NOTES

Commonly used ML algorithms

- Linear Regression → estimate real value based on continuous variables.
- Logistic Regression → estimate discrete values (0/1) based on given set of independent variable.
- Decision tree → classification problem.
- SVM - classification method → graph plot
- Naive Bayes - assumes presence of particular feature in a class is unrelated to presence of any other feature.
- kNN - classification & regression.
- k-means - clustering problem.
- Random forest - ensemble of decision trees.
- Dimensionality reduction.
- Gradient boosting algorithm.
  - ↳ GBM → used while dealing with plenty of data → boosting algo
  - ↳ XGBoost → high predictive power → boosting algo
  - ↳ LightGBM → fast training speed & higher efficiency → tree based learning algo.
  - ↳ CatBoost → open sourced ML

5. Python 2:55 hrs
- Introduction
- Variable to Datatypes
- Basic of Python syntax
- Other Python operator
} 1:01 hrs
- Conditional statements
- Python Functions
- Sequences
} 1:08 hrs
- Iterations
- Advance Python tools.
} 0:46 hrs

6. Advanced Statistical method in Python 4:54 hrs
- * Linear regression with Statsmodel
- * Multiple Linear regression with statsmodel
- * Linear regression with Sklearn
  - Practical example
- * Logistic Regression
- * Cluster Analysis
- * K-mean clustering
- * Other type of clustering

7. Mathematics 0:51 hrs.

8. Deep learning 13:06 hrs.
46m - Intro to Neural network
21m - How to build a neural n/w from scratch - Numpy
28m - Tensor Flow Intro
26m - Digging deeper into NNs - Deep Neural N/w
20m - Overfitting
8m - Initialization
21m - Digging into Gradient Descent & learning rate schedules
15m - Preprocessing
37m - Classifying on the MNIST dataset

Content - 30 hrs

1. Introduction
2. Field of Data Science - The various DS disciplines
3. The field of Data Science - Connecting the Data Science discipline

1. Introduction 2:09 hrs.
2. The field of Data Science
- Various DS disciplines
- Connecting the DS disciplines
- Benefits of each disciplines
- Popular DS techniques
- Popular DS tools
- Career in DS
- Debunking common misconception.

3. Probability 3:37 hrs
- Combinatorics
- Bayesian Inferences
- Distributions
- Probability in other field.

4. Statistics 3:49 hrs
- * Descriptive Statistics
- Practical example
} 1:04 hr
- * Inferential Statistics fundamentals } 0:22 hrs
- * Inferential Statistics: Confidence Intervals
- Practical Example
} 0:54 hrs
- * Hypothesis testing
- Practical example
} 0:55 hrs

39 - Business Case Example
17m - Conclusion

2h 9. Appendix
- 29m - Deep learning Tensorflow - Intro
- 16m - Classifying on MNIST dataset
- 56m - Business Case

30m 10. 30m Software Integration

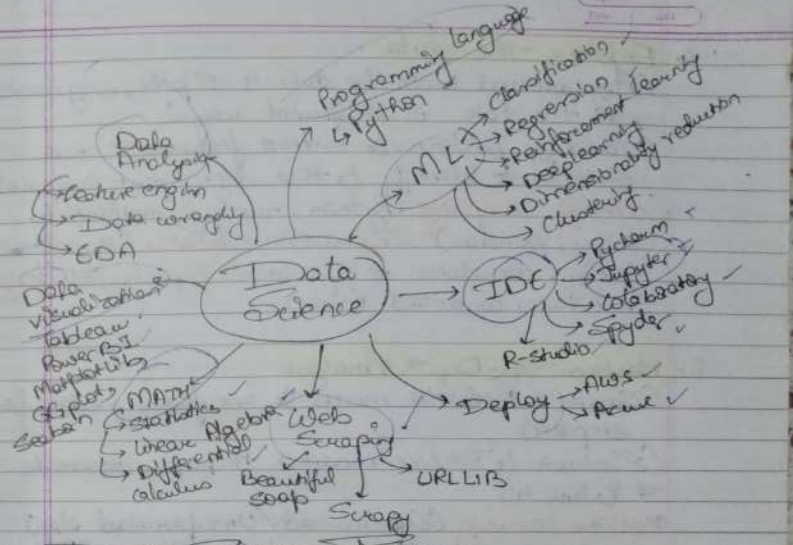
35m 11. Case Study.
- 10m - What's next
- 1:30 - Preprocessing the Absenteeism_data
- 1:17 - Applying ML to create the absenteeism_module
- 16m - Loading the absenteeism_module
- 23m - Analyzing predicted O/P in Tableau

1:04 12. Appendix.
- 40 - Additional Python tools
- 59 - Pandas Fundamentals

13. Bonus Lecture.

---

3. Process the Data
(Examine data at a high-level → Understand every column; identify errors, missing values & corrupt records)
(Clean the data → Throw away, replace, and/or filter corrupt/error prone/missing values)
Scripting language Python/R
Data wrangling & cleaning Python Pandas Library
Distributed Processing Hadoop MapReduce/Spark.



Data Science Process
1. Frame the problem (translate ambiguous req. into a concrete, well-defined problem. Identify business priorities & strategy decisions that will influence your work.
Domain knowledge (needed), Product Intuition (metrics), Business Strategy (priorities), Teamwork (people & resources)
2. ~~Identify~~ Collect raw data
(Identify all available datasets, extract data into usable format)
Database management (Systems: mySQL, postgreSQL, Oracle, MongoDB) Query Structured Database (SQL)
Retrieving unstructured info. (Information retrieval/text mining)
Distributed Storage (Hadoop HDFS, Spark, Flink)

4. Explore the data
(Play around with the data → Split, segment & plot the data in different ways)
(Identify patterns & extract features → use statistics to identify & test significant variables)
Scientific computing (Python, numpy, matplotlib, scipy, pandas) Inferential Statistics (Hypothesis testing, correlation vs causation) Experimental design (A/B tests, controlled trials)

5. Perform In-Depth Analysis
(Create a Predictive model → use feature vectors from step #4)
(Evaluate & Refine model → Perhaps return to # 2, 3 or 4)
Machine Learning (Supervised/Unsupervised algos) contextual problems) ML tools & library (Python: scikit-learn) Advanced math (linear algebra & multivariate calculus)

6. Communicate results
(Identify business insights → Return back to the business problem) (Visualize your findings → keep it simple & prioritize focus) (Tell a clear & Actionable story → Effectively communicate to non-technical audiences)
Business Acumen (Non-technical terminology)
Data Visualization tools (Tableau, D3.js, Google visualize, matplotlib, ggplot, seaborn
Data storytelling (presenting & speaking, reporting & writing)

Introduction.

Scatter Plot

K-mean clustering → Fan — High loyalty & high satisfaction
→ Supporter High loyalty & low satisfaction
→ Roamer Low loyalty & High satisfaction
→ Alienated Low loyalty & low satisfaction.

Improve shopping exp.
Loyalty cards, discount, raffle.

→ Cluster analysis - states the problem

Dendrogram
Heat map

Free Tableau training - 50% completion message

"Hi! I completed 50% of the course! Can I have my gift, please?" → Q/A section.

Field of Data Science

Data — Data team — Big data team
Business Intelligence — Data Science — Business analytics
Data Analytics.

Statistics → Datamining → Predictive analytics
Data Science.

* Analysis
Past
Explain
How? → Qualitative
Why? → Quantitative Data → How sale decreased

* Analytics
Future
explore potential future events.
Qualitative Intuition + analysis
Quantitative formulas + algorithm

3/2/22

Exploratory Data Analysis.

↳ Find out the work areas where you can work to make more profit.

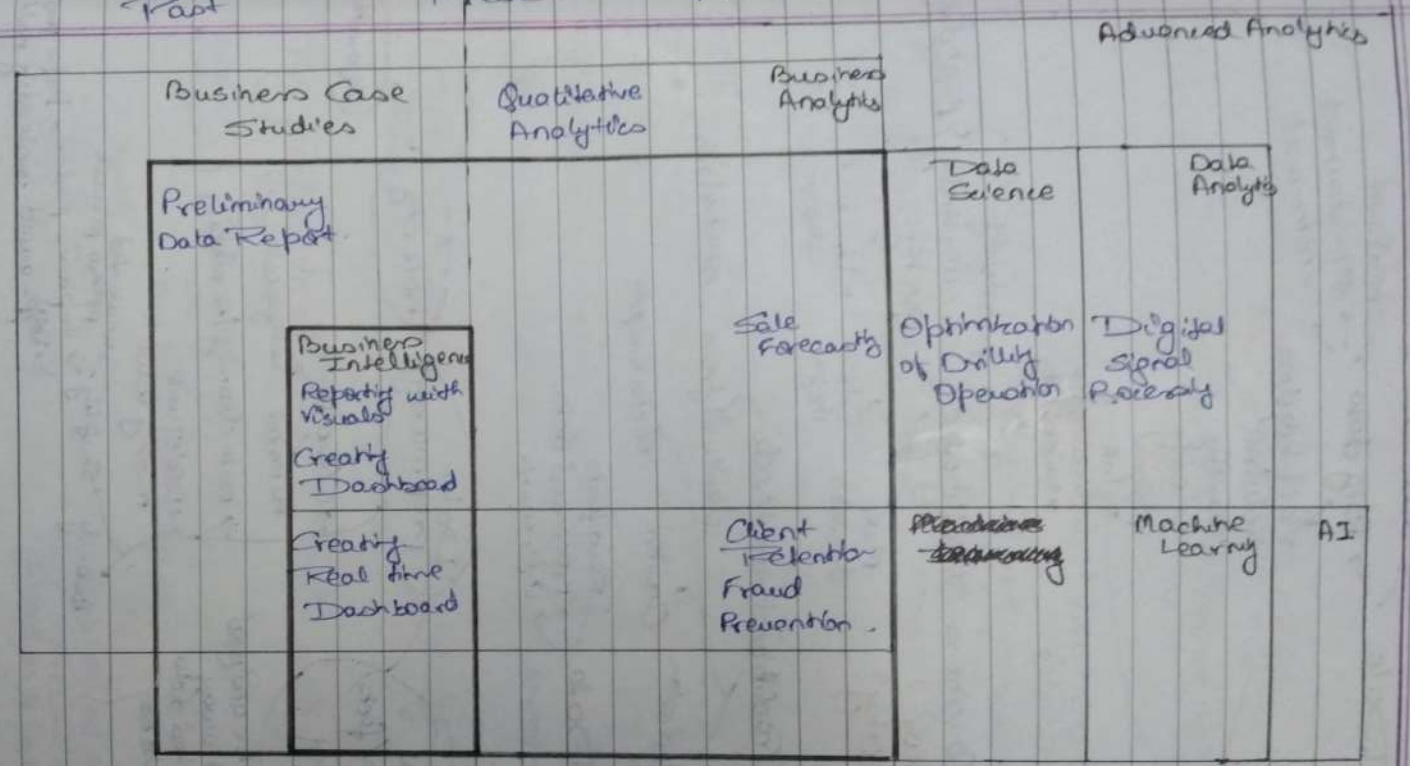
* Data science - discipline reliant on data availability while
Business analytics does not completely rely on data
* Data science can be used to improve the accuracy of predictions based on data extracted from various activities typical for driving efficiency

* Business intelligence
aims to explain past events using business data

- Machine learning → creating
→ implementing algo
→ let m/c receive data to use data to → Make prediction
→ analyse pattern
→ give recommendations

- Artificial Intelligence
Simulate human knowledge to decision making with computer

- Symbolic Reasoning → AI
↳ based on high level human-readable representation of problem & logic


Past
Present
Future.
Advanced Analytics
Business Case Studies
Qualitative Analytics
Business Analytics
Data Science
Data Analytics
Preliminary Data Report
Business Intelligence Reporting with visuals
Creating Dashboard
Sale Forecasting
Optimization of Drilling Operation
Digital Signal Processing
Creating Real time Dashboard
Client Retention
Fraud Prevention
Machine Learning
AI

Data
- Traditional structured
- Big data
  - structured
  - semistructured
  - unstructured

V's of big data
Volume
Variety
Velocity
Vision
Value
Visualization
Variability

- Business Intelligence
  - analysing
  - understanding
  - repeating

  } past data.

  How?
  Why?

  Make decision
  Extract insight
  Extract ideas

- Traditional methods → derived from statistics
  - Regression
  - Clustering
  - Factor analysis
- Data
  - Raw facts
  - Processed data
  - Information

Raw data / facts / primary data
→ Cannot be analysed straight away
→ untokened data
→ Data collection

→ Data preprocessing → Processing → Information

Data preprocessing:
- Class labelling
  - Numerical
  - Categorical
- Data cleaning / scrubbing
- Dealing with missing value
- Balancing
- Shuffling
  - unwanted pattern prevents
  - improves predictive performance
  - helps avoid misleading result

Big Data → financial trading data record

Data mining
Data masking
confidentiality preserving data mining.

Information → Business Intelligence (BI) analysis

Data Skills + business knowledge + intuition

Explain past performance
- what
- when
- how many
- which region

Quantification – the process of representing observations as numbers

Observations → Quantifications → Measures
- accumulation of observations to show some information

Measure → related to a simple descriptive statistics of past performance

Metric = Measure + Business meaning
- comparison

business goals → Key performance (KPIs) = Metrics + Business objectives
- indicator
- within timeframe
- generated only from user who have clicked on a link – ad campaign

Metric
- traffic of a page visited by any type of user

KPI
- traffic generated only from users who have clicked on a link provided

Link - edureka - Youtube
"Exploratory data analysis using Python | Python data analysis | Python training | Edureka"

5/2/22

Exploratory Data analysis.
↳ understand various aspects of data.

Objectives.
↳ identify faulty part.
↳ understand relationship b/w the variables

Steps

Understand data → Clean Data. → Analyse relationship b/w variables.

import seaborn ← visualization.

* Understand data

data.head(), data.tail() (range)

data.shape, data.describe()
(no. of rows, column) → count, mean, std, min, max.

data.columns.
data.nunique() → unique value.
data['___'].unique()

* Cleaning.
→ Check null — data.isnull().sum()
→ Delete redundant data — std = data.drop(['race', 'parental'], axis=1) (variable)
→ check outlier - data point that differ significantly from others.

finding std[(std['cgpa'] > 8.80) | (std['cgpa'] < 5.11)]
trimming new std = std[(std['cgpa'] < 8.80) & (std['cgpa'] > 5.11)]

* Relationship analysis

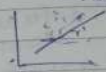
- Correlation matrix
  correlation = student.corr()
- use heatmap - visualize volume
  sns.heatmap(correlation, xticklabels=correlation.columns, yticklabels=correlation.columns, annot=True)
- pairplot - plot pairwise relationship in dataset
  sns.pairplot(student)
- Scatterplot
  sns.relplot(x='maths score', y='reading score', hue='gender', data=student)
- Histogram. - bar plot
  → Represent data in form of groups
  sns.distplot(std['writing score'], bins=9)
  ↑ no. of bars
  → sns.catplot(x='writing score', kind='box', data=student)

%matplotlib inline - interactive width backend to frontend
↳ plotting command below o/p cell of prev. plot.

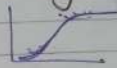
import os
provides func° for creating & removing a directory, fetching content, changing & identifying current directory.

16/2/22

Linear regression y=bx
↳ to quantify causal relationship among variables



Logistic regression → decision making
↳ 0's & 1's



Cluster analysis

$y = a + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$

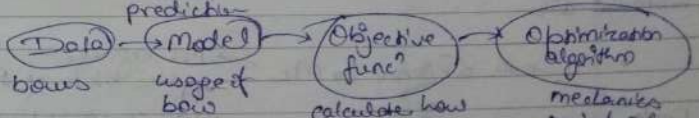
x → explanatory variable
→ Regressor
→ independent variable
→ predictor variable

* Factor analysis
↳ group by explanatory variable together

* Time series → plotting values against time

Machine learning → create an algo → trial & error process
→ find model
→ fit data
→ improve complex computational models
→ accurate predictions

eg



Training model → no rules
→ a final goal
→ measure inaccuracy
→ improve

Types of machine learning (ML)

- Supervised learning
  - training analgo.
  - labelled data
  - know target prior
  - SVM, NN, DL, Bayesian n/w
- Unsupervised learning
  - unlabelled data
  - large data → DL
  - K-mean
- Reinforcement learning
  - + reward system

Hadoop → a software - collection of programs.

Roles

- Data → DB Administrator.
  Handles this control of data mainly works with structural data
  - Data Architect
    Design the way data will be retrieved, processed & consumed
  - Data engineer
    Processes the obtained data so that it is ready for analysts
- Business intelligence (BI) → BI developer performs analyses specifically designed for company
  - BI Analyst
    perform analyses & reporting of past historical data
  - BI consultant
    external BI analyst
- Data Science & ML → ML engineer applies state of the art computational methods.
  - Data Scientist
    employ traditional statistical methods or unconventional ML techniques for making prediction.
  - Data Analyst
    prepares more advanced types of analyses

	Data		Data Science		
	TRADITIONAL	BIG	Business Intelligence	Traditional Methods	Machine Learning
WHEN it is applied	At the beginning of your analysis.		After the data has been gathered & organized	After BI reports have been created to discussed	
			PAST	NOW	FUTURE
Why you need it	data driven decisions require well organised & relevant raw data stored in digital format.		use data to create reports & dashboards to gain business insights.	Predictive Analytics access potential future scenarios by using advanced statistical methods	utilize artificial intelligence to predict behaviour in unprecedented ways
What techniques involved	DATA COLLECTION Preprocessing • class labeling (categorical vs numerical) • data cleaning • dealing with missing values Case specific eg- Balancing & Shuffling datasets ER diagram RDBMS	DATA COLLECTION Preprocessing • class labeling (no, text, digital images, digital video data, digital audio data) • data cleaning • dealing with missing values Case specific Text data mining confidentiality - Preserving data mining techniques	ANALYSE THE DATA Extract info. and present it in the form of: • Metrics • KPIs • Reports • Dashboard	Regression Logistic regression Clustering Factor Analysis Time Series	Supervised learning • SVMs • NNs • Deep learning • Random forest • Bayesian n/w Unsupervised learning • K-means • Deep learning Reinforcement learning similar to supervised learning, but instead of mining the labels, one maximizes rewards
Where it is used	Basic customer data Historical stock price data	Social media Financial trading data	Price optimization Inventory management	User experience (UX) (Customer analysis) Sales Forecasting (Time series)	Fraud Detection client retention
How using what tools	Programming lang. R, Python, SQL, MATLAB Software Excel, IBM SPSS	Programming lang. R, Python, Java, Scala Software HADOOP, APACHE HBASE, MongoDB	Prog. lang. R, SQL, Python, MATLAB Software EXCEL, PowerBI, SAS, Qlik, Tableau	Prog. lang. R, Python, MATLAB Software EXCEL, IBM SPSS, EViews, Stata	Prog. lang. R, Python, MATLAB, JAVA, Javascript, C, Scala, C++ Software Microsoft Azure, Rapidminer
Who	Data Architect Data Engineer Database Administrator	Big Data architect Big Data Engineer	BI Analyst BI Consultant BI Developer	Data Scientist Data Analyst	Data Scientist Machine learning engineer.
	Big Data - not just volume it defines Variety, Variability, Velocity, Veracity & others		Qualitative analysis SWOT Strength Weakness Opportunity Threats not used	Excel, SPSS, Stata - useful	DL is still a debate on why algo used to outperform all conventional

Probability

→ the likelihood of an event occurring → likely
→ unlikely
→ relatively more likely

higher prob. value (0.8, 0.9, 1) ⟹ higher likelihood.

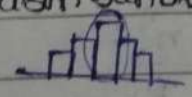
$P(A) = \frac{\text{preferred/favourable}}{\text{All of sample space}}$

Independent event $P(A \& B) = P(A)P(B)$

Expected value → used to make predictions about future based on past data
avg. outcomes we expect if we run an exp. many times
experimental probabilities → good approximation
→ easy to compute

$E(A)$ → outcome we expect to occur when we run an exp. $E(A) = P(A) \times n$

$E(X) = P(A) \cdot A + P(B) \cdot B + P(C) \cdot C$

Probability frequency distributions → collection of prob. for each possible outcomes
eg sum of 2 dices 

$P > 1$ ← double counting
$P < 1$ ← not counting fully

$P(A) + P(B) + P(C) = 1$
$A' = B + C$
$P(A') = 1 - P(A)$

Combinatorics → Permutation, Variations, Combination ? No. of fav. outcomes

↳ combination of objects from a ~~set~~ specific finite sets

Restrictions → Repetition, Order, other

- Permutation → no. of diff. possible ways we can arrange elements
  ↳ Order is completely upto us

$P_n = n!$

↳ -ve. no. don't have factorial

$(n+k)! = n! \times (n+1) \times (n+2) \times \dots \times (n+k)$

$(n-k)! = \frac{n!}{(n-k+1)(n-k+2)\times\dots\times n}$

$\frac{n!}{k!} = (k+1)(k+2)\dots n$

Variation - permutation with repetition

first pick & then arrange some elements

$\overline{V}_p^n = n^p$ p-many element

ABC [ ][ ] $\overline{V} = 3^2 = 9$

2

Variation w/o repetition

when arranging p elements out of total of n.

$\overline{V} = \frac{n!}{p!}$

Combination

↳ don't take into account - double counting

↳ set

↳ All different permutations of a single combination are different variations

$C_p^n = \frac{n!}{(n-p)!\,p!} = \frac{V_p^n}{P_p}$

Picking more elements leads to having fewer combinations

$^6C_4 = 15 \quad ^6C_5 = 6$

↳ Symmetry of combination

$p > n/2 > n-p$

- Solving combinations with separate sample spaces

eg $^3C_1 * ^2C_1 * ^5C_1$

↳ Determine the appropriate amount of time it would take for such a task to be completed

Set

$x \in A$ — element, set — x part of A

$A \ni x$ — set, element — A contains x

$\forall$: for all ; such that

event A, event B

Mutually exclusive - non overlapping

$A \cup B = A + B$

Independent event.

if $P(A) = P(A|B)$ then the two are independent and if not, A and B are dependent.

eg getting head after another head.

Conditional probability

$P(A|B) = \frac{P(A \cap B)}{P(B)}$ if $P(B) > 0$

$P(A) = P(A|B)$ ← independent

if any two events are independent $P(A \cap B) = P(A) \times P(B)$

$P(A|C) \neq P(A)$ ← A & C are also dependent

if $P(B) = 0$ → event B would never occur

$A|B$ → not interpretable.

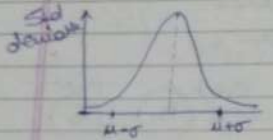
Law of total probability

$P(A) = P(A|B_1) * P(B_1) + P(A|B_2) * P(B_2) + \dots$

Bayes' Rule

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Possible values a variable can take & how freq. they occur



more congested the middle of distribution the more data falls within that interval

Probability distributions

Finite no. of outcomes → Discrete → True / False

Infinite many outcomes → Continuous → Slide

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

Variable | Type characteristics

Discrete distributions → finite distinct outcome

- Bernoulli Distribution $\sigma^2 = p(1-p)$ — events with only two possible outcomes (1 trial → p, 0 → 1−p)
- Binomial Distribution — carry out similar exp. several times in a study.
  - * 2 outcomes per iteration
  - * many iterations

  eg. of identical Bernoulli events $X \sim Bin(n, p)$
- Poisson Distribution — Test out how unusual an event freq. is for a given interval

$P(Y \leq y) = P[Y < (y+1)]$

Continuous distribution.

- Normal distribution — outliers
- Student's T distribution → accomodates extreme values significantly better
  - ↳ Any extreme value represents a much bigger part of population
- Chi squared
  - ↳ not often mirror real life events
  - ↳ used in Hypothesis testing
  - ↳ Asymmetric
  - ↳ non negative values
- Exponential distribution
  - ↳ events that are rapidly changing early on
- Logistic distribution
  - ↳ useful in forecast analysis
  - ↳ useful for determining a cut off point for a successful outcome

Uniform distributions

↳ all outcomes have equal probability i.e each outcome is equally likely

Drawback. - No predictive power (knowledge of expected value of var.)

* Both mean & variance are uninterpretable

Binomial Distribution

1 T/F question → Bernoulli event

Test of 10 T/F ques → Binomial distr

$X \sim B(10, 0.6)$ — no. of times we expect to get specific outcome

$B(n, p)$ — n → trials, x → no. of times

(n+1) bars

Expected values: $E(X) = p_1 x_1 + p_2 x_2 + \dots + p_n x_n$

$$P(y) = {}^{n}C_r \, p^r (1-p)^{n-r}$$

$$\text{Var } \sigma^2 = E(y^2) - E(y)^2 = np(1-p)$$

Poisson Distribution, $Y \sim P_o(\lambda)$

↳ freq. with which an event occur

↳ no. of occurance.

$$P(Y) = \frac{\lambda^y e^{-\lambda}}{y!}$$

e → euler's no. / Napier's const. 2.72

$$E(Y) = \lambda \qquad \sigma^2 = \mu = \lambda$$

Continuous distribution

↳ Infinitely many consequetive outcomes

↳ Sample space is infinite

PDF: $f(y) \geq 0$ ↳ associated prob. of every possible value 'y'

$P(y) = \frac{1}{\infty}$ ↳ likelihood of each individual one would be extremely small

Prob. for any individual value equal to 0.

Integral check → wolframalpha.com

Cumulative distribution Function (CDF)

$F(-\infty) = 0 \qquad F(\infty) = 1$

Prob. of intervals $P(b > x > a)$

$\int_a^b p(x)\,dx$

PDF —Integral→ CDF $\int_{-\infty}^{y} p(y)\,dy = F(y)$

PDF ←Derivative— CDF $p(y) = F(y)\frac{d}{dy}$

$$E(y) = \int_{-\infty}^{\infty} y\,p(y)\,dy$$

$$\text{Var}(y) = E(y^2) - E(y)^2$$

Normal Distribution → Bell shaped curve

↳ prob. distribution that is symmetric about the mean

68, 95, 99.7 law.

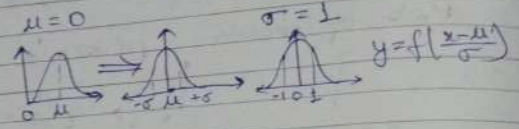
Standard normal distribution

Transformation — Way in which we can alter every element of a distribution to get a new distribution $(x, \div, -, +)$

$y = f(x)$ $\quad y = f(x+3)$ $\quad y = f(x-3)$ $\quad y = f(x \cdot c)$

$y = f(x^2)$

Standardize $E(x)=0$ $Var(x)=1$



$Z \sim N(0,1)$

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

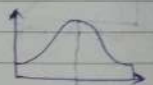
$Z = \frac{Y-\mu}{\sigma}$ → no. of standard deviations away from the mean

Drawbacks
- useful when we have normal distribution
- requires lot of data
- Risk of outliers drastically affecting our analysis

Student's T distribution

$t(k)$ ← degree of freedom

$Y \sim t(3)$



To accommodate the occurrence of values far away from mean

Small sample size approximation of a normal distribution

Certain characteristics → Sufficient data ⇒ Normal distribution

Certain characteristics + ~~Sufficient~~ data = Student's T distribution

If $k > 2$

$E(Y) = \mu$

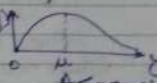
$Var(Y) = \frac{s^2 \cdot k}{k-2}$

- used for hypothesis testing with limited data
- CDF table (T-table)

Chi squared distribution

$X^2(k)$ $Y \sim X^2(3)$ ← degree of freedom

→ Statistical analysis
- ↳ Hypothesis testing
- ↳ Computing confidence intervals



↑ table of known values: N, T

$E(X) = k$

$Var(X) = 2k$

Exponential Distribution

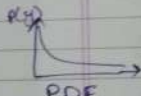
$Exp(\lambda)$ $X \sim Exp(\frac{1}{2})$

→ Start off high
- → Initially ↓
- → Eventually plateaus
- → Aggregate amount of views keep increasing
- → No. of new ones diminishes

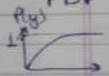
λ → Rate parameter.

$E(Y) = 1/\lambda$

$Var(Y) = 1/\lambda^2$



→ Most likely for a random viewer to have accurate close to the video's initial release



$Y \sim Exp(\lambda)$
$X = \ln(Y)$
$\Big\} \; X \sim N(\mu, \sigma^2)$

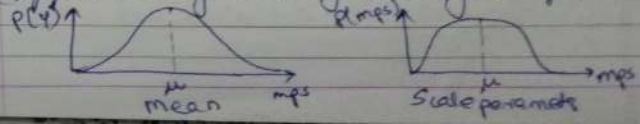
One of the most common transformation

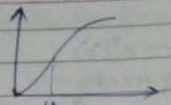
Logistic Distribution

$Logistic(\mu, s)$ ← location, ← Scale parameter

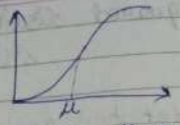
$Y \sim Logistic(6, 3)$

Application - forecasting victory or defeat





Once we reach values near the mean, the prob. drastically goes up



The later the graph start to pick up, but the quicker it reaches values close to 1.

$$E(Y) = \mu$$

$$Var(Y) = \frac{s^2 \pi^2}{3}$$

Excell

Histogram Tower/bar of histogram → bin

1. Select column
2. Insert → Statistical chart → Histogram
   - Bin width → X axis

Charts/Scatter plot

Select column → Insert → chart → Scatter plot
- exponential ← Trendline ← +

Sum → new column

= B2● → = B2+C1

Head Project manager responsibilities

1. Development of the game
   a) Balanced
   b) Realistic
2. Supervise the release
   a) Social media marketing
   b) Competitor analysis
3. Customer analysis.

Probability in Finance

↳ Option pricing

Option – An agreement b/w two parties for the price of a stock or item at a future point in time

↳ Allows one side to decide

↳ How much we are willing to pay to receive that pact? (Highest premium

Expected payoff

$E(P) < 0$ Disadvantageous (Avoid buying this option)

$E(P) = 0$ Fair Deal

You expect to make as much as you paid

$E(P) > 0$ Favourable deal

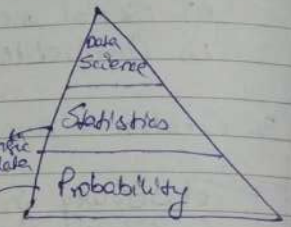
Go through the deal

Pricing an option → $E(P) \leq 0$ → unfavourable / fair deal

Probability with statistics

Statistics
- based on sample of data
- Uncertainty
  - Expected value
  - Prediction intervals

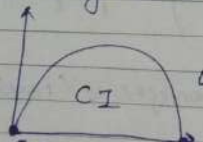
Characteristics
- based on population



analyses numeric & categorical data

Mean, variance & expected value

Experimental data
- Many useful concepts based on probability theory.
- Expresses the likelihood of the population mean being within that interval



confidential interval → To calculate CI we must know mean, variance & std. deviation

90%, 95%, 99%.

Approximate some margins for the mean of the entire population

* Stats used for hypothesis testing

Hypothesis → an idea that can be tested.

3 crucial requirements
- Mean, variance & type of distribution
- validate similar stmts to a specific degree of certainty

Distribution → Shape, Characteristics } Type
- e.g. σ^2

* Any distribution predicts a value for all points within our datasets.
* Distribution anticipates the actual data points

Mathematical modelling (Supervised ML)
- extension of stats that data scientists deal with

Probability in Data Science

Data Analysis → Analyse past data
- → Find insights
- → Make reasonable predictions

Mathematical modelling → we run artificial simulations to see how well our predictions match up to various possible future outcomes.
- → Monte Carlo simulation
  - → we generate artificial data to test the predictive power of our mathematical models
    - → Not completely random data
    - → follows restrictions
  - → We can tweak restrictions
  - → Many models look well when we train them
  - → Perform poorly with new data

Expected values
- ML → An extremely fast paced trial & error process
- The more predictions it makes, the more precise they become.

Page No.
Date: / /20

Forcasting
↳ ML & DL have very high predictive powers → not 100% certain
→ Even data scientists assign probabilities to their predictions

Data Science
An expansion of probability, statistics & programming that implements computational technology to solve more advanced questions

Statistics

Page No.
Date: / /201

- Population
  - collection of all items of interest
  - N → parameters
- Sample
  - subset of population
  - n → statistics
  - less time consuming
  - less costly
  - easy to observe
  - easy to contact
  - Representative
    - accurately reflect members of entire population
  - Random
    - chosen strictly by chance

Descriptive statistics

Types of data

- Categorical or Group
  - car brand
  - yes/No
- Numerical
  - Discrete
    - finite
    - eg grades, no. of objects
  - continuous
    - infinite
    - weight, height, area, distance, time

Measurement levels

- Qualitative
  - Nominal
    - categories
  - Ordinal
    - ●●●○○
    - order
- Quantitative
  - Interval
    - doesn't have true 0
  - Ratio
    - has a true zero

Categorical data

- Frequency distribution tables
- Bar charts
- Pie charts → Market shares
- Pareto diagram

Frequency distribution table

- Relative frequency - % of total freq. for each category

Pareto diagram

- ↳ special type of bar chart, category shown in descending order of freq.

cummulative freq - sum of relative freq.

* Pareto diagram 80/20 rule
  ↳ shows how subtotals change with each additional category to provide us with a better understanding of our data.

Excel (Take a tour

Lookups & Functions

* Add (i) =SUM(D4:D7)
  ii) "Alt =" then enter to select all values
  iii) =SUMIF(D11:D15,">50") enter
  ↳ add only if no other so
  iv) =SUM(D48, G48:G51, 100)
  v) =Sumif(D73:D77, ">50")

Excel → Data → from other sources

Home → auto sum

* Fill → select cursor to drag down/up/left/right.
  same to copy same item in rows

* Split
  Ctrl E → Flash Fill.
  Home → Fill → Flash fill.

* Based on delimiter.
  Select cells → Data → Text to col
  Next ← Tab/Comma ← Next ← Delimited

* Split a columns with formulas.
  = LEFT(C5, FIND(",", C5) - 1)

* Transpose Rows ↔ Columns
  Ctrl C → Ctrl + Alt + V
  Ctrl C → select cell → Home → Paste special, transpose
  • with formula
  = Transpose (C33:H34)

* Sort & filter with ease
Right click
select column > Home > Sort & Filter → sort A to Z
→ Largest to smallest
→ Custom filter.

• Filter → select all → Home → Sort & filter
Column ← Filter
Select ← filter
Ok ← Clear filter
number filter.

* Table
Select All value → Insert → Table

Total rows in tables
Select any cell → Table design → Total row
or Ctrl + Shift + T

* Drop downs
Select cell → Data → Data Validation
Ok ← List

Select cell → Ctrl Q → quickly make a chart

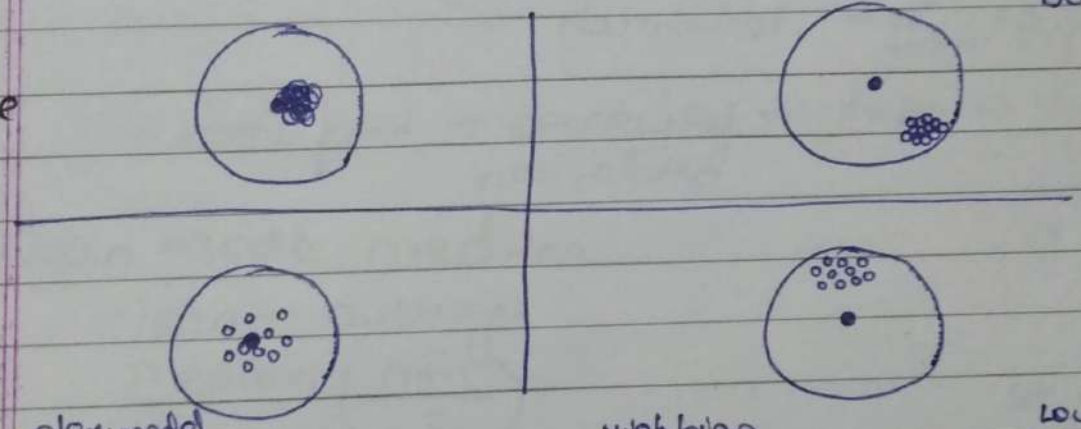
Bias-Variance Tradeoff

Bias → error b/w avg model prediction & ground truth
bias = $E[f'(x)] - f(x)$
→ tells us the capacity of underlying model to predict the values.

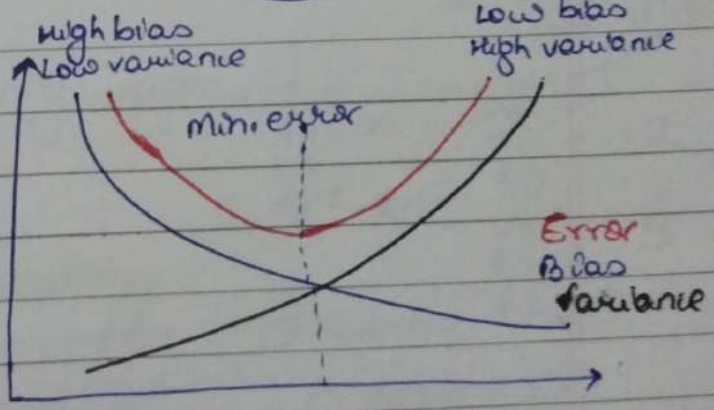
Variance → tells how much the func^n can adjust for the change in the data set.
Variance = $E[(f'(x) - E[f'(x)])^2]$

Low Bias

High Bias → over simplified model
→ Under fitting
→ high error on both test & train data.



Overly complex model
↳ overfitting
↳ Low error on train data
& high on test
↳ starts modelling the noise in the input



Relative freq = $\frac{\text{Freq}}{\text{Total freq}}$

Histogram - unequal interval

Outliers → go against the logic of whole dataset

Variables → Categorical → cross table, side by side bar chart

→ Numerical → Scatterplot

Scatter plot → somewhat vertical → two variables not related

Mean mode median.

Mean - average

Median $\left(\frac{n+1}{2}\right)$

Mode - frequently appearing

54500 7
+5
54615 30

49
51 12
53
54
55 5
62

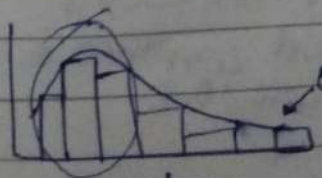
Measure of central tendency

Measures of asymmetry - skewness

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar{x})^2}^3}$$

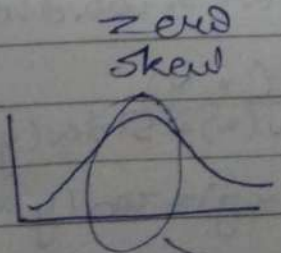
Skewness → checks whether the data is symmetric or asymmetric

mean > median



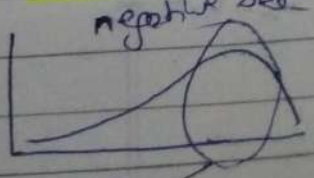
+ve skew.

mean = median = mode



zero skew

mean < median



negative skew

where the data is situated

Variability

Data → Sample data → $\frac{\sum_{i=1}^{n} x_i}{n}$ ← size of sample
→ Population data → $\frac{\sum_{i=1}^{N} x_i}{N}$ ← size of pop

Variance → dispersion around mean

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

→ (n-1) for sample variance

=Var.S
=var.p

Standard Var
deviation → variability / dispersion of data

Coeff. of var / Relative std deviation → compare 2 or more

$$\frac{\text{Standard deviation}}{\text{mean}}$$

Covariance - How two var are correlated
→ indicates the rel. of 2 var whenever one var changes

$$S_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$ ← sample

→ >0 two var move together
<0 " " move opposite
=0 " " are independent

Correlation - describe degree to which two var. move in coordination with one another

$-1 \leq C \leq 1$

$\frac{cov(x,y)}{stdev(x) * stdev(y)}$ Disregard correlation between 0 & 2

corr(x,y) = corr(y,x)

Titanic

What sorts of people were more likely to survive?
→ train.csv → test.csv
418 row entries

Solution → columns
PassengerId, Survived.

$\frac{2^2 + 2^2}{2} = 4$

causality - direction of causal relationship

Correlation does not ~~to~~ imply causation

Age - continuous variable

Level of measurement → Nominal, Interval

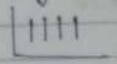
Data → Categorical, Numerical

Inferential Statistics Fundamentals.

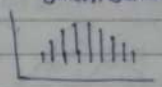
→ Probability
→ Distribution theory

Distribution - providing prob. of occurrence of diff. possible outcomes in an exp.
↳ How values are positioned in relation to each other

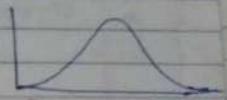
Uniform distn



Binomial distribution



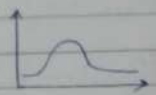
Normal distribut

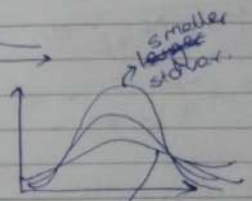


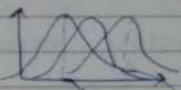
Gaussian distribution & Bell curve.
used in regression analysis

$N \sim (\mu, \sigma^2)$
mean var.

Student's T distr.







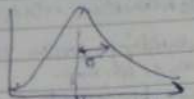
$Z = \frac{x - \mu}{\sigma}$ $N \sim (0,1)$
Origin

Standardization

- compare differently normally distributed datasets
- detect normality
- detect outliers
- create confidence intervals
- test hypotheses
- perform regression analysis

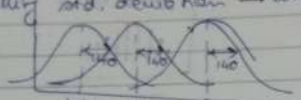
Types of Data
- Categorical
- Numerical
  - continuous
  - Discrete

Normal Distribution
Gaussian distribution - a bell curve

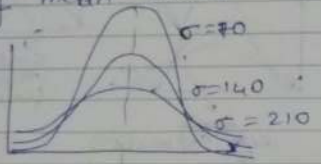


$N \sim (\mu, \sigma^2)$ → variance
↑ Distribution, mean

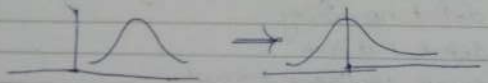
- centrality std. deviation → constant



Centrality mean → constant



- Standardization



$\sim N(\mu, \sigma^2) \longrightarrow \sim N(0,1)$

$z = \frac{x - \mu}{\sigma}$

Central limit theorem
Original distribution $(\mu\ \sigma^2)$ → Sampling distribution $\left(\mu, \frac{\sigma^2}{n}\right)$, $n > 30$

- CLT allows to → perform tests, solve problems, makes inference } using Normal distribution
  even when the population is not normally distributed

Standard error - The std deviation of distribution formed by the sample means

$= \sqrt{\frac{\sigma^2}{n}} = \frac{\sigma}{\sqrt{n}}$

→ shows variability of sample means
→ it shows how well you approximated the true mean

Std error ↑ Sample size ↓
∩

Estimates → points, confidence intervals
are like judges

most efficient estimator with smallest var → Unbiased estimator
expected value = population parameter

confidence level $1-\alpha$

[Point estimate − Reliability factor * Std error , Point estimate + rel. factor * std error]

$\left[\bar{x} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} , \bar{x} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right]$ ← table

Confidence interval

90% → $\alpha = 10\%$ or 0.10

$\alpha/2 = 0.05$

$1 - \alpha/2$

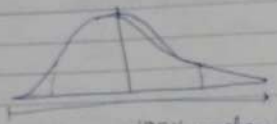
0.95

Z value table

When $1-\alpha$ is lower CI is smaller interval

" $1-\alpha$ is higher, CI is longer

95% → accepted norm

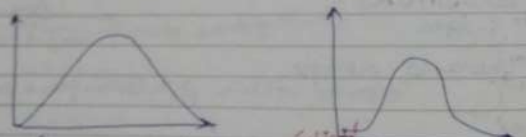


100% makes sense it comprises entire area

Student's T distribution

↳ inference through small sample

↳ unknown population variance



Normal distribution

Student's T distribution

$$t_{n-1,\alpha} = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

Degree of freedom = $n-1$

n → sample size

For big enough samples → Student's T distribution coincides with normal distr.

$$\bar{x} \pm t_{n-1,\alpha/2} \frac{s}{\sqrt{n}}$$ (s: std dev.) Population var. unknown

Margin of error ME

known: $\bar{x} \pm Z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$

unknown: $\bar{x} \pm t_{n-1,\alpha/2} \frac{s}{\sqrt{n}}$

Confidence interval : $\bar{x} \pm ME$

biggest ME → widest CI

smaller ME → narrower CI

↑ statistics ↑ ME

↑ std dev ↑ ME

↑ Confidence ME ↓

Samples

Dependent

- Before & after situation.
- cause & effect

Pooled variance formula.

$$S_p^2 = \frac{(n_x - 1)S_x^2 + (n_y - 1)S_y^2}{n_x + n_y - 2}$$

$$(\bar{x} - \bar{y}) \pm t_{n_x + n_y - 2,\alpha/2} \sqrt{\frac{S_p^2}{n_x} + \frac{S_p^2}{n_y}}$$

no. of variable.

Independent

- Population variance – known
- Population variance – unknown but assumed to be equal
- Population variance unknown but assumed to be diff.

$$CI = (\bar{x} - \bar{y}) \pm Z_{\alpha/2} \sqrt{\frac{\sigma_x^2}{n_x} + \frac{\sigma_y^2}{n_y}}$$

$$(\bar{x} - \bar{y}) \pm t_{v,\alpha/2} \sqrt{\frac{S_x^2}{n_x} + \frac{S_y^2}{n_y}}$$

$$v = \frac{\left(\frac{S_x^2}{n_x} + \frac{S_y^2}{n_y}\right)^2}{\left(\frac{S_x^2}{n_x}\right)^2/(n_x - 1) + \left(\frac{S_y^2}{n_y}\right)^2/(n_y - 1)}$$

Hypothesis Testing

→ idea that can be tested

Steps in data driven decision making

- Formulate a hypothesis
- Find the right test
- Execute the test
- Make a decision

hypothesis
→ one try to reject

Null hypothesis H_0 - one to be tested

Alternative hypothesis - other than null hypothesis

Significance level

α - prob. of rejecting the null hypothesis, if it is true 0.01, 0.05, 0.1

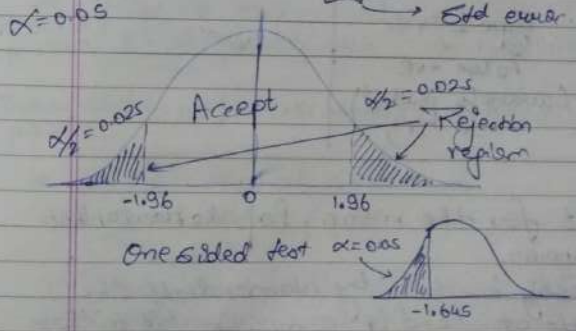
Z - test

$$z = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

$\bar{x}$ ← sample mean

μ ← hypothesized mean

$s/\sqrt{n}$ → Std error



Types of error.

Type 1 error.
↳ Reject a true null hypothesis
↳ False positive

Type II error
↳ accept a false null hypothesis
↳ False -ve

Rejecting a false null hypothesis $P = 1 - \beta$

Eg. H_0 - She doesn't like you.

	The truth	
	H_0: True She doesn't like you	H_0: False She likes you
Accept (H_0 not rej)	✓	Type 2 error (False -ve) missed your chance
Reject (Invite her)	Type 1 error False +ve (wrongly invited her)	✓

Test for the mean, Population variance is known.

Testing is done by standardizing the variable at hand & comparing it to the z

$$Z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}} \qquad Z \sim N(\bar{x} - \mu_0, 1) \\ \sim N(0, 1)$$

p-value universal concept works with every distribution
Smallest level of significance at which we can still reject the null hypothesis, given the observed sample statistic

Rule You should reject the null hypothesis if

$$p\text{-value} < \alpha$$

Test at 90% : $0.0001 < 0.1$
Test at 95% : $0.0001 < 0.05$
Test at 99% : $0.0001 < 0.01$
} Reject null hypothesis

Rule You should reject the null hypothesis if

$$p\text{-value} < \alpha$$

Test at 90% :
Test at 95% :
} Reject.

Test at 99% } Cannot reject

How to find p-value manually p-value online calculator.

One sided p-value: 1 - no. from the table $\Rightarrow 1 - 0.983 = 0.017$

Two sided p-value: (1 - no. from the table) $\times 2 \Rightarrow (1 - 0.983) \times 2 = 0.034$

Where & How p-values used.
→ Most statistical software calculates p-values for each test.
→ Researcher decide significance post-factum
→ p-values are usually found with 3 digits after the dot x.xxx
→ The closer to 0.000 the p-value, the better significance result
* p-value lower than level of sign - you reject null hypothesis

Test for the mean, Population variance unknown

Like confidence intervals with var. unknown & a small sample, the correct steps to use is the t-statistics

$$T = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

Decision rule

Accept if : The absolute value of the T-score < critical value

Reject if : " " " T-score > critical value

Accept if - p-value > α

Reject if p-value < α

Test for the means dependent samples.

$H_0 : \underbrace{\mu_B - \mu_A}_{D_0} \geq 0$

$H_0 : D_0 \geq 0$

$H_1 : D_0 < 0$

$$T = \frac{\bar{d} - \mu_0}{\text{st. err}}$$

($\bar{d}$ → mean diff; μ_0 → D_0; st. err → or pooled std.)

Test for the mean, independent sample known variance

Z-stats { Big sample, known variance → can also be used with unknown var. }

t-stats { Small sample, Unknown var. }

$$Z = \frac{\bar{x} - \mu_0}{\sqrt{\frac{\sigma_e^2}{n_e} + \frac{\sigma_m^2}{n_m}}}$$

Test for the mean, independent samples unknown variances pooled in variance

pooled var $s_p^2 = \frac{(n_x - 1)\, s_x^2 + (n_y - 1)\, s_y^2}{n_x + n_y - 2}$

pooled std dev $\sqrt{\frac{s_p^2}{n_x} + \frac{s_p^2}{n_y}}$

Rule of thumb: Reject the null hypothesis when the T-score is bigger than 2

Generally for Z & T, a value higher than 4 is extremely significant

Steps

- Find sample var
- State null Hypothesis
- Calculated pooled var
- Find t score
- Find p-value
- Interpret result.

$$T = \frac{(\bar{x} - \bar{y}) - (\mu_m - \mu_f)}{\sqrt{\frac{s_p^2}{n_m} + \frac{s_p^2}{n_f}}}$$

Once we have surpassed 50 degree of freedom the student's T ≈ Normal distribution

Part of AD. stats.

Overfitting & Underfitting

* Underfitting – The model has not captured the underlying logic of the data
low train accuracy, low test accuracy

* Overfitting
Our trainy has focused on particular trainy set so much it has 'missed the point'
High train accuracy, low test accuracy

* Good model
captures the underlying logic of the dataset
High train } accuracy
High test }

1. We will use our model to make predictions based on the test data.
2. We will compare those with the actual outcome
3. Calculate the accuracy
4. Create a confusion matrix.

Misclassification rate = # misclassified / # all elements

We use map func° Yes → 1 No → 0.

Confusion matrix

```
def confusion_matrix(data, actual_values, model):
    pred_values = model.predict(data)
    bins = np.array([0, 0.5, 1])
    cm = np.histogram2d(actual_values, pred_values, bins=bins)[0]
    accuracy = (cm[0,0] + cm[1,1])/cm.sum()
    return accuracy, cm
```

Cluster Analysis

The goal of clustering is to maximize the simi

Programming

→ Problem solving & abstract thinking
→ Mechanistic thinking
→ Good style of coding

Programming involves → formulating problems
→ Breaking them down into meaningful steps
→ Communicate these steps to computer

Python → technical advantage
case sensitive → practical application
→ open-source - free
→ General-purpose - interoperability with C, Java, R
→ High level lang.

Jupyter → facilitates communication tremendously

Ctrl + Enter - code execute
Shift + Enter Executes continue
Cut → Select cell → Press X
Paste → " → Press V
Copy → " → Press C
Cell above a cell → " → Press A
Cell below → " → Press B
Delete → " → Press D twice

Markdown cell documentation
↳ Press M
Markdown → code cell
Press Y

Variable

(a,b) = (1,2)

type(-6) → int

String

'George'

y = 10

print(str(y) + 'Dollar')

'I \'m fine'

↳ escape

16/3 → 5

5**3 → 125

float(16)/3 → 5.33

16.0/3 → 5.33

16%3 → 1

y = 125

y == 125 True

y == 126 False

Comment # Comment

Line continuation This is \ Rahul

'Friday'[4] → d

Logical operator → and, or, not Precedence Not > And > Or

Identity operator is, is not

function

def function_name(parameters):
    function body

Return can be used only once in a func°

Print → does not affect calculation of o/p

Return → does not visualize the o/p it specify what a certain func° is suppo to give back

Built in function

int(5.0) → 5

max(10,20,30) → 30 min(10,20,30) → 10

- abs() - allows you to obtain the absolute value of its argument

z = -20 abs(z) → 20

- sum() → sum of all elements
- round(x,y) → returns float of its argument (x) rounded to a specified no. of digit(y) after decimal.

2**10 → 1024 len() ← length

pow(2,10) → 1024

Methods obj name.ap method()

- .append()
- .extend()
- .index()
- .sort() → reverse=True (opposite)

- List slicing list_name[5:0] (from ← to position in list)

BiggerList = [list_1, list_2]

- Tuple - immutable (, ,)
  ↳ .split()

Dictionary
dict = {'k1': 'cat', 'k2': 'dog'}

eg Team = { }
Team['T1'] = 'Ram'
Team['T2'] = 'Shyam'

Team.get(T2) → Shyam.

# Iterable-
even = [0, 2, 4, 6, 8, 10, 20]
for n in even:

Range (start, stop, step)
↳ last + 1

step > start > stop

list (range(10)) → [1, 2, 3, 4, ---- 10]

Advance Statistical methods

* OOPs

Object = Data + Manipulation operations
Instance → float, integers, string, lists

Class defines the rules for creating that object
↓
Object → attributes/properties
→ methods

Function	Method
can have many parameters	the object is one of its parameters
exists on its own	belongs to a certain class
function()	object.method()

Package - directory of a collection of modules

Example: from math import *
from math import sqrt

For more info type → help(math)

Regression analysis cause → effect
Factor analysis.

Linear regression with stats model

linear approximation of a causal relationship b/w two or more variable.

$y = \beta_0 + \beta_1 x_1 + \varepsilon$ → error

- y → Dependant variable
- β_0 → constt value irrespective of x
- x_1 → independant variable
- β_1 → quantify the effect

$\hat{y} = b_0 + b_1 x_1$

$\hat{y}$ → estimated/predicted value

Correlation	Regression
Relationship	one variable affects the other
Movement together	cause & effect
$p(x,y) = p(y,x)$	one way
Single point	line.

* Regression line → best fitting line through the data point

- Package → import numpy - work with multidimensional array
  import pandas - allow us to organise data in table form & attach descriptive label to rows & col
  import scipy - scientific calculation - matlab, ML,
  import statsmodels.api - very good summaries
  import matplotlib - visualization of numpy comp.
  import seaborn - based on matplotlib & add attractive statistical graphics → better visualisation
  * import sklearn → ML libraries.

```
data = pd.read_csv('Filename.csv')
        ← pandas.
data.describe - features of data

plt.scatter(x1,y)
plt.xlabel('SAT', fontsize=20)
plt.ylabel('GPA', fontsize=20)
plt.show()
```

OLS - ordinary least squares regression

fit() - specific estimation to obtain the fit of the model

Regression
```
x = sm.add_constt(x1)
result = sm.OLS(y,x).fit()
result.summary()
```

OLS Regression →
- Model summary
- Coeff. table → values of b_0, b_1
- Some additional test.

Decomposition of variability

Anova framework. →

Sum of square total SST
$\sum_i (y_i - \bar{y})^2$ SST, TSS
dispersion of the observed variable around mean
- measure total variability of dataset

Sum of square regression SSR
$\sum_i (\hat{y}_i - \bar{y})^2$ ESS
- measures the exp variability by your line
→ measure how well our line fits the data

Sum of square error.
$\sum_{i=1}^{n} e_i^2$ SSE, RSS
measures the unexplained variability by the regression

$SST = SSR + SSE$

$\sum (y_i - \bar{y})^2 = \sum (\hat{y}_i - \bar{y})^2 + \sum e_i^2$
Total variability — explained var. — unexplained var.

Ordinary least squares → min SSE

Lower error ⇒ better explanatory power

↳ depict the closest to all points

$$S(b) = \sum_i (y_i - x_i^T b)^2 = (y - Xb)^T (y - Xb)$$

↳ OLS estimator of β for a simple linear reg.

Other method

↳ Generalized least sq.

↳ Max. likelihood estm

↳ Bayesian regression

↳ Kernel regression

↳ Gaussian process regression

R Squared

measure goodness of fit

$$R^2 = \frac{SSR}{SST}$$

0 ←——→ 1

0: your reg. explains none of the variability

1: your reg. explains the entire variability

It measures how much of the total variability is explained by our model

Steps linear regression

→ Import relevant libraries

Load data

Create regression

↳ Declare independent & dependent variable

Multiple linear regression

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon$$

Y: inferred value; β_0: intercept

⇓

Adjusted R^2 variable

↳ penalizes excessive use of variables

→ Basis for comparing models → same dependent var; same dataset

→ usually smaller than R^2

x = data [['SAT', 'year']]

When a new parameter is added if it increases R-squared but ↓ adjusted R-squared → variable can be omitted since it holds no predictive power

F statistics testing the overall significance of the model

$H_0: \beta_1 = \beta_2 = \beta_3 \cdots = \beta_k = 0$

H_1: at least one $\beta_i \neq 0$

if all betas are 0, then none of X_s matter & our model has no merit

* Lower the f-statistics, the closer to a non significant model

Larger f value → significant

Small p value

OLS assumption

(i) Linearity $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon$

(ii) No endogeneity $\sigma_{x\varepsilon} = 0 : \forall\, x, \varepsilon$

(iii) Normality & Homoscedasticity $\varepsilon \sim N(0, \sigma^2)$

(iv) No autocorrelation $\sigma_{\varepsilon_i \varepsilon_j} = 0 : \forall\, i \neq j$

(v) No multicollinearity $\rho_{x_i x_j} \neq 1 : \forall\, i, j \; ; \; i \neq j$

- Linearity $y = \beta_0 + \beta_1 x_1 + \cdots \beta_k x_k + \varepsilon$

if straight line linear regression is suitable.

← linear reg not appropriate.

Fix → run non linear reg.
- → exp. transformation
- → log transformation

- No endogeneity

$\sigma_{x\varepsilon} = 0 : \forall\, x, \varepsilon$

error is correlated with independent variable

Omitted variable bias → easiest way to detect is through the error term
- → you forget to include a relevant variable
- → Always diff
- → Always sneaky
- → only exp & advanced knowledge can help

- Normality & Homoscedasticity

$\varepsilon \sim N(0, \sigma^2)$ → zero mean

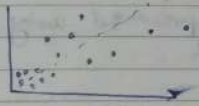
Normality t test & F-test work because we have assumed normality

Central limit theorem.

$\varepsilon \sim N(0, \sigma^2)$

Homoscedasticity $\sigma^2_{\varepsilon_1} = \sigma^2_{\varepsilon_2} = \cdots = \sigma^2_{\varepsilon_k} = \sigma^2$

→ to have equal variance



Prevention
- → Look for omitted variable bias
- → Look for outliers
- → transform - Log transformation
  - → take log of the variable
  - → perform regression

Semi log model $\hat{y} = b_0 + b_1 (\log x_1)$

or $\log \hat{y} = b_0 + b_1 x_1$

Log - log model $\log \hat{y} = b_0 + b_1 (\log x_1)$

elasticity

- No autocorrelation — no serial correlation

$\sigma_{\varepsilon_i \varepsilon_j} = 0 : \forall\, i \neq j$

autocorrelation - not observed in cross-sectional data.

Day of the week effect → high returns on Friday, low returns on mondays

Detection
- * If there are no patterns to be seen → no autocorrelation
- Durbin - watson falls b/w 0 & 4
  - 2 → no autocorrelation
  - < 1 & > 3 → cause an alarm.

No remedy → when in the presence of autocorrelation avoid the linear regression model.

Alternatives
- → Auto regressive model
- → moving avg. model
- → Auto reg. moving avg. model
- Auto reg. integrated moving avg. model

- No multicollinearity - easy to spot & easy to fix

$r_{x_i x_j} \neq 1 : \forall i,j \; i \neq j$

eg $a = 2 + 5b \qquad b = \frac{a-2}{5}$

$r_{ab} = 1 \rightarrow$ perfect multicollinearity

If a can be represented using b, there's no point in using both.

Fixes
- Drop one of the two variable
- Transform them into one
- keep them both

Prevention
↳ find the correlation b/w each two pairs of independent variables

$r_{x_i x_j}$ for $\forall i,j \; i \neq j$

- Dummy variable
  ↳ innitiation of categories with no.

SAT → GPA

Attendance

dummy var

```
data['Attendance'] = data['Attendance'].map({'Yes':1, 'No':0})
```

- Making prediction - linear reg.

Adding new data
```
new_data = pd.DataFrame({'const': 1, 'SAT': [1700, 1670], 'Attendance': [0,1]})
new_data = new_data[['const', 'SAT', 'Attendance']]
new_data
```

change name of index
```
new_data.rename(index={0:'Bob', 1:'Alice'})
```

predict
```
predictions = results.predict(new_data)
```
Predicts

Linear regression with Sklearn

Sklearn → Scikit-learn
- fast & efficient
- prefer working with array

Numpy, scipy, matplotlib

pandas dataframe → Statsmodel (great for learning)

Numpy array → Scikit learn (is preferred)

Advantages
- Incredible documentation
- Variety → Regression, Classification, Clustering, Support vector m/c, Dimensionality red.

Import from sklearn.linear_model import LinearRegression

GPA ← SAT

dependent var. — independent var

O/P target — I/P feature

Regression, itself
```
reg = LinearRegression()
reg.fit(x, y)
```
x → x_matrix. I/P; y → target

Reshaping matrix in 2D
```
x_matrix = x.values.reshape(-1,1)
```

Calculate the R-squared
```
reg.score(x_matrix, y)
```

L_1 norm — L_2-norm

least absolute deviations (LAD) / least squares / least absolute error (LAE)

$$S = \sum_{i=1}^{n} |y_i - f(x_i)| \qquad S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

find intercept { reg.intercept_

find the coefficient { reg.coef_

make predictions { reg.predict()

Rsquared slightly larger than adj R^2

$R^2 > R^2_{adj}$ → not penalizes a lot for the inclusion of 2 independent variables

⇒ Multiple Linear regression – Sklearn

$$R^2_{adj} = 1 - (1-R^2) * \frac{n-1}{n-p-1}$$

n → no. of observations

p → no. of predictors

```
def adj_r2(x,y)
    r2 = reg.score(x,y)
    n = x.shape[0]
    p = x.shape[1]
    adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
    return adjusted_r2
```

⇒ Feature selection

simplifies models, improves speed & prevents a series of unwanted issues arising from having too many features

If a variable has a p-value > 0.05 we can disregard it.

feature_selection.f_regression

→ creates simple linear regression of each feature & the dependent variable

* Note that for a simple linear regression the p-value of F-stat ≡ the p-value of the only independent variable

Feature selection.

from sklearn.feature_selection import f_regression

Calculate univariate p-values of the var

```
f_regression(x, y)
array(  ,  ), array(  ,  )
p_values = f_regression(x,y)[1]
p_values
p_values.round(3)
```

p-values

* Create summary table

```
reg_summary = pd.DataFrame(data = x.columns.values,
                           columns = ['Features'])
reg_summary['Coefficients'] = reg.coef_
reg_summary['p-values'] = p_values.round(3)
reg_summary
```

- Feature scaling / Standardization

transforming data into a standard scale

$$\text{Standardized variable} = \frac{x - \mu}{\sigma}$$

Weights - ML word for coeff.
→ Bigger the wt → Bigger the impact

Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x)

x_scaled = scaler.transform(x)
x_scaled
```

Create summary table

```
reg_summary = pd.DataFrame([['Bias'], ['SAT'], ['Rand 123']], columns=['Features'])
reg_summary['Weights'] = reg.intercept_, reg.coef_[i
                         reg.coef_[1]
```

Lec 221

Make pred with stand. coef.

```
new_data = pd.DataFrame(data=[[1700,2],[1800,1]], columns=['SAT','Rand 123'])
new_data

new_data_scaled = scaler.transform(new_data)
new_data_scaled

reg.predict(new_data_scaled)
```

Overfitting & Underfitting

Overfitting: Our training has focussed on the particular training set so much, it has missed a point
High train accuracy.
Low test accuracy

Underfitting: The model has not captured the underlying logic of the data
Low accuracy

→ We can create the regression on the train data but Test on the test data

Train - Test Split

Import relevant lib.

```
import numpy as np
from sklearn.model_selection import train_test_split
```

Generate data

```
a = np.arange(1,101)
```

np.arange([start,] stop, [step,]) returns evenly spaced value within a given interval.

```
b = np.arange(501, 601)
```

Split the data (splits arrays or matrices into random train and test subsets)

```
a_train, a_test = train_test_split(a, test_size=
shuffle=False, random_state=
```

Shuffle data same type every time

Explore the result

```
a_train.shape, a_test.shape
a_train
a_test
```

⇒ Linear Regression
- Clean data
- Relax assumptions
- Log transformation
- Create a model
- Create dummies

axis 0 → row
axis 1 → columns

Data preprocessing

data.describe(include='all') → categorically defining data

dropping columns { data1 = data.drop(['Model'], axis=1) → remove less important variable

give no. of null values in variable { data1.isnull().sum()

Rule of thumb:- if you are removing <5% of the observations, you are free to just remove all that have missing value

remove the rows that contains NULL value { data_no_mv = data.dropna(axis=0)

Exploring the prob. distr. func° (PDF)

for optimal results your model should be linear, looking for a normal distribution { sns.distplot(data_no_mv['Price'])

dealing with outliers * { q = data_no_mv['Price'].quantile(0.99)

One way to deal with outliers seemingly is to remove top 1% of observations

, data1 = data_no_mv[data_no_mv['Price']<q]

data1.describe(include='all')

resetting the index after cleaning { data_cleaned = data_cleaned.reset_index(drop=True)

* log transformation especially useful when facing exponential relationship

np.log(x) → returns the natural log of a no. or array of no.

log_price = np.log(data_cleaned['Price'])
data_cleaned['log_price'] = log_price

Multicollinearity
↳ one of the best ways to check for multicollinearity is through VIF (variance inflation factor)

→ data_cleaned.columns.values

→ from statsmodels.stats.outliers_influence import variance_inflation_factor
variables = data_cleaned[['Mileage', 'Year', 'EngineV']]
vif = pd.DataFrame()
vif['VIF'] = [variance_inflation_factor(variables.values, i) for i in range(variables.shape[1])]
vif['features'] = variables.columns

vif = 1 = no multicollinearity
1 < vif ≤ 5 = perfectly okay
10 < VIF = unacceptable

# drop features with no multicollinearity
data_no_mc = data_cleaned.drop['Year', axis=1]

Create dummy variables.
pd.get_dummies(df [, drop_first]) - spots all categorical variables & creates dummies automatically.

If we have N categories for a feature we have to create N-1 dummies

Steps
* Importing the relevant libraries
* Loading the raw data
* Preprocessing
  Explore the descriptive statistics of variables
  Determine the variables of interest
  Deal with missing values
  Explore the pdfs
  Deal with outliers
* Checking the OLS assumptions
  Relaxing the assumptions
    Multicollinearity
* Create dummy variables
  Rearrange a bit
* Linear regression model
  Declare the i/p & the targets
  Scale the data
  Train Test split
  Create the regression
  Finding weights & bias
  * Testing

* Scaling has no effect on the predictive power of dummies, once scaled, though, they lose all their dummy meaning

Train test split

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
    train_test_split(inputs_scaled, targets, test_size=
        random_state = 365
```

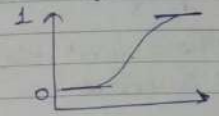
* Residual = Diff b/w the targets & the predictions
  ↳ estimates of errors

Logistic Regression

Categorical — non-linear data

$y = x_0 + b_1 x_1 + b_2$ dummy

Categ 1 → Categ 2



binary predictable.

Assumption → No endogeneity
→ Normality & homoscedasticity
→ No autocorrelation
No multicollinearity

→ Predict prob. of an event occurring.

Logistic model $p(x) = \frac{e^{(B_0 + B_1 x_1 + \cdots B_k x_k)}}{1 + e^{(B_0 + B_1 x_1 + B_2 x_2 + \cdots B_k x_k)}}$ rejected.

odds → $\left(\frac{p(x)}{1-p(x)}\right) = e^{(B_0 + B_1 x_1 + \cdots B_k x_k)}$

Logit model $\log \frac{p(x)}{1-p(x)} = B_0 + B_1 x_1 + \cdots B_k x_k$

Regression

```
x = sm.add_constant(x1)
reg_lgp = sm.Logit(y, x)
result_lgp = reg_lgp.fit() → result_lgp.summary
```

* Current func° value gives value of objective func° at 10th iteration

* Maximum likelihood estimation (MLE)
likelihood function - estimates how likely it is that the model at hand describes the real underlying relationship of the variables.

→ The bigger the likelihood func", the higher the prob. of our model is correct

→ MLE tries to maximize the likelihood func"

* Log likelihood - almost but not always -ve.
bigger log likelihood - higher the probability

* LL Null - (log likelihood - null)
The log-likelihood of a model which has no independent variable.

* LLR (log likelihood ratio)
measures if our model is statistically different from LL null aka a useless model

* Pseudo R-squared McFadden's R-squared.
A good pseudo R-squared is somewhere b/w 0.2 to 0.4.

→ This measure is mostly useful for comparing variables of the same model.

→ Different models will have completely diff. & incomparable Pseudo-R squares.

Logit model
SAT - Admittance regression

$$\log\left(\frac{\pi}{1-\pi}\right) = -69.91 + 0.042 * SAT$$

π → prob. of an event occurring.
$1-\pi$ → prob. of an event not occurring

$$\log\left(\frac{odd_2}{odd_1}\right) = 0.042(SAT_2 - SAT_1)$$ → 100 unit

$\frac{odd_2}{odd_1} = 66.6\%$

10 units

1 Difference of 1 unit of SAT.

$$\log\left(\frac{odd_2}{odd_1}\right) = 0.042 \qquad \frac{odd_2}{odd_1} = 152\%$$

$$\frac{odd_2}{odd_1} = 1.042$$

When the SAT score increases by 1, the odds of admittance ↑ by 4.2%.

$$\Delta odds = e^{b_k}$$

Dummies → linear regression
Binary predictors → logistic regression

⇒ Accuracy of the model
sm.LogitResults.predict() → returns the values predicted by our model

np.set_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})

results_log.predict()

sm.LogitResults.pred_table() → returns a table where compares predicted to actual values.

RGB - 16 777 216 colors

confusion matrix

```
cm_df = pd.DataFrame(results_log.pred[1])
cm_df.columns = ['Predicted 0', 'Predicted 1']
cm_df = cm_df.rename(index = {0: 'Actual 0', 1: 'Actual 1'})
cm_df

cm = np.array(cm_df)
accuracy_train = (cm[0,0] + cm[1,1]/cm.sum()
accuracy_train
```

Underfitting & Overfitting → After inferential statistics fundamentals

Cluster Analysis → unsupervised learning

The goal of clustering is to maximize the similarity of observations within a cluster & maximize the dissimilarity b/w clusters

eg → Market segmentation
eg → Image segmentation

Explore the data → identify patterns

supervised learning → linear & logistic regression
→ Classification predicting an o/p category
→ labelled data

Maths prerequisites → distance b/w 2 data points → euclidean distance $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$
→ centroids → mean position of a group of points

k-mean clustering
↳ no. of clusters.

1) Choose the no. of clusters
2) Specify the cluster seeds
3) Assign each point to a centroid
4) Adjust the centroids.

importing libraries

```
from sklearn.cluster import kMeans
```

Steps

Import libraries
Load the data
Plot the data
Select the features → dataframe.iloc(row indices, col. indices) gives the dataframe, given features to be kept
Standardize
Clustering
Clustering results
wcss
Elbow method.

no of clusters
kmeans = kmeans()
kmean.fit()

Sklearn.cluster.kmeans.fit_predict(x)
returns the cluster predictions in an array

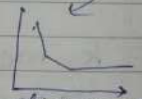
How to choose no. of clusters

Elbow method

within-cluster sum of squares
or wcss

observation = N	N	N
cluster = N	1	small
wcss = 0 (min)	max	low

wcss → kmeans.inertia_



elbow method.

```
wcss = []
for i in range(1,7):
    kmeans = kmeans(i)
    kmeans.fit(x)
    wcss_iter = kmeans.inertia_
    wcss.append(wcss_iter)
wcss
```

Elbow method

```
number_clusters = range(1,7)
plt.plot(number_clusters, wcss)
plt.title('The elbow method')
plt.xlabel('No. of clusters')
plt.ylabel('within cluster Sum of Squares')
```

Pros of k-mean
- Simple to understand
- Fast to cluster
- Widely available
- Easy to implement
- Always yield a result (although coming to decency)

Cons	Remedy
• We need to pick k	• The elbow method
• Sensitive to initialization	• k-means++
• Sensitive to outliers	• Remove outliers
• Produce spherical sol.	
• Standardization	

Standardization

The ultimate aim of standardization is to reduce the wt of higher no.

Standardization is trying to put all variables on equal footing

Market Segmentation
Brand loyalty → Churn rate
→ Retention rate
→ Customer lifetime value (CLV)

Standardize the variable

```
from sklearn import preprocessing
x_scaled = preprocessing.scale(x)
```

* sklearn.preprocessing.scale(x) -
Standardize with mean 0 & std dev of 1
each variable (column) separately

Types of Analysis

clustering mainly used

- Exploratory
  - get acquainted with the data
  - Search for pattern
  - Plots → Clustering
    - Data visualizable
    - Descriptive
- Confirmatory
  - Explain a phenomenon
  - Confirm a hypothesis
  - Validate prev. research
- Explanatory

Types of clustering

- Flat → k-means
- Hierarchical
  - Agglomerative (Bottom-up)
  - Divisive (Top-down)

Dendrogram
The longest blue line
Shows similarity b/w observations

Pros
- Shows all possible linkage b/w clusters
- understand data much better.
- No need to preset the no. of clusters
- Many methods do perform hierarchical clustering

Cons
- scalability problem

Heatmap

sns.clustermap(x_scaled / data)

Mathematics

Linear Algebra

Matrix. R x c

Scalars have 0 dimensions. [15], [2] [1x1]
(like point)

Vector
(like line) $\begin{bmatrix} 5 \\ -2 \\ 4 \end{bmatrix}$ [m x 1]

Scalar
$s = 5$

Vector
`v = np.array([5,-2,4])`

Matrices `m = np.array([[5,3,6],[-3,0,14]])`

`m.shape` → returns dimension of matrix

Tensor Flow.

- scalar → Tensor of rank 0 → 1x1
- vector → " " " 1 → mx1
- matrix → " " " 2 → mxn

addition $m_1 + m_2$ (matrices)

`v1+np.array([1])`

Transpose `m1.T` (m1: matrix) → transpose

Dot product

$\begin{bmatrix} 2 \\ 8 \\ -4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -7 \\ 3 \end{bmatrix} = [-66]$

`np.dot(x,y)` (x: matrix 1, y: matrix 2)

Matrix multiplication $m \times n$ with $n \times k$

$A_{100 \times 300} \cdot B_{300 \times 3} = C_{100 \times 3}$

- Application of linear algebra in DS
  - ↳ Vectorized code
  - → Image recognition
  - → Dimensionality reduction

Deep learning

Training an algo involves:-
Data → feed → model → compare → Objective Function → Optimization algo → vary

Types of ML → Supervised (classification & Regression) → Training model
→ unsupervised clustering
→ Reinforcement

The linear model.

$$f(x) = x\,w + b$$

x: input, w: wt, b: intercept

Linear model with multiple i/p & multiple o/p

$$y_1 = x_1 w_{11} + x_2 w_{21} + b_1$$
$$y_2 = x_1 w_{12} + x_2 w_{22} + b_2$$

Objective func° → used to evaluate how well the model's o/p match the desired correct values.
- Loss → ↓ loss func ↑ level of accuracy (supervised)
- Reward approach ↑, ↑ (reinforcement)

Loss
- Regression
  - L2-norm = $\Sigma_i (y_i - t_i)^2$
  - → OLS
  - lower the error the lower the loss
- classification → cross entropy
  - $L(y,t) = -\Sigma_i t_i \ln y$

Optimization algo.

Gradient Descent → can find the min by trial & error

$f'(x_i) \longrightarrow \nabla E_x(x_1, x_2, \ldots x_N)$

$x_{i+1} = x_i - \eta f'(x_i)$

η → learning rate

When the min is reached

$x_{i+1} = x_i$ - (no longer update)

Oscillate - repetitive variable around central value

Learning rate to be

High enough so we can reach the closest min. in a rational amount of time

Low enough so we don't oscillate around the min

n - Parameter Gradient descent

$x_i w + b \mp y_i \longrightarrow t_i$

$L(y,t) \longrightarrow$ loss

$C(y,t) \longrightarrow$ cost

$E(y,t) \longrightarrow$ error

$x_{i+1} = x_i - \eta f'(x_i)$

optimization

$\begin{cases} w_{i+1} = w_i - \eta \nabla_w L(y,t) \\ b_{i+1} = b_i - \eta \nabla_b L(y,t) \end{cases}$

$\nabla_w L = \sum_i \nabla_w \frac{1}{2}(y_i - t_i)^2 \longrightarrow x_i w + b$

$= \sum_i x_i \delta_i (y_i - t_i)$

$= \sum_i x_i \delta_i$

Supervised learning → inputs, weights, biases, outputs, targets

$w_{i+1} = w_i$

Simple linear Regression

Import the relevant libraries → Numpy (math, algebra, matplotlib), matplotlib.pyplot (nice graphs)

Generate random i/p data to train on

Create the targets we will aim at

Plot the training data

Create weights

Create biases

Set a learning rate

→ from mpl_toolkits.mplot3d import Axes3D (nice 3D graphs)

2) Generate random i/p data to train on.

np.random.uniform(low, high, size) - draw random value from interval (low, high) where each no. has an equal chance to be selected

Size = n (no. of observations) * k → no. of variables

np.column_stack(appropriate tuples) - takes a seq. of 1D array & stack them into a single 2D array.

3) Create the targets we will aim at.

→ noise

→ target eg $= 2*x_1 + 3*x_2 + 5 + \text{noise}$.

Gameplan for training:
i) calculate outputs
ii) compare o/p to targets (loss)
iii) adjust weights & biases

Train the model

for i in range(100):

$\{y = xw + b\}$ outputs = np.dot(inputs, weights) + biases

$\{(y_i - b_i)$ deltas = outputs - targets

L_2 norm loss formula

$\{(y_i - b_i)^2$ loss = np.sum(deltas**2) / 2 / observations

print(loss)

deltas_scaled = deltas / observations

$\{w_{i+1} = w_i - \eta \sum_i x_i \delta_i\}$ weights = weights - learning_rate * np.dot(inputs.T, deltas_scaled)

$b_{i+1} = b_i - \eta \sum_i \delta_i$ biases = biases - learning_rate * np.sum(deltas_scaled)

* In the memory of computer, the variables; weights, biases & outputs contain their optimized values of those from the last iterations of the loop

Data

Model

Objective func°

Optimization algo

Tensorflow → CPUs, GPUs, TPU
→ for neural n/w

Scikit-learn better for k-mean clustering & random forest

High level packages → PyTorch, Keras

Deep learning library

Tensors n-dimensional arrays

* np.savez(file name, arrays) saves n-dimensional arrays in .npz format, using a certain keyword (label) for each array)

tf.keras.Sequential() → func° that specifies how the model will be laid down ('stack layers')

Linear combination + O/P = layers

tf.keras.layers.Dense(output size) takes the i/p provided to the model & calculates the dot product of the i/p & wt & adds the bias

model.compile('optimizer', loss) → configures the model for training

SGD = Stochastic Gradient Descent

i) L2-norm loss = Least sum of squares (least sum of squared error)

ii) Scaling by # observations = avg (mean)

model.fit(inputs, targets) fit (trains) the model

Epoch = iteration over the full dataset

verbose = 0 → silent → no o/p about training is displayed
verbose = 1 → progress bar
verbose = 2 → one line per epoch

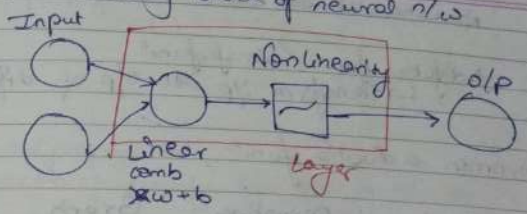
model.layers[0].get_weights() → extract weights

model.predict_on_batch(data) calculates the o/p given i/ps

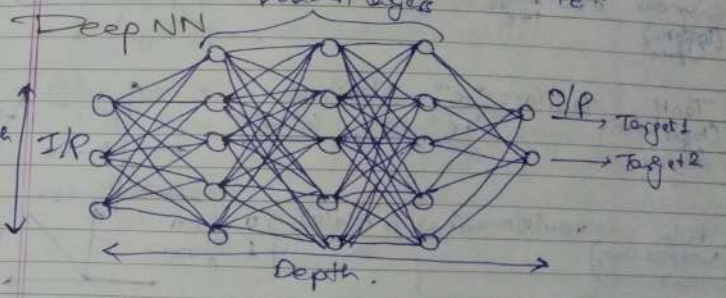
tf.keras.optimizers.SGD(learning_rate)
Stochastic gradient descent optimizer, includes support for learning rate, momentum, decay, etc.

Loss func° → alternative loss for regression is huber loss

- Layer – building block of neural n/w



Non-linear sigmoid $= \sigma(x) = \frac{1}{1+e^{-x}}$



Hyper parameter
Width, preset by us
Depth
Learning rate (η)

Parameter found after optimizing
wt (w)
biases (b)

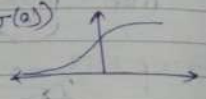
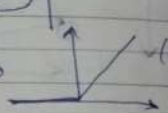
Non linearities
↓
Stacking layers
↓
Depth
↓
Deep learning

Batch size
Momentum coeff (α)
Decay coeff (ϵ)

Non linearities also called as activation func^n

Activation func^n / transfer func^n
↳ transform i/p into o/p of diff. kinds

Common activation func^n

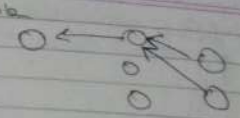
Name	Formula	Derivative	Graph	Range
Sigmoid (logistic) func^n	$\sigma(a) = \frac{1}{1+e^{-a}}$	$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1-\sigma(a))$		(0,1)
Tanh (hyperbolic tangent)	$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$	$\frac{\partial \tanh(a)}{\partial a} = \frac{4}{(e^a + e^{-a})^2}$		(-1,1)
Relu (rectified linear unit)	$relu(a) = \max(0,a)$	$\frac{\partial\, relu(a)}{\partial a} = \begin{cases} 0 & \text{if } x \le 0 \\ 1 & \text{if } x > 0 \end{cases}$		$[0,\infty)$
Softmax	$\sigma_i(a) = \frac{e^{a_i}}{\sum_j e^{a_j}}$	$\frac{\partial \sigma_i(a)}{\partial a(a_j)} = \sigma_i(a)(\delta_{ij} - \sigma_j(a))$	diff. everywhere	(0,1)

* All common activation func^n are monotonic, continuously differentiable

* The softmax transformation transforms a bunch of arbitrarily large or small no. into a valid prob. distribution

* Softmax is often used as activation of o/p layer in classification problems

- Back propagation



- Forward propagation
process of pushing i/p through the net

Overfitting – missed the points
Underfitting – not captured logic

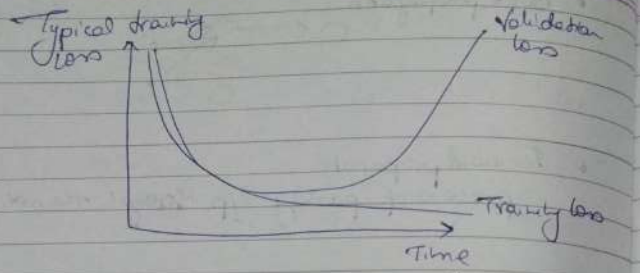
* Bias - variance tradeoff
The balance b/w underfitting & overfitting

Data Set → Training set – where learning happens
(80,10,10) (70,20,10) → Validation set – helps us prevent overfitting
→ Testing set – measure final prediction power of the model

N-fold cross validation → Training set + Validation set
→ Testing set

* We train the model until the loss func^n is minimized.

* Early stopping - prevent overfitting
1. → Preset no. of epochs
2. Stop when updates become too small. $X_{i+1} - X_i = 0.001$
3. Validation set strategy



	Preset epochs	Updates too small	Validation set
Solves the prob	✓	✓	✓
Certain that loss is minimized	✗	✓	✓
Doesn't iterate widely	✗	✓	✗
Prevent overfitting	✗	✗	✓

Best practice

Initialization - set initial values of weights

Xavier - Glorot initialization

Uniform xavier initialization - draw each wt w_i from random uniform distribution in $[-x, x]$

for $x = \sqrt{\frac{6}{i/p + o/p}}$

Normal xavier initialization - draw each wt w_i from a normal dist. with a mean of 0 & std dev

$$\sigma = \sqrt{\frac{2}{i/p + o/p}}$$

default initializer

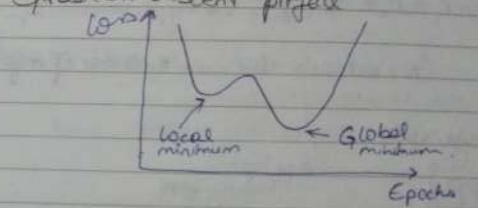
Stochastic Gradient Descent

-> Notably,

We updates the weights after each batch

→ we lossen bit accuracy

Gradient descent pitfall



$$w \leftarrow w(t) - \eta \frac{\partial L}{\partial w}(t) - \alpha \eta \frac{\partial L}{\partial w}(t-1)$$

current update — updates a moment ago

$\alpha = 0.9$

$\eta \rightarrow$ learning rate

→ small enough - gently descend instead of oscillating & diverging

→ Big enough so we reach it in a rational amount of time

Learning schedules

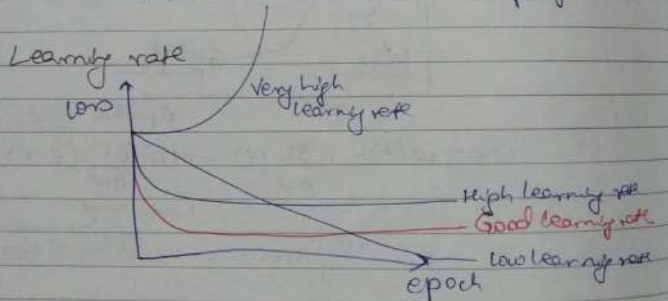
1. We start from a high initial ~~start~~ learning rate — First 5 epochs — $\eta = 0.1$
2. At some points we lower the rate to avoid oscillation — Next 5 epochs — $\eta = 0.01$
3. Around the end we pick a very small rate to get a precise ans — Until the end — $\eta = 0.001$

- exponential schedules

$\eta_0 = 0.1$ → current epoch

$\eta = \eta_0 e^{-t/r}$ ← const → hyperparameter.

No set rule But some order of mgp.



Learning rate schedules → Adagrad adaptive gradient algo.

$$G_i(t) = G_i(t-1) + \left(\frac{\partial L}{\partial w_i}(t)\right)^2$$

with beginning point $G_i(0) = 0$

$$w(t+1) = w(t) - \eta \frac{dL}{dw}(t)$$

$$\Delta w = -\eta \frac{dL}{dw}(t)$$

$$\Delta w_i(t) = -\frac{\eta}{\sqrt{G_i(t)+\epsilon}} \frac{\partial L}{\partial w_i}(t)$$

Adaptation metric.

- Adagrad → Smart
  - → Adaptive learning rate schedule
  - → Based on the training itself
  - → Per weight
- RMS prop → root mean square propagation.

$$\Delta w_i(t) = \frac{-\eta}{\sqrt{G_i(t)+\epsilon}} \frac{\partial L}{\partial w_i}(t)$$

$$G_i(t) = \beta G_i(t-1) + (1-\beta)\left(\frac{\partial L}{\partial w_i}(t)\right)^2$$

at beginning point $G_i(0) = 0$

β - yet another parameter usually around 0.9

- ADAM (Adaptive Moment estimator) → Learning rate + momentum

~~Adam~~ most advanced optimizer (very fast & efficient)

$$\Delta w_i(t) = -\frac{\eta}{\sqrt{G_i(t)+\epsilon}} M_i(t)$$

$$M_i(t) = \alpha M_i(t-1) + (1-\alpha)\frac{\partial L}{\partial w_i}(t)$$

$$M_i(0) = 0$$

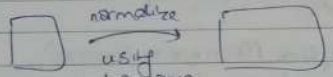
Preprocessing → compatibility
→ Order of Mag
→ Generalization
Data transformation
np.savez( )

Types → Relative
→ Logrithmic

Standardization / Feature scaling

Standardized variable $= \frac{x - \mu}{\sigma}$ (μ → mean, σ → std dev)

Normalization



PCA - principal component analysis
Dimension reduction technique used to combine several variables into bigger (latent) variable.

Whitening
performed after PCA remove unrelated correlation

Categorical Data
→ Binary encoding
→ Many category
→ One hot encoding
→ Few category

Classifying the MNIST dataset

MNIST → Very unusual problem
→ Extremely common
→ Easy to build up to CNN
→ Very big to preprocessed
✓ image recognition

Each photo 28×28 pixel
0 → Black
255 → White

MNIST action plan
1. Prepare our data to preprocess it. Create training, validation & test datasets
2. Outline the model to choose the activation func°
3. Set the appropriate advanced optimizer & the loss funct°
4. Make it learn
5. Test the accuracy of the model.

✓ Package import tensorflow_datasets as tfds

tfds.load(name) → loads a dataset from tensorflow datasets
as_supervised = True → loads the data in a 2-tuple structure [input, target]
with_info = True → provides a tuple containing info about version, features, # samples of the dataset

Preprocess the data

```
mnist_train, mnist_test = mnist_dataset['train'],
                          mnist_dataset['test']

num_validation_samples = 0.1 * mnist_info.splits
                         ['train'].num_examples

num_validation_samples = tf.cast(num_validation_samples,
                         tf.int64)
```

→ convert a var into given data type

```
* def scale(image, label):
    image = tf.cast(image, tf.float32)
    image /= 255.
    return image, label
```

dataset.map(*function*) → applies a custom transformation to a given dataset. It takes as i/p a func^n which determine the transformation

batch size = 1 - SGD
batch size = # samples - GD
1 < batch size < # sample - mini batch GD

dataset.batch(batch_size) → combines consecutive elements of a dataset into batches

when batching - find avg - loss.

Outline the model

```
input_size = 784
output_size = 10
hidden_layer_size = 50
```

- tf.keras.Sequential() → func^n that is laying down the model (used to stack layers)
- tf.keras.layers.Flatten(original shape)
  ↳ transforms (flattens) a tensor into a vector
- tf.keras.layers.Dense(output size)
  takes i/p, provided to the model & calculates the dot product of the i/ps & wts & add the biases
  This is also where we can apply an activation func^n

Select loss & the optimizer

model.compile(optimizer, loss) configures the model for training

What happens inside an epoch

- At the beginning of each epoch, the training loss will be set to 0.
- The algorithm will iterate over a preset no. of batches, all from train_data.
- The weights & biases will be updated as many times as there are batches.
- We will get a value for the loss func^n, indicating how the training is going.
- We will also see a training accuracy.
- At the end of the epoch, the algo will forward propagate the whole validation set.

width - hidden layer size

When we reach the max no. of epochs the training will be over

Create the model → fiddle with the hyperparameters → Check validation accuracy

Testing the model

model.evaluate()
returns the loss value & metrics value for the model in test model

Deep Neural Nlw for MNIST Classification

MNIST dataset → handwritten digit recognition.

"Hello World"

Dataset → 70000 images (28×28 pixel)
↳ 1 digit per image

Goal To write an algo that detects which digit is written (0, 1, 2, ... 9)

to build a neural nlw with 2 hidden layers

Import relevant libraries
↓
Load & preprocess data → train dataset, validation dataset, test dataset
extract train & validation data
↓
batch train & validation data
↓
batch test data
scale
map
Botch, Buffer size
shuffle

↓
Model
outline the model
define → input, output, hidden layer } size
then define how model will look
choose the optimizer & loss func"
↓
Training
↓
Test model

* width (hidden layer size) 50 → 200 → 500
validation accuracy significantly higher
it takes the algo much longer to train

* Depth ↑ → add another hidden layer to algorithm
↳ accuracy of the model does not necessarily improve
↳ Fiddling with the hyperparameter may not be enough
↳ Sometimes deeper nets needs to also be wider in order to have higher accuracy.

* Fiddle with activation functions: relu → sigmoid.
relu → cleans the noise in the data
Sigmoid → does not filter the signals as well as relu

* Fidde with activation func" → use different activations for different layers
↳ The result should not be significantly diff

* Adjust the batch size
  100 → 10,000
  ↳ Bigger the batch size results in slower training
  Notice that the validation accuracy starts from a low no. & with 5 epochs actually finishes at a lower no. That's because there are fewer updates in a single epoch.

* Adjust the batch size
  100 → 1
  Batch size 1 → results in SGD. It takes the algorithm very little time to process a single batch (as it's one data point) but there are thousands of batches (54000 to be precise) thus the algo is slow. Remember that it's depends on the no. of cores that you train on. If you are using a CPU with 4 or 8 cores, you can only train 4 or 8 batches at once. The middle ground (mini batch 100 samples) is optimal.
  Notice that the validation accuracy starts from a higher no. That's because there are lots updates in a single epoch. Once the training is over, the accuracy is lower than all other batch sizes (SGD was an approximation.

* Adjust the learning rate 0.0001
  ↳ Since the learning rate is lower than normal we may need to adjust the max epochs (to say 50)
  ↳ The result is basically the same, but we reach it much slower.
  ↳ While Adam adapts to the problem, if the orders of mag are too different it may not have enough time to adjust accordingly.

* Adjust learning rate 0.02
  ↳ While Adam adapts to the problem, if the order of mag. are too diff, it may not have time to adjust accordingly. We start overfitting before we can reach a real solution.

Business Case → Audiobooks

Action Plan

1. Preprocess the data
   - → Balance the dataset
   - → Divide dataset in train, validate & test } prevent overfitting
   - → Save the data in tensor friendly format. good old npz
2. Create a data set handles batching
3. Create the machine learning algo
   - → same structure different model.

np.delete(array, obj to delete, axis)

np.ndarray.astype() - create a copy of the array cast to a specific type

Early stopping → mechanisms
tensorflows → callbacks.

tf.keras.callbacks.EarlyStopping(patience)
decides how many consequitive ↑ we can tolerate

Test model.
model.evaluate() returns the loss value & metrics values for the model in test mode

* Test accuracy should be lower & equal to the validation accuracy

Import relevant libraries
↓
Data & preprocessing
↓
Model
outline, optimizer, loss, early stopping (extra)
↓
Test the model

Convolutional neural n/w → also flattening (CNN)

kernel → weights - 5×5

pooling: 24×24 → 12×12

No of kernels used → hyperparameters

↳ Mainly used in image recognitions
  ↳ spatial proximities are preserved
  ↳ a detail is looked for every where in a photo

Deep mind.

Recurrent neural n/w (RNN)

↳ Sequential data → NLP
  ↳ trading
  ↳ music

* The hidden layer for the second sample is a func^n of the new i/p & updated weights, and the hidden layer of from prev. iteration

Non neural N/w

* Discriminative models

I/p → Model → Prob whether o/p is correct

Random forest - not good at classification as it overfits a lot

many bad classifier ≡ Good classifier

* Generative models

I/P ⇄ Model ⇄ Prob. whether an o/p is correct → Target joint probability

eg↳ Hidden Markov model
↳ Bayesian N/w

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

Appendix

- Numpy - 3rd party package used for computations
  ↳ multidimensional array.
- Matplotlib 2D plotting package, especially designed for visualizing python & numpy computation.
- Tensorflow - Machine learning - deep learning.
  ↓
  Nodes → in graph represent mathematical operations
  graph edges represent multidimensional data array (tensors) communicated b/w them

  Data → Preprocess → save .npz file

  np.savez(file name, arrays) → used for saving n-dimensional arrays in npz format

  tf.placeholder → where we feed the data

  tf.variables → preserve their values across iterations while placeholders don't

  tf.matmul(A, B→) ≡ np.dot(A, B)

* Loss func : L2-norm loss of observation / 2

  tf.losses.mean_squared_error → method equal to avg L2-norm loss

Optimizable.
tf.train.GradientDescentOptimizer(LR) → learning rate
minimize (mean_loss)
↳ common optimization algos

Prepare for execution
tf.InteractiveSession() → used when we want to execute something

Initializing variables
tf.global_variables_initializer() → method that initializes all tensorflow objects marked as variable.
↳ run → initialize.

Running → iterations
curr_loss → actually return something for optimizer to minimize

'_' underscore → special symbol to disregard a return value of a function (method)

Plotting the data.

Extremely reusable code of tensorflow

* The huber loss is more appropriate than the L2-norm when we have outliers, as it is less sensitive to them.

* Change no. of observations 1000 → 100,000
↳ algo will take more time to solve the problem
↳ matplot lib cannot plot the data, as there are too many points.

* Change learning rate → 0.0001
↳ takes the algo same time to finish working.
↳ loss is not minimized.
↳ wts & biases are far from what we want them to be.
↳ More iterations are needed for this learning rate to solve the problem
↳ The problem is not solved.

* Change learning rate → 0.1 or 1
↳ It takes the algo the same time to finish working.
↳ loss diverges to infinity.
↳ wts & biases are completely random & entering big.
↳ More iterations would not solve the issue, as the loss is not converged
↳ The problem is not solved.

* Change loss function L2 norm → Huber loss
mean_loss = tf.losses.mean_squared_loss(labels=targets, predictions=outputs)/2
↓
mean_loss = tf.losses.huber_loss(labels=targets, predictions=outputs)
↳ Any func° that has the prop. to be least for better result & higher for worse result can be loss func°. This is called as
↳ Almost everything seems identical.
↳ The values of the loss are largely lower.
↳ Both huber & L2 norm converges equally well.
↳ Huber loss is used when we have lots of outliers

Classifying on the MNIST Dataset

Relevant packages

↳ from tensorflow.examples.tutorials.mnist import input_data → automatically downloads the MNIST dataset to the directory of Jupyter notebook

↳ mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

Outline the models

↳ tf.reset_default_graph() → clears the memory of all variables left from previous runs (reset the computational graph.

↳ tf.get_variable("name", shape) → func used to declare variables

↳ tf.nn → contains neural network support. Among other things, it contains the most commonly used activation func like tf.nn.sigmoid
tf.nn.tanh
tf.nn.relu
tf.nn.softmax

↳ tf.nn.softmax_cross_entropy_with_logits(logits, labels)
↳ function applies a softmax activation to calculate a cross entropy loss.
Logits → unscaled probabilities.

↳ tf.reduce_mean() → finds the mean of elements of a tensor across a dimension.

Prediction Accuracy

In what % of the cases the o/p of algorithm matched the target

↳ tf.equal() checks if two values are equal

- The true early stopping will come if the validation loss starts increasing

↳ mnist.train.next_batch (size of batch)
func" that comes with the MNIST data provider which loads the batches one after the other

Tensorflow → code → reusable.

Business case

Preprocessing the data
- Extract the data from csv
- Balance the dataset
- Standardize the inputs
- Shuffle the data
- Split the dataset into train, validation (test)
- Save the 3 datasets in .npz

self → method → instance method.

- A ReLu activation of the first hidden layer will cause many hidden units to be zeroed

Improve model accuracy
- Improve preprocessing
- Fine-tune the model
- Play around with the activation func"
- Fiddle with the batch size
- Experiment with learning rate / optimizer
- Check out tf.contrib
- Take your own data & wisely

Software Integration

Data
form of symbols 1s and 0s, other digits, letters, special characters, etc.
→ it can be collected, measured, analyzed, processed further.

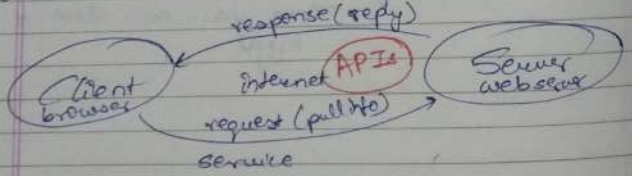
Servers
a combinations of hardware & software responsible for storing, managing & processing large amount of data.

Web servers - web pages

Database servers - database queries

FTP servers - files

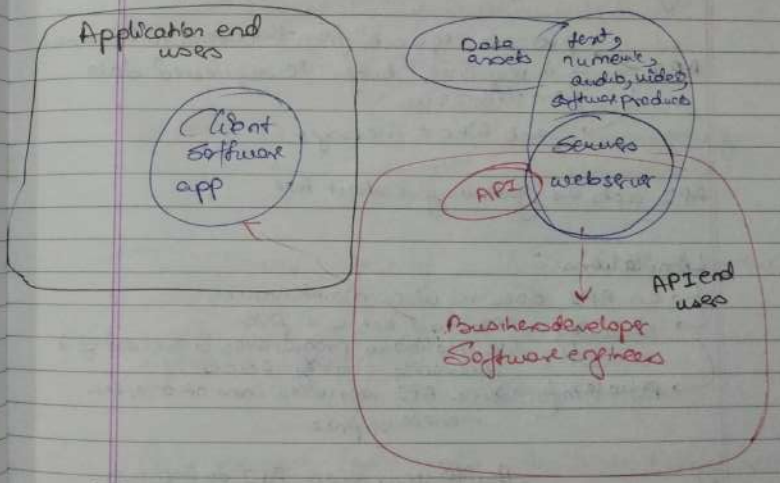
+ Client - Server model.



Data connectivity ⟶ APIs.
regards the ability to connect clients & servers, ensuring the swift & voluminous transfer of information b/w them.

- API (Application Programming Interface)
a contract allowing software to share data with each other.
lets device & software applications communicate in real time

API applications
a collection of endpoints to which developers can attach, and then extract specific information that can be used by those who are working with the apps.



Data assets
data that is expected to have some value in the future = forms of data or software related to the processing of information.
- Intellectual property ⟶ patent, songs
- Databases
- Websites . codes

- Application
  a program designed to perform a specific set of operations for the end user, be it a person or another application
  - web browsers
  - video editing programs
  - Database management software

- Programming
  → refers to the functions the app performs
  → regards the process of converting i/p into o/p

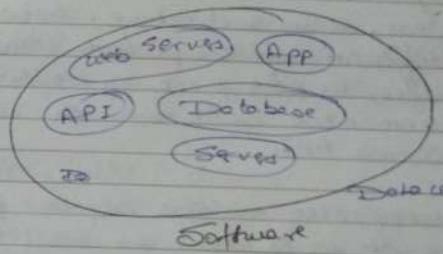
API → process requests from the clients to the server
  → in response, brings the requested data from the server
  → acts like a messenger

API acts as gateway extract text

Limitations
↳ an API does require maintenance
↳ Security → APIs must act as a filter (vs. malicious programmes to overloading to guarantee quality access)
↳ private info: → some API services come at a certain monetary price

Your Server → Build your own API to improve communication with apps & people
  ↳ Having your own interface helps you improve the functionality of the services you are offering

Benefits of API
↳ data is changing extremely quickly
↳ all you need is a specific bit from entire dataset
↳ don't forget how huge the database of an external source could be
↳ Saves enormous amounts of your time
↳ provides access to numerous databases
↳ is very efficient when delivering precisely the info. you need.



* Different piece of software (software products) communicate through text files

• json → Javascript Object Notation
  ↳ just an ordinary text file

Only message (you need) has been turned inside common text format
→ All modern programming languages can read, export or work with JSON
→ Developers find it a lot easier to translate, or to parse a json string

- APIs act like a messenger b/w the end user of the app & the server
- APIs are like a gateway to the server (or a bridge b/w the two pieces of software)
- Interfaces help you establish connections

Integration

→ a system, or an architecture, composed of a few different software products, various prog. languages, or other pieces of software which can communicate with each other w/o API or a common API

→ a situation where multiple software products can be set up to work as one tool

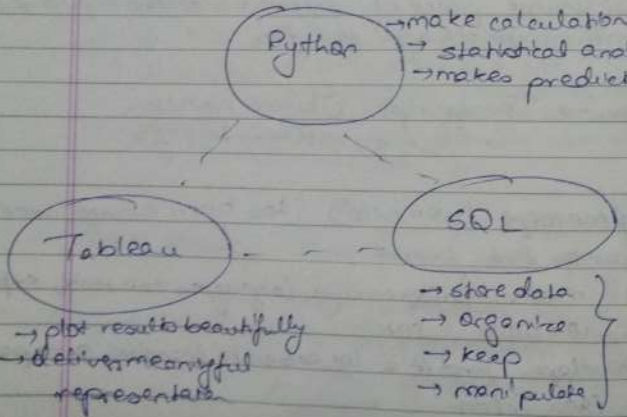


Tableau - - - SQL

- Driver
  a software driver can drive a certain software product with another one
  - It is a piece of software. It's function is to mediate b/w two products
  - Tableau has built in drivers

* Tableau & Python share the same code & function through a common API
  - Tabpy server allows you to execute Python code remotely (& on the fly) thanks to a built in driver

Tableau built in drivers
↳ may leave the user completely unaware of their existence
↳ allow Tableau to centralize the advantages of using SQL relative to Python computers

Case Study

Good Business analyst must

SQL → be able to manage information

Python → have a substantial amount of mathematical & statistical tools

Tableau → present results in the most intuitive way

Business logic → be able to tackle the problem from a fundamentally business perspective.

The exercise will address ABSENTEEISM at a company during work time ↳ absence from work

Problem

higher competitiveness → increased pressure

unachievable business goals → raised stress levels

elevated risk of becoming unemployed → raised stress levels

DATA

Primary
data you have created

eg - Survey questionnaire
- Sales operations
- Supply inventory
- accounting ledger data
(business intelligence)

Secondary
an already existing dataset that somebody else has created
NOT YOURSELF

eg - free data from a website
- paid data from organizations

Data preprocessing
a group of operations that will convert your raw data (the data you have been given) into a format that is easier to understand and hence useful for further processing & analysis

- fix the problems that can inevitably occur with data gathering
- organize your info in a suitable & practical way

Preprocessing

1. Importing the data

import pandas → alias used to work with panel data

Panel → data handle data in tabular form

- read_csv() → assign info from initial .csv file to this variable

2. Checking the content of the dataset

* You must always make a copy of your initial dataset

df = data.copy()

df → dataframe

→ Display all items

pd.options.display.max_columns = None

pd.options.display.max_rows = None.

display(df)

→ df.info() → gives summary of dataset.

- variable in economics → characteristics / quantity } that may change.
  feature, attribute

variable in programming - storage location.

maths → scalar → vector → matrix

↓

Python → array

Variables → dependent → target

→ independent → predictors

Logistic regression → 0 False
→ 1 True

→ Dropping column in dataframe

df.drop(['ID'], axis=1) → temporary o/p

df = df.drop(['ID'], axis=1)

→ Analyzing the reasons for absences

df[specify column].min() / .max() } finding min & max of column
eg: 'Reason'

pd.unique(df['Reason for Absence'])
↳ distinct values in the columns

→ or df['Reason'].unique()

len() → no. of elements

to find missing element use
sorted() → sorted list

Quantitative analysis → add numeric meaning to our categorical nominal values → dummy variables

explanatory binary variable

1 → certain categorical effect that

0 → absent

- get_dummies() - converts categorical variable into dummy variables

- Checkpoints
In programming creating checkpoints refers to storing the current version of your code, not really the content of a variable

→ Create a temporary save of your work so that you reduce the risk of losing important data at a late stage.

- Timestamp - used for values representing dates & time
a classical date type found in many prog. languages
pd.to_datetime() → converts values into timestamp

When doing this conversion, you must specify the proper format of date values you will be working on

, format - 'string' allows you to take control over how python will read the current dates, so that it can accurately understand which no. refer to days, month, year

%d → day
%m → month
%Y → Year
%H → Hour
%M → minute
%S → second

(format = '%d/%m/%Y')

- weekday() → returns an integer corresponding to the day of the week

travel expenses = fuel + parking + meals + transportation + other

Absenteeism reasons → Transportation expense
→ Distance to work
→ Age
→ Daily work load avg
→ Body mass Index
→ Children / Pets

- value_counts() → count no. in different categories

→ The more manual way of doing preprocessing gives you a higher level of control over your data

Applying ML to create absenteeism_module

Logistic regression
random forest
Neural n/w.

→ import libraries
→ load preprocessed data
Logistic regression → Create targets → above which excess
below which moderate

np.where(cond, value if True, value if False)

.median() gives mean of code
↳ Using median as a cut off line is numerically stable & rigid.

→ Select o/p for the regression
→ A balance of 45-55 % is almost always sufficient

df.iloc[: , :] → col. indices
row
→ slice data by position of column

→ Standardize the data

absenteeism_scaler = StandardScaler()

absenteeism_scaler.fit(df)

↳ calculate & store the means & std dev.

.transform() → does the actual scaling

→ Split data into train & test data set

↳ to avoid overfitting & underfitting

80 → 20 split ↳ shuffle

random_state → make shuffle pseudo random

→ Logistic regression with sklearn

* statsmodels are not always numerically stable, for more complicated models.

↳ so use sklearn

Import

Train the model

ML
- reg.fit(x,y) → fits the model acc. to given training data
  - LogisticRegression()
- reg.score(i/p, targets) → returns the mean accuracy on the given test data & labels

Manually checking

reg.predict() → predicts the class labels for given i/p samples

→ Creating a summary tables with coeff & intercept.

reg.intercept_

reg.coef_

→ A feature is not particularly important:

↳ if its coefficient is around 0

↳ if its odds ratio is around 1

→ A wt (coeff) of 0 multiplies that no matter the feature value, we will multiply it by 0

→ For a unit change in the standardized feature, the odds increase by a multiple equals to the odds ratio. (1 = no change)

odds	× odds ratio	= New odds
5:1	× 2	= 10:1
5:1	× 1	= 5:1

* When we standardized we also standardized the dummy → that a problem

→ The further away from 0 a coefficient is, the bigger its importance.

→ Standardized models (almost) always yield higher accuracy

+ ML engineers - prefer models with higher accuracy, so they normally go for standardization

+ Econometricians & Statisticians prefer less accurate but more interpretable models because they care about underlying reasons behind different phenomena.

→ Data Scientists may be in either position. Sometimes, they need higher accuracy, other times - they must find the main drivers of a problem.

→ The intercept or the BIAS calibrates the model

→ Backward elimination.

* The idea is that we can simplify our model by removing all features which have close to no contribution to the model.
* When we have the p-values, we get rid of all coefficients with p-values > 0.05.

* If the weight is small enough, it won't make a difference anyway.

→ List comprehension is a syntactic construct which allows us to create a list from existing lists based on loops, conditionals, etc.

Testing the model

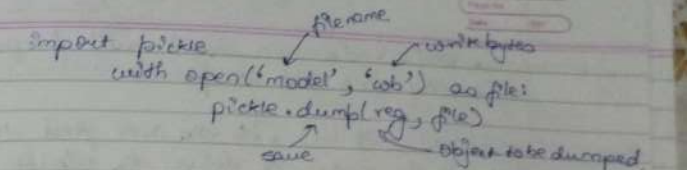
→ Often the test accuracy is 10-20% lower than the train accuracy (due to overfitting)

reg.predict_proba(x) → returns the probability estimates for all possible o/p (classes)

Saving the model

Saving the model means saving the reg object

pickle[module] → is a python module used to convert a python object into a character stream



pickle → std python tool → serialization
→ deserialization

pickling → converting python object → string of characters
unpickling → converting string of characters → python object

- Pickle & python version may differ
- Pickle is slow
- Pickle is not secure
  → Never unpickle data received from an untrusted & unauthenticated source.
  → reason - anything can be pickled so you can easily unpickle malicious code.

Deployment of the model through a module

→ storing code in a module will allow us to reuse it w/o trouble.

Model	vs	Module
- the analytical tool applied to solve the business problem		A software component containing the code that will help us execute the model → a file containing python definitions & statements, with the suffix .py.

Scaler → it contains the statistical parameters needed to adjust the magnitude of all numbers we have in this data set

• load_and_clean_data() will preprocess the entire data set we provide

• predicted_outputs() - its role is to feed the cleaned data into the model, to deliver the O/p we discussed

Tableau

Slope = scatter plot

Additional Python tools

template .format() → very useful tool for preprocessing (specify)
{} {} → text data
→ applicable to string values only
→ will have formatted the values as text first regardless of their initial datatype

in template act as placeholder

We can use both index values & keys at the same time

Iterating over range objects.

help(range)   range (start, stop, step)

List comprehension

i+j for i in range(2) for j in range(5)

lambda func° → creating anonymous functions

raise to power2 = lambda x : x**2

raise to power2(3)

Or   lambda (x : x/2) (11) Ans 5.5 (check)

Shift + Tab → documentation
(tab + enter: to find the range of already defined)

Lambda function → one or many parameters; contains single expression only
→ Can be applied to the larger exp. they have been written in

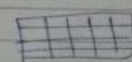
Pandas

Functions	methods
independent entity	- can have access to the objects data
squeeze = True	- can manipulate the objects state

pd.series( )
.idxmax() → returns index label corresponding to highest value in a series
.max()
.head() none
.describe()
len()
.unique() → array → delivers the values in order they have appeared in the data set
.nunique() → no. of unique elements
.sort_values() → ascending = False →

Series
Single column data
1-D

Dataframe
multicolumn data
2-D
collection of series



Every column from the Dataframe is a Series object itself

array → Dataframe
↓ ()
numpy
pandas.

- iloc[] - purely integer location based indexing for selection by position.
  - iloc[r, c]
  - iloc[r, :] → entire row.

- loc[]