Module - 5

Ch- Feature Engineering

→ Featurization & feature engineering:

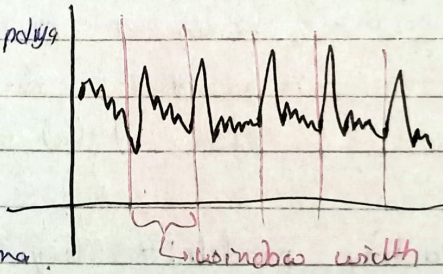
Till now we have seen, how to convert text data to numerical data, categorical data to numerical data. But there are some other types of data which needs to be converted to numerical data.

(i) Time-series data (ii) → heart rate, stock price etc
(ii) Image data
(iii) Video data.

We will learn how to deal with these type of data.
Some other biggie of data is graph data.
There are tons of types of data.
featurization is to convert that data to numerical data.

☆ Moving windows for time series data:

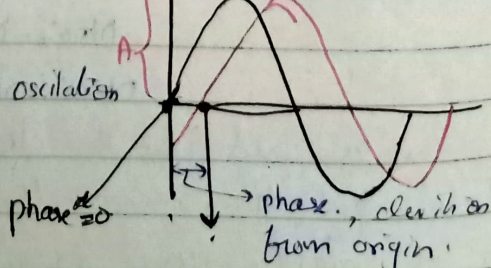
This is simplest featurization of time series data.
In each window you can choose any polys
(i) Mean, std-dev of window
(ii) Media, Quantiles " "
(iii) Ma, Min
(iv) No of local minima & local maxima



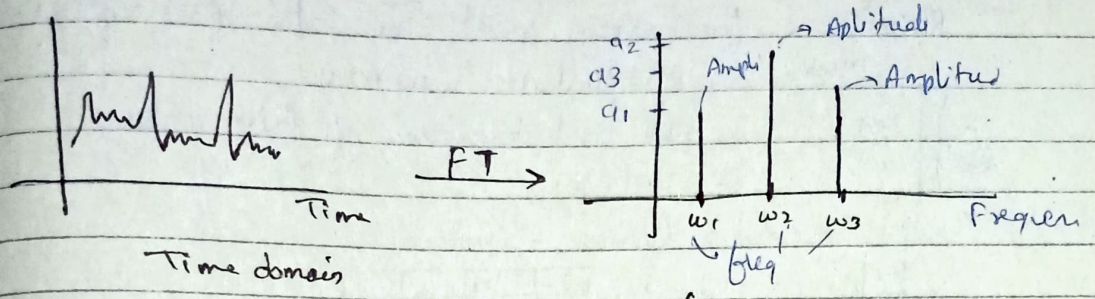
step 1: Define window width
step 2: Identify the useful features.
step 3: put all the features as x_i & identify y_i according to task

☆ Fourier Decomposition / Transform:- It is another method to represent Time-series data.

Amplitude:- height of wave.
period:- Time taken to finish one oscilation
frequency- $\frac{1}{\text{Time period}}$.



If we have a repeating pattern, we can decompose it as sum of sin wave.



Some time it is useful to represent data in freq domain.

∴ Now we get

Freq	Amp
f1	a1
f2	a2
f3	a3

we can create a feature vector as $[f_1, f_2, f_3, a_1, a_2, a_3]$

This type of representation is called fourier representation of time-series data.

* **Deep learnt features:-**

Previously, we had domain expert to identify the feature in data from time-series etc. They designed special features for each domain. But the problem is feature designed in one domain may not work in other data. This was the common practice untill 2012-2013.

But now the thing changes.

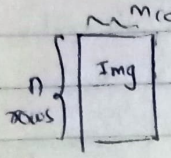
Deep learning automatically (almost) learns the best featurization for the data specially in time-series, text and image data. Only req. is it needs lots of data.

LSTM is one such model to automate featurization.

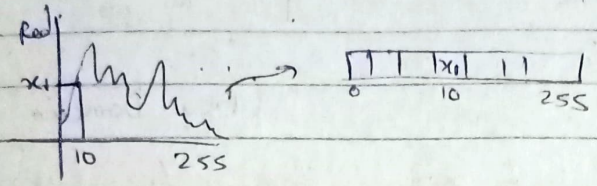
* **Image histogram:-** we deal with lot of image in real world. Hence it is more important to deal with image data. Today the best way to learn image featurization in Deep learning is CNN. But as of now we learn a old technique called image histogram

There are 2 type of image histogram (i) color histogram (ii) edge hist

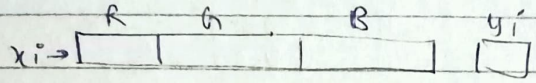
(i) Color histogram:- we know every pixel have 3 value R G B, for every pixel we take Red values.



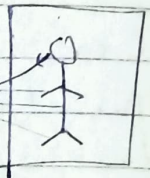
Red value = n x m;
plot a histogram of these value.



similarly do for B x G colur. Hence for a image we have 3 vectors.



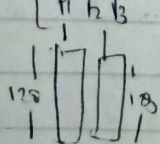
(ii) Edge histogram: each edge can have different direction. hence we can assign angle to each edge. Break up the big image into grid and ask a question, in a grid if there is a edge in cell with 0°, 45°, 90°.... etc. If there are no assign the angle to that cell/region ∴ for each region/cell we get a edge value/angle. Now plot a histogram of these value, And using the histogram convert it to vector.

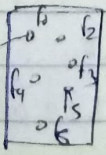


★ Scale Invariant feature Transform (SIFT)

This is a very popular image processing featurization tehn It is very useful in object detection. It identify some key points of image as feature and create a vector of 128 dim for each feature

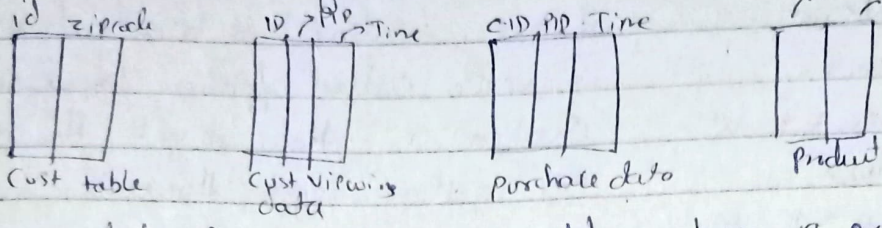
Scale invariant here means that if the image is big of small the key points will not change SIFT is also rotation invariant





* Relational data & featurization: Some time data may be in data base table, we need to featurize them.

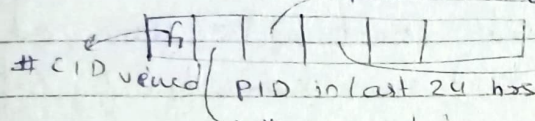
Ex:-



* Task:- predict if a customer would purchase a product in the next 7 days.

custid, Pcode → 1/0 → buy / Not buy.

So how to we come up with feature:



In general we use SQL to featurize Relational data.

* Graph - data & featurization:

In a graph based data, we need concepts from graph theory like no. of paths b/w two nodes, No. of neighbours etc. It depends on type of task.

* Feature Engineering: Indicator variable:-

Assume we have height as feature. Since height is a real-valued feature, I can leave it as is. One other approach is we can change it to indicator variable:-

if h > 150 → 1

else 0

* height is now converted to a binary value feature i.e. indicator variable feature. because value of height indicates something that if h=1 then height is 150 more than 150. * It is a problem specific and there is no hard and fast rule.

Example ① categorical feature :- country

if country = INDIA or country = USA then x

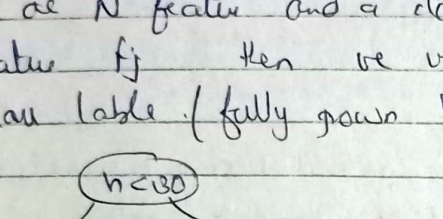
else then y;

* Feature binning:- Feature binning is a logical extension of indicator variable:- we create bins here

Ex:- If height<120 — we are creating 4 bins/category.
xtra 1 — It is also called feature bucketing.
If h>120 & h<150 — Question is How to get the right threshold
xtra 2 — Ans getting right threshold is problem spe
If h>150 & h<180
xtra 3
else xtra 4

We can also use decision Tree to make bins of a particular feature. Suppose there are N feature and a class label, we want to bin feature fj then we will creat a DT using only fj and class label (fully grown DT) using ~~same~~ fj only.

Ex:



Now we can creat bin.

* Interaction variable:- Assume our task is to predict gender using height, weight, hair length, eye color.
h, w, hL, ec → gender.
Ex: (i) if (h<150) AND (w<50kg) // Two way interaction variable
{ create a new feature f5=1 }
we can use any mathematical operator instead of AND/OR
we can also have 3-way, 4-way ... n way interaction.
Q) But the question is, given a task how to find good interaction feature.
Sol) Again use Decision Tree: Ex: h, w, hL → gender.
train a decision tree using these 3 features and for every leaf node of DT we can creat a interaction variable.

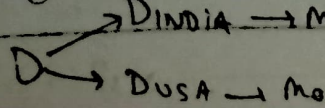
* Mathematical transforms:-
Suppose we have a feature x, we can apply log(x), e^x, sqrt(x) x^2, x^3 ... x^1y^1 ... sin x, cos x, tan x.
The Question what is the best transform: Ans:- It is very very problem spe

* model specific featurization: some type of featurization works on some model, some other type of " works better on some other model. Hence in that case we need model specific featurization. It depends on the model we are using.

* feature orthogonality:-
* The more different/orthogonal the feature are, the better would our models be.
let we have 3 feature f_1 f_2 $f_3 \rightarrow y$
if f_1 correlate f_2 f_2 correlated f_3. f_3 correlate f_1, if feature are highly correlated among themselves the overall input will be less.
* can we add a feature f_4 which is correlated with y but not correlated with f_1, f_2, f_3 :-
idea :- ① build a model on $f_1, f_2, f_3 \rightarrow y$
$\tilde{y}_i$ (prediction by model)
② calculate errors $y_i - \tilde{y}_i = e_i$, Now while designing f_4, try to correlate f_4 with e_i (error)
This methode is very much similar to GBDT. (Boosting)

* Domain specific featurization :- It is always imp to research and study existing featurization that are designed by Domain experts. Every domain is different and hence featurization using Domain knowledge is very important.

* feature slicing:- let's assume we want to predict gender.
h, w, hair, eye color, country → gender
(e.g. INDIA, USA)
Suppose 80% of data is of Indian & 20% is of USA. If we try to build the model on this imbalanced data. Model tends to perform better on people from India because more data is available for Indian. In that case slice your data using country feature and train different models for each category



The only criteria for feature slicing
(i) Each category should behave differently
(ii) Each category should be have enough data points.

CH- MISCELLENEOUS TOPICS

★ **Calibration of Model:-** Calibration of model means getting the probability of the result with result itself. for example suppose we have a 2-class classification model and it give output for x_q, $y_q = 1$ or $y_q = 0$ but calibration means we need the probabily of $P(y_q = 1 / x_q)$ or $P(y_q = 0 / x_q)$

So why do we need probabili, probabily is needed in log loss calculation.

★ **Calibration plots:-** $f \rightarrow D_n$ (model trained on D_n)

$D_{cv} = \{x_i, y_i\}$ ↳ cross validation data.

for each x_i in D_{cv}, send it to f and get $\hat{y}_i$

$x_i, \hat{y}_i, y_i \rightarrow$ True class label.
↳ model predicted value

x_1	$\hat{y}_1$	y_1
x_2	$\hat{y}_2$	y_2
⋮	⋮	⋮
x_n	$\hat{y}_n$	y

Step 1: Build the table.

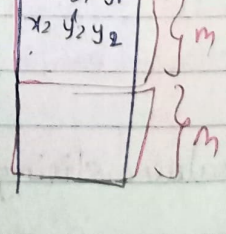
Step 2: sort the table in increasing order of $\hat{y}_i$

Step 3 break the table into peices of size m

Step 4: calculate average y_i & $\hat{y}_i$ in each peice.

$\hat{y}^{(i)}$ = average of ith chunk

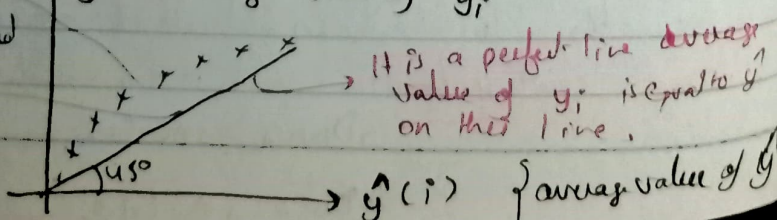
$y^{(i)}$ = " " " "



# of peices = $\frac{\text{Points in } D_{cv}}{m}$

$D_{calib} = \{\hat{y}^{(i)}, y^{(i)}\}$

Using this calibration data we can plot calibration plots.



Now the whole challange of calibration model comes down to we are given $\hat{y}_i$ predict y_i

platt Scaling / sigmoidal calibration.

Task is : ~~when~~ we have a calibration dataset of $\hat{y}_i, y_i$ and given a $\hat{y}_i$ we need to predict a y_i

$D_{calib} = \{\hat{y}_i, y_i\}$

$\hat{y}_i$ predicted by model → $p(y_i = 1 | x_i)$

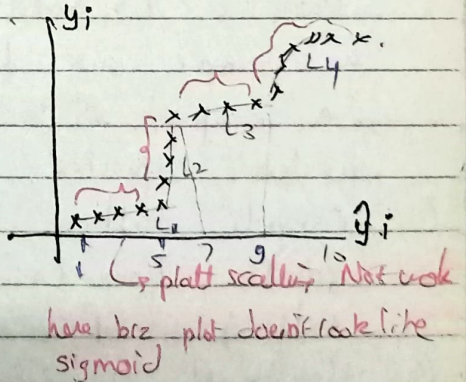
Acc to platt scaling, $p(y_i = 1 | x_i) = \frac{1}{1 + \exp(A f(x_i) + B)}$

$f(x_i) \rightarrow \hat{y}_i$

Use the D_{calib} data to find the perfect value of A & B

But platt scaling only works when calibration plot looks like sigmoidal shape.

Isotonic Regression:- This technique works even if your calibration plot doesn't look like sigmoidal curve.



platt scaling Not work here bcz plot doesn't look like sigmoid

The idea behind isotonic Regression is: Break the plot in small regions and ~~plot~~ fit a line in each region. These are called pices wise linear model.

If $\hat{y}_i$ in (1, 5) use line L_1

If $\hat{y}_i$ in (5, 7) use L_2

If $\hat{y}_i$ in (7, 9) use L_3

∴ how do we determine these boundaries and lines. Here we try to minimize a optimization problem

The same time it also try to take care that we don't have too many lines.

Isotonic Regression needs large no. of data points in D_{calib}

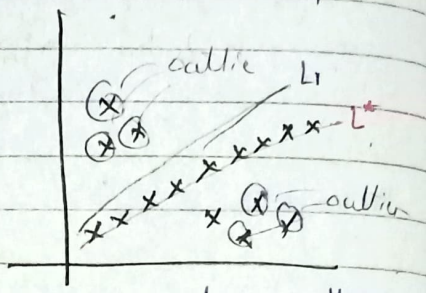
If D_{calib} is very small then use simple platt scaling.

★ Random Sampling Consensus (RANSAC)

it is statistical technique which is widely used in modern ML. It tries to ans, can we build a robust model in the presence of outliers.

for simplicity let assume our model is linear regression. In Lr. Reg we try to fit a line/plane in the data.

$D_{Tr} = \{x_i, y_i\}$ $x_i \in R$, $y_i \in R$



if we train a model on D_{Train} we might end of getting L_1 as our line because Lr. Reg try to minimize the squared loss for each points.

But problem is, some of the points are very far, then the sq-loss will be larger hence they tend to pull the line towards them. Hence L_1 we got is a non-robust because it is impacted by outlier.

RANSAC is the solution.

① Given D_{Train}, get a D_0 = Random sample(D_{Train})
Build a Lr. model on D_0, let we get a line L_1. When we sample our data, probability of an outlier point in sampled data is very less. Hence we would get a less impacted line due to less No. of outlier.

② Compute outliers based on L_1 (by calculating distance of each points from L_1, points which have high distance are called outlier) let we get a set $O_1 = \{\text{outliers}\}$.

③ Construct a new data set $D^1_{Train} = D_{Train} - O_1$:-

④ Construct a Data set D_1 = random sample (D^1_{Train}), make a model L_2 & find outlier $O_2 = \{\}$
Repeat the step.

Let we get a model L_i in i^{th} step & L_{i+1} in $(i+1)^{th}$ step. If they are very much similar then we will stop. Now we make a final model on $D^{(+)}_{Train}$.

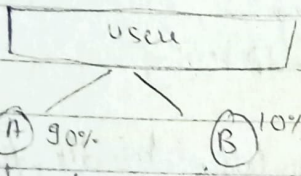
Retraining model periodically:-

A/B Testing :- It is also known as Bucket Testing or split run or controlled experiment.

Ex assume we are building a model for Google to predict which ad to show for search query.

So how does A/B testing work here

Suppose we have a old model and we have millions of user. Divide them in two part



No of Click, No of add shown

No of clicks ; No of add shown.

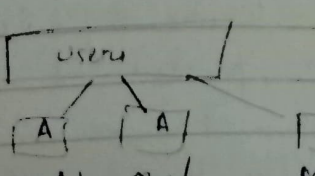
Now we get accuracy of both model & $\frac{\#\text{ of clicks}}{\#\text{ of add shown}}$

If accuracy A < accuracy B

Now we repeat this with changing the percentage to 80, 50 - 50 etc. and if we are very sure then only we can deploy model new.

But one thing to note is, Accuracy may not give correct interpretation. Hence instead of using Accuracy, we can use confidence interval. And if confidence interval of A & B are not overlapping then we can be sure about the result.

There is also a concept called A A testing:-



Data Science / ML project life-cycle :-

1) Understand business requirements :- define the problem, identify customer
2) Data aquisition :- ETL (Extract, transfor, load.), SQL [DB, DW, log-files, Hadoop]
3) Data preparation :- cleaning, pre-processing.
4) Exploratory data analysis : plots, viz, hypothesis testing slice & dice data.
5) Modeling, Evaluation, & interpretation.
6) Communicate results : clear & simple, 1-pagers., 6-pagers.
7) Deployment : engineering
8) Real world testing : A/B testing
9) Customer / business buy-in
10) operations : retrain-models, handle failures, process
11) optimization : improve models, more data, more feature, optimize code.

VC-dimension :-

Q) Is there any methode to measure how powerful a class of models is

is there any numerical value that can be assigned to model which tells about power of class of model.

→ Linear models
→ RBF-SVM
→ Boosting algo
etc.

VC-dimension is one such concept
↳ Vapnik > Chervonenkis

VC-dimension is a theoritical concept, but it is not use much in practical scenio.

Ex: take a linear models : line / plane / hyperplane are decision surface Ex:- logistic Reg, Linear SVM

Imagine we have only 2 points + & ve

+ -

in whichever they are placed, we can always find a way to split the space to separate these points

Now, let we have 3 points, even in this case we can separate them using line/hyperplane.
Now what about 4 points: + − we can't
− +

Separate them.
There exist atleast one configuration s.t there 4 points cannot be shattered/separated
Hence VC dimension of linear model is 3
VC-dim(Linear_model) = 3
VC-dim(RBF-SVM) = ∞
theoritically RBF-SVM are super powerful but practically it is not the case.

★ Productionizing models

We have the model f, in the runtime we keep getting x_q and we need to send y_q or Pror(y_q=1) or P(y_q=0)
productionizing models mean Running the model in real world.
There are 2 methods for this
① ~~model persistance~~ Using sklearn.
model persistance: store the model in HDD and when you need it, load it to ram and just give query points. We can persit a model in ① string ② file.
sklearn stores model in a pickle file and we can save this .pkl file to disk

② Custom implementation approach:-
(a) store all of the parameters of model to a file.
(ex: logistic Reg - store weights.
So, why do we need it, because python is slow we can use the calculated parameters and create a calculate y_q using C/C++/Java which are fast languages