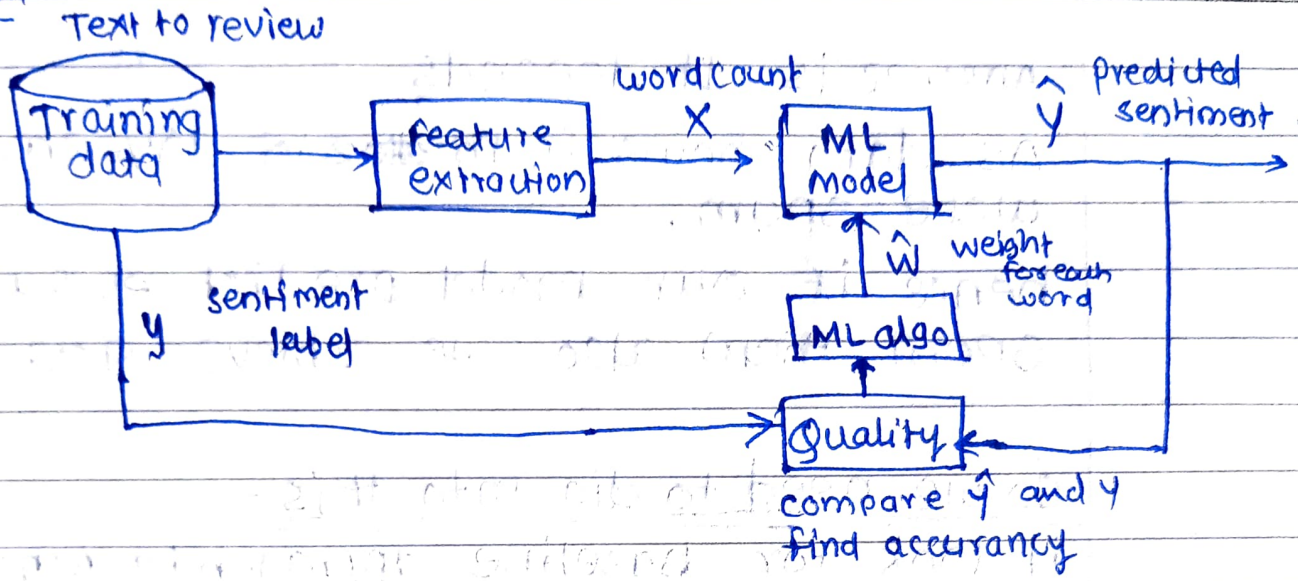


Error = # wrong prediction / # total sentence

Accuracy = # Right Pred / # total Sen

Accuracy = 1 − Error

Block diagram



* We evaluate the model by finding error and accuracy.

What is good accuracy?

When we have Binary classification randomly selecting gives 0.5 accuracy
For k classes accuracy is $\boxed{\text{accuracy} = \frac{1}{k}}$

So as with random guessing we get such accuracy we least, least, least beat the accuracy getting by Random guessing.

e.g. If we have 3 classes we have accuracy $= \frac{1}{3} = 0.333$ for random guessing.
∴ we must be having at least
$\boxed{\text{Accuracy} > 0.333}$

Is 90% accuracy is good?

Answer is It depends

As data shows from 2010 90% emails sent were spam.

hence if our model predict every email as spam, then also we achieve 90% accuracy.

So we need to dig into this -

i) look for baseline approach i.e. random guess majority class i.e. in email case.

ii) what is the impact of mistakes / errors if important enough email sent to spam if disease get undetected.

Types of mistakes

True label \ Predicted label	+	−
+	True positive	False negative
−	False positive	True negative

* Different mistakes have different impac

1) True positive
2) True negative

} Accura

3) False positive
4) False negative

} error mistk

For email filtering

False negative → Spam email in Inbox → Annoyi

False positive → Important emails lost → Harmfu

for disease detection

False negative → Disease untreated → Harmful
False positive → wasteful treatement → less harm

* Hence we need to look for the cost of wrong predictio

Confusion matrix

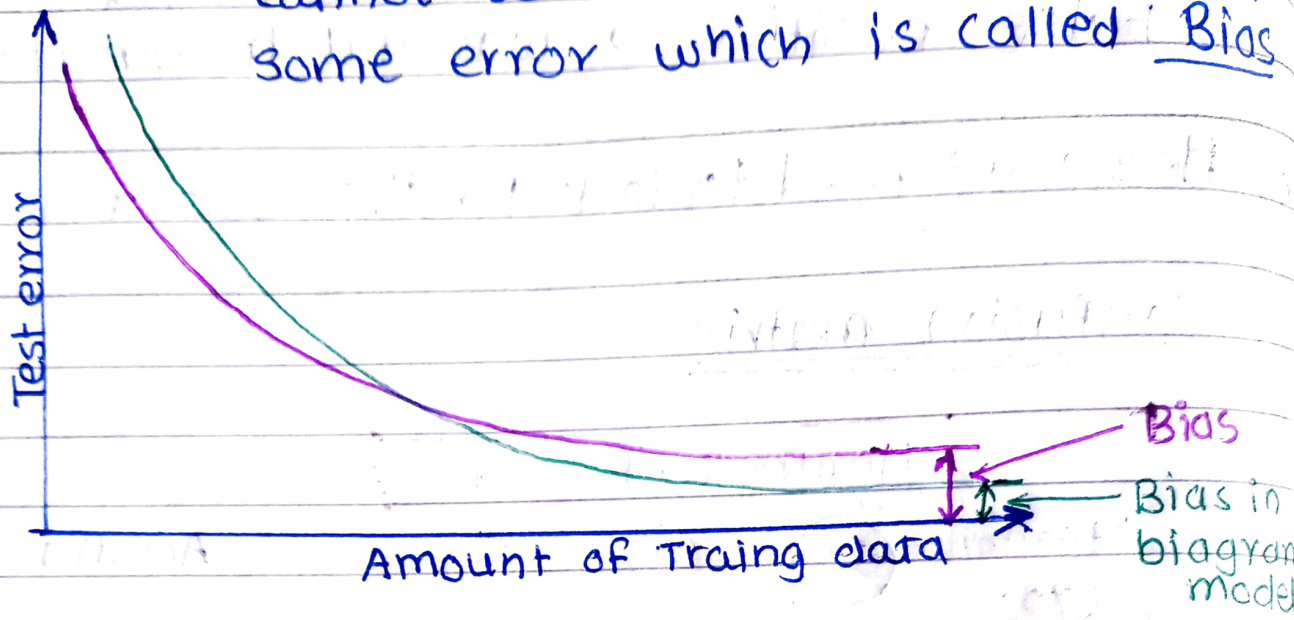
	Healthy	Cold	Flue
Healthy (70)	60	8	2
Cold (20)	4	12	4
Flue (10)	0	2	8

$$\text{Accuracy} = \frac{60+12+2}{100}$$

$$= 0.80$$

Here we need to look for the wrong prediction making what impact and what is consquences which will cost most and prepare model accordingly.

Bias - Even if we provide infinite data error cannot be 0 and their is always some error which is called **Bias**



In our model of sentiment classifier sigle words are ok but when it comes to combination of two words model makes mistake.

e.g. Sushi was not good

Hence we need more complex model as

word	weight
good	+1.5
Not good	-2.1

← Biagram model

↓

less errors
more accurate
less Bias

↓

But Not completely accurate

↓

Still have some Bias.

(unsupervised)

Clustering and Retrieval

IANT

Date:
Page:

PP — To find similar articles like documents

etrieval — 1) we find distance between Query document and
s nearest — all other documents.
neighbour — 2) Nearest neighbours returned.
search

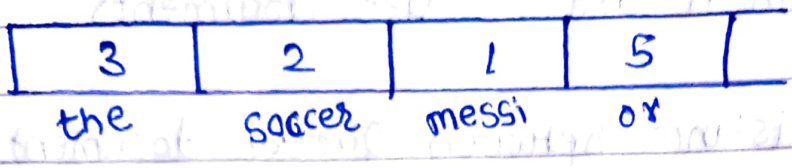
lgorithms — 1) 1 NN algorithm - most neighour nearest only on

2) KNN algorithm. - group/set of nearest neighbou

Nearest Neighbor Search

1) We get one article out of corpous
2) Specify distance of each article with every article and form distance metrics.
3) articles having lesser distances retrieved for Query article.

Word count document Representation.



We count # instances for each word and form vector for each documents and look for similarity of documents based on word counts vector products

Similarity for Doc 1 & 2 & 3

Doc 1 ⇒ 1 0 0 0 5 3 0 0 1 0 0 0 0
Doc 2 ⇒ 3 0 0 0 2 0 0 1 0 1 0 0 0

} similarity = Doc1 x Doc2 = 13

DOC 1 ⇒ 1 0 0 0 5 3 0 0 1 0 0 0 0
DOC 3 ⇒ 3 0 0 0 6 4 0 0 2 0 0 0 0

} similarity = Doc1 x Doc3 = 45

hence similar Document for 1 is 3.

Issues with word count

1) As their are common words like 'the', 'or', 'and' which are present in all documents

2) As short document have less count of words as compare to large documents hence it hampers the results.

2) To handle short and long documents we use method Normalization

Doc 1 ⇒

1	0	0	0	5	3	0	0	1	0	0

normalization ⇒ $\sqrt{1^2+5^2+3^2+1^2} = 6$

Normalised Doc ⇒

$\frac{1}{6}$	0	0	0	$\frac{5}{6}$	$\frac{3}{6}$	0	0	$\frac{1}{6}$	0	0

we use this normalised vector for document retrieval as regardless of document size it treat document counts equivalently.

1) To handle frequency of rare words & common words use method called -
TF-IDF - Term Frequency - Inverse Doc Frequency

The most common words which comes in every document must be downgraded like 'the', 'and' 'also' etc.

Common locally Rare globally

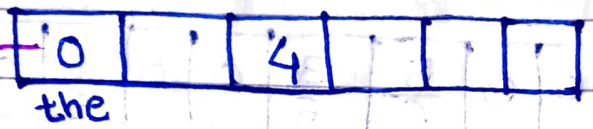
Rare words ⇒ words appear in few doc
common words ⇒ words appear in every doc

Common globally

Calculating TF-IDF

TF - Term frequency

IDF - Inverse document frequency



$$\log \frac{\#\text{Doc}}{1+\#\text{Doc having word 'the'}}$$

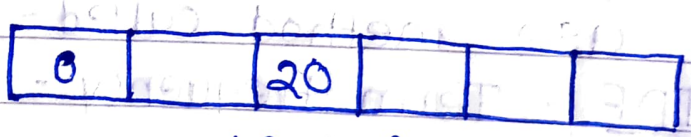
words in many document → log = large #
1+#large

words in few documents → log = large #
1+#few

* hence globally common words are downgraded and Rare words upgraded by TF-IDF

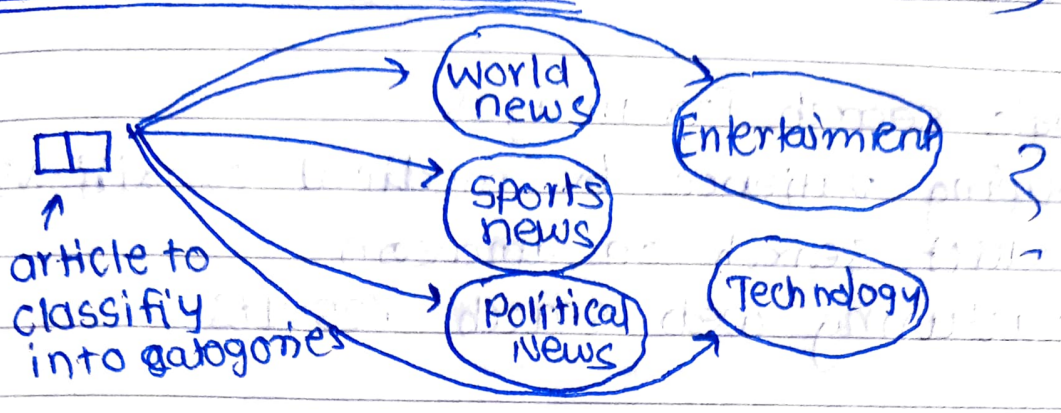
Now TF-IDF = TF * IDF

TF * IDF =



use for Document retrieval.

Multiclass classification (supervised)



Here we need to give labels to clusters which causes it becomes supervised learnings.

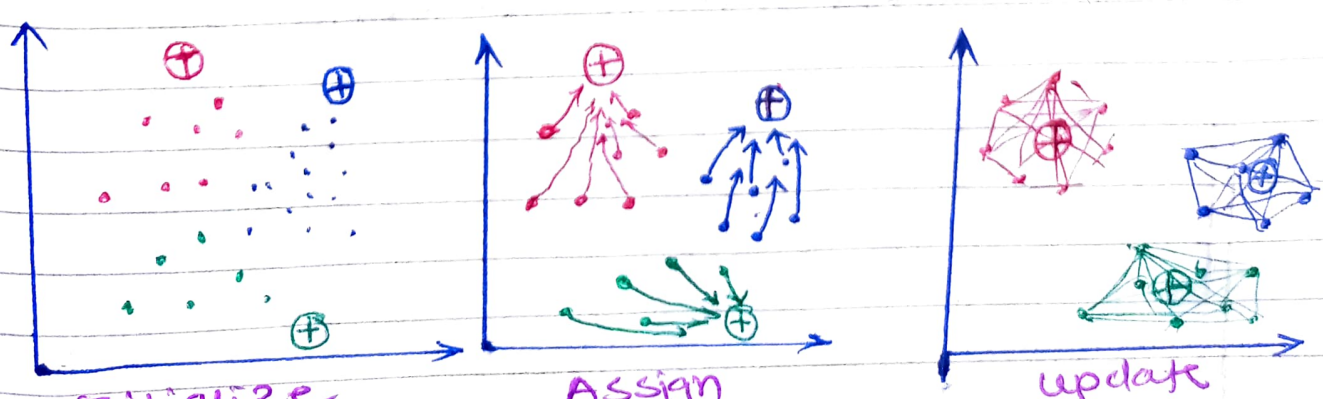
K-means Algo -

0) initialize cluster centres randomly wherewere we want.

1) Assign observations to nearest cluster centre

2) Revise and update cluster centres as mean of observations in cluster.

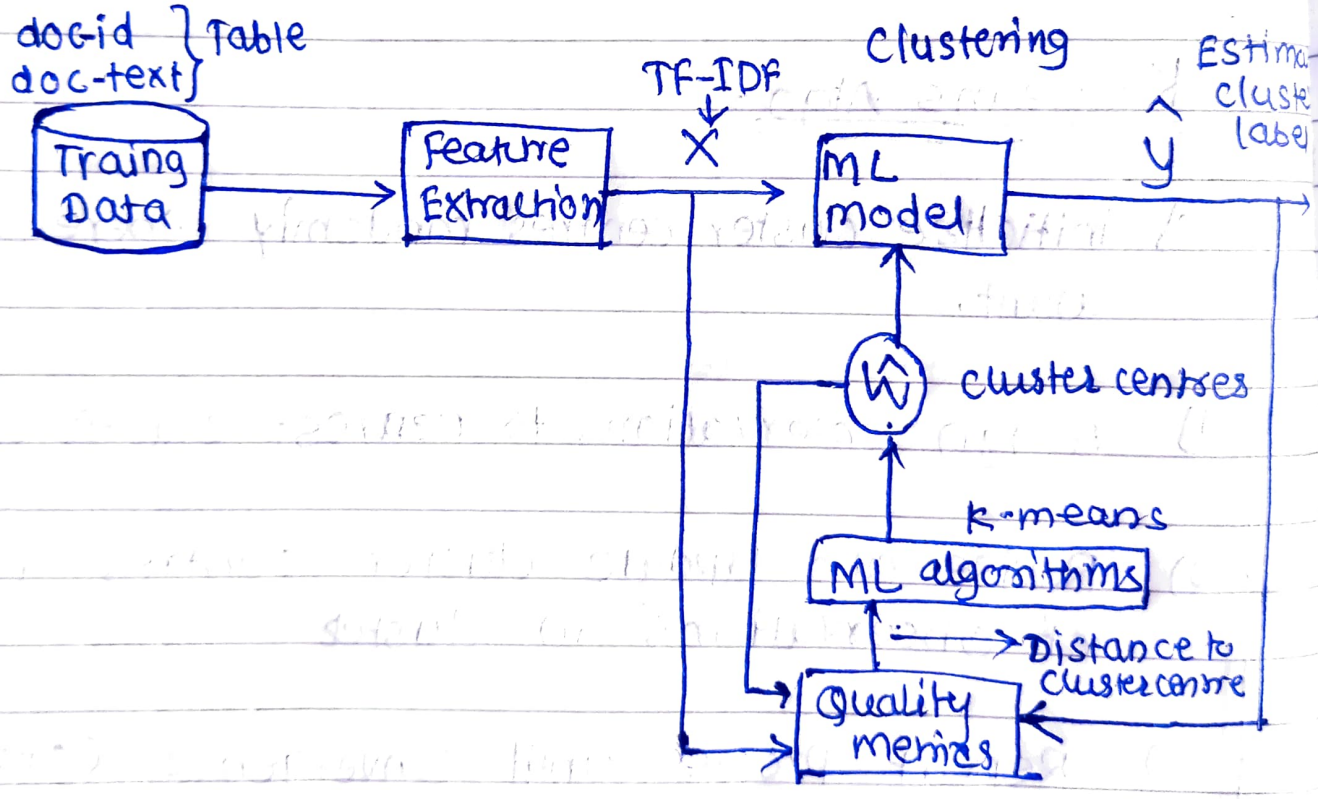
3) Repeat Process until convergence (step 1 & 2)



Examples

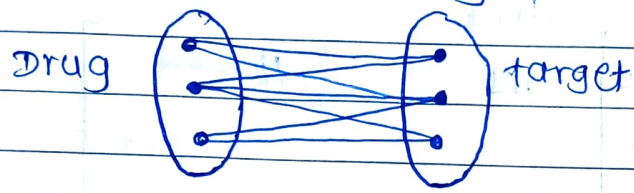
1) Google search for images
2) Grouping patience by medical conditions
3) product search on amazon
4) Structuring web search results.

Clustering Block diagram



Recommender Systems

Application - Youtube, Netflix, Amzon, music, friend suggestions, Drug target interactions i.e. if one medicine is useful in treatement of one disease it can be useful treating some other diseases.

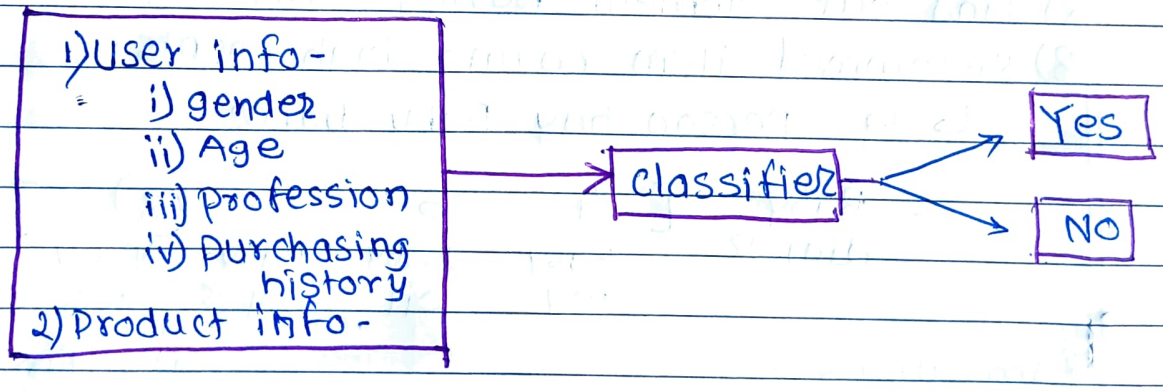


Building Recommodar System

Approach 0 - Based on popularity products are ranked and suggested accordingly (global popularity) which has limitations for personalised suggestions.

Approach 1 - classification model.

- To find wheather person would like or not purchasing the product.

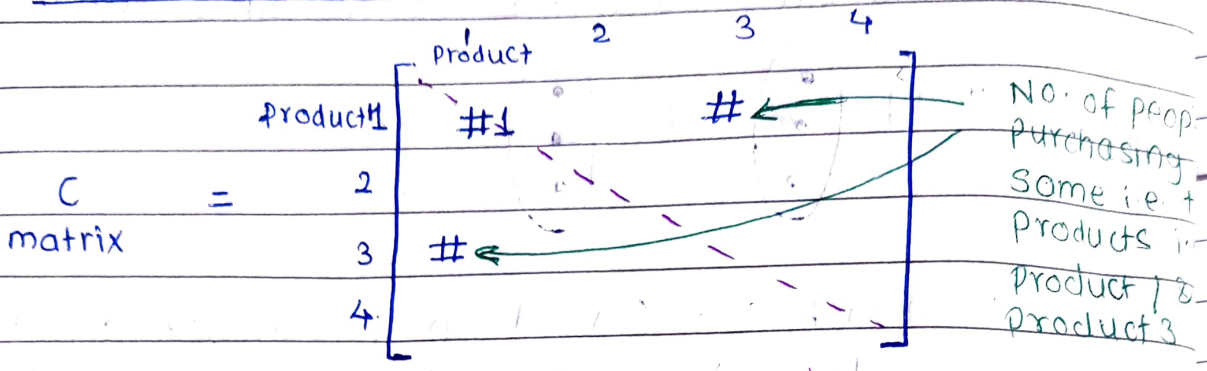


Limitations - features like user info, product info may not be available.

Approach 2 - Collaborative filtering -

- people brought this also buy ---

Co-occurance matrix



given matrix is a symmetric matrics

Hence no. people Buying product1 and no. people Buying product3 is same. $C_{ij} = C_{ji}$

To recommed the product

1) we go to product row
2) find out element having most/largest counts
3) Recommed item having high counts.
   ↳ e.g. person buy Baby diaper

Baby diaper [4 Baby toy, 5 Baby oil, 7 Baby wipes, (8) Baby food]

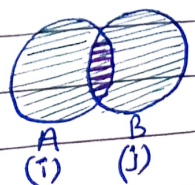
This item is more likely to be reco

Limitations -

But this causes Recommendation which is based on popularity and drown down oth effects hence we need to normalisation of popular as follows -

Normalization -

$$\frac{\#\text{ Purchased } i \text{ and } j}{\#\text{ Purchased } i \text{ or } j}$$



Weighted average of purchased items.

user bought items {diaper, milk}

$$\text{weighted score for (Baby wipes)} = \frac{1}{2}(S_{\text{babywipe, diaper}} + S_{\text{baywipe, milk}})$$

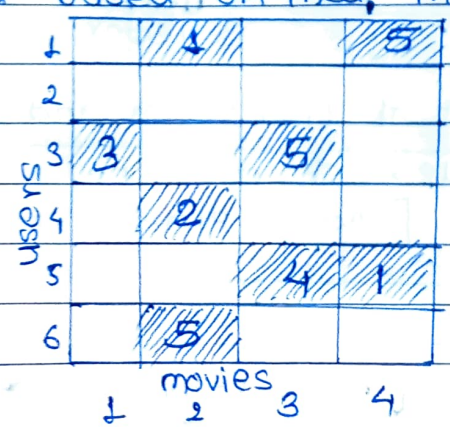
1) We look for purchased history i.e. diaper, milk.
2) We look for row of this products
3) If any item is to be recommended from cofactor matrix we look for it for e.g baby wipe.
4) We find # People purchased baby wipe & baby diaper and # people purchased baby wipe & milk and get average for weighted score.

Limitations

It doesn't look for other features like time of day, user's information & product information.
cold start problem - when their is no history.

Approach 3 - Discovering hidden structure by matrix factorization.

Let some users have watched some movies and rated those movie we have matrix with user and their rating for movie but not every movie rated by every user and this blank space can be filled up and based on that movies recommended.



Now white spaces filled up by looking for movies user rated and predicted the rating for unrated movie
Completed matrix used further for Recommendation

Let user 1 Rated movie 4 as 5 rating.

movie-5 [2 3 4 1]

Action Romance Comedy drama

vector of featur and their weight is known

user -1 [2 3 4 1]

Action Romance Comedy drama

vector of feature and their weight of that liked by user

for movie Rating of unknown movie

	Action	Romance	comedy	drama
movie 4 [	1	3	4	5]
user 1 [	2	3	4	1]

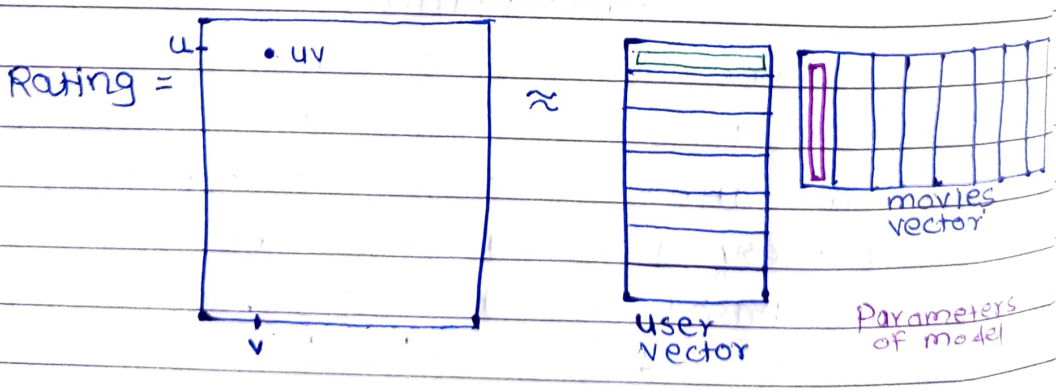
Rating that user may give is decided by produ vectors.

Rating for movie 4 = movie4 - [1 3 4 5] } = 1X2 + 3X3 + 4X4
user 1 - [2 3 4 1] }

= 32

Based on the this white spaces filled.

Predictions in matrix form.



When we get product of User vector and movies vector we get all the values of metrix.

User vector element = L_u = [Action Romance comedy drama]

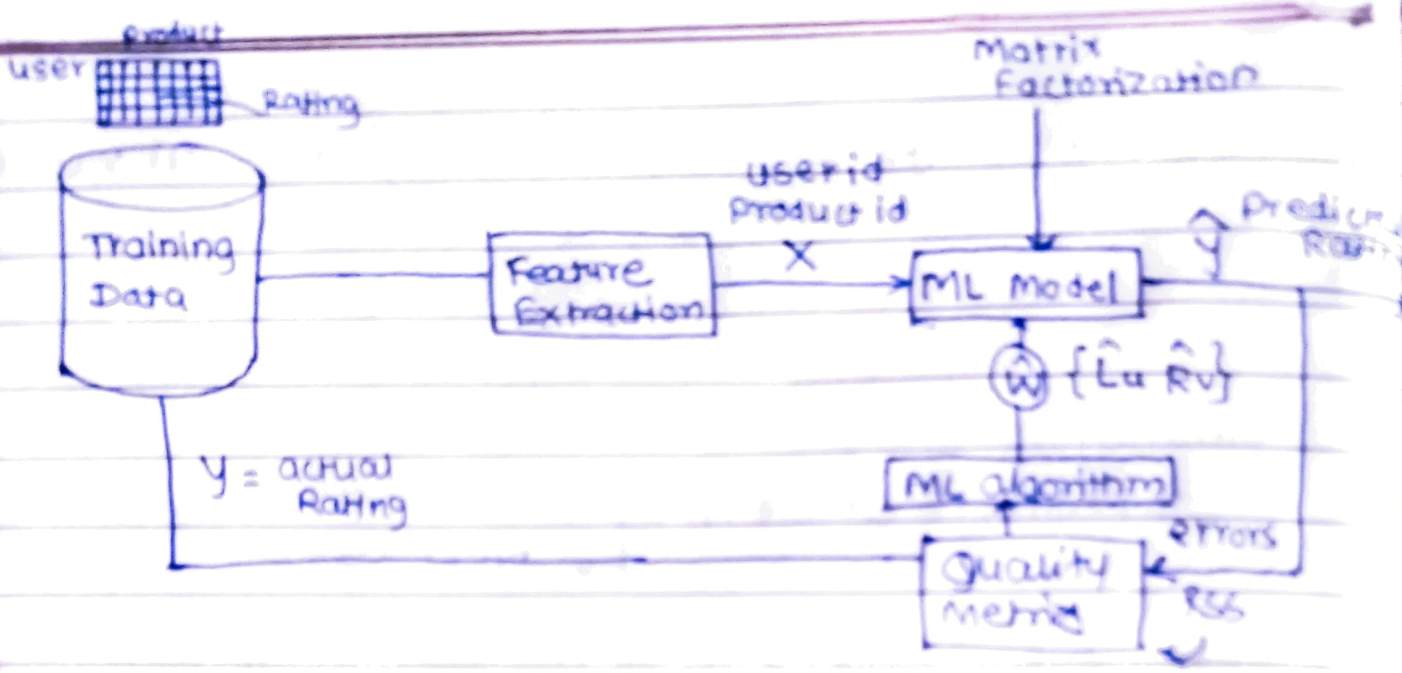
Movie vector element = R_v = [Action Romance, comedy drama]

$$RSS(LR) = (\text{Rating given} - [L_u R_v])^2 + \dots \text{all rated movies}$$

Rating given: actual Rating
$[L_u R_v]$: Predicted Rating

Limitation

cold start problem. → New movie
→ New user


Product
User
Rating
Training Data
Feature Extraction
User id
Product id
X
Matrix Factorization
ML Model
ML algorithm
Quality Metric
Errors
RMS