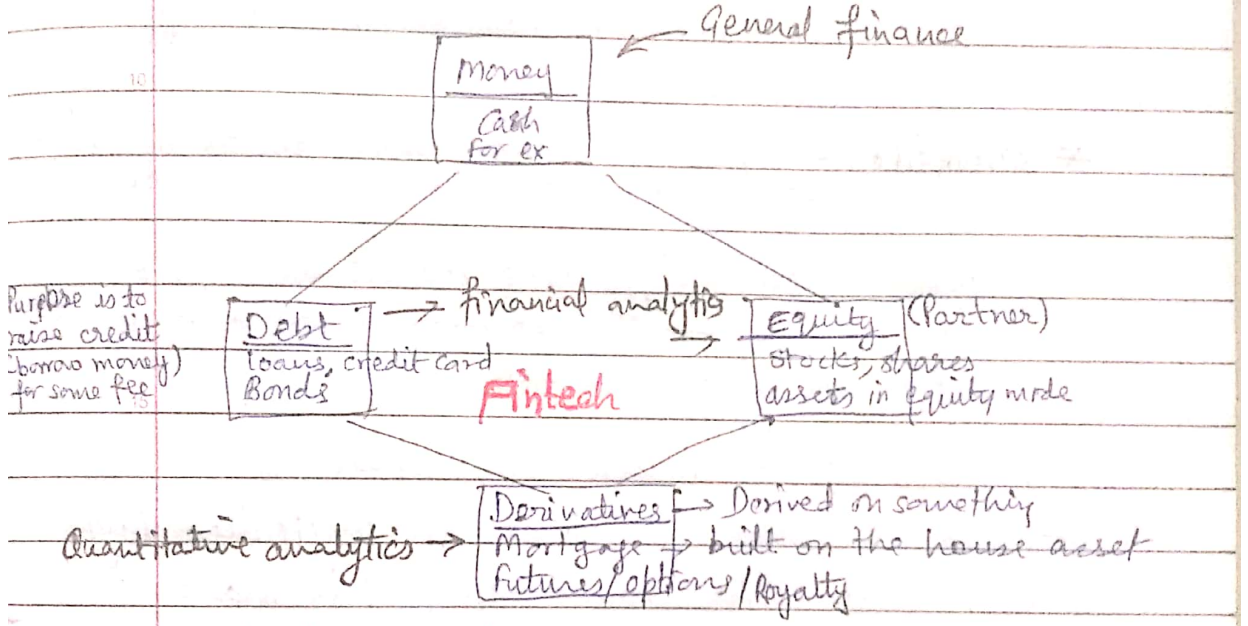
Finance & Risk Analytics

Abhinanda Sarkar

Base is money - shared, traded for other things
money moving around (forex)

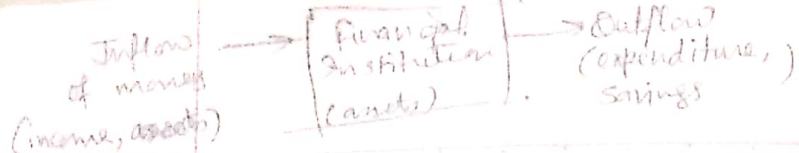
Liquidity risk - will you be able to buy/sell anything
- ex need to sell apartment, but not getting buyer
- You wanted cash, but you can't have it



Fintech - technologies that are used to facilitate all the above boxes.
for ex, Paypal, paytm, security of a transaction visa/master. Anything which enables financial transactions, say internet banking.

Policy nature of finance (rules & regulations)
RBI, CFPB etc

Only govt. has rights to create money, that's why on a note, RBI governor declaration is ~~done~~ given.
However, its debatable now whether other than govt anybody else should have rights to create money or not like bitcoin, blockchain (can money be created this way? Is it valid?)



Risk - Uncertainity, not just -ve, can be an opportunity also (Something may happen increases unpredictability)
- Downside *

Hedging - Have another source of income apart from salary - ~~Sale~~ to pay the loan. Salary is uncertain, though another source of income is also uncertain, but both together would help in reducing the risk of not being able to pay the loan.

* Downside - future plans issues due to uncertainty
- getting a picture of future, what could happen.

Both uncertainity and downside helps to take the decision → This exercise is done in banks or financial institution as well

Risks of Financial Institutions

- Risk associated with what an institution buys or sells. For banks they take money from customer (in terms of savings/FD) and sells back to customer again (in terms of loans, card).
Bank's profit = selling - buying.
Banks biggest risk is that the trading business would fail - Known as credit Risk
Credit means somebody owing someone something.

2 ~~Ring~~ Kind of credit risk

1) As a buyer - bank owes you money
2) As a seller - Somebody owes bank money
Bank may transfer its risk to 3rd party but as a consumer, bank owes you money and you cant share your risk with anyone

Default/Credit Risk is when any of the party is getting default

Market Risk — changes associated with the market, eg. Policy changes, interest rates, state of share market (deals with what a market is). Market — Real/Virtual env where buying/selling/trading is happening
For IT professional — State of IT industry is market risk. Not in control of an individual
Things ~~can~~ that can be controllable —
can be put in credit risk
Things that can ~~be~~ not be controlled —
can be put in market risk
Operational risk — Not associated with any financial institutions. It is just the risk of doing business. ~~Reg~~ eg. Attrition of employees (could be in control / could not be in control)

Liquidity risk can or cant be treated (often treated) as market risk as most of the time it is generated by market.
However its implications are on credit risk

Main analysis in credit Risk
① Probability of default — if the person is going to default or not
② Exposure at default — when something goes wrong how much is the impact (30%, 40%—) how much it is going to hurt me.

LR - Whether the person would default or not (0 or 1)
DA - What are the special characteristics of a 0 person or a 1 person so when a new person comes in, does he/she have characteristics of a 0 or 1 (Statistical method for differentiation)

(3) Loss given default - Default has happened. It has impacted me. So how much is the final value now (could be a number or %age)

Specific kind of market risk -

- Stock market - lot of uncertainity, so you need to know what goes in there

* Data on multiple companies - 3 yrs data of weekly closing stock price

- Work with returns (not the actual price)

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

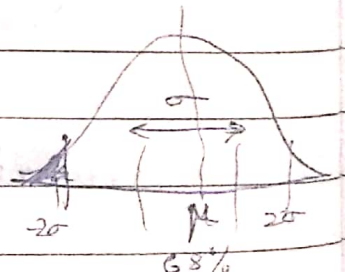
P = Price
R = Return

So convert every price value (weekly values) into returns and then work with distribution of returns further

- Distribution of returns

$$R \sim N(\mu, \sigma^2)$$

↑ normal distribution

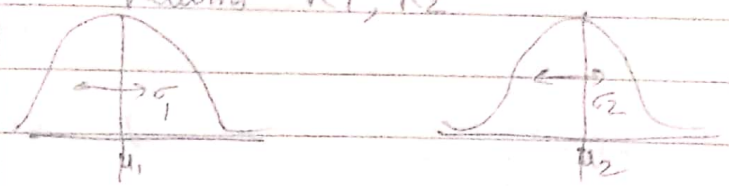


For portfolio optimization, if we understand the distribution of returns, we can take better decisions like I know average return (μ), 1σ of it has 68% possibility, 2σ (has higher variance) but possibility of loss is also high and possibility of gain is also high

Portfolio Management

2 Stocks in Portfolio

Returns R_1, R_2



← variance

Expected return $E(R_1) = \mu_1$ $\quad Var(R_1) = \sigma_1^2$

$E(R_2) = \mu_2$ $\quad Var(R_2) = \sigma_2^2$

Correlation b/w R_1, R_2 = $Corr(R_1, R_2) = \rho$

Now you want to find in which proportion, we need to invest in 1st stock and 2nd stock

Suppose this proportion invested for stock 1 = wt 'w' on R_1

Weight w on R_1 and $(1-w)$ on R_2

$$0 < w < 1$$

Portfolio Return (R) $= wR_1 + (1-w)R_2$

Expected Return on R $E(R) = w E(R_1) + (1-w) E(R_2)$

Variance on R $Var(R) = Var(wR_1 + (1-w)R_2)$

$= w^2 var(R_1) + (1-w)^2 Var(R_2)$

$+ 2w(1-w) Cov(R_1, R_2)$

$$\rho = Corr(R_1, R_2) = \frac{Cov(R_1, R_2)}{sd(R_1)\, sd(R_2)}$$

Variance of portfolio

$$Var(R) = w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\rho\sigma_1\sigma_2$$

When 2 returns are uncorrelated,

i.e $\rho = 0$, $var(R)$ depends on 1st 2 terms

If ρ = +ve, var on portfolio increases

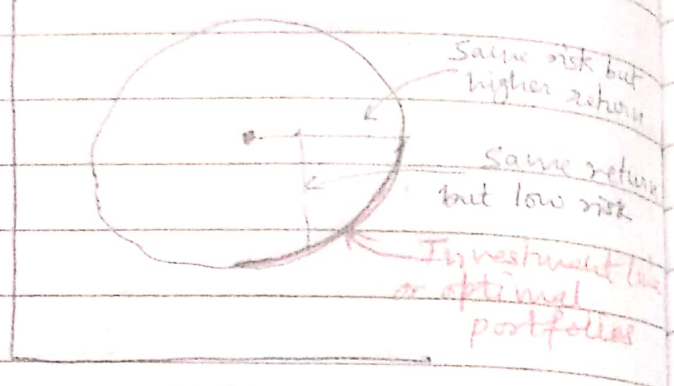
if ρ = -ve, var on portfolio decreases

So we should pick -vely correlated stocks

i.e if one stock goes up, other goes down

this reduces the variability and in turn Risk

— ρ = the implies more risk but higher returns as well



$E(R) = w\mu_1 + (1-w)\mu_2$

$sd(R) = \sqrt{w^2\sigma_1^2 + (1-w)^2\sigma_2^2 + 2w(1-w)\rho\sigma_1\sigma_2}$

Being on border is better than being in the circle

If I pick 2 stocks with

$\mu_1, \sigma_1 = 5, 2$

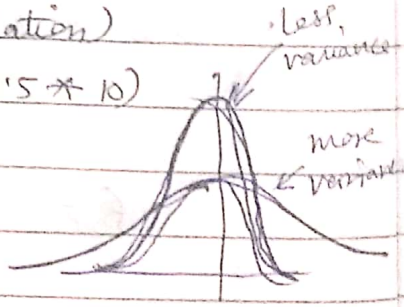
$\mu_2, \sigma_2 = 10, 5$

Now if $w = 0.5$ - (50% in R_1, 50% in R_2)

$\rho = 0$ (no correlation)

$E(R) = 7.5$ $(0.5 * 5 + 0.5 * 10)$

$sd(R) = 2.6925$



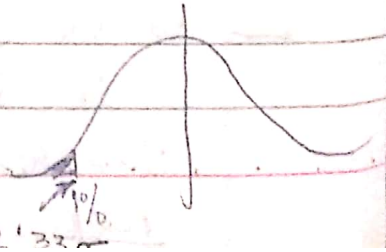
Now if $\rho = -0.5$

$sd(R) = 2.179$ (with same return as $\rho = 0$)

So how to do the trade off b/w risk and returns

So I dont want to play very safe ~~but~~ in terms of returns but at the same time I dont want to sink down (save from worst case)

Say grey area says 1% chance of losing, 99% chance of gain



So qt(norm) qnorm(0.01) = −2.3263
i.e −2.326 σ point corresponds to lowest 1% risk point. Also known as Value at risk (VAR) — estimate of worst case scenario
So we need to maximize VaR i.e
$\mu - 2.33\sigma$
Two ways to maximize are — either increase μ or decrease σ
VaR — Best of the worst
↳ a measure of "Typically" Market Risk
So in example μ is given, σ is given you need to find weights of portfolio which would maximize VaR.
To convert price matrix into return matrix

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

Acc.ts = as.ts(Acc) ← Convert data into time series data
LAcc.ts = Lag(Acc.ts, 1)
Acc.ts − LAcc.ts ← change in Price
trans = (LAcc.ts − Acc.ts) / Acc.ts
mean(trans) = 0.00606
sd(trans) = 0.04113
mean(BHEL) = −0.00020
sd(BHEL) = 0.0434
Cor(trans, BHEL) = 0.3412508 ← the correlation
It shows high risk
Sim = 0.5 trans + 0.5 BHEL
mean(Sim) = 0.002927 mean is less than individual
sd(Sim) = 0.034 stocks but sd also decreases

To find VaR - $\mu - 2.33\sigma$

mean(sim) − 2.33 * (sd(sim)) = −0.07782

So 7% is worst case scenario on week on week date of 3 yrs horizon, that is 0.5 ACC and 0.5 BHEL. So 1% chance that I can loose more than 7%.

Nao we need to try with different weights and choose the one which is giving best VaR.

Get Covariance matrix of whole portfolio (Σ)

- var(cbind(trains, BHEL))

Mean Return of portfolio ↓

$$[w_1, w_2 \text{ ---- }, w_{10}] \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{10} \end{bmatrix} = \mu \text{ (mean return of portfolio)}$$

$$w' \Sigma w = \text{Var}(R) \quad \text{(variance of portfolio return)}$$

w' ↗ transpose of weight vector; Σ ↑ Covariance matrix

Can be applied for number of shares (as weights) instead of investment proportion.

Componando / Dividando problem

average of ratios is NOT same as ratios of average

Easy way to automate it is by "Optimization"

- Convert prices into returns in excel
- Create column of weights (w_1 — w_{10})
- Get average returns and multiply be weights

mean of each column

- for variance, $Var(w_1R_1 + w_2R_2 + \cdots w_n + R_n)$
- $sd(R) = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + \cdots + w_n^2\sigma_n^2 + 2w_1w_2\rho\sigma_1^2 + \sigma_2^2 + \cdots}$

Where σ_1 is sd of 1st column

- Maximize return $w_1\mu_1 + w_2\mu_2 + \cdots + w_n\mu_n$
- Now setup criterias VaR

$\mu - 2.33\sigma$

- Objective is VaR (maximize)
- wts b/w 0 and 1
- sum(wts) = 1 (100%)

VaR computation – 2 techniques

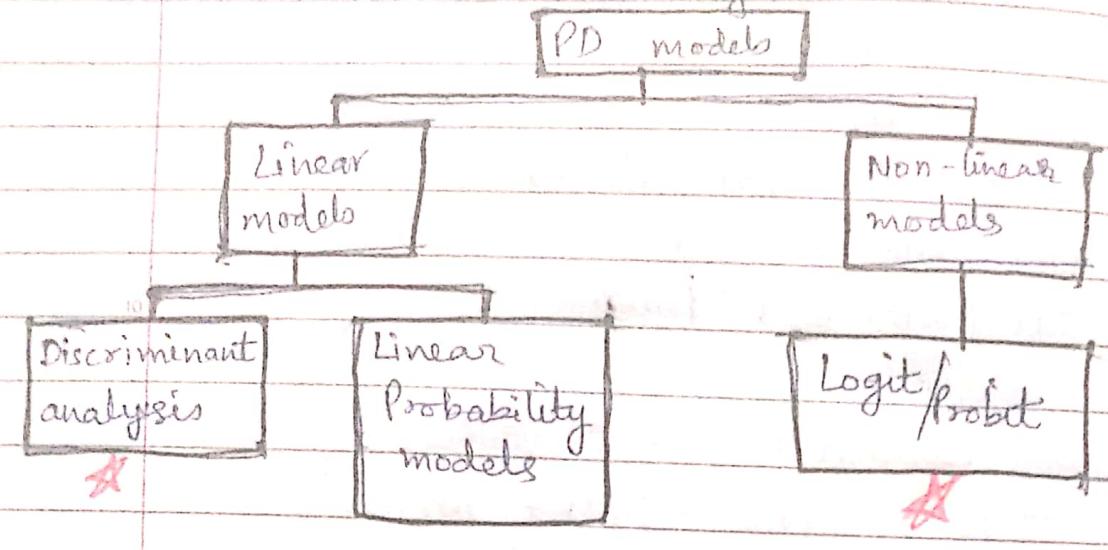
- Parametric – Delta normal
- Historical Simulation

Credit Risk

- How decisions are taken in financial industry
- How ML is adopted in taking the decisions
  - Is the person capable of paying back
    - If yes, how much should I give
    - If not, how much should I give or should not give
    - For what period, what special conditions
- At No wrong activity has happened so far, You need to assess the future based on current scenario and then take a decision

- Look for key attributes
- Check various models

PD models - statistical models in credit risk modelling



A person may voluteerily go default which is termed as Fraud

~~Fra~~ As a ML algorithm, you try to catch the pattern of defaulters/fraudulators. However, fraudulators also at the same time learn to discover new patterns (in vode old ones)

For ex. if a rule says that if a person is using ATM card more than 30% beyond 30 km circle, mark the person suspicious.

Now that person would purposefully make the transactions within 30 kms (say 29 kms), if he gets to know about this internal rule.

LR - finds the probability of 0 and 1. In this case probability of default

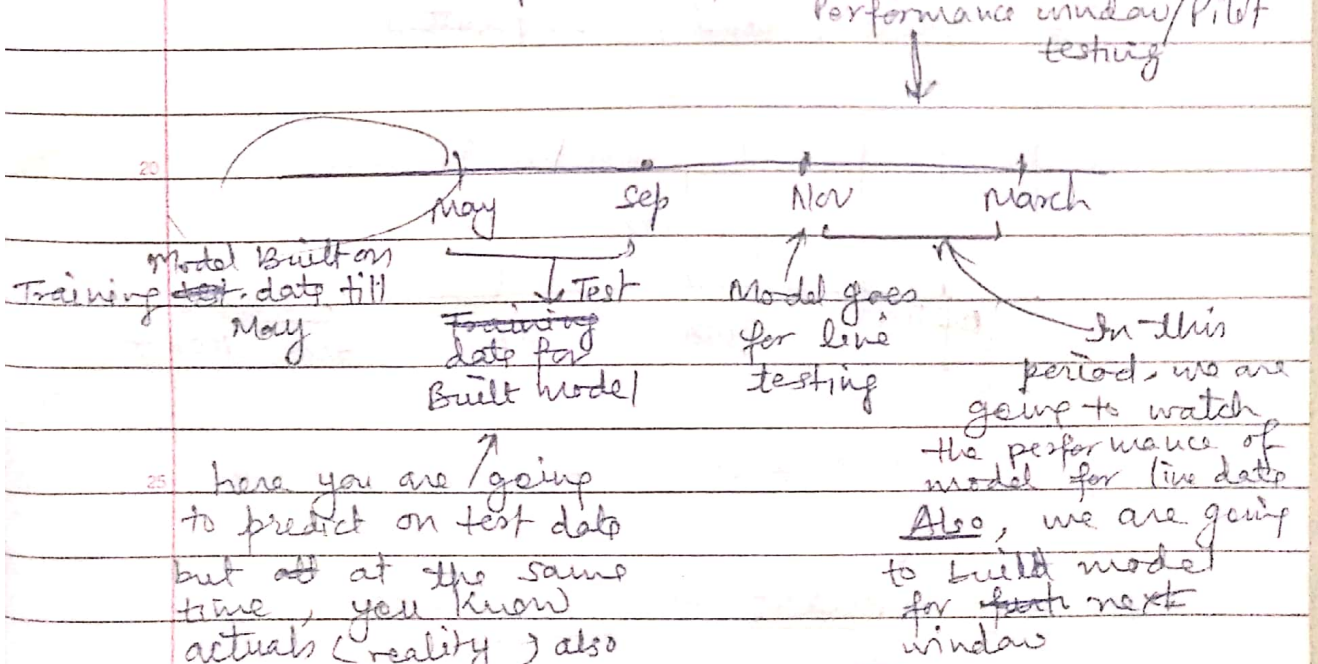
DA - Gives probability of 1 given various ratios (posterior probability). So we use prior probability to get posterior probability

Prior & Posterior Probability

Posterior probability is the probability of assigning to groups "given" the data (condition)

Prior probability is the probability that an observation will fall into a group before you collect the data. For ex, when you know 60% males get default and 40% females get default before running any model (historical), it is called prior probability.

However, after working on different attributes you find out ~~that~~ some different probability, after running models and various algorithms, it is called posterior probability



Generally, analytics team keep building the model every quarter. (If old one is failure, to build again. If success, to build better)

PD - Linear models

① Altman's Z score model (1968)

$Z(Public) = 1.2X_1 + 1.4X_2 + 3.3X_3 + 0.6X_4 + 1.0X_5$

where $Z > 2.99$ is healthy

$Z < 1.81$ is unhealthy

$1.81 < Z < 2.99$ is intermediate

X_1 = Working capital / total assets

X_2 = Retained earnings / Total assets

X_3 = Earnings Before Interest and Taxes / Total assets

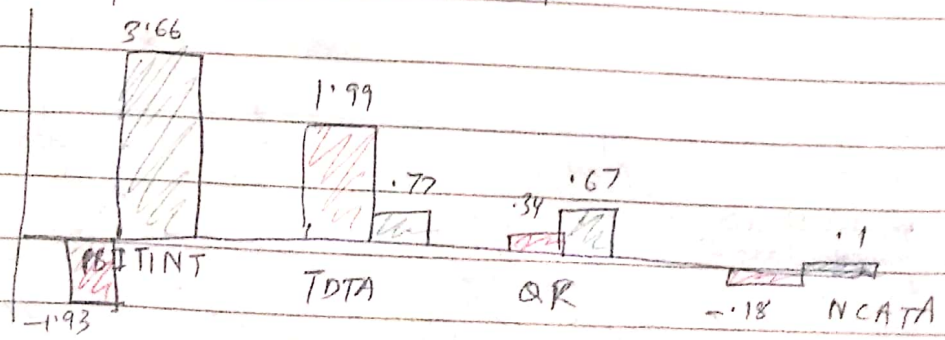
X_4 = Market value of equity / Total assets

X_5 = Net Sales / Total assets

Based on above, he did DA to seperate out a good company and a bad company.

→ Deep dive on variables ∴

clear seperation in mean values of variables b/w defaulters and non-defaulters



• - default ○ - non default

② Discriminant function (2000)

India Z score model

$Z(Public) = 1.06 + 0.01\ PBITINT - 3.12\ TDTA + 0.48.QR + 4.61.NCATA$

PBITINT – Profit before Interest & Taxes / Total assets

TDTA – Total borrowings / Total assets

WR – Current assets – Inventories / Current liabilities & provisions

NCATA – Profit after tax & depreciation / Total assets

However these kind of clear cut rules (1 and 2) might not work as ppl can easily break them (by understanding and working to satisfy it)

Challenges of DA

- Zone of indifference (Intermediate)
- Sensitivity to industry
- Dealing with new companies
- Loss of predictive power across time – use of penalty functions
- Multi collinearity of variables
- Some variables provide the same information and are highly correlated
- Assumption of multivariate normality of variables.

(3) Logistic model

Logit (score) = $-6.9965 + 4.8879 *$ Total Borrowings / Total assets $- 0.455 *$ Net working capital / current liabilities $- 6.8605 *$ Net cash accruals / Total assets $+ 1.6317 *$ Current liabilities / Current assets $+ 0.1978 *$ Total borrowings / total liabilities

When to use logit and when to use RF.

- logit gives probability of default but at the same time, also gives the coefficients of variables. So answers questions like why this variable is important or 1 unit increase may impact how much.
- RF gives imp variables (based on max splits) but does not answer to what extent or why this variable is imp. However, gives good prediction
- SVM, xgboost

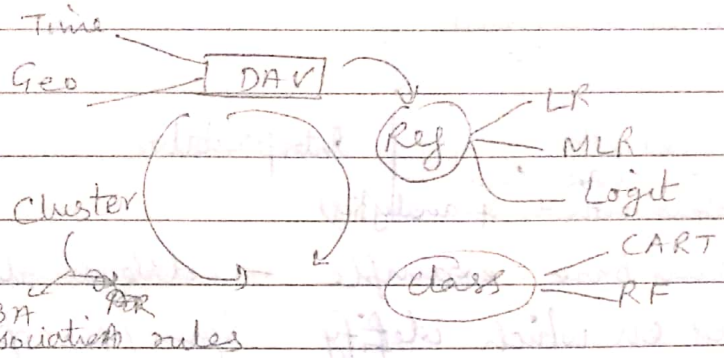
Classification - What is the probability of dependent var ~~being~~ being 0 or 1

Discrimination - Given that the dependent variable is 0 or 1, what are the significant characteristics in discriminating 2 groups.

annavinit@gmail.com 9980780995
vinit.t@greatlakes.edu.in

Web and Social media Analytics

- Text mining (Text as data) — Images, text, videos, audio
- Google Analytics
- Campaign management



Google to determine traffic congestion - analyses number of android phones on that road, ppl tweeting, sending messages on social media and much more. Analysis of all of this gives true picture of traffic.

20-75-5 rule for Corporate data
Numerical | Text data | Tacit (Intuition)

Problems with Text data
- Loose structure
- Poor spellings
- Non traditional grammer
- Multi lingual
- Ambiguity (I put my bag in car. It is big & blue. what? car or bag?
- Context - Homonyms, metaphors, Sarcasm

BLINK - by michael Gladwell - art of thinking without thinking

GLM allows binomial, binomial and multinomial also

P1	Extract the data clean it Get Text	Corpus
P2	→ Corpus - Stopwords - Stem document - sparse matrix	DTM/TDM
P3	DTM/TDM logistic, CART (use various methods of analytics)	Interpretation

Phase 3 Spam base example - data is already a DTM, based on which identify Spam / No spam mail.

As we have DTM, as data set

1) Perform logistic regression (In reality Text mining and logit does not work)

```
spamModelLR = glm(Spam~., data = spam, family =
                                    "binomial")
spamPred = predict(spamModelLR, data = spam,
                                    type = "response")
table(spam$Spam, spamPred > 0.5)
```

↳ will get confusion matrix

2) Built classification tree (for small data, it works)

```
SpamCART = rpart(Spam~., data = spam, method = "class")
prp(spamCART, extra = 2)
```

However in general industry use case there would by >3000 columns, so rpart is not a very good idea for classification. Best is RF.

3) Random forest

```
spamRF = randomforest(Spam~., data = spam).
varImpPlot(spamRF)
```

Packages in - text mining
Snowball &
wordcloud - visualization
Syuzhet - Automated Sentiment Discovery
(Sayushat)

So idea is to get Important variables from RF and use them in CART to fine tune the model further. Suppose there are 3000 columns in DTM, RF will pick $\sqrt{3000}$ columns each time (randomly) on a random sample (with replacement) and build a tree. Say it will build 200 such trees and find the top 20/30 variables which gives best Gini index (Imp variables). It means these are the imp variables which are causing various splits. Once we are sure of this causal effect, we can use these 20/30 variables to fine tune CART. (No point in using these selected variables and build RF again as it would further reduce imp vars to say 5)
However, we can build RF of 3000 vars, get 300 imp vars and again build RF to give 20/30 vars (In iteration)

Phase 2 — Text mining applications

- Web and Social media, Discovery of
  - Sentiments
  - Opinions
  - Emoticons
  - Topics
- Fraud and Irregularity detection
- Spam identification
- Business Intelligence
- Content enrichment

Phase 2 — Take clean set of data and create DTM for further text analysis

example - Apple tweets (Colums Tweet and avg rating)

Till 2015, apple used to extract say 1000 tweets and 5-6 linguists used to manually analyze the tweets and give rating from -5 to +5 ~~on a scale of~~ and then average of all 5/6 ratings were taken for a particular tweet. Now it is generated by Machines

To get pure character set of tweets

stringAsFactors = FALSE while importing csv

tweets $Negative = as.factor(tweets $ Avg =< -1)

table(tweets $ Negative)

As apple wants to analyze only -ve tweets Could have been classified as +ve, -ve and neutral Now avg column is not of any interest for us. Only -ve tweets along with tweets (Polarity)

Create corpus (a bag of words after extracting Text only from tweet to apply further techniques)

corpus = Corpus(VectorSource(tweets $ Tweets)

ex - corpus of automobile companies chairman speech or say Harry potter books review

wordcloud(corpus, colors = rainbow(7), max.words = 50)

(However better visualization is to use N/w diagram instead of wordcloud as it tells connections b/w words instead of plotting unigrams)

wordcloud just ~~repo~~ gives an idea of corpus. Converting corpus into Lower using "tolower" is optional. It is useful for unedited text where users can write in any form.

However, where ever data is fully edited, for ex Chairman speech where every capitalization/ lower case has a meaning or say TOI news,

- corpus = tm_map(corpus, tolower)
  similarly remove Punctuation, numbers, special characters etc.
- stopwords ("english")[1:10] (169 words in english)
- corpus = tm_map(corpus, removeWords, c("apple", "iphone"))

However, again in some cases stopwords are required like Healthcare industry
Another approach is, for my iphone use replace with my-iphone.

- Stem the document by finding root of words
  corpus = tm_map(corpus, stemDocument)
- Now create DTM
  frequencies = DocumentTermMatrix(corpus)
- Analyze frequencies
  inspect(frequencies[1000:1005, 505:515])
- check for sparsity
  findFreqTerms(frequencies, lowFreq = 20)

If DV is in column, go for DTM
If DV is in row, go for TDM and take a transpose of it

Sparcity - Say there are 1000 tweets, and if a word is not appearing in atleast 5 tweets, we will remove them. Will not consider those words which are appearing in < 5 tweets

- sparse = removeSparseTerms(frequencies, 0.995)
that means considers terms which are in atleast 5 terms out of 1000 (not in 995 terms)
- Now in sparse we have got 309 columns
So a Ground rule is after removing sparsity, columns should not be more than 400.
- tweetsSparse = as.data.frame(as.matrix(sparse))

Imp - If there are any Col names which starts with numbers or any special character which R does not approve, it would throw errar as we process tweetsSparse
So better is to convert them into R friendly names (function would append X_ <col name>)

colnames(tweetsSparse) = make.names(colnames(tweetsSparse))

- Add dependent variable again
tweetsSparse $Negative = tweets $ Negative

↳ Now use analytics models as step 1

Example of topic identification based on various feeds (Need to find out crude feeds)
— Its not as simple as finding crude word in the feed.
- There are other words / text which model learns to classify it as crude
An important function to find associated words after creating DTM/TDM is "findAssocs"

findAssocs (frequenciesRT, "Oil", 0.5)

DTM → word where association with others need to be found ← where correlation is more than 0.5

Output — $ Oil

barrel	Crude	price
0.55	0.52	0.51

↑ Corr. of Oil with barrel

↑ Corr. of Oil with Crude

↑ Corr. of Oil with Price

Phase 1 To get data from twitter and process it,

packages twitteR, RCurl, httr, syuzhet, tm, wordcloud

Get API Key, API secret Key, Access token and Access secret token

→ Extract data

Setup twitter oauth (authorization)

Once it is established using above 4 keys, data can be extracted

```
tweets = searchTwitter("CBI", n=200, lang="en")
tweets_df = twListToDF(tweets)
```

→ Clean data (Using gsub function)

gsub = global substitute

```
tweets_df $ text = gsub("&amp", "", tweets_df $ text)
```

similarly other strings like

"(RT|via)((?:\\b\\W*@\\w+))" } replace with ""

"http\\wt"

"[[:punct:]]"

Imp → tweets_df $ text ← iconv(tweets_df $ text, "UTF-8", "ASCII", sub="")

Convert everything which is UTF or ASCII so that R can handle.

Very Important - To get sentiment automatically (which earlier linguists were doing and then average reading was being taken)

```
Sent.value <- get_sentiment(tweets_df$text)
nrc_sent.value <- get_nrc_sentiment
                  (tweets_df$text)
```

- Sent.value gives sentiment score of each tweet (earlier we did manually on a scale of -5 to +5 5/6 linguists and then averaging out). It gives weighted average value say 0.3
- nrc_sent.value gives score of various human emotions (anger, frustation, trust, ---). It gives weighted avg scores of these individual emotions
- 2 more values possible with Syuzhet Package. (Not covered here)
- Diff b/w Sentiments and emotions

Sentiments = cascious value. If you are writing a tweet, you consciously think what to write, how to write (may be direct, may be indirect)

Emotions = Sub casscious. Your emotions comes from within in your comment. You purposefully dont write it

Syuzhet - How it works - Stanford proj

As there was no stability in human analysis of tweets (may depend on mood, fatigue etc) automation of sentiment score analysis was required. Stanford analyzed 4M documents

OCR Package in R to read images

Linguistically

Ran 1940 to 2003 → manually analysed → extracted sentiment score ⇒ built package which can now identify sentiments by analysing words in various context

- Now as we have got sentiment score, create corpus and go to Phase 2

corpus = Corpus(VectorSource(tweets.df$text))

Read Complete Documentation of Syuzhet Package

Banglore's Police CCTNS - crimes and criminals tracking Networking System developed by Wipro

Dont use words like CBI (otherwise any sentence where CBI appears would come)
Use # (hashtags for specific topics)
Use @ UserName for specific users

R (Packages) and Python (libraries) are not very suitable for text mining.
There are various other softwares/Tools available
- Wordij (Univ of california Product)

Wordij

- Suite of products, a combination of multiple softwares
- **Wordlink**
  - * Takes data and prepares DTM
  - * Also finds out biagrams
  - * Gives their frequencies in the text
  - * Generates 8 files everytime
- **VISIJ**
  - * takes care of visualizations
  - * Creates N/W diagram of word cloud
  - * Gives the frequency that which word is followed by which word and in what frequency
  - * Arrows will vary in thickness which visij directly tell which words are always followed by other like "Wells", "Fargo".
  - * Link would be visible which would tell Wells is followed by Fargo but Fargo is never followed by Wells (Directional nature)
- **OptiComm**
  - * Seed word and target words are used to find connection b/w 2 words which do not appear just next to each other (otherwise Visij would give * "info")

word1 — — — — — word2 (Any connection b/w 2 words)

some words b/w

- **BARNet**

Analyzes data in different time frames for ex, tweets of day 1, tweets of day 1+7, tweets on day 1+15 — — — .

and we want to find out the variation in tweets (in corpus) over a period of time, we can use DANet

- Z Utilities
  - Gives statistical inferences
  - Coefficients of correlation b/w 2 terms, or 2 N-grams etc
  - for ex, we used FindAssocs in R
  - Gives z statistics of any term
- Utilities
  - Proper Nouns
    - identify Proper Nouns so that you can convert it into ~~unigrams~~ strings (say Harry and Potter is converted into Harry Potter)
    - Now they can be used in Wordlink or any N/W diagram.
  - TimeSlice
    - Suppose out of complete text mining data, we want to analyze data b/w specific time (say Jan 1998 to Jan 1999), we can extract/filter it using TimeSlice. However, we need timeSeries data for it. (based on text collection timestamp)

Real Example using Tweet.txt

Wordlink - pick source file as tweet.txt
pick "droplist" from the documentation folder where you have kept "wordij" folder. It gives stop words for english language

default setting says "Drop words appearing less often than" 3 and "Drop pairs appearing less often than" 3

= Analyze Now

For huge data also, it takes few seconds

8 files are generated

Gives details about frequency of various words / pairs, their propotion in the dataset

VISij - generate N/W diagram by adding tweets.txt.net (in folder) file

Here you can play with # of nodes and changing node settings to picking nodes to analyze

☆ Analyse connections to word "not" or "dont", Here you can easily get idea that sentence why not apple iphone is being build up.

☆ Also it helps in doing customized stemming by finding out similar words

- Can build list of N/ws of different days
- N/w diagram for jumbled up paragraph would be identical.

Opticom - select wtg file and ptg file give seed and target words (ex. apple and freak) gives connection of these 2 words no matter whether they are n words apart

[0.217] apple → new → iphone → freak (28.3333)

Above all gives a picture whether to use any word as stop word (for deletion or not), whether we can combine words, whether we can utilize N/w and optscom info for stemming specific words etc.

Utilities - finds Proper Nouns
- Select tweets.txt input file
- mention an output file say trial.txt
- Output string Replace file - gives hyphenated file
- Wherever in b/w the sentence, it finds capital it considers it as "Proper Nouns"
- It does not consider first word of the sentence which by default is capital.

Word Tree Tool (IBM's)

www.jasondavies.com/wordtree

take any text (here copied complete text from tweets.txt) and paste in Paste Text box
→ Generate wordtree

Search "freak" in search box →
gives connection of all words with freak in one go
→ It was part of IBM Watson (now orphaned product)

stopped now. Team is disbanded. Part of the team started BigML.com

IBM Watson – main feature of it is "Ask a question". Not mainly a chatbot but if u suppose give a HR data, ~~it would~~ and if you ask "Why attrition is so high" it would do the regression in the background do some factor analysis and give you the reasons of attrition being high.

However chatbot is just an application of AI which gives predefined answers.

Big ML.com

A tool developed by IBM Watson team, very user friendly, can do all sort of supervised/unsupervised techniques (results match very closely to the manual versions of analysis)

BigML conference in Spain in Mar/Apr timeframe

1) Data is uploaded as "Sources"
2) cloud icon gives option to ready "1-click dataset" or various other options. 1-click dataset analyses Outliers, Null values, do date processing provides information (! sign) for non-recommended columns etc
3) Here you get "DataSource" which has ready data with distribution of various columns.
4) Choose "Model" for CART
5) Choose "Ensemble" for Random Forest
CART and Random Forest are proprietry and can be used by R only and not any other tool or language
6) Choose Model → Objective field (dependant variable) → create ML
Complete Model can be exported in tableau
7) back to dataset, choose "Logistic Regression" → Build model → Evaluate (It would give Confusion Matrix
8) For text mining, this tool creates only Word cloud

Google Analytics

Timeline feature of Google
→ ~~timeline.google.com~~
google.co.in/maps/timeline
gives all history of your travel

Google analytics certified – level 1
- Who are the people coming on your website
- What are they doing on your website
- Are they looking for specific product/page
- What kind of mobile / Service provider they are using
- Browser, its language they are using
- All information about a user coming on your website

analytics.google.com

googlemerchandisestore.com

get tracking ID ~~after~~ after registering on google analytics.

Now place this tracking ID on every page of your website so that google can track every action of website (put that in css).

Now google saves a small cookie on the machine and captures data from your machine.

This is how data for a specific website is accumulated over a period of time (using TID) and browser and other user

specific information is collected (using cookies)

Webserver Master – Means you are admin of your website

Real time

- Who are on my website right now
- From which part of the world they are coming
- What are they doing
- What browser they are using etc
- helpful is creating chat box where you can directly utilize this information

Audience

- Who are the people who have ever come to your website from date of inception till today
- Where they have come from, their gender & age (if declared by them), which country, city, day-time they visited
- Active users
- Lifetime value (if they have bought anything from you)
- Cohort analysis (what type of ppl coming in)
- Their Demographics
- Their interest levels (Google asks for specific interests)
- Geography details
- Behaviour (how much time spending on page, images or text they are interested in etc)
- Technology, Service Provider, M/c they use

Acquisition

- If users end up buying something, how did they end up,
- did they come from youtube, google, any orgaic search, referral, adword which they have put somewhere etc
- If website pay google $5 per click, it helps in determining where the traffic come from.

Behaviour

- How users interact with my system
- What is the flow, from which page to which page
- Any page showing 404 error or 505 error
- Is there any page which is becoming exit page (after users not liking it)
-

Conversions

- About money, how sales is happening
- Conversion rates

Bounce Rate – You spent money to bring ppl to your website (to acquire them), they came to website and without doing anything they disappered. (Youtube bounce rate is very low)

Analysis of google merchandising site (1st oct - 31st Dec 2017)

- shows user peak on Dec 12
- So we check user vs bounce rate, it shows high peak of users but high peak of bounce rate too
- That means customers are not attracted (engaged) they did not traverse through a single page.

10 types of customers/users

- Suspect - I dont Know them ~~bad~~, they dont Know them (sending mail to testers by B school for BA course)
- Prospect - I do something, and they came to me
- Customer -
- Lead
- opportunity
- Conversion

- No of users coming from India = 6.67%
- Out of which MH is highest buyer as state (17%) and Bengaluru is highest ~~buyer~~ user city (11.9%)
- However conversion (Revenue) is coming more from mumbai (though it is less in terms of total number of users)
- 1st 5000 users based on revenue, In tableau create scatter plot and correlation b/w revenue and session. Perform cluster analysis

In analysis tab → uncheck "Aggregate Measures" to create scatter plot of individual values.

- Which age group has higher % of new users
- Age gp of indians showing highest number of users. Its revenue? What can be said about this?
- Compare chinese users (~~not based on country but~~ those who have set up their browsers to Simplified chinese) on the basis of country. What revenue patterns can be seen?

in audience → geo → go to language
select zh-cn

zh-cn = 357.99
China = 298.07

Text mining

challenges

- Gender Identification - Big challenge in the field of ~~text~~ mining. Finding Proper Nouns from text (Ex Capitalize letters etc) But how to find out gender from names?

Limit for free 1000 identifications per day

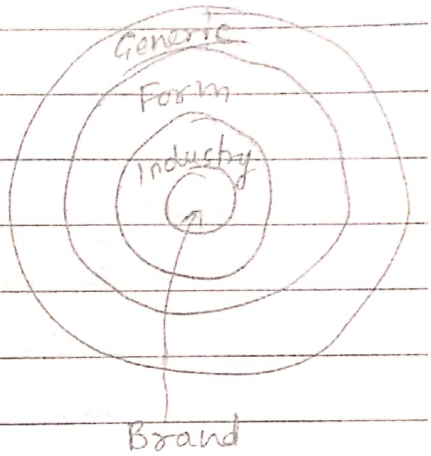
An useful API - genderize.io
R Package - GenderizeR (It ~~is~~ is an R Package which goes to above API only to get info)

Campaign Management

- Nature of competition - 4 types

Brand - Direct substitutes

ex of buying a car, people suggesting various cars based on their brand value (maruti, hundai, honda etc)



Industry - Same product or class of product.

ex - first hand maruti car vs Mercedes benz of 1964 (2nd hand) as a possibility so competition is not just b/w various brands of that product but within other class of products in industry

Form - Same end benefit

eg want to buy a car but looking for other options like bullet or harley davidson which is very different from basic need but solving the purpose.

Generic - Compete for same sales volume/money

McFarlan Grid (1984) - tells about relationship b/w IT strategy and business & operations

Factory	Strategic
Support	Turnaround

Support — applications that improve management & performance but are not critical to business. eg. Many companies open their account on google analytics, google supports but then they dont help business much (no focus on proper usage)

Factory — applications that are critical to sustaining existing business. For ex google taking care of your automated data managing instead of manual team

Strategic - appcations that are critical for future success. Eg. Placing strategic ad-words and attracting people

Turnaround - applications that may be future strategic decisions.

Strategic Alignment Model

Have the right ppl, bring the right job, getting the right reservation and using the data as currency

— Generating reports is ok but that to do, how to interpret those reports

Humans of internet

- Historians - they search only intended things
- Mavericks - they find something different (adventurous)
- Confused - they do not what they are searching but still keeps searching
- Gamblers - No specific Goal, just find what interests them.
- Time wasters

- Facebook data is also possible to download but is paid.
- So better to download your own facebook data in a JSON file and analyse using tableau (Facebook, settings, download my data)

Google adwords - Specific keywords which someone searches and your website is shown as "Ads" at the top of google. If someone clicks on that ad (URL of my website), I will pay google some $

If there are 2 or more websites with same adwords/search words, google auctions them. Whoever pays more price will get Top Ad slot.

Women clothing example

- Based on customer reviews, perform sentimental analysis
- Check if there is any relation b/w comments and rating i.e if somebody has really talked bad about the brand, is rating also low.
- -ve ratings are seen by more number of people and +ve ratings are seen by less number of people?
- Which products have got -ve ratings only. Which products have got mixed ratings (+ve, -ve, neutral)
-

Python for data science and Machine learning

Camlin Page
Date / /

Popular DS libraries

- NumPy
- SciPy
- Pandas
- Seaborn
- Scikit-learn
- Matplotlib
- Plotly
- PySpark

To convert notebook to .py → use nbconvert
or save as .py file

Anaconda Virtual environments [allows to set up virtual installation]

- allows to set up virtual installations of python and libraries on your computer
- Can have multiple versions of python or libraries
- These env can be easily activated/deactivated
- There is a "virtualenv" library for normal Python distribution
- Anaconda has built in virtual environment manager for easy process

NumPy

- NumPy is Linear Algebra library for Python,
- All libraries in PyData ecosystem rely on NumPy as one of the building blocks
- Incredibly fast as it has bindings to C libraries
- If you have anaconda, go to command prompt
  conda install numpy or
  pip install numpy

- Numpy arrays are the main way we will use Numpy throughout the course.
- Numpy arrays → Vectors, Matrices. — Two flavors of Numpy arrays — Both can be called arrays

Numpy Arrays

- are the main way we're going to be using the Numpy library
- Creating Numpy array from Python object like list

```
my_list = [1,2,3]
import numpy as np:
np.array(my_list)
array([1,2,3])
```

→ array as container for list object
→ Casting of normal python list into array

or

```
arr = np.array(my_list)
arr
array([1,2,3])
```

2 d list or matrix

→ list of list

```
my_mat = [[1,2,3], [4,5,6], [7,8,9]]
np.array(my_mat)
array([[1,2,3],
       [4,5,6],
       [7,8,9]])
```

Casting of list of lists to numpy array

★ However, we will be using numpy own built in generation methods to actually create arrays a lot faster and simpler

```
np.arange(0,10)
array([0,1,2,3,4,5,6,7,8,9])
np.arange(start, step, step)
```

```
from numpy.random import randint
randint(2,10)
3
```

Functions

To generate specific type of arrays

① np.zeros(3)

array([0., 0., 0.]) → rows → colums.

② np.zeros((5,5)) → tuple np.ones((5,5))

array([[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.]])

③ np.linspace(1,5,10) → Into 10 evenly spaces point

array([1, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5])

④ Identity matrix – useful matrix when dealing with linear algebra problems.

np.eye(2) → It is two dimensional square matrix

array([[1., 0.]
[0., 1.]])

→ diagonal of 1 and all 0's

→ Single digit as argu

⑤ Arrays of random numbers.

np.random.rand(5)

array([0.9225, 0.0715, 0.6829, 0.9956, 0.23256])

↳ creates array of random no. b/w 0 to 1

* If we want to return a sample / samples from a standard normal distribution / Gaussian instead of using rand, we can use randn

np.random.randn(2) → 1d

array([0.1748905, -0.917685])

↳ numbers from a stnd. normal dist.

np.random.randn(2,2) → 2d

np.random.randint(1,100,10) → gives 10 nos

start, end (not inclusive)

import randint

alternatively

array([80, 10, 12, 50, 60, 66, 70, 88, 13, 93])

attributes and methods of array

```
arr = np.arange(4)
arr
array([0,1,2,3])
arr.shape
(4,)
```
↳ 4, indicates it is 1d arr

```
arr.reshape(2,2) → reshape in 2d array
array([[0,1],
       [2,3]])
```

```
ranarr = np.random.randint(0,10,8)
ranarr
array([1,6,9,5,4,3,7,2])
```

ranarr.max() → returns max value
eg or np.max(arr)

ranarr.min() → returns min value

ranarr.argmax() → gives index loc^n of max value
2 .idmax()

Data type in array
arr.dtype
dtype('int32')

Numpy Indexing and Selection

```
import numpy as np:
arr = np.arange(0,11)
arr
array([0,1,2,3,4,5,6,7,8,9,10])
```
arr[8] gives 8

We can use normal python methods to get specific element in numpy array

arr[:6] gives array([0,1,2,3,4,5])

arr[5:] gives array([5,6,7,8,9,10])

Numpy array differ from normal Python list because of their ability to brodcast

```
arr[0:5]=100
arr
array([100,100,100,100,100,5,6,7,8,9,10])
```

slice_of_arr = arr[0:6]

```
slice_of_arr
array([0,1,2,3,4,5])
```

slice_of_arr[:] = 99 → assign 99 to all values.

```
slice_of_arr
array([99,99,99,99,99,99])
```

2 formats for grabbing elements from a 2d array Matrix

```
arr_2d = np.array([5,10,15],[20,25,30],[35,40,45])
arr_2d
array([[5,10,15],
       [20,25,30],
       [35,40,45]])
```

arr_2d[0][0] → gives 5 ★ This is double bracket notation

2 ways to access 30

① Double bracket notation

arr_2d[1][2]

② Comma Single bracket notation → recommened.

arr_2d[1,2]

To Get Sub matrix from matrix / 2d array slicing

Ex1

```
arr_2d[:2,1:]
array([[10,15],
       [25,30]])
```

Ex2

```
arr_2d[:2]
array([[5,10,15],
       [20,25,30]])
```

arr = np.arrange (1,11)

arr

arr ([1,2,3,4,5,6,7,8,9,10])

bool_arr = arr > 5 → comparison of array & single digit using boolean operator

bool_arr (Step1)

array ([False, False, False, False, False, True, True, True, True, True], dtype = bool)

boolean Masking or masking

Above can be used to actually do conditional selection

(Step 2)

arr [bool_arr]

array ([6,7,8,9,10]) → get results where this boolean array is true

or

Combination of Step 1 & 2

arr [arr > 5] also gives

array ([6,7,8,9,10])

Numpy Operations

Array with Array

Array with Scalars

Universal Array functions

① Array with Array

arr = np.arange (0,3)

arr

array ([0,1,2])

arr + arr — Addition of arr

array ([0,2,4])

arr - arr

arr * arr

Array with Scalar

→ Scalar mean just a single numbers and Numpy does is broadcast that number to every element in array

arr+100

array ([100, 102, 104])

works same with '-' & '*'

1/0 in phython gives error, However, in numpy it is does not give error but warning.

arr/arr ← notice 0/0 here gives null and warning in

array ([nan, 1., 1., 1., 1.,]) RuntimeWarning: invalid value encountered in true_divide.
(0/0)

Similarly, 1/arr gives

array ([inf, 1., 0.5, 0.33, ... 1])

IMP :-

Universal Array Fnc^n

① np.sqrt(arr).

② np.ext(arr)

np.max(arr) or ~~arr.max~~ arr.max()

np.sin(arr)

np.log(arr)

↳ for all universal functions.

docs.scipy.org/doc/numpy/reference/ufuns.html

Ques mat

array([[1,2,3,4,5],
[6,7,8,9,10],
[11,12,13,14,15],
[16,17,18,19,20],
[21,22,23,24,25]])

mat[1:3, 1:2] gives array([[2],
[7],
[12]])

Although removing 2 ~~will give~~ should also give same result but it gives array([2,7,12])

PANDAS

① Open source library
② Build on top of NumPy.
③ Allows for fast analysis, data cleaning and preparation.
④ It excels in performance and productivity.
⑤ It also has built-in visualization feature.
⑥ It can work with data from a wide variety of sources.
⑦ Panda installation
    Conda install pandas
    pip install pandas.

① Fast Analysis
② Data cleaning
③ Data Preparation

Series

① Data type
② Similar to numpy array.
③ Build on top of numpy array object.
④ Difference b/w numpy array & panda series is that can access labels meaning it can be indexed by label.

```
import numpy as np
import pandas as pd
labels = ['a', 'b', 'c']
my-data = [10,20,30]
ass = np.array (my-data)
d = {'a':10, 'b':20, 'c':30}
```

pd.Series (data = my-data) ← only passing 1 par data

```
0   10
1   20
2   30
```

} looks like numpy array except here we have an index 0,1,2 and actual data 10,20,30

* In panda series, we can actually specify what we want index to be

Ex. pd.series(data = my-data, index = labels)

	a	10	
label of index	b	20	→ label index series
	c	30	→ data points

pd.Series(my-data, labels)
↳ Since data and index are actually in order as parameter

Another way:

pd.Series(arr, labels)

a	10
b	20
c	30

pd.Series(d) →

a	10
b	20
c	30

**

Can hold variety of objects as its data point

pd.Series(data = labels)

0	a
1	b
2	c

pd.Series(data = [sum, len, print])

0	<built-in function sum>
1	< , , len>
2	< , , print>

dtype = object

ser1 = pd.Series([1,2,3,4], [USA, USSR, INDIA, SL])

USA	1	USA
USSR	2	0
INDIA	3	
SL	4	

dtype = INT64

ser1['USA'] = 1

Basic operations in series done based on index

Ser 2 = pd.Series([1,2,4,5], ['US', 'ITALY', 'USSR', 'NORWAY'])

Ser1 + Ser2 gives

US	2.0
ITALY	NaN
INDIA	NaN
USSR	6.0
NORWAY	NaN
SL	NaN

dtype: float64

IMP while performing operations with Panda series, integers are converted into floats in order to retain all possible informa

DataFrame

→ bld of dataframe using series object.

```
import numpy as np
import pandas as pd
from numpy.random import randn
np.random.seed(101)
```

→ to get the same random numbe

```
df = pd.DataFrame(randn(5,4), ['A','B','C','D','E'],
                  ['W','X','Y','Z'])
df
```

→ axis=1

	W	X	Y	Z
A				
B				
C				
D				
E				

axis=0 ← data frame

W, X, Y, Z → is panda's series

df → is just bunch of series that share an same index.

```
df['W'] or df.W
```

gives →

A	2.70
B	0.65
C	-2.01
D	0.18
E	0.19

→ returns series

Name: W, dtype: float64

pd.asmatrix(df) → would convert dataframe back to numpy 2D array.

```
type(df['w'])
pandas.core.series.Series
type(df)
pandas.core.frame.DataFrame
```

df[['w','x']] → returns dataframe with multiple columns.

df['new'] = df['w'] + df['y'] → Creates 'new' column

df.drop('new') → gives KeyError as it does not 'new' column in axis=0

df.drop('new', axis=1) → deletes 'new' column

→ This does not happen in place.

df.drop('new', axis=1, inplace=True)

→ actually remove 'new' column.

df.drop('E', axis=0) → drops row 'E'

Why rows are refered to as zero axis and column as 1 because it is directly taken from the shape.

Df are just fancy index markers on top of a numpy array.

```
df.shape
(5,4)
 ↓    → 1 index
0 index
```

0 index has rows
1 " " columns.

Selecting rows:- 2 ways to select rows in df

① df.loc['A']

```
W    2.7
X    0.62
Y    0.90
Z    0.50
Name: A, dtype: float64
```

→ returns Series.

Imp Con — Not only are columns series but rows are also series.

(2) df.iloc[2] → numerical based index.

Subset of row & column.

df.loc['B', 'Y']
-0.8542

df.loc[['A', 'B'], ['W', 'Y']] → gives subset of rows and columns.

CONDITIONAL SELECTION USING DATAFRAME

df > 0

	W	X	Y	Z
A	T	F	F	T
B	T	F	T	T
C	F	T	F	F
D	T	F	F	T
E	T	T	T	T

Step 1 booldf = df > 0

booldf

Step 2 df[booldf]

	W	X	Y	Z
A	2.7	0.6	0.9	0.5
B	0.6	NaN	NaN	0.6
C	NaN	0.74	0.52	NaN
D	0.188	NaN	NaN	0.9
E	0.19	1.9	2.6	0.68

Step 1 and Step 2 can be combined to get same result as above df[df > 0]

df['W'] > 0

A True
B True
C False
D True
E True

Name: W, dtype = bool

On data frame stacking command such as bracket notation on top of that such as return x column where that's true

df[df['W'] > 0]

	W	X	Y	Z
A	2.7	0.6	0.9	0.5
B	0.6	-0.31	-0.8	0.6
D	0.18	-0.75	-0.9	0.9
E	0.19	1.9	2.6	0.6

df[df['W'] > 0]['X']

A 0.6
B -0.3
D -0.7
E 1.9

Name: X, dtype: float64

boolser = df['W'] > 0
result = df[boolser]
mycols = ['Y', 'X']
result[mycols]

plt.pl.plot('scatter')

For 2 conditions, df[(df['W']>0) & (df['Y']>1)]

df.reset_index() → reset index to a column and you'll get the actual index then to be numerical.

	index	W	X	Y	Z
0	A	2.7			
1	B				
2	C				
3	D				
4	E				

newind = 'CA NY WY OR CO'.split()

newind

['CA', 'NY', 'WY', 'OR', 'CO']

df['States'] = newind. → adds new column.

	W	X	Y	Z	States
A					CA
B					NY
C					WY
D					OR
E					CO

If we want States column to be new index.

df.set_index('States'). we need to add in-place='TRUE' to retain these changes.

	W	X	Y	Z
States (new index)				
CA				
NY				
WY				
OR				
CO.				

Multi index and index hierarchy

```
import numpy as np
import pandas as pd
outside = ['G1', 'G1', 'G1', 'G2', 'G2', 'G2']
inside = [1, 2, 3, 1, 2, 3]
hier_index = list (zip (outside, inside))
hier_index = pd.MultiIndex.from_tuples (hier_index)
df = pd.dataframe (randn (6,2), hier_index, ['A', 'B'])
```

	Num	A	B
Groups			
G_1	1		
	2		
	3		
G_2	1		
	2		
	3		

```
df.loc ['G1']
```

	A	B
1		
2		
3		

```
df. loc ['G1']. loc[1]
A
B
Name.
```

To give names to index

```
df. index. names = ['Groups', 'Num']
```

Cross section

```
df.xs
```

To access G1, there are two ways -

```
df. loc ['G1'] or df.xs ('G1')
```

df.xs(1, level='Num')

	A	B
Groups		
G1	0.3	1.69
G2	0.16	.18

} gives row 1 of both G1 & G2

MISSING DATA

- Panda will fill missing value in your dataset with NaN value

```
import numpy as np
import pandas as pd
d = {'A': [1,2, np.nan]}, 'B': [5, np.nan, np.nan],
                          'C': [1,2,3]}
df = pd.DataFrame(d)
```

df →

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

df.dropna() → * drops complete rows with one or more missing values. In case you want to delete missing values from dataset.

* default axis=0, operation occurs along the row.

	A	B	C
0	1.0	5.0	1

df.dropna(axis=1) → drops column with null values

	C
0	1
1	2
2	3

df.dropna(thresh=2) → ~~It would not delete those drops rows~~ rows which ~~does not~~ have at least 2 NonNa values

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2

df.fillna(value = 'FILL VALUE')

	A	B	C
0	1	5	1
1	2	FILL VALUE	2
2	FILL VALUE	FILL VALUE	3

df['A'].fillna(value = df['A'].mean())

		o/p →
0	1.0	1
1	2.0	2
2	NaN	1.5

Name: A, dtype: float64

GroupBy

→ to tied a group rows of data together and call aggregate functions

→ Groupby allows you to group together rows based off of a column and perform an aggregate function on them

Comp Name EE

byComp = df.groupby('Company')

<pandas.core.groupby.DataFrameGroupBy object at 0x0007

byComp.mean() → gives mean sales
.std() →
.sum()

Comp	sales
FB	75
GOOG	56
MSFT	152

.sum().loc['FB'] or df.groupby('Company').sum().loc['FB']

Sales 593
Name: FB, dtype: int64.

Basic Data

	Company	Person	Sales
0	GOOG	Sam	200
1	GOOG	Charlie	120
2	MSFT	Amy	340
3	MSFT	Vanessa	124
4	FB	Carl	243
5	FB	Sarah	350

`df.groupby('Company').count()`

Company	Person	Sales
FB	2	2
GOOG	2	2
MSFT	2	2

`df.groupby('Company').max()`

Company	Person	Sales
FB	Sarah	350
GOOG	Sam	200
MSFT	Vanessa	340

NOTE :- It also returns the person because Python is able to store things in alphabetical order

`df.groupby('Company').min()`

Carl	243
Charlie	120
Amy	124

`df.groupby('Company').describe()`

Company		Sales
FB	count	
	mean	
	std	
	min	
	25%	
	50%	
	75%	
	max	

`df.groupby('Company').describe().transpose()`

Q1

Merging, Joining and Concatenating data frames

```
df1 = ( { 'A': ['A0', 'A1', 'A2'],
          'B': ['B0', 'B1', 'B2'],
          'C': ['C0', 'C1', 'C2'] },
          index = [0,1,2])

df2 = ( { 'A': ['A3', 'A4', A
          'B': ['B3', 'B4', B
          'C': ['C3', 'C4', C
          index = [3,4,5])

df3 = ( { 'A': ['A6', 'A7', 'A8'],
          'B': ['B6', 'B7', 'B8'],
          'C': ['C6', 'C7', 'C8'] },
          index = [6,7,8])
```

Concatenation - basically glues together dataframes.
Dimension should match along the axis you are concatenating on.

```
pd.concat ([df1, df2, df3])
```

By default concatenation is done on axis=0, however same can be done with axis=1.

```
pd.concat ([df1, df2, df3], axis=1)
```

Ex

left

	A	B	key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2
3	A3	B3	K3

Right

	C	D	key
0	C0	D0	K0
1	C1	D1	K1
2	C2	D2	K2
3	C3	D3	K3

Merging

Merge function allows you to merge Dataframes together.

pd.merge(left, right, how='inner', on='key')

	A	B	key	C	D
0	A0	B0	K0	C0	D0
1	A1	B1	K1	C1	D1
2	A2	B2	K2	C2	D2
3	A3	B3	K3	C3	D3

Merging, In case of multiple key column, innerjoin is based on combination of keys

pd.merge(left, right, on=['key1', 'key2'])

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2

left →

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

Right

	key1	key2	C	D
	K0	K0	C0	D0
	K1	K0	C1	D1
	K1	K0	C2	D2
	K2	K0	C3	D3

Outer Join pd.merge(left, right, how='outer', on=['key1','key2'])

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN
5	NaN	NaN	K2	K0	C3	D3

Right join

pd.merge(left, right, how='right', on=['key1', 'key2'])

A	B	key1	key2	C	D
A_0	B_0	K_0	K_0	C_0	D_0
A_2	B_2	K_1	K_0	C1	D1
A_2	A_2	K_1	K_0	C2	D2
NaN	NaN	K_2	K_0	C3	D3

left join

pd.merge(left, right, how='left', on=['key1', 'key2'])

A	B	key1	key2	C	D
A_0	B_0	K_0	K_0	C_0	D_0
A_1	B_1	K_0	K_1	NaN	NaN
A_2	B_2	K_1	K_0	C_1	D_1
A_2	B_2	K_1	K_0	C_2	D_2
A_3	B_3	K_2	K_1	NaN	NaN

JOINING – method for combining the columns of two potentially differently-indexed dataframes into single result DataFrame.

left

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

right

	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

left.join(right) → by default it is inner join

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2

left.join(right, how='outer')

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

Operations.

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz.

① `df['col2'].unique()`

→ `array([444,555,666], dtype=int64)`

② `df['col2'].nunique()`

→ 3

③ `df['col2'].value_counts()`

444	2
555	1
666	1

Name: Col2, dtype: int64.

④ Selecting data

⑤ Applied method

```
def times2(x):
    return x*2
```

`df['col1'].apply(times2)`

0	2
1	4
2	6
3	8

Name: col1, dtype: int64

Built-in functions can also be applied.

`df['col3'].apply(len)`

0	3
1	3
2	3
3	3

Apply is powerful when you combine it with lambda expression as we do not need to define entire function.

`df['col3'].apply(lambda x: x*2)`

0	888
1	HHO
2	1332
3	888

Name: col2, dtype: int64

To drop columns

```
df.drop('col1', axis=1)
```

	Col1	Col2	Col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

→

	Col1	Col2
0	1	444
1	2	555
2	3	666
3	4	444

df.columns → returns index object which is list of columns.

Index (['col1', 'col2', 'col3'], dtype='object')

★★ This is going to be useful when you are trying to index a column and may be the spacing is weird or you can't figure out what the string is.

df.index → gives info of the index.
since it is range index it actually just reports back the start, stop and step size

RangeIndex (start=0, stop=4, step=1)

Sorting & Ordering data frame

df.sort_values('col2') or df.sort_values(by='col2')

	Col1	Col2	Col3
0	1	444	abc
3	4	444	xyz
1	2	555	def
2	3	666	ghi

df.isnull() → gives no values in your data frame

	Col1	Col2	Col3
0	False	False	False
1	False	F	"
2	F	F	"
3	F	F	"

Pivot table

df

	A	B	C	D
0	foo	one	x	1
1	foo	one	y	3
2	foo	two	x	2
3	bar	two	y	5
4	bar	one	x	4
5	bar	one	y	1

df.pivot_table(values='D', index=['A','B'], columns=['C'])

		C	x	y
A	B			
bar	one		4.0	1.0
	two		NaN	5.
foo	one		1.0	3.0
	two		2.0	NaN

Data Input and Output

Data Sources

① Csv → conda install BeautifulSoup4 — html & sql

② Excel → conda install lxml

③ HTML → conda install html5lib

④ SQL → conda install sqlalchemy

* Below command assumes files are at same loca" where jupyter notebook is installed

Read / write csv files

pd.read_csv('Example.csv')

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

df = pd.read_csv('example.csv')

df.to_csv('My_output')

pd.read_csv('My_output') →

	Unnamed:0	a	b	c	d
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15

~~pd.read_csv('My_output', index=False)~~

pd.to_csv('My_output', index=False)

pd.read_csv('My_output') →

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

R/W Excel files

- Pandas can only import the data.
- It can't import formulas, images or things like macros. It will crash pandas
- → You might have to install
  conda install xlrd
- → pd.read_excel('Excel_Sample.xlsx', sheet_name='Sheet1')
- → df.to_excel('Excel_Sample.xlsx', sheet_name='New_She

R/W HTML files

data = pd.read_html('https://www.fdic.gov/...../xxx.html')
type(data)
list

↓ Panda find every table element that is in this html fi

~~data[0]~~

* Pandas read_html function will read tables off of a webpage and return a list of DataFrame object

data[0]

Bank Name City ST CERT ..... Update Date

data[0].head() → gives top 5 rows

R/W SQL files

→ pandas.io.sql provides collection of query wrappers to both facilitate data retrieval and to reduce dependency on DB-specific API.

→ Database abstraction is provided by SQLAlchemy

→ psycopg2 - for PostgreSQL } driver library for
  pymysql for MySQL } database

→ SQLite - Included in Python's standard library by default.

Sal. JobTitle [sal ['EmployeeName'] == 'Joseph Driscoll']

→ We can find an overview of supported drivers for each SQL dialect in the SQLAlchemy docs.
→ If SQLAlchemy is not installed, a fallback is only provided for sqlite (and for ~~myy~~ mysql for backwards compatibility, but this is deprecated and will be removed in a future version). This mode requires a Python database adapter which respect the Python DB-API.

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:')
df.to_sql('my-table', engine)
sqldf = pd.read_sql('my-table', con=engine)
sqldf
```

	index	a	b	c	d
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15

df.info() → gives number of entries against individual columns.

* Employee with Job title 'JOSEPH DRISCOLL'

```
Sal [sal['EmployeeName'] == 'JOSEPH DRISCOLL']['JOBTITLE']
```

(3) (2) (1) condition

(2) This gives ~~that~~ specific dataframe i employee name Joseph Driscoll
(3) Adding this gives the job title.

* Person with highest salary

```
Sal[sal['TotalPayBenefits'] == sal['TotalPayBenefits'].max()]
                                                ['EmployeeName']
```

Or

```
Sal.loc[sal['TotalPayBenefits'].idxmax()]
```

Matplot Lib

Matplotlib.org → Gallery

- Most popular plotting library
- gives control over all aspects of figure
- designed to have similar feel to matlab's graphical plotting
- Seaborn is built over MatPlot Lib

```
!pip install matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

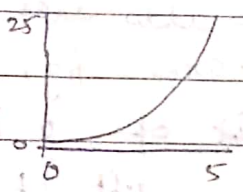
→ To allows to see visualizations in jupyter notebook

2 ways of plot creation → functional & object oriented

```
x = np.linspace(0, 5, 11)
y = x**2
```

Functional

```
① plt.plot(x,y)
② plt.show()
```



mainly useful if you are not plotting in jupyter

① would give plot as output in all

② would print the plot as "kind of markdown"

```
plt.plot(x, y, 'r-')
```

Colour, line style

↖ adds colour (Red)

```
plt.xlabel('X Label')
plt.ylabel('Y label')
plt.title('Title')
```

Multiplot on same canvas

```
plt.subplot(1,2,1)
```

← 1st plot of the row

← Plot num you are referring to

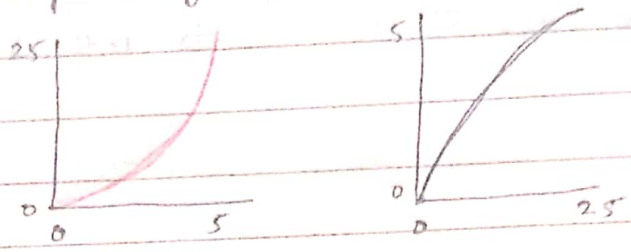
# of rows

# of columns

plt.imshow(mat, cmap = 'Colormap')

red

plt.plot(x, y, 'r')
plt.subplot(1, 2, 2) ← 2nd Plot of the row
plt.plot(y, x, 'b') ← blue

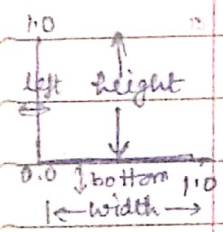


Object oriented API method ↑ better way

means we are going to instatiate figure objects and then call methods or attributes from that object.

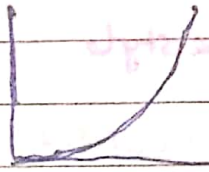
fig = plt.figure() ← creates figure object blank at this moment, can add attributes now.

axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
↑ list of 4 inputs
left, bottom, width, height (ranges from 0 to 1)
Basically the % of canvas you want to take



axes.plot(x, y)

x = np.linspace(0, 5, 11)
y = np.sqrt(x)

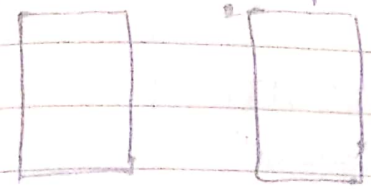


axes.set_xlabel('X Label')
axes.set_ylabel('Y Label')
axes.set_title('Set Title')

fig = plt.figure()
axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3])

Subplots → tuple packing.

```
fig, axes = plt.subplots(nrows=1, ncols=2)
```



fig, axes are defined in previous page.

* plt.tight_layout() → fixes overlap in layout
* axes → is a list of axes object meaning you can iterate through it. ↳ axes1 axes2

```
for current_ax in axes:
    current_ax.plot(x,y)
```

plotting by iterating.

Since we can iterate thru axes, we can also index it.

```
axes[0].plot(x,y)
axes[1].plot(y,x)

axes[0].set_title('First Plot')
axes[1].set_title('second Plot')
```

plotting by indexing.

Figure Size, Aspect Ratio and DPI → dots per inch

```
fig = plt.figure(figsize=(3,2), dpi=100)
ax = fig.add_axes([0,0,1,1])
ax.plot(x,y)
```

figsize → inches

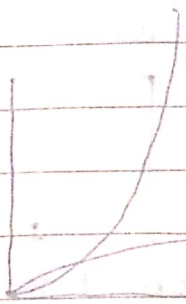
For subplots,

```
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(3,2))
axes[0].plot(x,y)
axes[1].plot(y,x)
```

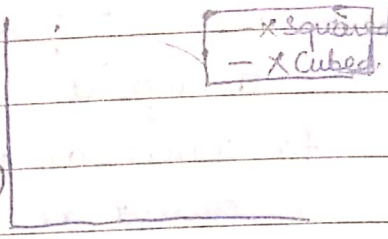
* Matplotlib can be used to generate high quality outputs in a number of format png, jpeg, svg, Tgf, pdf

fig.savefig('my-picture.png')
.jpg

# fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(x, y, label='x squared')
ax.plot(y, x, label='x cubed')



ax.legend()
or
ax.legend(loc=0)
ax.legend((loc=0.1, 0.1))



Plot Appearance

ax.plot(x, y, color='green')
='#FF8C00')

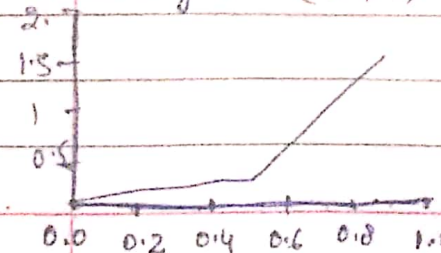
ax.plot(x, y, color='purple', linewidth=2)

ax.plot(x, y, color='green', linewidth=2, alpha=0.5)
or lw
↓ transparency of line

linestyle='--'
or ls ='-.'
=':'
='steps'

markeredgewidth=3, marker='o', markersize=1 (10, 20), markerfacecolor='yellow'
markeredgecolor='green', ='+'
='*'
='1'

ax.set_xlim([0,1]) → sets lower and upper bound of x axis
ax.set_ylim([0,2])



Special Plot Types

- Boxplots, histograms, scatter plots can also be created.
- Most of these plots will actually be created using Seaborn, a statistical plotting library for Python

IMP - write tips plotting in jupyter notebook.

- ~~fig should~~ All commands for plotting fig should go in same cell.

SEABORN → very nice visualization library

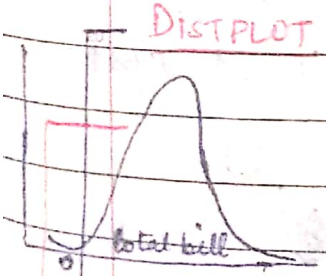
- statistical plotting library and built on top of matplotlib
- has beautiful default styles
- designed to work very well with pandas dataframe objects.

Distribution Plots → Diff. plot types with seaborn that allow us to visualize the distribution of a data set

```
import seaborn as sns
% matplotlib inline
tips = sns.load_dataset('tips')
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	F	No	SUN	Dinner	2
1	10.34	1.61	M	No	SUN	"	3
2	21.01	3.50	M	No	SUN	"	3
3	23.68	3.31	M	No	SUN	"	2
4	24.59	3.61	F	No	SUN	"	4

DISTPLOT




```
sns.distplot(tips['total_bill'], kde=False)
```

→ Distplot allow us to show the distribution of a univariate set of observations

→ kde → Kernel density estimation

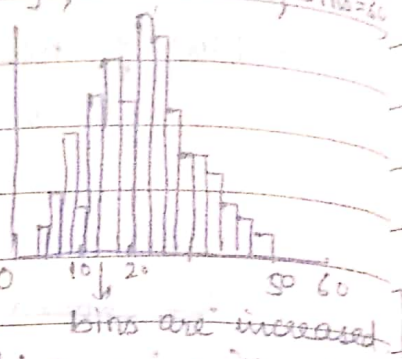
1. This plot is without kde.

2. kde = False

→ Most of total_bills are b/w 10$ & 20$

To get little more information

sns.distplot(tips["total_bills"], kde=False, bins=40)



bins are increased

If we keep bins=100, i.e. too high, we will get weird scenario where we are beginning to plot every instance of total bill for every single price point.

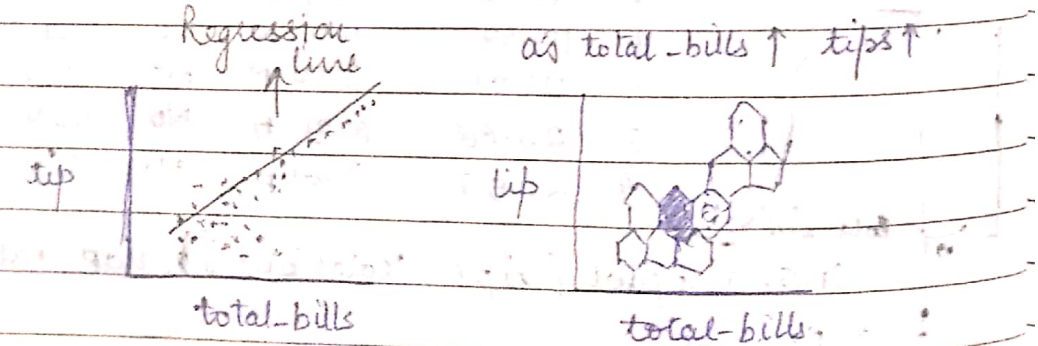
Joint plot – allows to match up 2 dist plots for bivariate data.

sns.jointplot meaning we can combine two different distribution plots

kind parameter - allow us to choose how we actually want to compare these two distributions

To compare distribution of total_bill Vs tip size

sns.jointplot(x='total_bill', y='tip', data=tips)



Joint plot gives additional argument para 'kind'

Kind – Allows you to affect what's actually going on inside of this joint plot. By default it is scatter

sns.jointplot(x='total_bill', y='tip', data=tips, kind='hex')

helps you make hexagon distribution representation

* If hexagon has a certain no. of points it gets darker.
" " " less " " " " lighter

If kind = 'reg', which stands for regression
In this case Seaborn, is actually going to draw a regression line on it.

If kind = 'kde' — allows to have 2d 'kde' which shows you the density where points match up most.
→ Joint plot is usually used with scatter, as it is easiest to read and gives quite information.

Pair plot — Plots pairwise relationship across an entire data frame at least for the numerical columns.
- pair plot creates joint plot for every single combination of the numerical columns in this data frame

sns.pairplot(tips)

→ larger the dataframe, longer the pairplot takes

→ size | size | size
total_bill | tip

- Nice way to quickly visualize data.

sns.pairplot(tips, hue='sex') → always categorical

'hue' arguements color data points based on column you put in as hue. Green points are female and blue points are male

sns.pairplot(tips, hue='sex', palette='coolwarm')

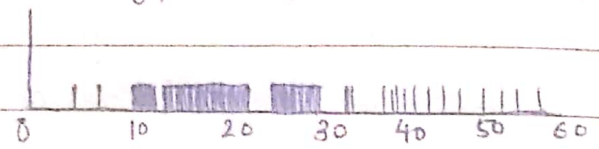
Allows to actually color this with some specific color palette.

Rug Plots

```
sns.rugplot(tips['total_bill'])
```

- It just draws a dash mark for every points on this uniform or univariant distribution.
- Rug plot are used to explain kde plot using coding.

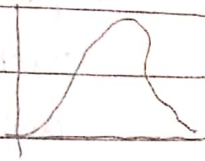
```
sns.rugplot(tips['total_bill'])
```



- Diff. b/w histogram and rug plot is that the histogram essentially has bins and it counts how many dashes were in that bin and then shows it as a number up here.

To kdeplot.

```
sns.kdeplot(tips['total_bills'])
```



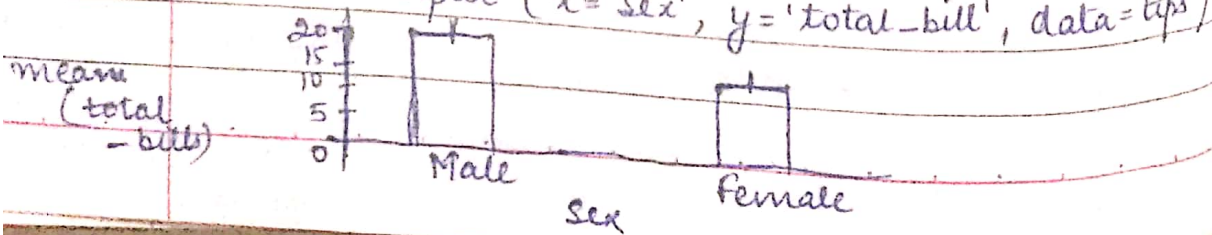
Categorical Plot using Seaborn

- For categorical plots, we will see distribution of a categorical column with numerical columns or another categorical columns.

Bar plot

- General plot that allows to aggregate the categorical data based of some function (default mean())

```
sns.barplot(x='sex', y='total_bill', data=tips)
```



- Default value of estimator object is mean, however we can put in our own aggregate functions.

```
import numpy as np
sns.barplot(x='sex', y='total_bill', data=tips,
            estimator=np.std)
```

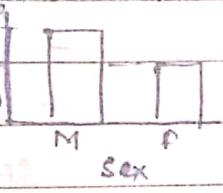
- Estimator - is a function that is aggregate func

Count Plot

- same as bar plot except the estimator is explicitly counting the number of occurences.

```
sns.countplot(x='sex', data=tips)
```

Wherein in bar plot estimator is mean.



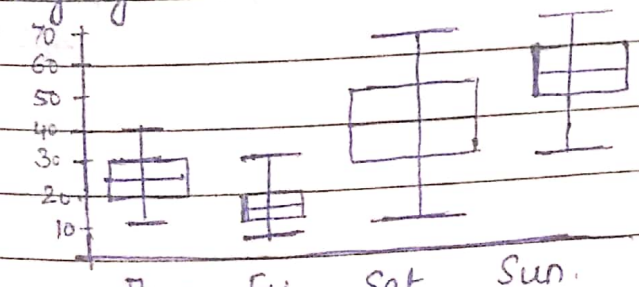
Box Plot and Violin Plot

- Used to show distribution of categorical data
- Box Plot is sometimes known as Box and Whisker plot
- Shows distribution of quantitative data in a way that hopefully facilitates comparison b/w variables.

```
sns.boxplot(x='day', y='total_bills', data=tips
```

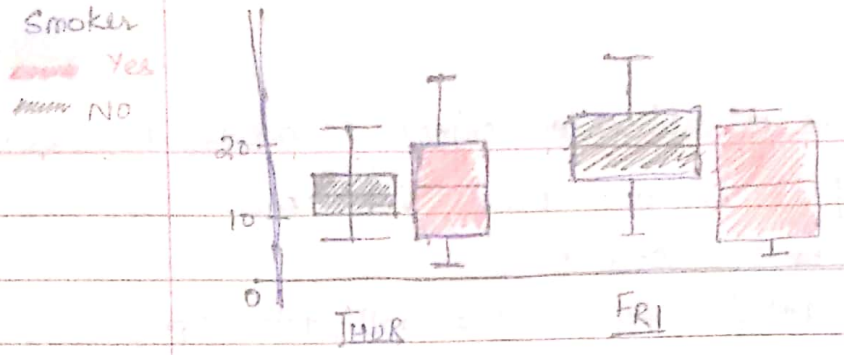
trying to see distribution of total_bills per day.

palette='rainbow'




```
sns.boxplot(x='day', y='total_bills', data=tips, hue='smoker')
```

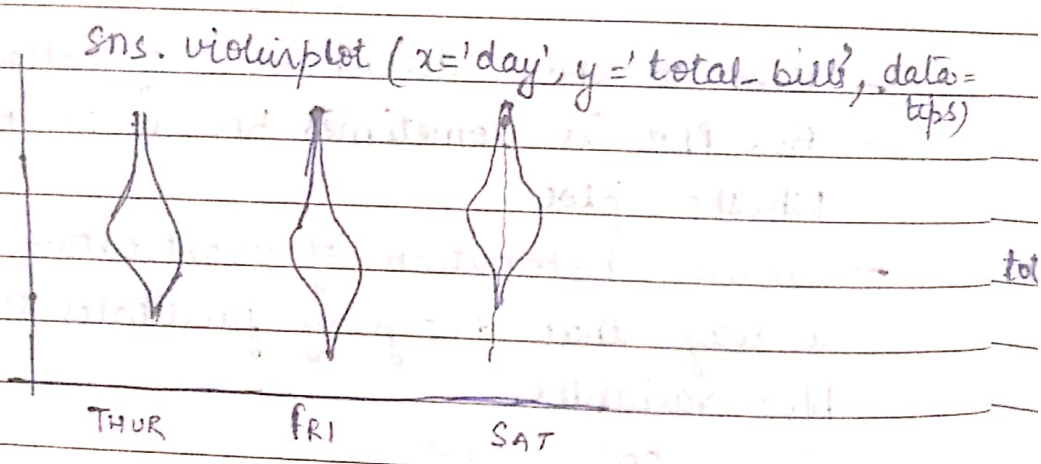
By adding another categorical column, we can split up these box plots even further.



★ IMP → Above shows day by day behavior for total bill but also whether or not this person is a smoker.

VIOLIN PLOT

- Similar to Box plot
- Going to show the distribution of data across some category

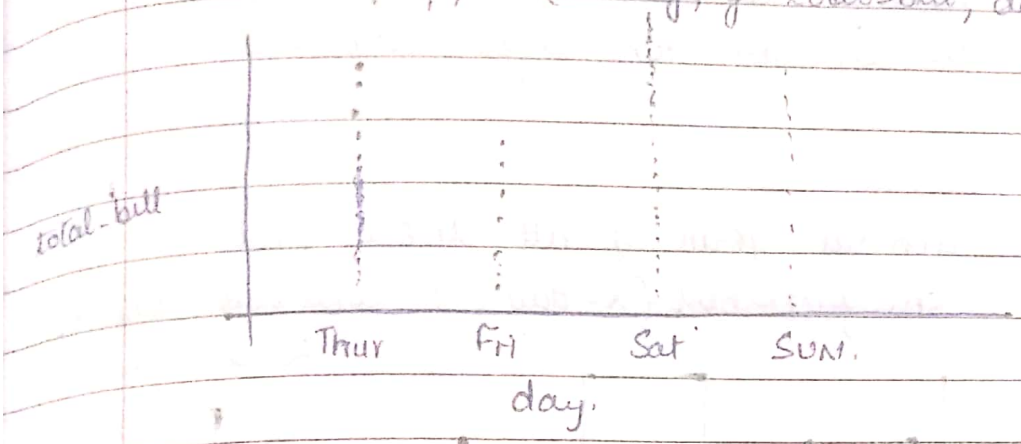


sns.violinplot(x='day', y='total_bill', data=tips, hue='sex', split=TRUE)

STRIP PLOT

- Draws scatterplot where 1 variable is categorical

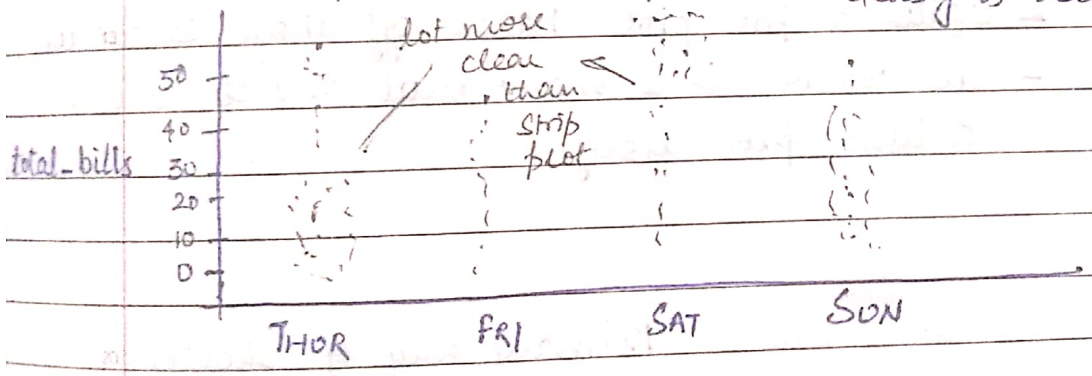
sns.stripplot(x='day', y='total_bill', data=tips)



- Issue with Strip plot – can't tell how many points are stacked on top of each other

sns.stripplot(x='day', y='total_bill', data=tips, jitter=TRUE)

JITTER – Add a little bit of random noise to separate some of these stacked points as density is better

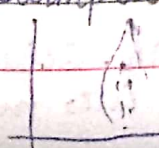


sns.stripplot(x='day', y='total_bill', data=tips, jitter=True, hue='sex', split=True)

SWARM PLOT – COMB^N OF STRIP AND VIOLIN PLOT

- Similar to strip plot except points are adjusted so that they don't overlap. This gives better representation of values.

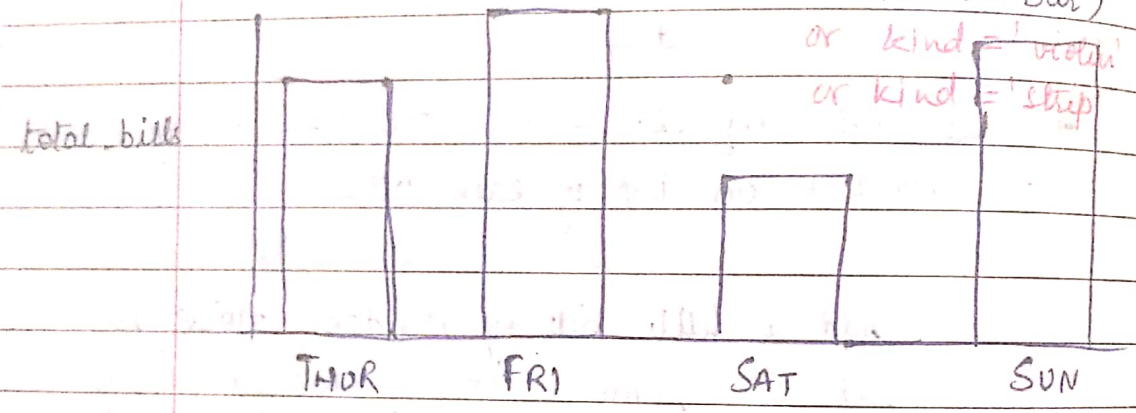
sns.swarmplot(x='day', y='total_bill', data=tips)



Disad
- Sometimes they don't actually scale to very large numbers both in terms of the ability to show all the points.
- Not advisable for large data sets.

Factor Plot
- General form of all these plots.

sns.factorplot(x='day', y='total_bill', data=tips, kind='bar')
or kind='violin'
or kind='strip'



- Creates a bar plot based off these categories
- However, it is individual choice to call actual plot itself.

Matrix Plots
Heat Maps – Primary way of showing matrix plot

Condition – Data should be in matrix form

Matrix form – Index name and column name match up so that the cell value actually indicates something that is relevant to both of those names. i.e. I need to have both variables on the columns and the rows. This can be done either thru a pivot table or correlation data as row is not an actual variable so, we need to get it into matrix form.

creating matrix tc=tips.corr()

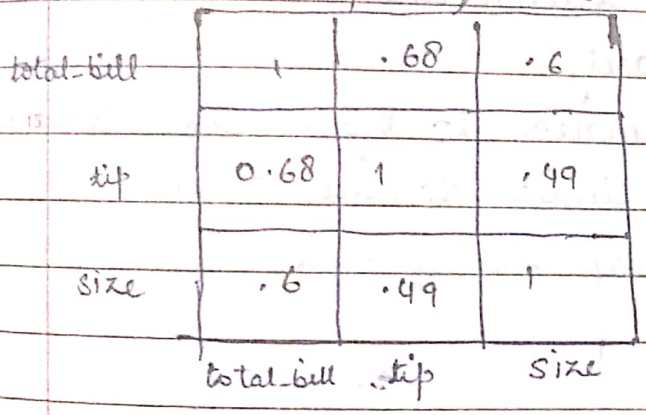
	total_bill	tip	size
total_bill	1.000	.67	.59
tip	0.67	1.0	.489
size	0.59	.48	1.00.

matrix form of tips dataset

flight.head()

	year	month	passengers.
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr.	129.

sns.heatmap(tc, annot=True, cmap='coolwarm')



matrix form of flights data

fp = flights.pivot_table(index='month', columns='year', values='passenger')

year	1949	1950	1951	1952	-	-	-	-	1960.
month									
Jan	112	115	145						
Feb									
Mar									
Apr									
May	121	125	172						

Above, if we see no's are correlated row. Year Vs Month

Heat Map can be used now to quickly visualize the data

sns.heatmap(fp)

* Looking at heat map we can easily tell which month of which year has most flights.

sns.heatmap(fp, cmap='magma', linecolor='white', linewidth=1)

CLUSTER MAP (SECOND MATRIX TYPE PLOT)

- Cluster map uses hierarchal clustering to produce a clustered version of heat map.

sns.clustermap(fp)

- This ~~key~~ cluster column and rows together based off their similarity.
- Clusters of information to try to show columns and rows that are similar to each other.

sns.clustermap(fp, cmap='coolwarm', standard_scale=1)

GRIDS

- How to use sideboards grid's ~~ability~~ Capability to automate subplots based of features of our data

```
import seaborn as sns
% matplotlib inline
iris = sns.load_dataset('iris')
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	"
2	4.7	3.2	1.3	0.2	"
3	4.6	3.1	1.5	0.2	"
4	5.0	3.6	1.4	0.2	"

Simplified version of pair grid
→ Does lot of stuff automatically

PAIR PLOT — sns.pairplot(iris)

PAIR GRID → We have to do little more but gives lot more control.

sns.PairGrid(Iris) — It takes all numerical colums and grids them up.

g = sns.PairGrid(iris) → creates empty grid.

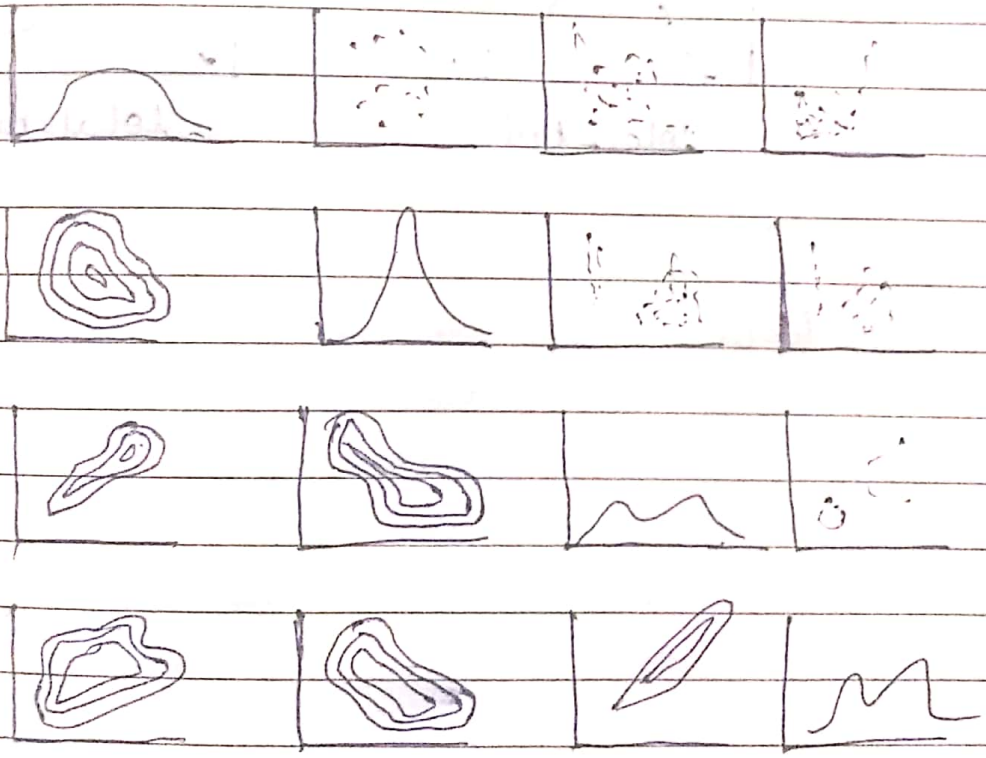
g.map(plt.scatter) → map func^n maps scatter func^n of plt library to ~~complete dataset~~ grid obj.

Since, pair grid gives more control, you can plot as below,

g.map_diag(sns.distplot)

g.map_upper(plt.scatter)

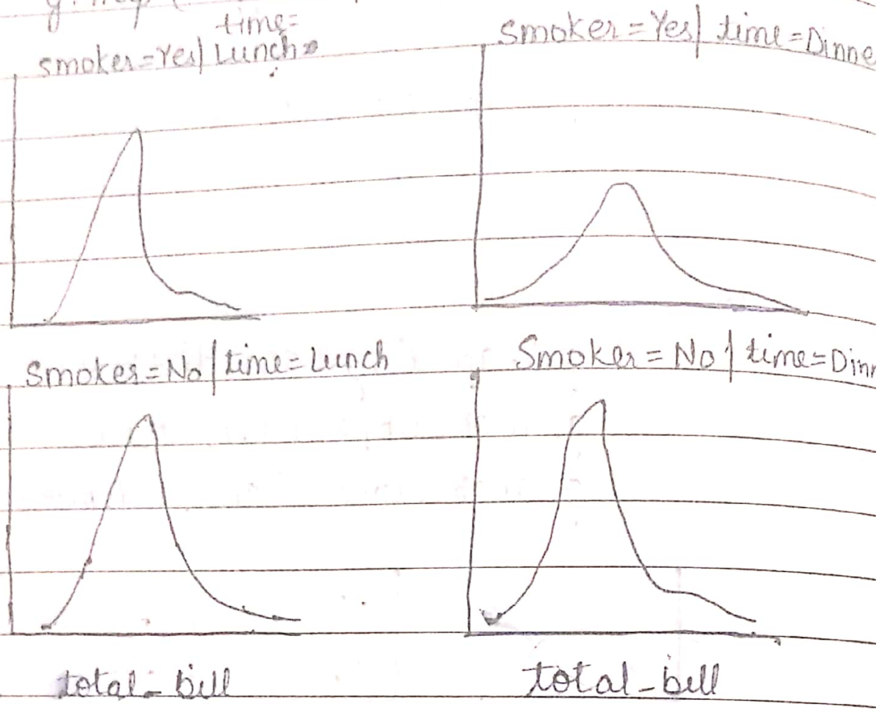
g.map_lower(sns.kdeplot)



FACET GRID

g = sns.FacetGrid(data=tips, col='time', row='smoker

g.map(sns.distplot, 'total_bill')

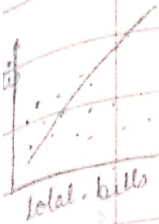


Rug

REGRESSION PLOTS #SIDDAY

#1 LMPLOT

Allows to display linear models with Seaborn.




```
sns.lmplot(x='total_bill', y='tip', data=tips)
```

hue='sex', ← separation based on categorical feature

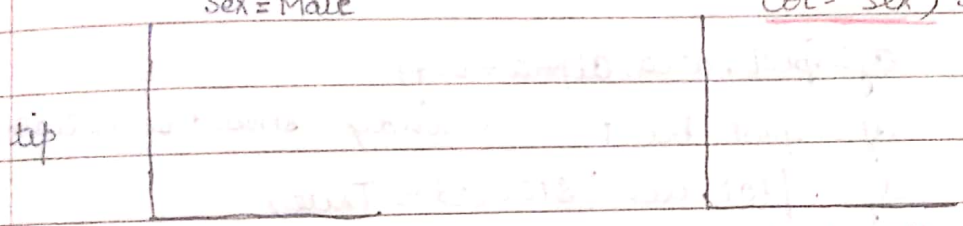
```
markers=['o','v'],
scatter_kws={'s':100})
```

↓ size of marker

Same can be done using grid instead of separating by Hue:

```
sns.lmplot(x='total_bill', y='tip', data=tips,
```

col='sex') Sex=Male | Sex=Female




```
sns.lmplot(x='total_bill', y='tip', data=tips, col='day', hue='sex',
           aspect=0.6, size=8)
```

Style and Color

```
sns.set_style('whitegrid')
sns.countplot(x='sex', data=tips)
sns.despine(left=True, bottom=False)
```

Data

Pandas In-built Visualization

To get a histogram of all the values in A column of DF1

```
df1['A'].hist(bins=30)
import seaborn as sns
import numpy as np
import pandas as pd
% matplotlib inline
df1 = pd.read_csv('df1', index_col=0)
df1['A'].plot(kind='hist', bins=30)
df1['A'].plot.hist()

df2.plot.area(alpha=0.4)
df2.plot.bar() → index should be categorical
df2.plot.bar(stacked=True)
df1.plot.line(x=df1.index, y='B', figsize=(12,3), lw=1)
```

	A	B	C	D.
2000-01-01				
2000-01-02				
2000-01-03				
2000-01-04				

```
df1.plot.scatter(x='A', y='B')
```

We can use style sheet using following commands.

```
plt.style.use()
```

- plot - bmh
- plot - fivethirtyeight
- plot - ggplot and many more

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
df['a'].plot.hist(---) as usual
```

ISHI

AANYA

To select specific number of rows.
df3.ix[0:30].plot.area(alpha=0.4)

PLOTLY AND CUFFLINKS

→ for more details visit github
— for financial analysis

PLOTLY

- Interactive visualization library.
- Cufflinks connects plotly with pandas.
  ↳ It is productivity tool that allows you to quickly call plot's just as you would with pandas for instance a data frame dot plot method. Instead you would actually create an interactive visualization using plotly.

```
import pandas as pd
import numpy as np
from plotly import __version__
print(__version__)
```

↳ 1.12.9

```
import cufflinks as cf
from plotly.offline import download_plotlyjs, iplot,
                           init_notebook_mode, plot,
init_notebook_mode(connected = True) → connects
                                       javascript to notebook.
```

as plotly essentially connects pandas and python to it interactive javascript library. And this is going to allow your notebooks to access these visualizations.

```
cf.go_offline()
df = pd.DataFrame(np.random.randn(100,4),
                  columns = 'ABCD'.split())
df.head()
```

Using dict, for creating another dataframe

```
df2 = pd.DataFrame({'Category': ['A', 'B', 'C'],
                    'Values': [32, 43, 50]})
```

df2 →

	Category	Values
0	A	32
1	B	43
2	C	50

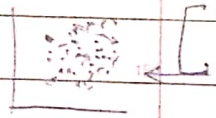
	A	B	C	D
0	.83	2.4	-2.4	.43
1	-.28	-0.7	.26	1.27
2	-1.2	1.86	.99	.50
3	-0.4	-0.34	.099	-1.7
4	0.001	1.41	-0.20	-1.5

df.plot() → pandas will by defaut take matplotlib

df.iplot() → matplot is converted to plotly interactive, Zoom in, save & download the plot as png, filter data.

How to use iplot to create other plots:-

1. df.iplot(kind='scatter', x='A', y='B') → By default it will join points with lines



df.iplot(, mode='markers', size=20)

Bar: df2.iplot(kind='bar', x='Category', y='values')

df.count().iplot(kind='bar')

df.sum().iplot(kind='bar')

Box: df.iplot(kind='box')

3D Surface plot

```
df3 = pd.DataFrame({'x': [1,2,3,4,5],
                    'y': [10,20,30,20,10],
                    'z': [500,400,300,200,100]})
df3.iplot(kind='surface', colorscale='rdylbu')
```

Histogram: df['A'].iplot(kind='hist', bins=25)

df.iplot(kind='hist', bins=25)

Spread → for stock data

df[['A','B']].iplot(kind='spread')

↳ used in world GDP comparison to population, happiness factor, United Nations Report

Bubble Plot – similar to scatterplot except that it will change size of the points based off another variable

```
df.iplot(kind='bubble', x='A', y='B', size='C')
```

Scatter matrix plot → similar to seaborne pair plot

→ it just creates a scatter matrix of all columns it can

```
df.scatter_matrix() → IMP - All columns should be numerical to make this work
```

* There are some technical analysis capabilities with cufflinks that are built in —(under beta)

Geographical Plotting

- challenging due to various formats data can come in
- We will use plotly for plotting
- Matplotlib also has a basemap extension.

Choropleth Maps

- This allows to plot out information either on a global scale or any nation wide scale.

```
import plotly.graph_objs as go
import plotly, plotly.plotly as py
from plotly.offline import download_plotlyjs, iplot
                           init_notebook_mode, plot,
init_notebook_mode(connected=True) → this initializes notebook mode
data = dict(type='choropleth', → type of geographical plot
            locations=['AZ','CA','NY'],
            locationmode='USA-states',
            colorscale='Portland',
            text=['text1','text2','text3'],
            z=[1.0,2.0,3.0],
            colorbar={'title':'Colorbar Title Goes Here'})
```

data variable

layout variable

```
layout = dict(geo={'scope':'usa'})
choromap = go.Figure(data=[data], layout=layout)
iplot(choromap)
```


```
import pandas as pd
df = pd.read_csv('2011_US_AGRI_EXPORTS')
```

https://plotly/python/reference/#choropleth

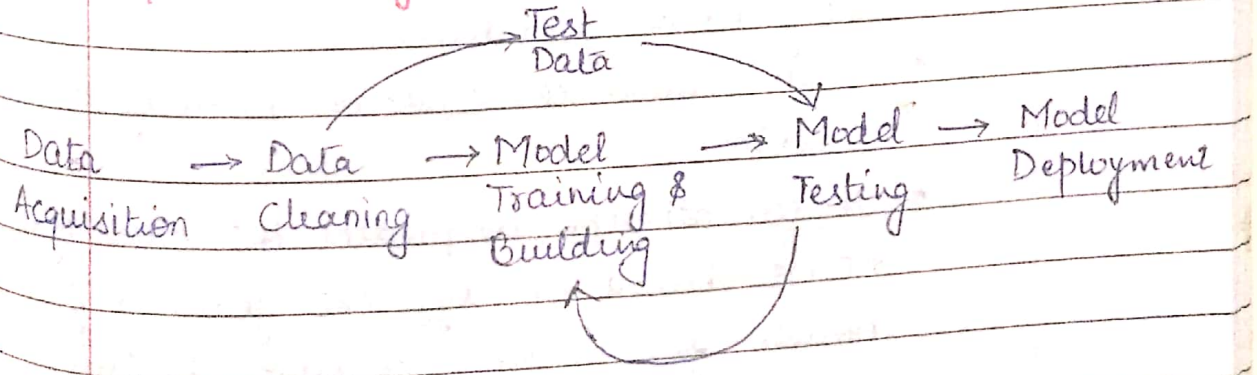
Machine learning:-

- It a method of data analysis that automates analytical model building.
- Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.

Usage of M/L

1. Fraud detection.
2. Web search results
3. Real time ads on web pages.
4. Credit scoring and next-best offers.
5. Prediction of equipment failures.
6. New pricing models.
7. Network intrusion detection.
8. Recommendation engines
9. Customer Segmentation
10. Text sentiment analysis
11. Predicting customer churn.
12. Pattern and image recognition.
13. Email spam filtering.
14. Financial modeling

M/L learning Process



3 main types of M/L algorithms

1. Supervised Learning.
2. Unsupervised "
3. Reinforcement "

1. We have labeled data and are trying to predict a label based off known features.
2. You have unlabelled data and are trying to group together similar data points based off of features.
3. Algorithm learns to perform an action from experience.

1.a. Algorithm learns by comparing its actual o/p with correct outputs to find errors.

1.b. It then modifies the model accordingly.

1.c. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on addition unlabeled data.

1.d Commonly used in applications where historical data predicts likely future events.

1.e. Ex → * it can anticipate when C.C. transactions are likely to be fraudulent.
* which insurance customer is likely to file a claim.
* It can attempt to predict the price of a house based on different features for houses for which we have historical price data.

Unsupervised learning -

- used against data that has no historical labels.
- System is not told the 'right answer'. The algorithm must figure out what is being shown.
- The goal is to explore the data and find some structure within or it can find the main attributes that separate customer segments from each other.
- Techniques includes self-organizing maps, nearest-neighbor mapping, k-means clustering and singular value decomposition.
- These algorithms are also used to segment text topics, recommend items and identify data outliers.

Reinforcement learning

- used for robotics, gaming and navigation.
- With reinforcement learning, the algorithm discovers through trial and error which actions yield the greatest rewards.
- It has 3 primary components:
  1. the agent (the learner or decision maker)
  2. the environment (everything the agent interacts with)
  3. actions (what the agent can do)
- The objective is for the agent to choose actions that maximize the expected reward over a given amount of time.
- The agent will reach the goal much faster by following a good policy.
- So, the goal in reinforcement learning is to learn the best policy.

Camlin Page
Date

Scikit Learn package - has lot of algorithms built
Every algorithm is exposed in scikit-learn via an "Estimator"

from sklearn.family import Model

Example

↑ Estimator object

from sklearn.linear_model import LinearRegression

Next, step is to instantiate that model or estimator

Estimator parameters - All the parameters of an estimator can be set when it is instantiated, and have suitable default values

Example:-

```
model = LinearRegression (normalize = True)
print (model)
LinearRegression ( copy_X = True, fit_intercept = True
                   normalize = True)
```

Splitting Train and Test data

x → features
y → labels

```
import numpy as np
```

↑ can be applied on features and labels

```
from sklearn.cross_validation import train_test_split
```

• model_selection

```
X, y = np.arange(10).reshape((5,2)), range(5)
X
```

list(y)

[0,1,2,3,4]

```
array([[0,1],
       [2,3],
       [4,5],
       [6,7],
       [8,9]]).

X_train, X_test, y_train, y_test = train_test_split (X,y
                                   test_size = 0.3)
>>X_train
array([[4,5],
       [0,1],
       [6,7]])

y_train
[2,0,3]

X_test
array([[2,3],
       [8,9]])

y_test
[1,4]
```

Train / fit our model

After splitting data, train / fit model on training data

```
model.fit (X_train, y_train)
```

↓

Model is ready to predict labels or values on the test set.

Predicted values

```
predictions = model.predict (X_test)
```

We can now evaluate our model by comparing our predictions to the correct values.

The evaluation methods depends on sort of ML algorithm (Ex. Regression, Classification, Clustering etc.)

Available in all Estimators

- model.fit() : fit training data.
- for SL appls, model.fit (X, y)
- for USL " , model.fit (X)

In supervised estimators

- model.predict (X_new)
- model.predict_proba() → for classification problems, some estimators also provide this method, which returns the probability that a new observation has each categorical label. In this case, the label with the highest probability is returned by model.predict()
- model.score() → for classification or regression problems, most estimators implement a score method. Scores are b/w 0 and 1, with a larger score indicating a better fit.

In Unsupervised estimators

- model.predict() → predict labels in clustering algorithms
- model.transform() → given an unsupervised model, transform new data into the new basis. Accepts one argument X_new, and returns the new representation of the data based on the unsupervised model.
- model.fit_transform → some estimators implement this method, which more efficiently performs a fit and a transform on the same input data.

Choosing an Algorithm (download cheat sheet - scikit-learn algorithm)

```
from sklearn.preprocessing import MinMaxScaler
scaler_model = MinMaxScaler()
scaler_model.fit(dataframe)
scaler_model.transform(dataframe)
```

Linear Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline
USAhousing = pd.read_csv('USA_Housing.csv')
USAhousing.head()
USAhousing.info() → gives column details (no & data type)
                  → no. of entries
USAhousing.describe() → count
                        mean
                        std
                        min
                        25%
                        50%
                        75%
                        max.
USAhousing.columns
```

EDA

```
sns.pairplot(USAhousing) → Does plotting b/w
sns.distplot(USAhousing['Price']) numerical variables.
                ↳ Creates dist. plot of Price colum
sns.heatmap(USAhousing.corr())
                ↳ draws correlation plot
```

Training a Linear Regression Model

```
X = USAhousing[[ .... 'features columns' ... ]]
y = USAhousing[['Price']]
```

Train Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=101)
```

(test_size → % of test data)

Optional — This essentially ensures a specific set of random splits on your data because this train split is going to occur randomly in order to make sure that your random is same as mine

Creating and Training the model

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit() → to train or fit my model on training data
lm.fit(X_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True,
                 n_jobs=1, normalize=False)
print(lm.intercept_)
-2640159.79.
lm.coef_
array([ 2.15, 1.64, 1.22, 2.233, 1.515])
X_train.columns
Index([ column names ], dtype='object')
cdf = pd.DataFrame(lm.coef_, X.columns,
                   columns=['Coeff'])
cdf
```

	Coeff
Avg. Area Income	21.52
Avg. Area House Age	164883.28
Avg. area no. of Rooms	122368.67
Avg. area no. of Bedrooms	2233.80
Area Population.	15.15

Ex. Importing ~~data~~ real data from sklearn

```
from sklearn.datasets import load_boston
boston = load_boston()
boston.keys()
dict_keys(['data', 'feature_names', 'target', 'DESCR'])
print(boston['DESCR'])
```

- 'DESCR' → data of housing prices
- 'feature_names' → Column names
- 'target' → target prices

Predictions from test set

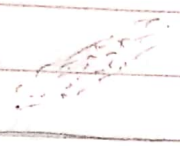
```
predictions = lm.predict(X_test)
predictions
array([ 1260960.7, ..... , ])
```

→ predicted prices of the house

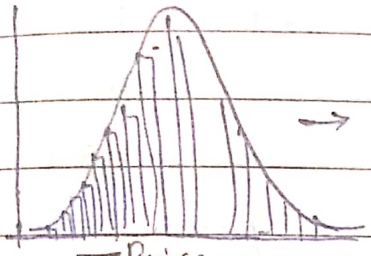
y_test → contains correct prices of house

Now, we want to know how far off are the predictions from the test prices the actual prices. This can be done by scatter plot.

```
plt.scatter(y_test, predictions)
sns.distplot((y_test - predictions))
```



→ Histogram of the distribution of our residuals (Ytest - predicted value)



→ Residuals looked to be normally distributed. It means model was a correct choice for the data.

→ If not, normally distributed, in ~~that case we might have to see~~ if a linear regression model was the correct choice for the dataset.

Regression Evaluation Metrics

There are 3 evaluations metrics for regression problems:

Mean Absolute Error (MAE)

- is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Mean Squared Error (MSE) is the mean of the squared errors:

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

- Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

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

MAE	MSE	RMSE
is the easiest to understand, because it's the average error.	is more popular than MAE, because MSE "punishes" large errors, which tends to be useful in the real world.	is even more popular than MSE, because RMSE is interpretable in the "y" units

★ All are loss functions, because we want to minimize them.

```
from sklearn import metrics
metrics.mean_absolute_error(y_test, predictions)
metrics.mean_squared_error(y_test, predictions)
np.sqrt(metrics.mean_squared_error(y_test,
                                   predictions))
```

R^2 can be calculated by
↳ a measurement of how much variance model explains.

① `metrics.explained_variance_score(y_test, predictions)`
r^2 in this case is 0.98 i.e. 98% of variance is explained by our model on test data.

or

② `lm.score(y_test, predictions)`

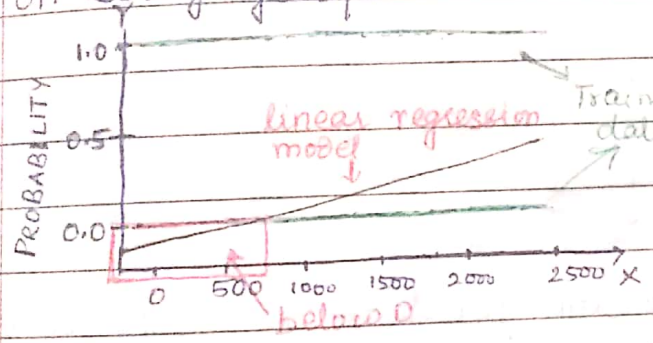
LOGISTIC REGRESSION (Sec 4-4.3)

- It is method for 'Classification'
- Ex:- ① Spam vessus "Ham" emails
  ② Loan default (yes/no)
  ③ Disease diagnosis

Binary classification

- ML and statistics classification is the problem of identifying to which of a set of categories a new observation belongs to based off of training data.
- It allows to solve classification problems where we are trying to predict discrete categories.

IMP - We can't use a normal linear regression model on binary groups. It won't lead to a good fit.



Y → probability of belonging to a particular group.

This plot is trying to predict likelihood of paying back loan.

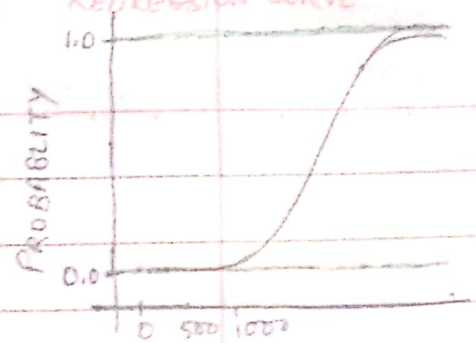
0 → 0% probablity as defaulting on their loan meaning they have a 0% probablity of being able to pay back their loan.

1 → 100% probablity as fully paying back their loan

IMP - As paycheck goes lowes you have a closer to zero % probablity that you're going to be able to pay back your loan and as your paycheck value gets higher you didn't have closer to 100% probablity of paying back loan.

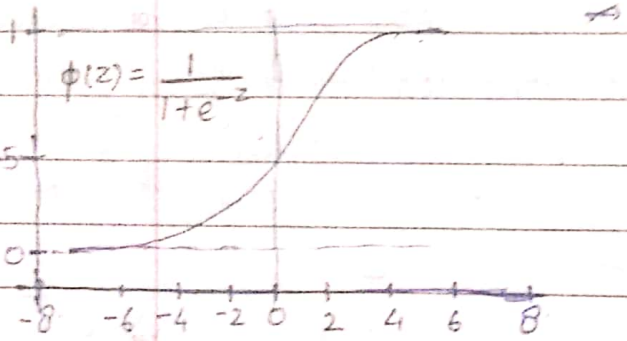
On training data (marked green) if we try to use a linear regression model we will get bad fit and end up predicting probabilities below zero % which does not make any sense.

LOGISTIC REGRESSION CURVE



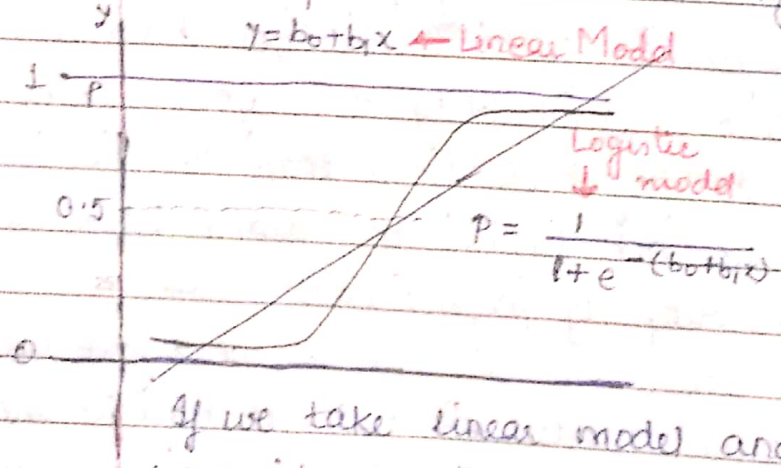
Therefore we can transform our linear regression to a logistic regression curve and logistic regression curve can only go b/w 0 and 1

SIGMOID FUNC^N OR LOGISTIC → takes in any value and outputs it to be between 0 and 1.



IMP - Logistic ~~regression~~ function is going to be key to understanding using logic regression to perform a classification

IMP This means we can take our Linear Regression solution and place it into the sigmoid func^n



If we take linear model and put it into the sigmoid func^n, we are able to Transform linear regression to a logistic model.

** We can set a cutoff point at 0.5

$P < 0.5 \Rightarrow 0$

$P > 0.5 \Rightarrow 1$

However, in some cases 0.5 can't be right cut off. (in imbalance data)

Model evaluation

CONFUSION MATRIX — is used to evaluate classification models. Model performance is evaluated on some test data for which the true values are already known.

n=165	PREDICTED: NO	PREDICTED: YES	
ACTUAL: NO	TN = 50	FP = 10 (TYPE I ERROR)	60
ACTUAL: YES	FN = 5 (TYPE II)	TP = 100	105
	55	110	

Ex:- Test for presence of disease

NO = negative test = False = 0

YES = positive test = True = 1

$$\text{ACCURACY} = \frac{TP + TN}{\text{TOTAL}} = \frac{100 + 50}{165} = .91 = 91\%$$ accurate.

$$\text{MISCLASSIFICATION RATE (ERROR RATE)} = \frac{FP + FN}{\text{TOTAL}} = \frac{15}{165} = 0.09 = 9\%$$ misclassifi or error rate

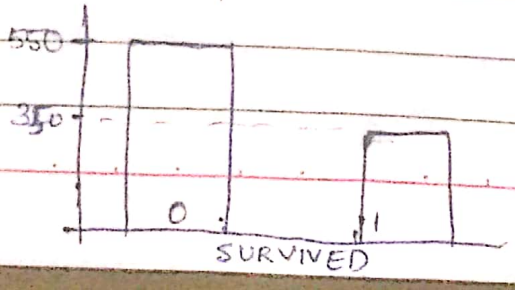
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
% matplotlib inline
train = pd.read_csv('titanic_train.csv')
train.head()
train.isnull() → False if it not null.
                 True if it is null.
sns.heatmap(train.isnull(), yticklabels=False,
            cbar=False, cmap='viridis')
```

PassengerId
Survived
Pclass
Name
Age Sex
~~Age~~ Sex
Cabin
Embarked

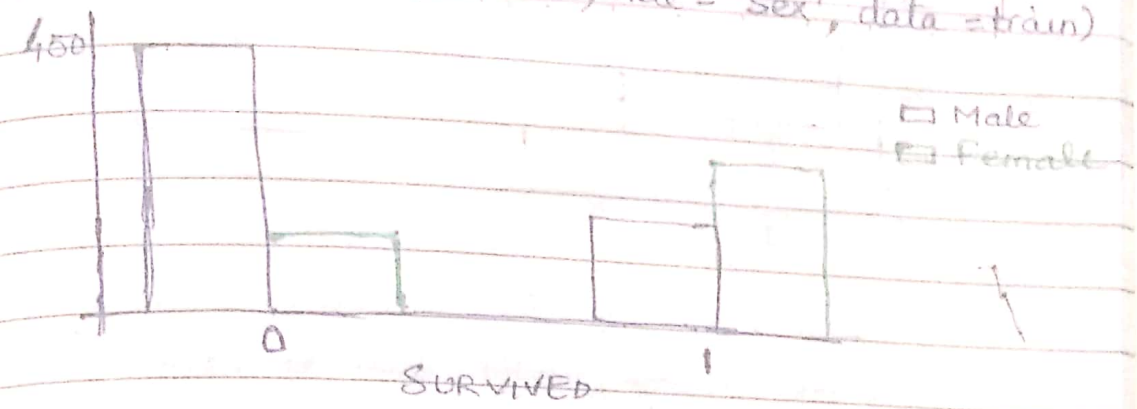
→ 20% age data missing
Red indicates TRUE meaning null values

Age data can be imputed by using the ~~Age~~ knowledge of other columns to fill in reasonable values for that age column.
Cabin ~~dat~~ column - missing too much of data to do something useful with it. We will probably drop this later or change it to 1 or 0.

```
sns.set_style('whitegrid')
sns.countplot(x='Survived', data=train)
```

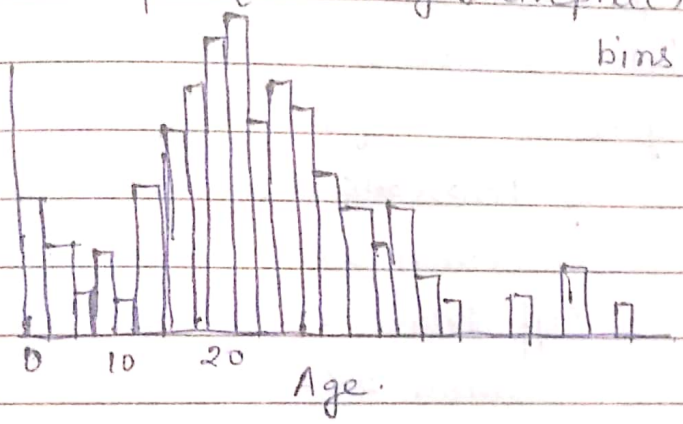


sns.countplot (x='Survived', hue='Sex', data=train)



Distribution Plot → Idea of the age of ~~palette~~ people

sns.distplot (train['Age'].dropna(), kde=False, bins=30)



OR train['Age'].plot.hist(bins=35)

sns.countplot (x='SibSp', data=train)

train['Fare'].hist(bins=40, figsize=(10,4))

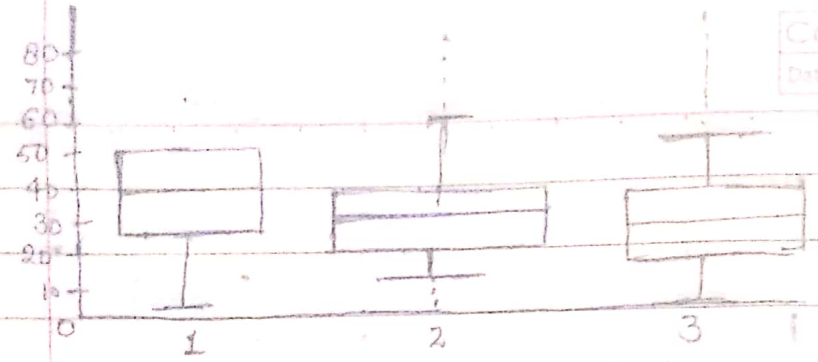
```
import cufflinks as cf
cf.go_offline()
train['Fare'].iplot(kind='hist', bins=50)
```

using interactive APIs.

Filling missing data for age column

1. Filling in the mean age of all the passengers. This is known as Imputation
2. Checking the average age passenger class.

```
plt.figure(figsize=(10,7))
sns.boxplot(x='Pclass', y='Age', data=train)
```



These average values can be used in order to impute the age based off of the past year data

```
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1].
    if pd.isnull(Age):
        if Pclass == 1:
            return 37
        elif Pclass == 2:
            return 29
        else:
            return 24
    else:
        return Age.

train['Age'] = train[['Age', 'Pclass']].apply(
                    impute_age, axis=1)
sns.heatmap(train.isnull(), yticklabels=False, cbar=False,
                    cmap='viridis')
```

Passenger
Survived
Pclass
Name
Sex
Age
SibSp
Parch
Ticket
Fare
Cabin
Embarked

In cabin column, there are too many missing points/information. Therefore, drop that cabin column.

```
train.drop('Cabin', axis=1, inplace=True)
```

Now, there is only 1 missing value in Embarked. Dropping any more missing value

```
train.dropna(inplace=True)
```

we filled few missing values and dropped few

Converting categorical features into dummy variables using Pandas

Sex & Embarked → Categorical

```
pd.get_dummies(train['sex'])
```

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
5	0	1

If ML algorithm is fed both columns, it will know that if it's zero female it can predict perfectly it's going to be 1 male. This is known as multi-collinearity and will mess up the algorithm. Therefore,

```
Sex = pd.get_dummies(train['sex'], drop_first=True)
embark = pd.get_dummies(train['Embark'], drop_first=True)
train = pd.concat([train, sex, embark], axis=1)
```

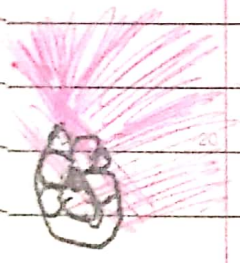
★ Ticket and Name column are not much useful, but we can do feature engineering in features However, in this scenerio we will go ahead and drop these columns.

```
train.drop(['Sex', 'Embarked', 'Name', 'Ticket'], axis=1, inplace=True)
train.drop(['PassengerId', axis=1, inplace=True)
```

```
X = train.drop('Survived', axis=1)
y = train['Survived']
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.80	0.91 (148/163)	0.85	163
1	0.82	0.65 (68/104)	0.73	104
avg/total	0.81	0.81	0.80	267

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, predictions)
array([[148, 15],
       [36, 68]])
```



		PREDICTED 0	PREDICTED 1	
Actuals	0	148	15	163
	1	36	68	104
		184	83	267

$$\text{Accuracy} = \frac{148+68}{267} = \frac{216}{267} = .8089$$

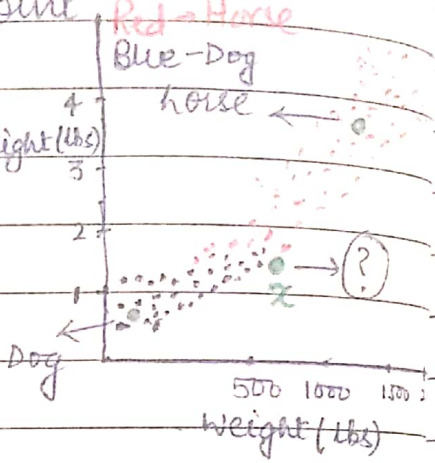
precision of (1) $\frac{68}{83} = .82$

(0) $= \frac{148}{184} = .80$

KNN Theory (Chap 4 of Statistical Learning)

→ Classification algorithm.

Ex: We want to know if we get an animal where we only know the height and weight, we want to predict if that new data point is a horse or a dog. We are trying to classify the new data point Based on where the green point lies we can predict whether it is horse or dog.



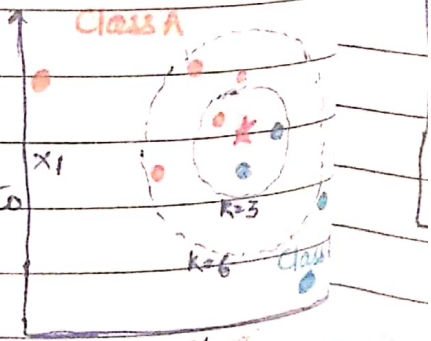
Training Algorithm

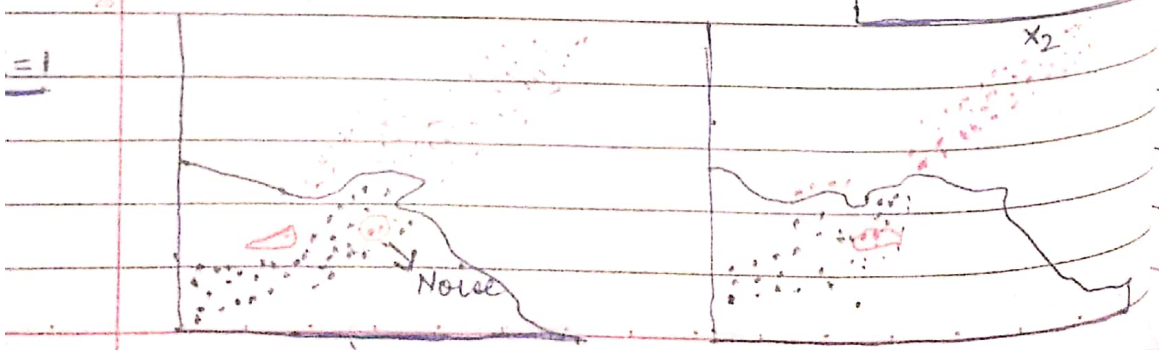
Store all the data.

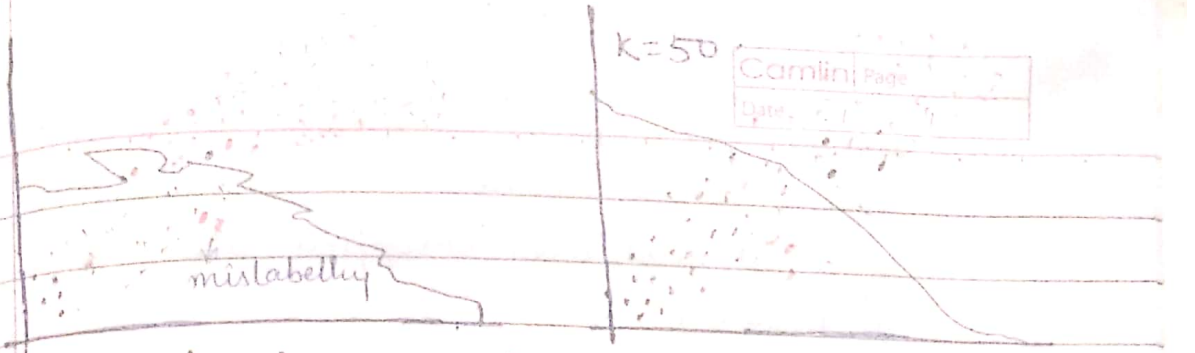
Prediction Algorithm

1. Calculate distance from x to all points in your data
2. Sort the points in your data by increasing distance from x.
3. Predict the majority label of the "k" closest points.

Choosing k will affect what class a new point is assigned to







As With increase in K values, we notice it smooth out and create more of a bias in ~~the~~ model or get a clear cut off at the cost of mislabeling some points.

PROS

1. Very simple
2. Training is virtual (sort data)
3. Works with any number of classes
4. Easy to add more data.
5. Few parameters
   - K
   - Distance Metric (Dist. b/w test point and training point)

CONS

1. High Prediction Cost (worse for large data sets)
2. Not good with high dimensional data
3. Categorical features don't work well (as we can't cal distance)
   a) as you have to sort entire large data set.
   b) High prediction cost.
   c) Testing out what points are closest to this data point.
   → Throw off ability to measure distances in various dimensions.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline
df = pd.read_csv('Classified Data', index_col=0)
df.head()
```

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ	TARGET CLASS
0											1
1											0
2											1

Standardizing everything to same scale

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df.drop('TARGET CLASS', axis=1))
scaled_features = scaler.transform(df.drop('TARGET CLASS', axis=1))
```

↓ peform standardization by centering and scaling.

↳ Gives scaled data

```
df_feat = pd.DataFrame(scaled_features, columns=df.columns[:-1])
```

↳ Standardized data

```
from sklearn.cross_validation import train_test_split
X = scaled df_feat.
y = df['TARGET CLASS']
X_train, X_test, y_train, y_test
        = train_test_split(X, y, test_size=0.3,

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=3
                     metric='minkowski',
                     metric_params=None, n_jobs=1,
                     n_neighbors=1, p=2, weights='uni
```

```
pred = knn.predict(x_test)
```

evaluation of knn model,

```
from sklearn.metrics import classification_report,
                            confusion_matrix
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[151  8]
 [ 15 126]]
```

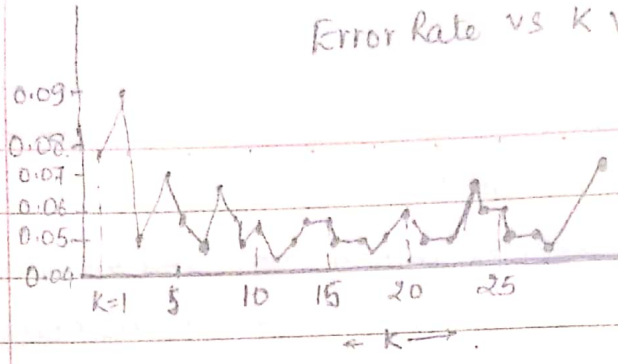
	precision	recall	f1-score	support
0	0.91	0.95	.93	159
1	0.94	0.89	.92	141
avg/total	0.92	0.92	.92	300

Elbow method to choose k value

Error rates is an empty list. We will iterate many models using many different values and plot out their error rate and see which one has the lowest error rate.

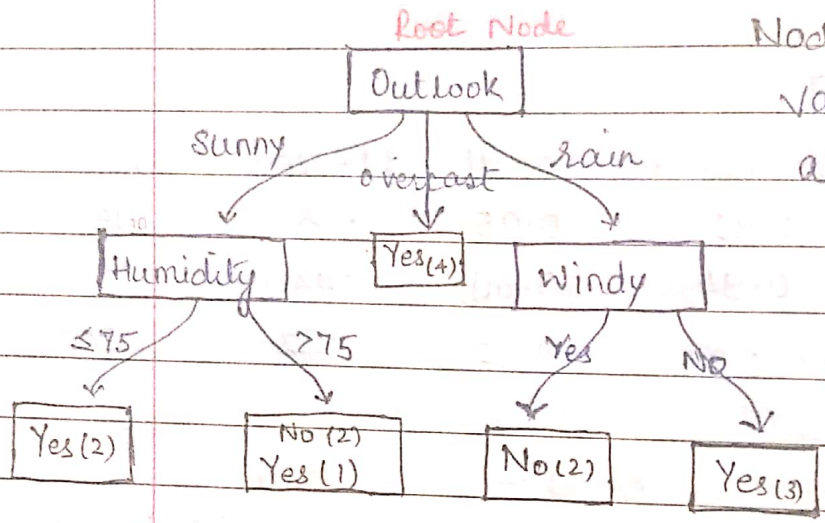
```
error_rate = []
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize=(10,6))
plt.plot(range(1,40), error_rate,
         color='blue', linestyle='dashed',
         marker='o', markerfacecolor='red',
         markersize=10)
plt.title('Error Rate vs K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Error Rate vs K Value



for k=1, error rate is $\frac{23}{300} = 0.076$

DECISION TREE (Chapter 8)



Nodes – Split for the value of a certain attribute.

Root – The node that performs the first split

Leaves – Terminal nodes that predict the outcome.

- Decision trees don't have the best predictive accuracy. This is partially due to the high variance meaning that different splits in the training data can lead to very different trees.
- Bagging is a general purpose procedure for reducing the variance of ML method.

Random Forests → used for very fast classification model

To improve performance, we can use many trees with a random sample of features chosen as the split.

1. A new random sample of features is chosen for every single tree at every single split.
2. For classification, m = $\sqrt{p}$, where p = features

#1. Using one very strong feature in "bagged trees as top split, will result in an ensemble of similar Trees that are highly correlated.

#2. Averaging highly correlated quantities does not significantly reduce variance.

#3. However, if we leave some features from each split, Random forests "decorrelates" the trees, This will result in reducing the variance of the resulting model by averaging process.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df = pd.read_csv('kyphosis.csv')
df.head()
```

	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

Number - is a number of vertebrate involved in the operation

Start - Number of the first or top most of vertebrate that was operated on.

df.info() gives 81 entries

```
sns.pairplot(df, hue='Kyphosis')
from sklearn.cross_validation import train_test_sp
X = df.drop('Kyphosis', axis=1)
y = df['Kyphosis']
X_train, X_test, y_train, y_test = train_test_split
                                   (X, y, test_size=0.3
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
predictions = dtree.predict(X_test)
from sklearn.metrics import classification_report
                           confusion_matrix
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

[[17 5] misclassifying 6 from test set
[1 2]]

	precision	recall	f1-score	support
absent	0.94	0.77	0.85	22
present	0.29	0.67	0.40	3
avg/total	0.87	0.76	0.80	25

Comparing above result with Random forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(x_train, y_train)
rfc_pred = rfc.predict(X_test)
print(confusion_matrix(y_test, rfc_pred))
print(classification_report(y_test, rfc_pred))
```

[[19 3]
[1 2]]

	precision	recall	f1-score	support
absent	0.95	0.86	0.90	22
present	0.40	0.67	0.50	3
avg/total	0.88	0.84	0.86	25

little hard to tell which performed better because it really depends whether we value precision or recall or being present or absent.
It is more important to realize that it is still present Vs absent. But it again really depends on the situation and the patient itself and what costs are associated with those decisions.

However, if we compare based on confusion matrix, overall it looks like Random Forest a little better.
Also, as data sets gets larger and larger RF will always do better than single decision tree.
In above ex, dataset was too small therefore RF was not able to outshine decision tree as much

```
- df['Kyphosis'].value_counts()
absent  64   ||  → dataset is unbalanced and
present 17.  ||    that also affect models
```

TREE VISUALIZATION USING DT

Scikit learn has some built in visualization capabilities for decision trees. However, ~~the~~ they are not used as most often RF are used which are difficult to visualize as there are 100s of trees Vs 1 in DT.

```
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot → is not included in anaconda dist.

features = list(df.columns[1:])
features
['Age', 'Number', 'Start']
```

additional library that will take the graph object and create picture of it from

```
dot_data = StringIO()
export_graphviz(dtree, out_file = dot_data,
                feature_names = feature,
                filled = True, round
```

Go to www.graphviz.org to install

```
graph = pydot.graph_from_dot_data
        (dot_data.getvalue())
Image(graph[0].create_png())
```

↳ require graphviz lib

First choice for very fast classification model to see what's kind of a baseline accuracy or precision or recall you can get a model before using other models or tuning stuff.