

Sardar Patel University, Balaghat (M.P)

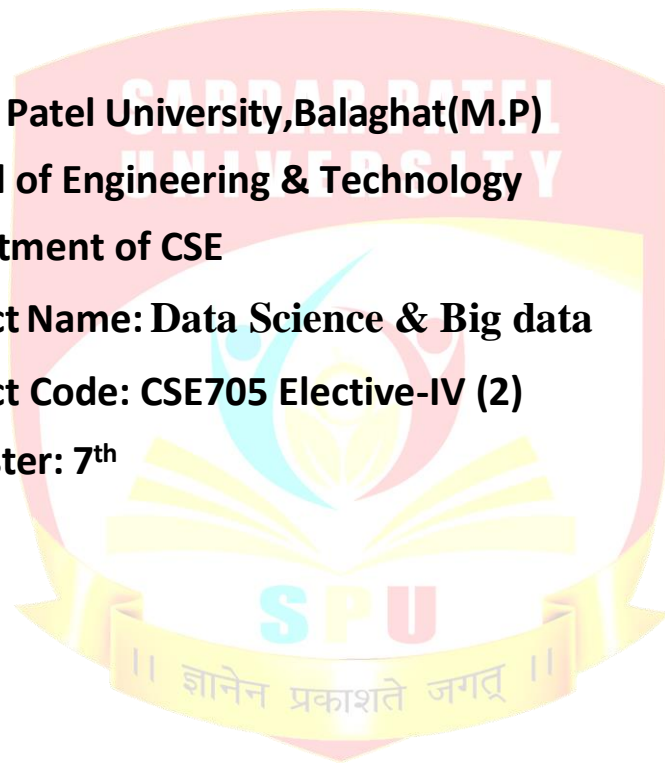
School of Engineering & Technology

Department of CSE

Subject Name: Data Science & Big data

Subject Code: CSE705 Elective-IV (2)

Semester: 7th



Unit-1

Topics to be covered

Understanding Data: Data Wrangling and Exploratory Analysis, Data Transformation & Cleaning, Feature Extraction, Data Visualization. Introduction to contemporary tools and programming languages for data analysis like R and Python.

Understanding of Data: Data Wrangling

Data wrangling is the process of cleaning, structuring and enriching raw data into a desired format for better decision making in less time. Data wrangling is increasingly ubiquitous at today's top firms. Data has become more diverse and unstructured, demanding increased time spent culling, cleaning, and organizing data ahead of broader analysis. At the same time, with data informing just about every business decision, business users have less time to wait on technical resources for prepared data.

This necessitates a self-service model, and a move away from IT-led data preparation, to a more democratized model of self-service data preparation or data wrangling. This self-service model allows analysts to tackle more complex data more quickly, produce more accurate results, and make better decisions.

Data Wrangling in Practice:

There are typically six iterative steps that make up the data wrangling process.

1. **Discovering:** Before you can dive deeply, you must better understand what is in your data, which will inform how you want to analyze it. How you wrangle customer data, for example, may be informed by where they are located, what they bought, or what promotions they received.
2. **Structuring:** This means organizing the data, which is necessary because raw data comes in many different shapes and sizes. A single column may turn into several rows for easier analysis. One column may become two. Movement of data is made for easier computation and analysis.
3. **Cleaning:** What happens when errors and outliers skew your data? You clean the data. What happens when state data is entered as CA or California or Calif.? You clean the data. Null values are changed and standard formatting implemented, ultimately increasing data quality.
4. **Enriching:** Here you take stock in your data and strategize about how other additional data might augment it. Questions asked during this data wrangling step might be: what new types of data can I derive from what I already have or what other information would better inform my decision making about this current data?
5. **Validating:** Validation rules are repetitive programming sequences that verify data consistency, quality, and security. Examples of validation include ensuring uniform

distribution of attributes that should be distributed normally (e.g. birth dates) or confirming accuracy of fields through a check across data.

6. **Publishing:** Analysts prepare the wrangled data for use downstream – whether by a particular user or software – and document any particular steps taken or logic used to wrangle said data. Data wrangling gurus understand that implementation of insights relies upon the ease with which it can be accessed and utilized by others.

Understanding of Data: Exploratory Analysis

Exploratory Data Analysis refers to a set of techniques originally developed by John Tukey to display data in such a way that interesting features will become apparent. Unlike classical methods which usually begin with an assumed model for the data, EDA techniques are used to encourage the data to suggest models that might be appropriate. Statpoint Technologies products provide many EDA techniques, scattered throughout the statistical procedures. Some of the most important exploratory data analysis methods include:

Box-and-Whisker Plots

Box-and-whisker plots are graphical displays based upon Tukey's 5-number summary of a data sample. In his original plot, a box is drawn covering the center 50% of the sample. A vertical line is drawn at the median, and whiskers are drawn from the central box to the smallest and largest data values. If some points are far from the box, these "outside points" may be shown as separate point symbols. Later analysts have added notches showing approximate confidence intervals for the median, and plus signs at the sample mean.

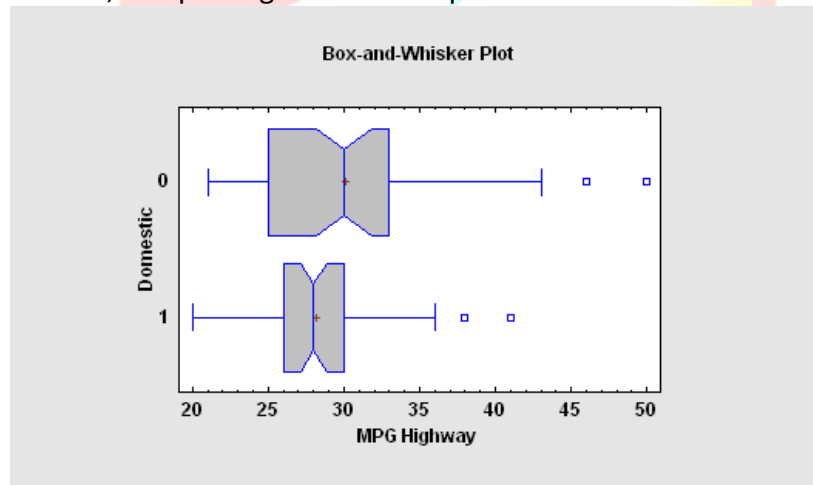


Figure 1: Box-and-Whisker Plots

Stem-and-Leaf Display

Stem-and-leaf displays take each data value and divide it into a stem and a leaf. For example, the temperature of the first subject in the data sample to the left had a body temperature of 98.4 degrees. The first two digits ("98") are called the stem and plotted at the left, while the third digit ("4") is called the leaf. Although similar to a histogram turned on its side, Tukey thought that the stem-and-leaf plot was preferable to a barchart since the data values could be recovered from the display.

Stem-and-Leaf Display for Temperature: unit = 0.1 1|2 represents 1.2

```

LO| 96.3 96.4

 2   96|
 6   96| 7789
19   97| 0111222344444
40   97| 556666777888888899999
(38) 98| 000000000001112222222222333334444444444
52   98| 55566666666666777777777888888888899
19   99| 000001112223344
 4   99| 59
 2  100| 0

HI| 100.8

```

Figure 2: Stem-and-Leaf

Rootogram

A rootogram is similar to a histogram, except that it plots the square roots of the number of observations observed in different ranges of a quantitative variable. It is usually plotted together with a fitted distribution. The idea of using square roots is to equalize the variance of the deviations between the bars and the curve, which otherwise would increase with increasing frequency. Sometimes, the bars are suspended from the fitted distribution, which allows for easier visual comparison with the horizontal line drawn at 0, since visual comparison with a curved line may be deceiving.

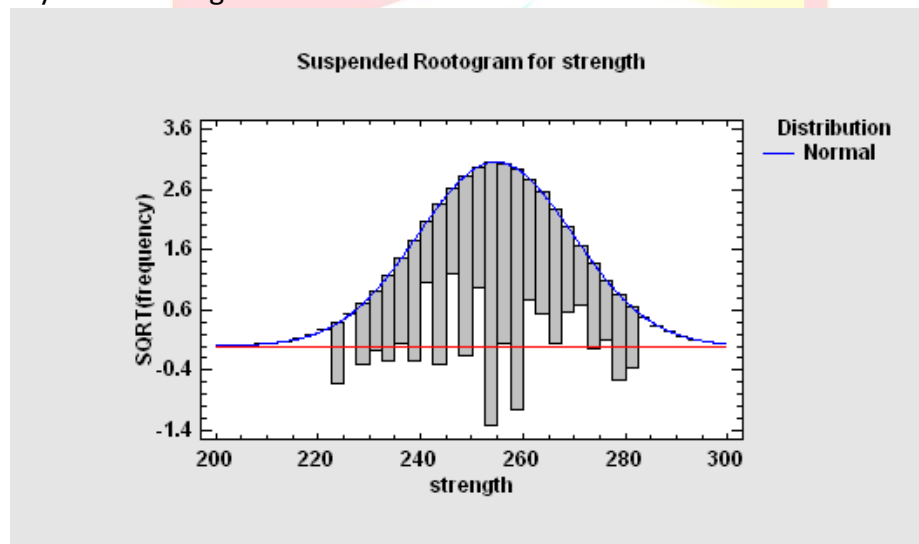


Figure 3: Rotogram

Resistant Time Series Smoothing

Tukey invented a number of nonlinear smoothers, used to smooth sequential time series data, that are very good at ignoring outliers and are often applied as a first step to reduce the influence of potential outliers before a moving average is applied. These include 3RSS, 3RSSH, 5RSS, 5RSSH, and 3RSR smoothers. Each symbol in the name of the smoother indicates an operation that is applied to the data.

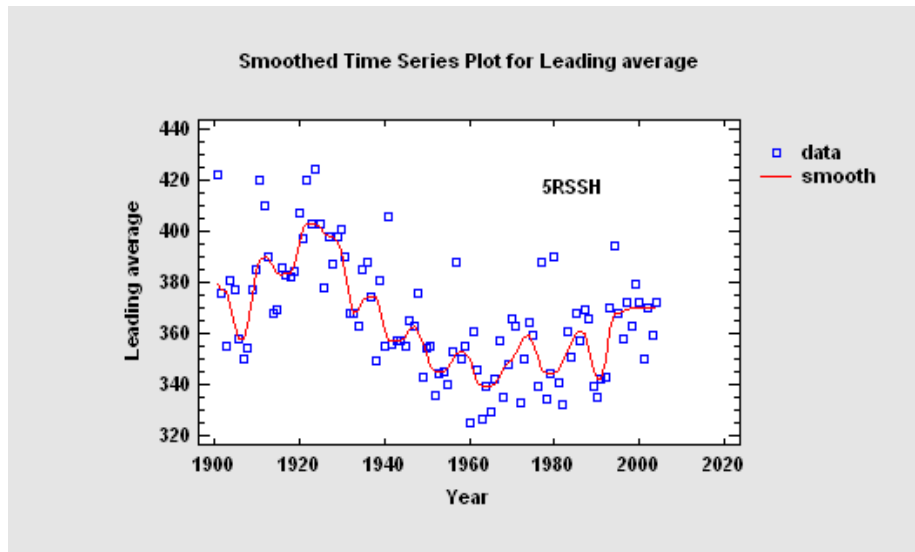


Figure 4: Resistant Time Series

Scatterplot Smoothing

X-Y scatterplots may be smoothed using any of several methods: running means, running lines, LOWESS (locally weighted scatterplot smoothing), and resistant LOWESS. Smoothers are useful for suggesting the type of regression model that might be appropriate to describe the relationship between two variables.

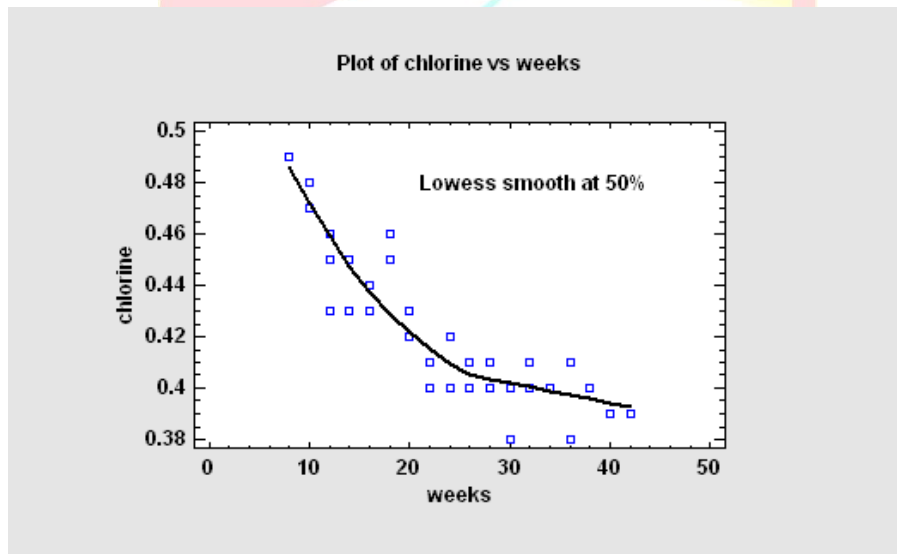


Figure 5: Scatterplot

Median Polish

The Median Polish procedure constructs a model for data contained in a two-way table. The model represents the contents of each cell in terms of a common value, a row effect, a column effect, and a residual. Although the model used is similar to that estimated using a two-way analysis of variance, the terms in the model are estimated using medians rather than means. This makes the estimates more resistant to the possible presence of outliers.

Polished Table

Sweeping 3 times.

Cause	None	Grams 1_14	Grams 15_24	Grams 25	Row effect
Lung cancer	-0.5	-0.2025	0.2	0.86	0.1175
Upper resp. cancer	0.0	0.0275	0.0	-0.02	-0.4525
Stomach cancer	0.24	0.0875	-0.16	-0.09	-0.2825
Colon cancer	0.0025	0.0	-0.1575	0.0725	-0.015
Prostrate caner	0.405	0.0125	-0.015	-0.035	-0.3075
Other cancer	-0.015	-0.0375	0.015	0.135	0.2025
TB	-0.06	-0.0025	0.03	0.0	-0.3925
Bronchitis	-0.125	-0.0575	0.055	0.245	-0.2075
Other respiratory	0.24	-0.0025	0.0	-0.28	-0.0025
Thrombosis	-0.305	0.0125	-0.015	1.235	4.0725
Cardiovascular	0.0925	-0.09	0.2425	-0.1175	1.685
Hemorrhage	0.0875	-0.085	-0.1525	0.1775	1.47
Ulcer	-0.0175	0.02	0.0525	-0.0275	-0.435
Violence	-0.125	0.1725	-0.185	0.125	0.0925
Other	0.035	0.2925	-0.035	-0.075	0.9625
Column effect	-0.09375	0.00875	-0.00375	0.13625	0.54625

Figure 6: Median Polish

Bubble Chart

The Bubble Chart is an X-Y scatterplot on which the value of a third and possibly fourth variable is shown by changing the size and/or color of the point symbols. It is one way to plot multivariate data in 2 dimensions.

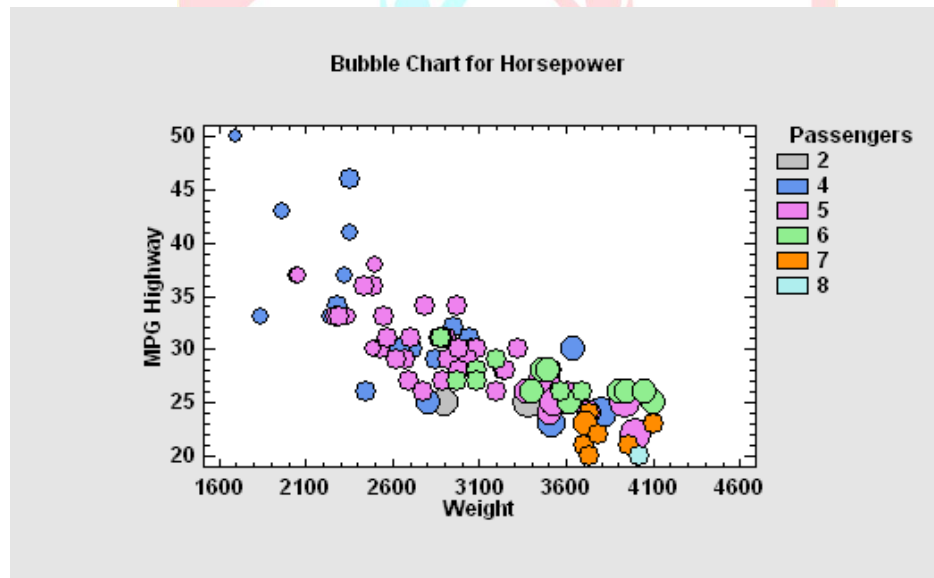


Figure 7: Bubble Chart

Resistant Curve Fitting

Tukey proposed a method for fitting lines and other curves that is less influenced by any outliers that might be present. Called the method of 3 medians, the data are first divided into 3

groups according to the value of X. Medians are then computed within each group, and the curve is determined from the 3 medians.

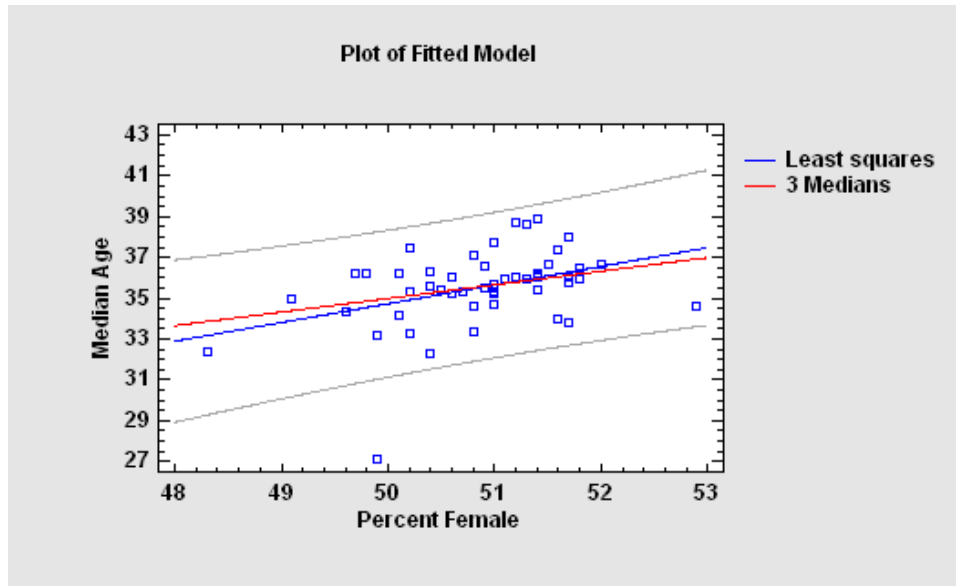


Figure 8: Resistant Curve

Multi-Vari Chart

A Multi-Vari Chart is a chart designed to display multiple sources of variability in a way that enables the analyst to identify easily which factors are the most important. This exploratory data analysis technique is commonly used to display EDA data from a designed experiment prior to performing a formal statistical analysis.

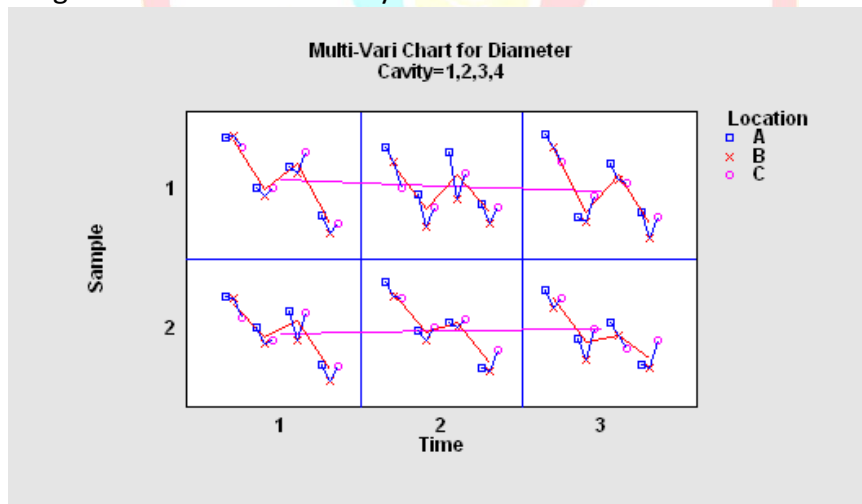


Figure 9: Multi-Vari Chart

Violin Plot

The *Violin Plot Statlet* displays data for a single quantitative sample using a combination of a box-and-whisker plot and a nonparametric density estimator. It is very useful for visualizing the

shape of the probability density function for the population from which the data came. A separate procedure is available for creating violin plots for multiple samples.

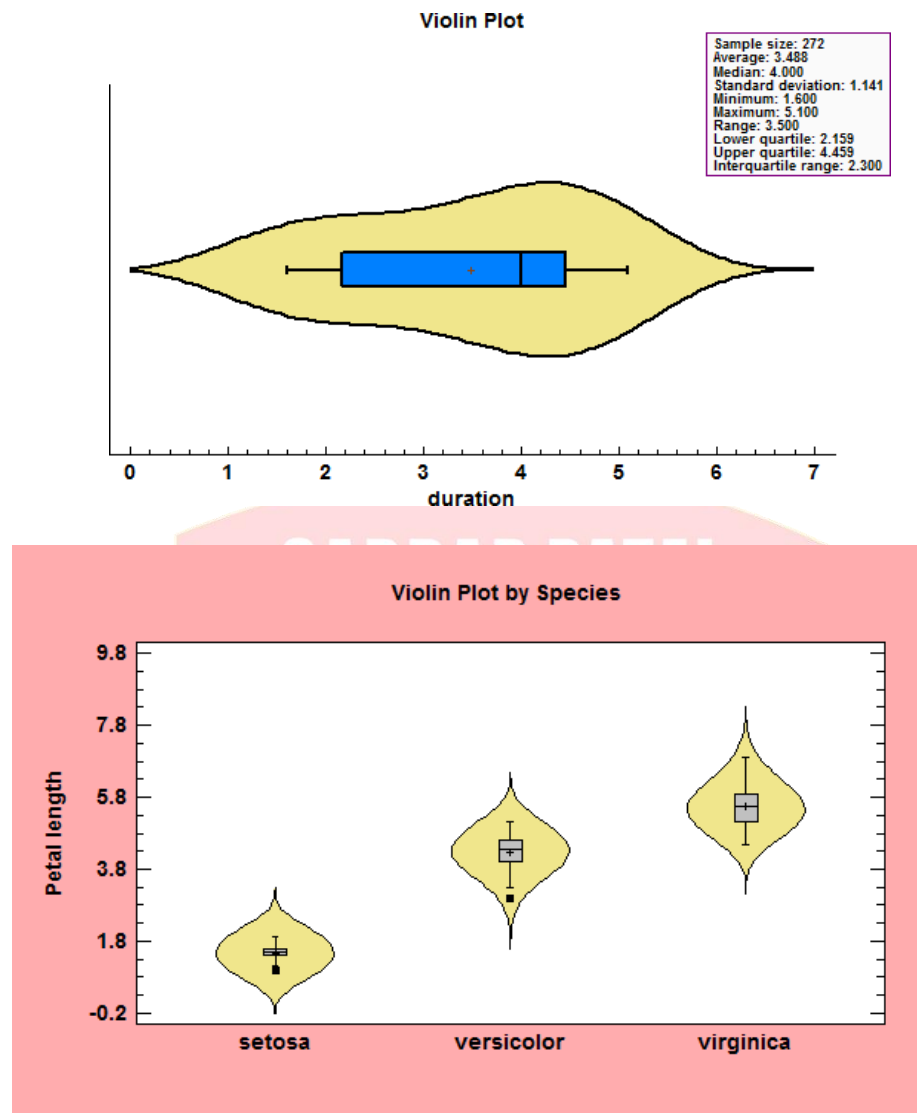


Figure 10: Violin Plots

Wind Rose

The *Wind Rose Statlet* displays data on a circular plot, depicting the frequency distribution of variables such as wind speed and direction. It may be used to display the distribution at a single point in time, or it may show changes over time in a dynamic manner.

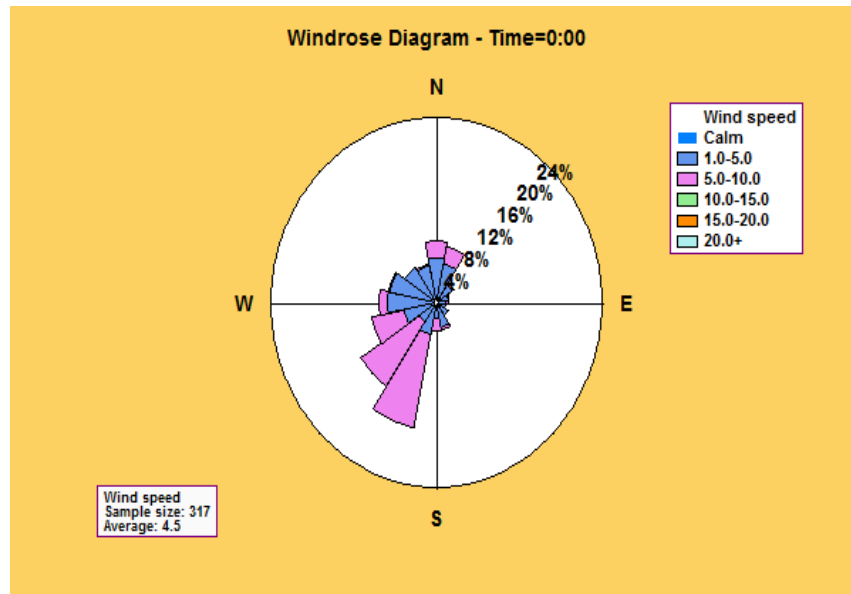


Figure 11: Wind Rose

Diamond Plot

The Diamond Plot procedure creates a plot for a single quantitative variable showing the n sample observations together with a confidence interval for the population mean. A separate procedure is available for creating diamond plots for multiple samples.

Multiple Diamond Plot by Type

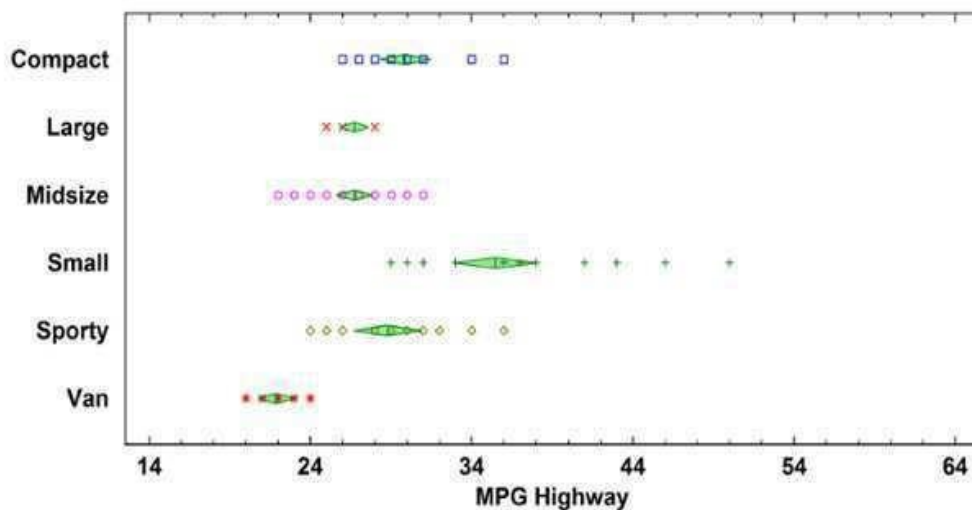


Figure 12: Diamond Plot

Heat Map

The Heat Map procedure shows the distribution of a quantitative variable over all combinations of 2 categorical factors. If one of the 2 factors represents time, then the evolution of the

variable can be easily viewed using the map. A gradient color scale is used to represent values of the quantitative variable.

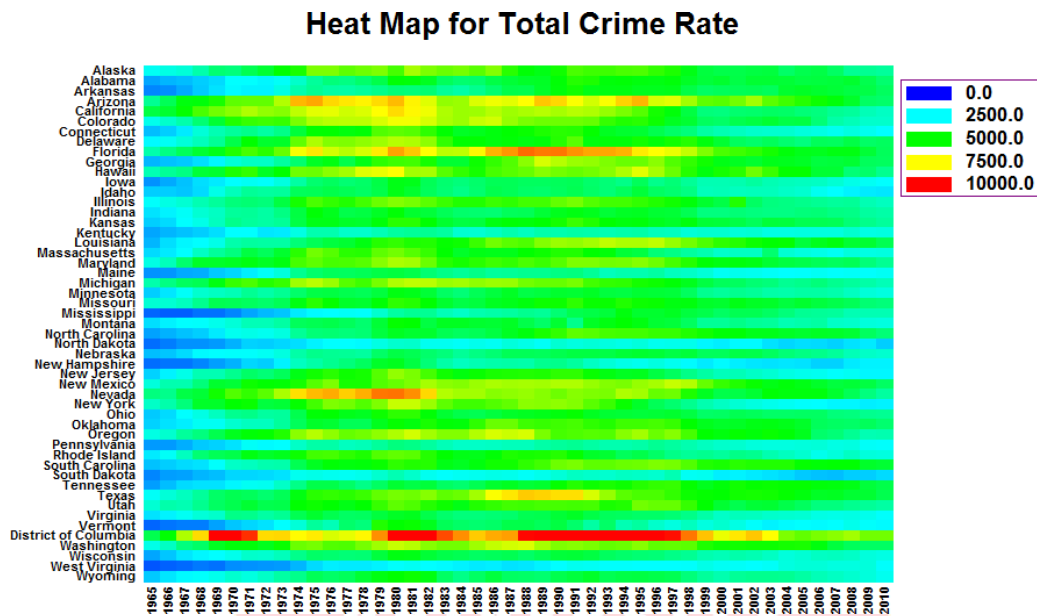
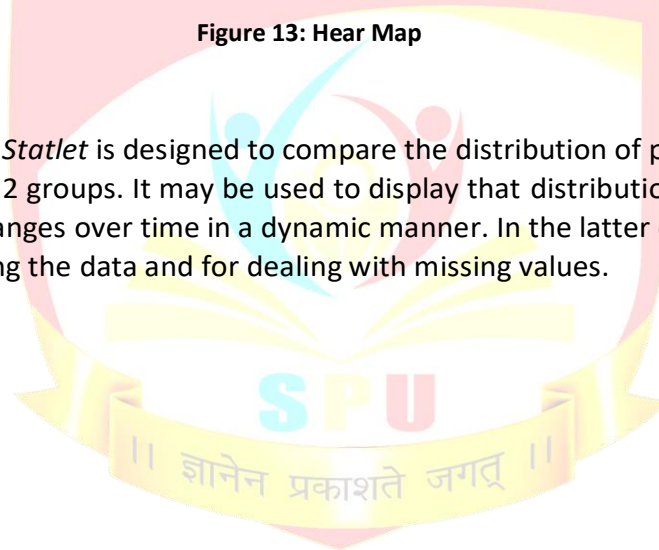


Figure 13: Hear Map

Population Pyramid

The *Population Pyramid Statlet* is designed to compare the distribution of population counts (or similar values) between 2 groups. It may be used to display that distribution at a single point in time, or it may show changes over time in a dynamic manner. In the latter case, various options are offered for smoothing the data and for dealing with missing values.



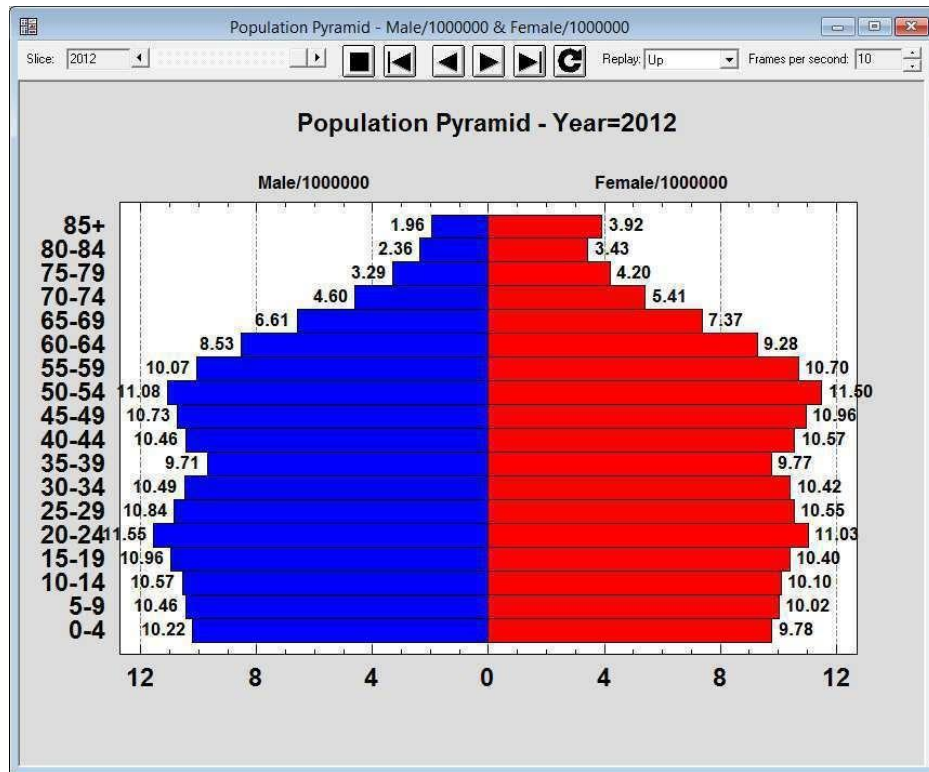


Figure 14: Population Pyramid

Sunflower Plot

The *Sunflower Plot Statlet* is used to display an X-Y scatterplot when the number of observations is large. To avoid the problem of overplotting point symbols with large amounts of data, glyphs in the shape of sunflowers are used to display the number of observations in small regions of the X-Y space.

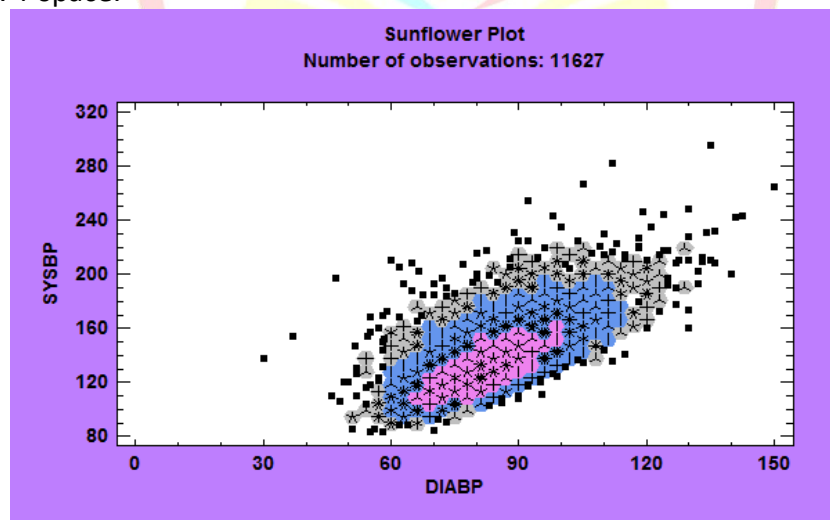


Figure 15: Sunflower Plot

Data Transformation & Cleaning, Feature Extraction

Pre-processing of data involves a set of key tasks that demand extensive computational infrastructure and this in turn will make way for better results from your big data strategy. Moreover, cleanliness of the data would determine the reliability of your analysis and this should be given high priority while plotting your data strategy.

1. Data pre-processing techniques

Since the extracted data tend to be imperfect with redundancies and imperfections, data pre-processing techniques are an absolute necessity. The bigger the data sets, more complex mechanisms are needed to process it before analysis and visualization. Pre-processing prepares the data and makes the analysis feasible while improving the effectiveness of the results. Following are some of the crucial steps involved in data pre-processing.

2. Data cleansing

Cleansing the data is usually the first step in data processing and is done to remove the unwanted elements as well as to reduce the size of the data sets, which will make it easier for the algorithms to analyze it. Data cleansing is typically done by using instance reduction techniques.

Instance reduction helps reduce the size of the data set without compromising the quality of insights that can be extracted from the data. It removes instances and generates new ones to make the data set compact. There are two major instance reduction algorithms:

Instance selection: Instance selection is used to identify the best examples from a very large data set with many instances in order to curate them as the input for the analytics system. It aims to select a subset of the data that can act as a replacement for the original data set while completely fulfilling the goal. It will also remove redundant instances and noise.

Instance generation: Instance generation methods involve replacing the original data with artificially generated data in order to fill regions in the domain of an issue with no representative examples in the master data. A common approach is to relabel examples that appear to belong to wrong class labels. Instance generation thus makes the data clean and ready for the analysis algorithm.

Tools you can use: Drake, DataWrangler, OpenRefine

3. Data normalization

Normalization improves the integrity of the data by adjusting the distributions. In simple words, it normalizes each row to have a unit norm. The norm is specified by parameter p which denotes the p -norm used. Some popular methods are:

StandardScaler: Carries out normalization so that each feature follows a normal distribution.

MinMaxScaler: Uses two parameters to normalize each feature to a specific range – upper and lower bound.

ElementwiseProduct: Uses a scalar multiplier to scale every feature.

Tools you can use: Table analyzer, BDNA

4. Data transformation

If a data set happens to be too large in the number of instances or predictor variables, dimensionality problem arises. This is a critical issue that will obstruct the functioning of most data mining algorithms and increases the cost of processing. There are two popular methods

for data transformation by dimensionality reduction – Feature Selection and Space Transformation.

Feature selection: It is the process of spotting and eliminating as much unnecessary information as possible. FS can be used to significantly reduce the probability of accidental correlations in learning algorithms that could degrade their generalization capabilities. FS will also cut the search space occupied by features, thus making the process of learning and mining faster. The ultimate goal is to derive a subset of features from the original problem that describes it well.

Space transformations: Space transformations work similar to feature selection. However, instead of selecting the valuable features, space transformation technique will create a fresh new set of features by combining the originals. This kind of a combination can be made to obey certain criteria. Space transformation techniques ultimately aim to exploit non-linear relations among the variables.

Tools you can use: Talend, Pentaho

5. Missing values imputation

One of the common assumptions with big data is that the data set is complete. In fact, most data sets have missing values that's often overlooked. Missing values are datums that haven't been extracted or stored due to budget restrictions, a faulty sampling process or other limitations in the data extraction process. Missing values is not something to be ignored as it could skew your results.

Fixing the missing values issue is challenging. Handling it without utmost care could easily lead to complications in data handling and wrong conclusions.

There are some relatively effective approaches to tackle the missing values problem. Discarding the instances that might contain missing values is the common one but it's not very effective as it could lead to bias in the statistical analyses. Apart from this, discarding critical information is not a good idea. A better and more effective method is to use maximum likelihood procedures to model the probability functions of the data while also considering the factors that could have induced the missing values. Machine learning techniques are so far the most effective solution to the missing values problem.

6. Noise identification

Data gathering is not always perfect, but the data mining algorithms would always assume it to be. Data with noise can seriously affect the quality of the results, tackling this issue is crucial. Noise can affect the input features, output or both in most cases. The noise found in the input is called attribute noise whereas if the noise creeps into the output, it's referred to as class noise. If noise is present in the output, the issue is very serious and the bias in the results would be very high.

There are two popular approaches to remove noise from the data sets. If the noise has affected the labeling of instances, data polishing methods are used to eliminate the noise. The other method involves using noise filters that can identify and remove instances with noise from the data and this doesn't require modification of the data mining technique.

7. Minimizing the pre-processing tasks

Preparing the data for your data analysis algorithm can involve many more processes depending on the application's unique demands. However, basic processes like cleansing,

reduplication and normalization can be avoided in most cases if you choose the right source for data extraction. It's highly unlikely that a raw source can give you clean data.

Data Visualization

Data visualization can be helpful in many ways and just in case if you are wondering where it is being used. Then are some of the popular sectors:

- By using data visualization it became easier for business owners to understand their large data in a simple format. The visualization method is also time saving so business does not have to spend much time to make a report or solve a query. They can easily do it in a less time and in a more appealing way.
- Visual analytics offers a story to the viewers. By using charts and graphs or images a person can easily exposure the whole concept. As well the viewers will be able to understand the whole thing in an easy way.
- The most complicated data will look easy when it gets through the process of visualization. Complicated data report gets converted into a simple format. And it helps people to understand the concept in an easy way.
- With the visualization process, it gets easier to the business owners to understand their product growth. Market competition in a better way. The visualization tools can be very helpful to monitor an email campaign. Or company's own initiative regarding something

Here are the list of popular data visualization techniques as follows:

Two-dimensional (2D) area

Such visual forms are mostly geospatial, which means they represent some certain geographical location on the globe.

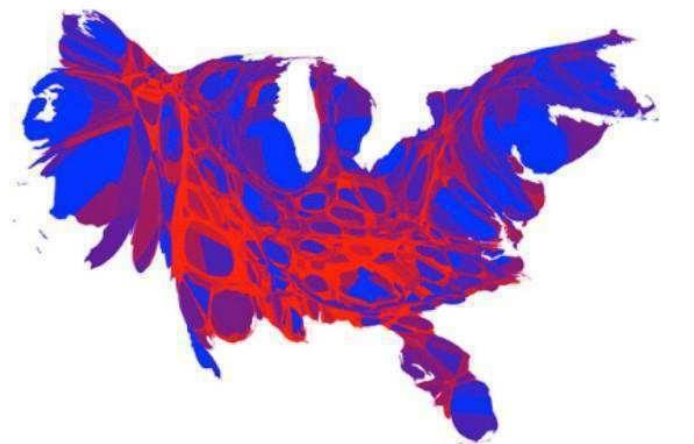


Figure 16: 2-D Dimensional area

- **Area or distance cartograms** are the copies of some parts of maps, depicting some additional parameters like demography, population size, travel times and any other variables.



Figure 17: Area Cartogram

- **Choropleth** is a map colored with different colors depending on the level of the examined variable, like the sales level per state or the biggest inventory stocks per state.



Figure 18: Choropleth

- A **dendrogram** is an illustration of a hierarchical clustering of various data sets, helping to understand their relations in an instant.

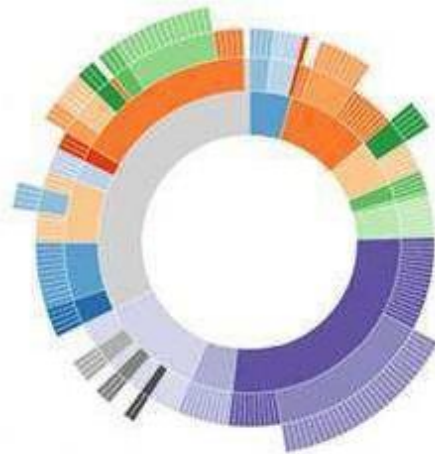


Figure 19: Dendrogram

- **Connectivity charts** show the links between phenomena or events. The chart below, for example, shows the connections between machinery failures and their triggers, as well as the strength of these connections.

Introduction to contemporary tools and programming languages for data analysis like R and Python

The Case for Python

Key quote: *“I have this hope that there is a better way. Higher-level tools that actually let you see the structure of the software more clearly will be of tremendous value.”* – Guido van Rossum
Guido van Rossum was the creator of the Python programming language.

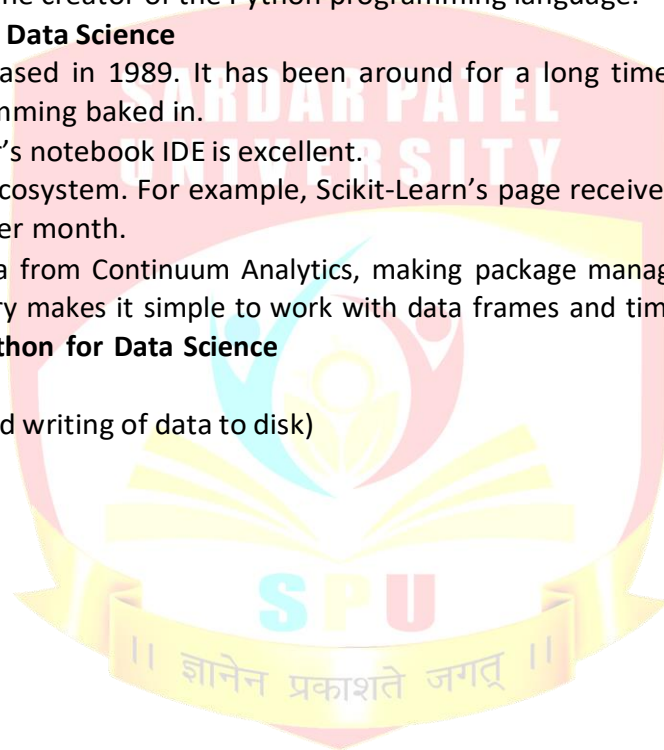
Why Python is Great for Data Science

- Python was released in 1989. It has been around for a long time, and it has object-oriented programming baked in.
- IPython / Jupyter’s notebook IDE is excellent.
- There’s a large ecosystem. For example, Scikit-Learn’s page receives 150,000 – 160,000 unique visitors per month.
- There’s Anaconda from Continuum Analytics, making package management very easy.
- The Pandas library makes it simple to work with data frames and time series data.

Advances in Modern Python for Data Science

1. Collecting Data

Feather (Fast reading and writing of data to disk)



- Fast, lightweight, easy-to-use binary format for filetypes
- Makes pushing data frames in and out of memory as simply as possible
- Language agnostic (works across Python and R)
- High read and write performance (600 MB/s vs 70 MB/s of CSVs)
- Great for passing data from one language to another in your pipeline

Ibis (Pythonic way of accessing datasets)

- Bridges the gap between local Python environments and remote storages like Hadoop or SQL
- Integrates with the rest of the Python ecosystem

ParaText (Fastest way to get fixed records and delimited data off of disk and into RAM)

- C++ library for reading text files in parallel on multi-core machines
- Integrates with Pandas: `paratext.load_csv_to_pandas("data.csv")`
- Enables CSV reading of up to 2.5GB a second
- A bit difficult to install

bcolz (Helps you deal with data that's larger than your RAM)

- Compressed columnar storage
- You have the ability to define a Pandas-like data structure, compress it, and store it in memory
- Helps get around the performance bottleneck of querying from slower memory

2. Data Visualization

Altair (Like a Matplotlib 2.0 that's much more user friendly)

- You can spend more time understanding your data and its meaning.
- Altair's API is simple, friendly and consistent.
- Create beautiful and effective visualizations with a minimal amount of code.
- Takes a tidy DataFrame as the data source.
- Data is mapped to visual properties using the group-by operation of Pandas and SQL.
- Primarily for creating static plots.

Bokeh (Reusable components for the web)

- Interactive visualization library that targets modern web browsers for presentation.
- Able to embed interactive visualizations.
- D3.js for Python, except better.

Geoplotlib (Interactive maps)

- Extremely clean and simple way to create maps.
- Can take a simple list of names, latitudes, and longitudes as input.

3. Cleaning & Transforming Data

Blaze (NumPy for big data)

- Translates a NumPy / Pandas-like syntax to data computing systems.
- The same Python code can query data across a variety of data storage systems.
- Good way to future-proof your data transformations and manipulations.

xarray (Handles n-dimensional data)

- N-dimensional arrays of core pandas data structures (e.g. if the data has a time component as well).
- Multi-dimensional Pandas dataframes.

Dask (Parallel computing)

- Dynamic task scheduling system.
- “Big Data” collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments.

4. Modeling

Keras (Simple deep learning)

- Higher level interface for Theano and Tensorflow

PyMC3 (Probabilistic programming)

- Contains the most high end research from labs in academia
- Powerful Bayesian statistical modeling

The Case for R

Key quote: *“There should be an interface to the very best numerical algorithms available.”* – John Chambers

John Chambers actually created S, the precursor to R, but the spirit of R is the same.

Why R is Great for Data Science

- R was created in 1992, after Python, and was therefore able to learn from Python’s lessons.
- Rcpp makes it very easy to extend R with C++.
- RStudio is a mature and excellent IDE.
- CRAN is a candyland filled with machine learning algorithms and statistical tools.
- The Caret package makes it easy to use different algorithms from 1 single interface, much like what Scikit-Learn has done for Python

Advances in Modern R for Data Science

1. Collecting Data

Feather (Fast reading and writing of data to disk)

- Same as for Python

Haven (Interacts with SAS, Stata, SPSS data)

- Reads SAS and brings it into a dataframe

Readr (Reimplements read.csv into something better)

- read.csv sucks because it takes strings into factors, it’s slow, etc
- Creates a contract for what the data features should be, making it more robust to use in production
- Much faster than read.csv

JsonLite (Handles JSON data)

- Intelligently turns JSON into matrices or dataframes

2. Data Visualization

ggplot2 (ggplot2 was recently massively upgraded)

- Recently had a very significant upgrade (to the point where old code will break)
- You can do faceting and zoom into facets

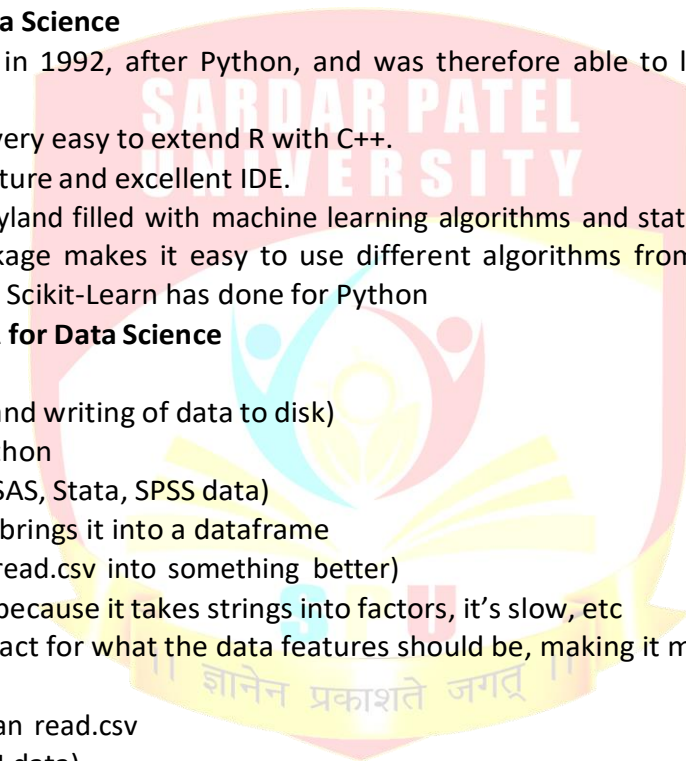
htmlwidgets (Reusable components)

- Brings of the best of JavaScript visualization to R

Leaflet (Interactive maps for the web)

- Nice Javascript maps that you can embed in web applications

Tilegramsr (Proportional maps)



- Create maps that are proportional to the population
- Makes it possible to create more interesting maps than those that only highlight major cities due to population density

3. Cleaning & Transforming Data

Dplyr (Swiss army chainsaw)

- The way R should've been from the first place
- Has a bunch of amazing joins
- Makes data wrangling much more humane

Broom (Tidy your models)

- Fixes model outputs (gets around the weird incantations needed to see model coefficients)
- tidy, augment, glance

Tidy_text (Text as tidy data)

- Text mining using dplyr, ggplot2, and other tidy tools
- Makes natural language processing in R much easier

4. Modeling

MXNet (Simple deep learning)

- Intuitive interface for building deep neural networks in R
- Not quite as nice as Keras

TensorFlow

- Now has an interface in R



Subject Name: **Data Science & Big data**

Subject Code: **CS-7005**

Semester: **7th**



Unit-2

Topics to be covered

Statistical & Probabilistic analysis of Data: Multiple hypothesis testing, Parameter Estimation methods, Confidence intervals, Bayesian statistics and Data Distributions.

Statistical analysis of data

Statistics is basically a science that involves data collection, data interpretation and finally, data validation. Statistical data analysis is a procedure of performing various statistical operations. It is a kind of quantitative research, which seeks to quantify the data, and typically, applies some form of statistical analysis. Quantitative data basically involves descriptive data, such as survey data and observational data.

Statistical data analysis generally involves some form of statistical tools, which a layman cannot perform without having any statistical knowledge. There are various software packages to perform statistical data analysis. This software includes Statistical Analysis System (SAS), Statistical Package for the Social Sciences (SPSS), Stat soft, etc.

Data in statistical data analysis consists of variable(s). Sometimes the data is univariate or multivariate. Depending upon the number of variables, the researcher performs different statistical techniques.

If the data in statistical data analysis is multiple in numbers, then several multivariates can be performed. These are factor statistical data analysis, discriminant statistical data analysis, etc. Similarly, if the data is singular in number, then the univariate statistical data analysis is performed. This includes t test for significance, z test, f test, ANOVA one way, etc.

The data in statistical data analysis is basically of 2 types, namely, continuous data and discrete data. The continuous data is the one that cannot be counted. For example, intensity of a light can be measured but cannot be counted. The discrete data is the one that can be counted. For example, the number of bulbs can be counted.

The continuous data in statistical data analysis is distributed under continuous distribution function, which can also be called the probability density function, or simply pdf.

The discrete data in statistical data analysis is distributed under discrete distribution function, which can also be called the probability mass function or simple pmf.

We use the word 'density' in continuous data of statistical data analysis because density cannot be counted, but can be measured. We use the word 'mass' in discrete data of statistical data analysis because mass cannot be counted.

There are various pdf's and pmf's in statistical data analysis. For example, Poisson distribution is the commonly known pmf, and normal distribution is the commonly known pdf.

These distributions in statistical data analysis help us to understand which data falls under which distribution. If the data is about the intensity of a bulb, then the data would be falling in Poisson distribution.

There is a major task in statistical data analysis, which comprises of statistical inference. The statistical inference is mainly comprised of two parts: estimation and tests of hypothesis. Estimation in statistical data analysis mainly involves parametric data—the data that consists of parameters. On the other hand, tests of hypothesis in statistical data analysis mainly involve non parametric data—the data that consists of no parameters.

Traditional methods for statistical analysis – from sampling data to interpreting results – have been used by scientists for thousands of years. But today's data volumes make statistics ever more valuable and powerful. Affordable storage, powerful computers and advanced algorithms have all led to an increased use of computational statistics.

Whether you are working with large data volumes or running multiple permutations of your calculations, statistical computing has become essential for today's statistician. Popular statistical computing practices include:

- **Statistical programming** – From traditional analysis of variance and linear regression to exact methods and statistical visualization techniques, statistical programming is essential for making data-based decisions in every field.
- **Econometrics** – Modeling, forecasting and simulating business processes for improved strategic and tactical planning. This method applies statistics to economics to forecast future trends.
- **Operations research** – Identify the actions that will produce the best results – based on many possible options and outcomes. Scheduling, simulation, and related modeling processes are used to optimize business processes and management challenges.
- **Matrix programming** – Powerful computer techniques for implementing your own statistical methods and exploratory data analysis using row operation algorithms.
- **Statistical visualization** – Fast, interactive statistical analysis and exploratory capabilities in a visual interface can be used to understand data and build models.
- **Statistical quality improvement** – A mathematical approach to reviewing the quality and safety characteristics for all aspects of production.

Statistical analysis in Big Data Processing

Data is piling up and people are wondering what can be done with it. In this age of Information, there is no scarcity of data; data is overpowering. The key lies in sifting through the

overwhelming volume of data that's available to businesses and organizations, thereby interpreting its implications correctly. Perhaps, a few statistical analysis methods can help find some nuggets of gold buried in all that noise.

There obviously are thousands of big data tools, all promising to save your time and money, and also uncover unprecedented business insights. While all of that may be true, navigating the maze of big data tools could be quite overwhelming and tricky. We suggest you start your data analysis efforts with a handful of basic, yet effective, statistical analysis methods for big data, before advancing to the more sophisticated techniques.

Listed here are five fundamental statistical analysis methods that you can start with, along with the pitfalls that you must watch out for.

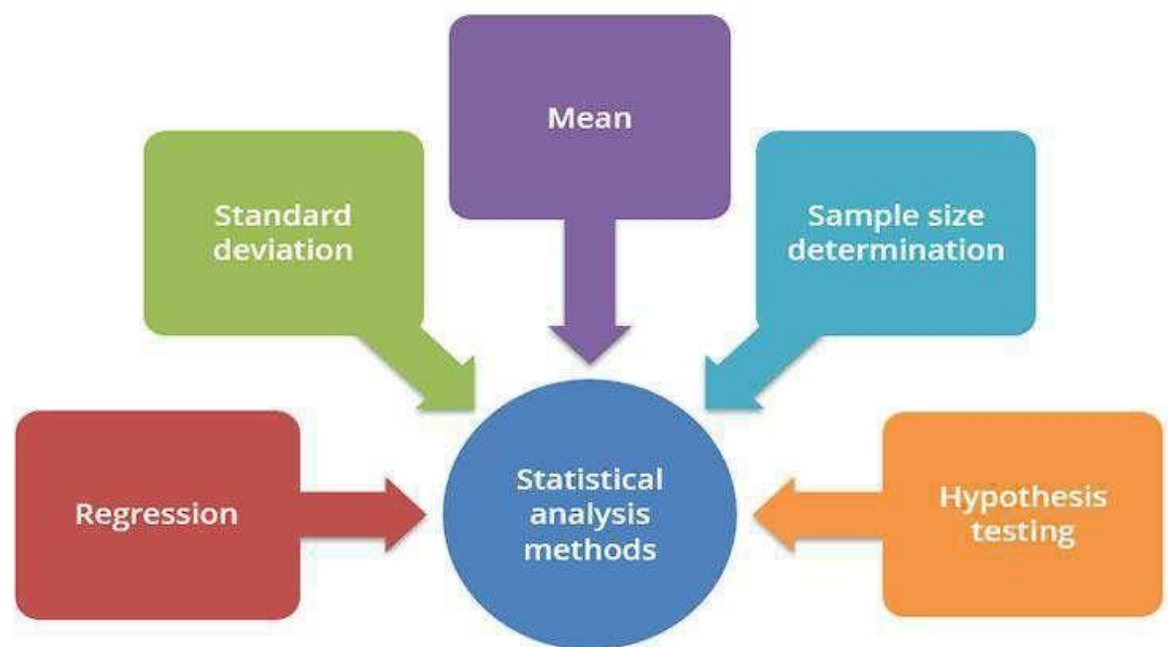


Figure 1: Fundamentals statistical methods

Various data statistical analysis methods used in data science are as follows:

1. Mean

More commonly known as the average, the arithmetic mean is the sum of a list of numbers divided by the number of items on the list. Using the method of mean you can determine the overall trend of a data set or obtain a rapid snapshot of your data. This method offers the advantage of simplistic and quick calculation.

Pitfall

If used alone, the mean is a dangerous tool, and in some data sets, it is also closely related to the mode and median. Remember, in a data set with skewed distribution or a high number of outliers, mean simply do not provide the kind of accuracy that's needed for a nuanced decision.

2. Standard deviation

This is a measure of the spread of data around the mean. While a high standard deviation means the data spreads widely from the mean, a low deviation signals that most data aligns with the mean. This statistical analysis method is useful for quickly determining the dispersion of data points.

Pitfall

Similar to mean, the standard deviation too, is deceptive if taken alone. For example, if the data has many outliers or a strange pattern such as a non-normal curve, standard deviation won't give you all the information needed.

3. Regression

The relationship between dependent and explanatory variables is modeled using the method of regression. The regression line helps determine whether those relationships are strong or weak, as well as the trends over time.

Pitfall

Regression is not very nuanced and the outliers on a scatter plot (as well as the reasons for them) matter significantly. For example, an outlying data point may represent your highest selling product. The nature of regression line is such that it tempts you to ignore these outliers.

4. Sample size determination

A sample does the job just as well when the data set is large and you don't want to collect information from each element of the dataset. The trick lies in determining the right size for the sample to be accurate.

Pitfall

When analyzing a new, untested variable in a data set, you need to rely on certain assumptions, which might be completely inaccurate. If such an error passes onto your sample size determination, it can affect the rest of your statistical data analysis.

5. Hypothesis testing

This method is about testing if a certain premise is actually true for your data set. The results of this test are statistically significant if the results couldn't have happened by a random chance.

Pitfall

To be rigorous, watch out for the placebo effect, as well as the Hawthorne effect. These statistical analysis methods add a lot of insight to your decision-making portfolio. Missing out on these methods for a lot of other fancy tools and techniques will be non-judicious on your part.

Probabilistic analysis of data

A widespread application of such an analysis is weather forecasting: for more than a century, hundreds of weather stations around the world record various important parameters such as the air temperature, wind speed, precipitation, snowfall etc. Based on these data, scientists build models reflecting seasonal weather changes (depending on the time of the year) as well as the global trends – for example, temperature change during the last 50 years. These models

are used to provide weather forecast for government and commercial organizations. In a typical forecast, the predicted values are not assigned probability: "In May, the maximum daily air temperature is expected to be 22 degrees Celsius."

In contrast to the predictions based on time series analysis, when performing probabilistic analysis, you get not just a single value as a forecast, but a probabilistic model that accounts for uncertainty. In this scenario, you would obtain a continuous range of values and assigned probabilities. Of course, for real world applications, it is more practical to deal with specific values, so the probabilistic models are used to obtain predictions at fixed probability levels. Considering the above example, a forecast might look like: "In May, the maximum daily air temperature will be 22 degrees Celsius with 95% probability."

So can distribution fitting be useful when analyzing time series data? The answer depends on the goals of your analysis – i.e. what kind of information you want to derive from your data. If you want to understand the connection between the predicted values and the probability, you should fit distributions to your data (just keep in mind that in this case the "time" variable will be unused). On the other hand, if you need to identify seasonal patterns or global trends in your data, you should go with the "classical" time series analysis methods.

An important domain of statistics is called hypothesis testing. The basic idea is that you think that there is a trend in your data, and you want to assess how likely it is to be a real world phenomenon versus just a fluke. Hypothesis testing doesn't address the question of how strong a trend is; it's designed for situations where the data is too sparse to quantify trends reliably. A prototypical example for hypothesis testing is this: I have a coin that might or might not be fair. I throw it 10 times and get 9 heads. How likely it is that this coin is loaded?

The first thing to realize is that we can't answer the question "how likely is this coin to be loaded?" In order to get that number, we would need some a priori knowledge or guess about how likely the coin is to be loaded, plus how loaded it would actually be, and there's no principled way to acquire that. So, here is the first big idea in classical statistics: instead of saying how likely the dice is to be loaded given the data, we ask how likely the data is given that the coin is not loaded. Basically, is it plausible that nine heads is just a fluke? We assume a fair coin, compute how likely this skewed data is, and use that to gauge whether the fair coin is believable. So, let's assume a fair coin. Every coin flip can be either heads or tails, and they are all independent. This means that there are $2^{10} = 1024$ possible ways the other important caveat is that you will get false positives. Say you do many hypothesis tests in your life, and you always use 5% as your threshold.

Then of the times when there really is no pattern, 1 in 20 of those times you will find something statistically significant. This phenomenon is a huge problem in the scientific literature, where researchers will often declare victory as soon as the p-value dips low enough to publish a paper.

All probabilistic analysis is based on the idea that (suitably trained and intelligent) people can at least *recognize* good probabilistic arguments presented by someone else, or discovered or

thought of by themselves, but not necessarily generate good assessments. The very fact that there was correspondence about the gambles—and occasionally some disputes about them—indicated that people do not automatically assess probabilities in the same way, or accurately (e.g., corresponding to relative frequencies, or making good gambling choices).

The von Neumann and Morgenstern work, however, does involve some *psychological* assumptions that people can engage in 'good' probabilistic thinking. First, they must do so implicitly, so that the choice follows certain 'choice axioms' that allow the construction of an expected utility model—i.e., a model that *represents* choice as a maximization of implicit expected utility; that in turn requires that probabilities at the very least follow the standard axioms of probability theory.

It also implies considering conditional probabilities in a rational manner, which is done only when implicit or explicit conditional probabilities are consistent with Bayes' theorem. Thus, the von Neumann and Morgenstern work required that people be 'Bayesian' in a consistency sense, although that term is sometimes used to imply that probabilities should at base be interpreted as degrees of belief.

Another way in which probability assessment must be 'good' is that there should be some at least reasonable approximation between probabilities and long-term relative frequencies; in fact, under particular circumstances (of interchangeability and indefinitely repeated observations), the probabilities of someone whose belief is constrained by Bayes' theorem must approximate relative frequencies.

Are people always coherent? Consider an analogy with swimming. People do swim well, but occasionally we drown. What happens is that there is particular systematic bias in attempting to swim that makes it difficult. We want to hold our heads above water. When, however, we raise our heads to do so, we tend to assume a vertical position in the water, which is one of the few ways of drowning (aside from freezing or exhaustion or being swept away in rough waters). Just as people drown occasionally by trying to hold their heads above water, people systematically deviate from the rules of probabilistic thinking. Again, however, the emphasis is on 'systematic.'

For example, there is now evidence that people's probabilistic judgments are 'sub additive'—in that when a general class is broken into components, the judgmentally estimated probabilities assigned to disjoint components that comprise the class sum to a larger number than the probability assigned to the class. That is particularly true in memory, where, for example, people may recall the frequency with which they were angry at a close friend or relative in the last month and the frequency with which they were angry at a total stranger, and the sum of the estimates is greater than an independent estimate of being angry period (even though it is possible to be angry at someone who is neither a close friend or relative nor a total stranger). The clever opponent will then bet against the occurrence of each component but on the occurrence of the basic event, thereby creating a Dutch Book.

Multiple Hypothesis Testing

In real situations, we often have several hypotheses that we want to test and see if any one of them is correct. For example, we might test 20 different ads out on different groups of users and see if any one of them increases the click-through rate (relative to a known baseline click-

through probability) with confidence threshold 0.05. That is, we will calculate the p-value for each ad and consider all those with p-value < 0.05 to pass our test.

The problem is that while each individual ad has only a 5% chance of passing our test by dumb luck, there is a very good chance that some ad will pass by dumb luck. So, the idea is that we must tighten p-value constraints for the individual ads, in order to have a p-value of 0.05 for the overall test.

A standard, conservative, and surprisingly simple solution to this problem is called the Bonferroni correction. Say you have n different ads, and you want a p-value of α for your overall test. Then you require that an individual ad passes with the smaller p-value of α/n . We then see that if the null hypothesis is true (i.e., no ad is actually an improvement), then our probability of errantly having a test pass is:

$$\begin{aligned}\Pr[\text{Some ad passes}] &= 1 - \Pr[\text{Every ad fails}] \\ &= 1 - \left(1 - \frac{\alpha}{n}\right)^n \\ &= 1 - \left(1 - n \left(\frac{\alpha}{n}\right) + \{\text{higher order terms}\}\right) \\ &= \alpha\end{aligned}$$

The Bonferroni correction is the most common multiple hypothesis correction that you will see, but you should be aware that it is unnaturally stringent. In particular, it assumes that every test you run is independent of every other test you run: if test X does not pass, then test Y is still as likely to pass as it ever was. But that often isn't the case, especially if your different tests are related in some way. For example, say that you have a database of people's height, weight, and income, and you are trying to see whether there is a statistically significant correlation between their income and some other feature.

Height and weight are strongly correlated between people, so if the tall people aren't richer, then the heavy people won't be either: they are more or less the same people! Intuitively, you want to say that we are testing 1.3 hypotheses or something similar, rather than two. Bonferroni would apply if you tested height on one sample of people and weight on an independent set. There are other, complicated ways to adjust for situations such as this, but data scientists are unlikely to need anything beyond Bonferroni corrections.

Parameter Estimation

Hypothesis testing is, as I mentioned earlier, about measuring *whether* effects are present, but not about quantifying their magnitude. The latter falls under the umbrella of "parameter estimation," where we try to estimate the underlying parameters that characterize distributions.

In parameter estimation, we assume that the data follows some functional form, such as a normal distribution with mean μ and standard deviation α . We then have some method to estimate these parameters, given the data, as follows:

In this case, these numbers are called the sample mean and the sample variance. To step back and cast this in statistical terminology, μ and σ are both test statistics that we calculated from the data. If we threw out our dataset and gathered a new one of the same size from the real-world distribution, μ and σ would be a little bit different. They have their own probability distributions and are themselves random variables. The question then is how well we can take these random variables as indicators of the actual μ and σ .

We will talk about the more general problem of confidence intervals later, but for now, there are two pieces of terminology you should know whenever you are talking about estimators:

- Consistency: An estimator μ^\wedge is “consistent” if, in the limit of having many different points in your dataset, it converges to the real μ .
 - Bias: An estimator is “unbiased” if the expectation value of μ^\wedge is the real μ .
- Most estimators you might cook up on your own are probably going to be consistent. The μ and σ are both consistent. However, only μ is unbiased; σ is on average an underestimate of the true standard deviation.

This surprised me the first time I learned it, so let me explain. It’s easiest if you imagine that there are only two data points that we use to find the mean and standard deviation. Then μ will be located exactly in-between them, at the place that minimizes:

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

If we had set μ to be anywhere else, then equation value would have been somewhat larger. However, the real μ is somewhere else, because our μ^\wedge is only an estimate of μ . It is an unbiased estimate.

so μ is equally likely to be more or less than μ^\wedge , but in either case equation would get larger. Basically, we are making μ^\wedge an overly good fit for our data, so on average, and the sample deviation will be less than the real-world deviation.

An unbiased estimator of the standard deviation is:

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

This will always be somewhat larger than our original expression, but they converge to the same number as N goes to infinity.

Confidence Intervals

When we are trying to estimate a parameter of a real-world distribution from data, it is often important to give confidence intervals, rather than just a single best-fit number. The most common use for confidence intervals is in calculating the mean of the underlying distribution. If you want to calculate something else, you can often massage your data to turn it into a calculation of the mean. For example, let's say you want to estimate the standard deviation of a random variable X . Well, in that case, we can see that:

$$\sigma_X = \sqrt{E[(X - E[X])^2]}$$

$$\sigma_X^2 = E[(X - E[X])^2]$$

so we can estimate the standard deviation of X by finding the mean of $(X - E[X])^2$ and taking the square root. If we can put confidence intervals on taking a mean, we can put confidence intervals on a lot of other things we might want to estimate.

The typical metric to use is the “standard error of the mean” or SEM. If you see a mean reported as 4.1 ± 0.2 , then generally 4.1 is the best-fit mean that you get by averaging all of the numbers, and 0.2 is the SEM. You can calculate the SEM in Python as follows:

```
>>> from scipy.stats import sem
>>> std_err = sem(my_array)
```

If you assume that the underlying distribution is normal, then the SEM will be the standard deviation of the estimate of the mean, and it has a lot of nice mathematical properties. The most notable is that if μ^{\wedge} is your sample mean, then the interval:

$$[\mu^{\wedge} - z * \text{SEM}, \mu^{\wedge} + z * \text{SEM}]$$

which will contain the mean 95% of the time, 99% of the time, or any other confidence threshold you want, depending on how you set the coefficient z . Increase that coefficient and you can increase your confidence. The following table shows several typical confidences that people want and the corresponding coefficients:

Confidence (%)	Coefficient
99	2.58
95	1.96

It is very tempting to say that there is a “95% chance that the mean is within our interval,” but that’s a little dicey. The real-world mean is within your interval or it isn’t – you just happen not to know which one. The more rigorous interpretation is to say that if the distribution is normally distributed (no matter its mean and standard deviation), and you take a sample of N points and calculate the confidence interval from them, then there is a 95% chance that the interval will contain the actual mean. The randomness is in the data you sample, not in the fixed parameters of the underlying distribution.

Of course, all of this hinges on the assumption that the real-world distributions are normal. All of the theorems about how to interpret the confidence intervals go out the window when you let go of that assumption, but in practice, things still often work out fine. Statisticians have worked out alternative formulas for the intervals that give the same sorts of guarantees for other types of distributions, but you don’t see them often in data science.

The SEM confidence interval is based on the idea that we want to make sure that the true value of the parameter is contained in the interval some known percentage of the time. Another paradigm that you see is that we want the interval to contain all “plausible” values for the parameter. Here “plausible” is defined in terms of hypothesis testing: if you hypothesize that your parameter has some value and calculate an appropriate test statistic from the data, do you get a large p -value?

Bayesian Statistics

Bayesian statistics, similar to classical statistics, assumes that some aspect of the world follows a statistical model with some parameters: similar to a coin that is heads with probability p . Classical statistics picks the value of the parameter that best fits the data (and maybe adds in a confidence interval); there is no room for human input. In Bayesian statistics though, we start off with a probability distribution over all possible values of the parameter that represents our confidence that each value is the “right” one. We then update our confidence as every new data point becomes available. In a sense, Bayesian statistics is the science of how to refine our guesses in light of data.

The mathematical basis for Bayesian statistics is Bayes’ theorem, which looks innocuous enough. For any random variables X and T (which may be vectors, binary variables, or even something very complex), it says that:

$$P(T|D) = \frac{P(T)P(D|T)}{P(D)}$$

As stated, Bayes’ theorem is just a simple probability theorem that is true of any random variables T and D . However, it becomes extremely powerful when we have T be a real-world

parameter that we are trying to guess from the data. In this case, T isn't actually random – it is fixed but unknown, and the “probability distribution” of T is a measure of our own confidence, defined over all the values T , could possibly take.

The left-hand side of Bayes' theorem gives us what we were unable to have in classical statistics: the “probability” that the real-world parameter is equal to a certain value, given the data we have gathered. On the right-hand side, we see that it requires the following:

1. $P(T)$: Our confidence about the parameter before we got the data
2. $P(D|T)$: How likely the data we saw was assuming that the parameter had a particular value. This often is known or can be modeled well.
3. $P(D)$: The probability of seeing the data we saw, averaged over all possible values the parameter could have had.

To take a concrete example, say that we have a user who may be a male or female, but we don't know which. So, we set our prior guess to say there is a 50% confidence they are male and 50% that they are female. Then say, we learn that they have long hair, so we update our confidence to:

$$\begin{aligned}
 P(\text{Female}|\text{Long Har}) &= \frac{P(\text{Female})P(\text{Long Har}|\text{Female})}{P(\text{Long Har})} \\
 &= \frac{\frac{1}{2} P(\text{Long Har}|\text{Female})}{\frac{1}{2} P(\text{Long Har}|\text{Female}) + \frac{1}{2} P(\text{Long Har}|\text{Male})} \\
 &= \frac{P(\text{Long Har}|\text{Female})}{P(\text{Long Har}|\text{Female}) + P(\text{Long Har}|\text{Male})}
 \end{aligned}$$

Personally, I sometimes have trouble remembering Bayes' theorem: the formula for the updated probability is just a little clunky. For me, it's much easier to think about the *odds*, that is, the *relative* probability of female or male. To update the odds, you just multiply by the relative probability of long hair:

$$\frac{P(\text{Female}|\text{Long Har})}{P(\text{Male}|\text{Long Har})} = \frac{P(\text{Female})P(\text{Long Har}|\text{Female})}{P(\text{Male})P(\text{Long Har}|\text{Male})}$$

Data Distributions

1. Model based distributions

- A. Models predict more than the range of values a group of points will fall into. They also predict the *distribution* of the points.
 - a) A distribution contains information about probabilities associated with the points. For example, distributions reveal which values are typical, which are rare, and which are seemingly impossible. A distribution also reveals the “best guess”

for predicting future values, as well as how certain or uncertain that guess may be.

- b) You can think of a distribution as the “boundary” conditions on a variable’s values.

2. Visualizing Distributions

- A. The easiest way to understand a distribution is to visualize it.
 - a) `library(mosaic); library(lattice)`
 - b) `dotPlot()`
 - c) `histogram()`
 - d) `freqpolygon()`
 - e) `densityplot()`
- B. So far we have examined a continuous variable. A continuous variable can take any value on a segment of the continuous number line. Other variables are *discrete*, the values of a discrete variable fall in a countable set of values that may or may not have an implied order.
 - a) Visualize a discrete distribution with a bar graph.
 - b) `bargraph()`

3. Statistics

- A. The most useful properties of a distribution can be described with familiar statistics.
 - a) Statistic defined - a number computed with an algorithm, that describes a group of individuals
 - b) Motivating examples
- B. Typical values
 - a) `median()` - “typical” value
 - b) `mean()` - best guess
 - c) `min()` - smallest value
 - d) `max()` - largest value
 - e) Describe a prediction as $\bar{Y} + \epsilon$, where ϵ denotes the structure of the distribution.
- C. Uncertainty
 - a) The “more” a variable varies, the less certain you can be when predicting its values. The spread of the distribution quantifies this uncertainty.
 - I. `range()`
 - II. `var()`
 - III. `sd()`
 - IV. `IQR()`
 - V. `bwplot()`

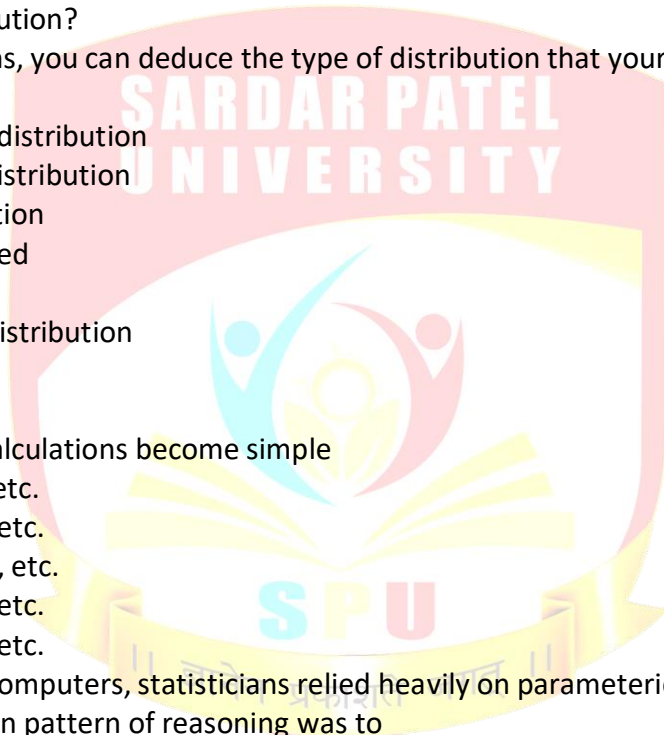
4. Probability

- A. What is the relationship between a distribution and an individual case?
- B. You can use probability to reason from a distribution to individual cases

- a) Probability defined as a frequency. Each variable takes each value with a certain frequency.
- b) If the next observation is similar to the observations in your distribution, it will have the same probability of taking each state as the previous observations.
- C. Simulation. You can use the frequencies of a distribution to simulate new values from the distribution, a technique known as monte carlo simulation.
 - a) `sample()`
 - b) `resample()`
 - c) `do()`
 - d) prediction intervals
 - e) `quantile()`

5. Parametric distributions

- A. How do you know that you've collected enough data to have an accurate picture of a variable's distribution?
- B. In some situations, you can deduce the type of distribution that your data follows
 - a) `plotDist()`
 - b) Binomial distribution
 - c) Normal distribution
 - d) t distribution
 - e) Chi squared
 - f) F
 - g) Poisson distribution
 - h) uniform
 - i) etc.
- C. In these cases, calculations become simple
 - a) `rnorm()`, etc.
 - b) `pnorm()`, etc.
 - c) `xpnorm()`, etc.
 - d) `dnorm()`, etc.
 - e) `qnorm()`, etc.
- D. Before modern computers, statisticians relied heavily on parametric distributions.
 - a) A common pattern of reasoning was to
 - I. Assume that data follows a distribution
 - II. Try to disprove the assumption:
 - 1. `qqmath()`, `xqqmath()`, `qqplot()`
 - 2. Goodness of fit tests
 - III. Proceed as if the assumption were true
 - b) But this reasoning has a weakness: it relies on an assumption that has not been proven. Moreover, we're tempted to believe that the assumption is true because the assumption will help us.
 - I. Unfortunately, tests that would disprove the assumption if it were false are not very powerful



- II. Once you assume a distribution, all of your results are “conditional” on the assumption being correct. This makes it very difficult to interpret parametric probabilities.
- E. Modern computing makes it easier to avoid parametric assumptions
 - a) It is easier to collect more data
 - b) It is easier to calculate probabilities computationally, which reduces the need to calculate them analytically (from a distribution)
 - c) However, it is important to understand parametric distributions because many methods of data science rely on them

5. Variation

- A. Distributions provide a tool for working with variation.
 - a) Variation is what makes the world dynamic and uncertain.
 - b) As a scientist, your goal is to understand and explain variation (natural laws explain how a variable varies), which makes distributions a very important tool.
- B. Variation defined - the tendency for a value to change each time we measure it
- C. Variation is omnipresent
 - a) Example of speed of light
- D. Causes of variation
 - a) unidentified laws
 - b) randomness
- E. Models partition the variation in a variable into explained variation and unexplained distribution.
 - a) The distribution predicted by the model captures the unexplained variation
 - b) As you add components of the law to your model, you will transfer unexplained variation to explained variation. As a result, the distributions predicted by the model will become narrower, and the individual predictions more uncertain.
 - c) Eventually, a distribution will collapse to a single point as you include the entire law in your model.

6. Summary

- A. Scientists search for natural laws. A complete law provides a function between values and a value. An incomplete law is a model. It provides a function between values and a distribution.
- B. Distributions reveal useful information, but the information is probabilistic.
- C. As models improve (include more components of the underlying law), the distributions that they predict become narrower.
 - a) As a result, their individual predictions become more certain.
 - b) If a model includes all of the components of a law, its distributions will collapse into single points. The model has become a law, a function from a set of values to a single value.
- D. Unfortunately, the nature of distributions makes models a little more difficult to work with than laws. Recall the two steps of the scientific method.
 - a) Identify laws as patterns in data

- I. Which function is “correct”?
 - b) Test laws against new data
 - I. Does the data “disprove” the model?
- E. Due to these problems, data scientists must modify the tactics they use to implement the strategy of the scientific method. We’ll examine these tactics in the rest of this Part.



Subject Name: **Data Science & Big data**

Subject Code: **CS-7005**

Semester: **7th**



Unit-3

Topics to be covered

Introduction to machine learning: Supervised & unsupervised learning, classification & clustering Algorithms, Dimensionality reduction: PCA & SVD, Correlation & Regression analysis, Training & testing data: Overfitting & Underfitting.

Introduction to Machine Learning: Supervised Machine Learning

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. Supervised learning problems can be further grouped into regression and classification problems.

- **Classification:** A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- **Regression:** A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

Unsupervised Machine Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answer and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

- **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Some popular examples of unsupervised learning algorithms are:

- k-means for clustering problems.
- Apriori algorithm for association rule learning problems.

Semi-Supervised Machine Learning

Problems where you have a large amount of input data (X) and only some of the data is labeled (Y) are called semi-supervised learning problems. These problems sit in between both supervised and unsupervised learning.

A good example is a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled. Many real world machine learning problems fall into this area. This is because it can be expensive or time-consuming to label data as it may require access to domain experts. Whereas unlabeled data is cheap and easy to collect and store.

You can use unsupervised learning techniques to discover and learn the structure in the input variables. You can also use supervised learning techniques to make best guess predictions for the unlabeled data, feed that data back into the supervised learning algorithm as training data and use the model to make predictions on new unseen data.

Summary

- **Supervised:** All data is labeled and the algorithms learn to predict the output from the input data.
- **Unsupervised:** All data is unlabeled and the algorithms learn to inherent structure from the input data.
- **Semi-supervised:** Some data is labeled but most of it is unlabeled and a mixture of supervised and unsupervised techniques can be used.

Classification Concepts

In classification, the idea is to predict the target class by analysis the training dataset. This could be done by finding proper boundaries for each target class. In a general way of saying, Use the training dataset to get better boundary conditions which could be used to determine each target class. Once the boundary conditions determined, the next task is to predict the target class as we have said earlier. The whole process is known as classification.

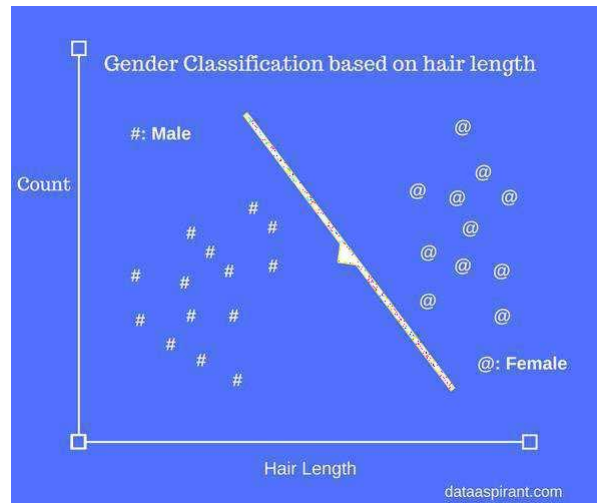


Figure 1: Classification

Target class examples:

- Analysis the customer data to predict whether he will buy computer accessories (Target class: Yes or No)
- Gender classification from hair length (Target classes: Male or Female)
- Classifying fruits from each fruit feature like colour, taste, size, weight (Target class: Apple, Orange, Cherry, Banana)

Let's understand the concept of classification with gender classification using hair length. To classify gender (target class) using hair length as feature parameter we could train a model using any classification algorithms to come up with some set of boundary conditions which can be used to differentiate the male and female genders using hair length as the training feature. In gender classification case the boundary condition could be the proper hair length value. Suppose the differentiated boundary hair length value is 15.0 cm then we can say that if hair length is less than 15.0 cm then gender could be male or else female. Some classification algorithms listed below.

Classification Algorithms

1. Logic Regression

Don't get confused by its name! It is a classification not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).

Again, let us try and understand this through a simple example.

Let's say your friend gives you a puzzle to solve. There are only 2 outcome scenarios – either you solve it or you don't. Now imagine that you are being given wide range of puzzles / quizzes in an attempt to understand which subjects you are good at. The outcome to this study would be something like this – if you are given a trigonometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you.

Coming to the math, the log odds of the outcome is modeled as a linear combination of the predictor variables.

$\text{odds} = p / (1-p)$ = probability of event occurrence / probability of not event occurrence

$\ln(\text{odds}) = \ln(p/(1-p))$

$\text{logit}(p) = \ln(p/(1-p)) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 \dots + b_kX_k$

Above, p is the probability of presence of the characteristic of interest. It chooses parameters that maximize the likelihood of observing the sample values rather than that minimize the sum of squared errors (like in ordinary regression).

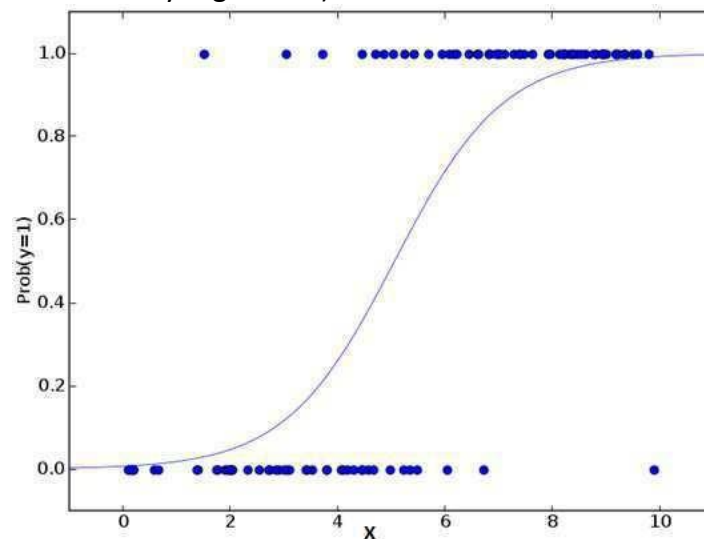
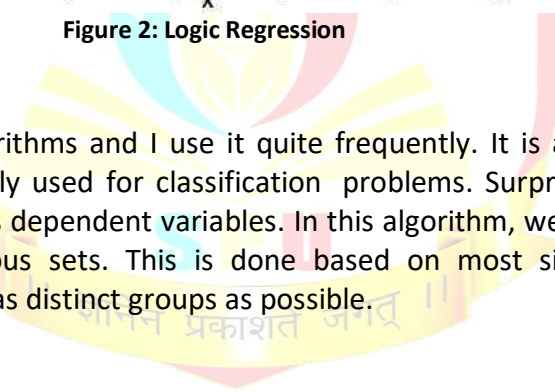


Figure 2: Logic Regression

2. Decision Tree

This is one of my favorite algorithms and I use it quite frequently. It is a type of supervised learning algorithm that is mostly used for classification problems. Surprisingly, it works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets. This is done based on most significant attributes/independent variables to make as distinct groups as possible.



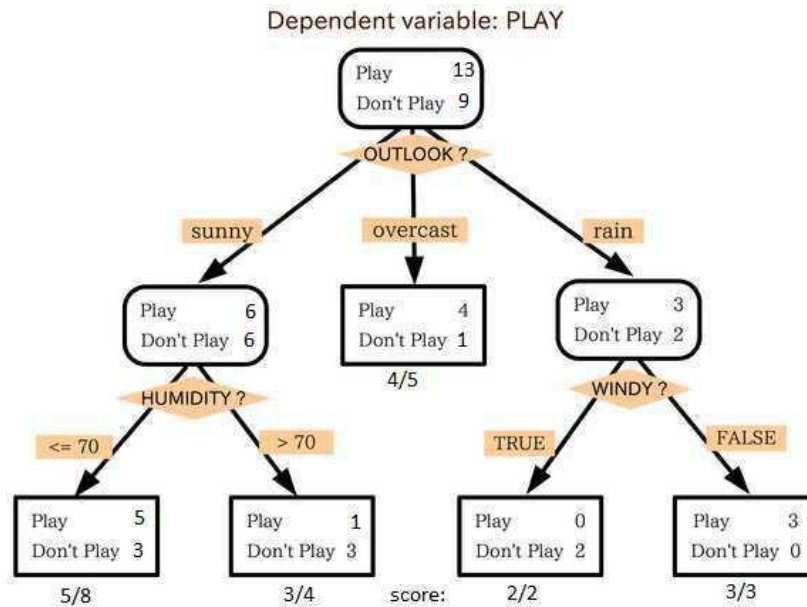


Figure 3: Decision Tree

In the image above, you can see that population is classified into four different groups based on multiple attributes to identify 'if they will play or not'. To split the population into different heterogeneous groups, it uses various techniques like Gini, Information Gain, Chi-square, entropy.

The best way to understand how decision tree works is to play Jezzball – a classic game from Microsoft (image below). Essentially, you have a room with moving walls and you need to create walls such that maximum area gets cleared off without the balls.

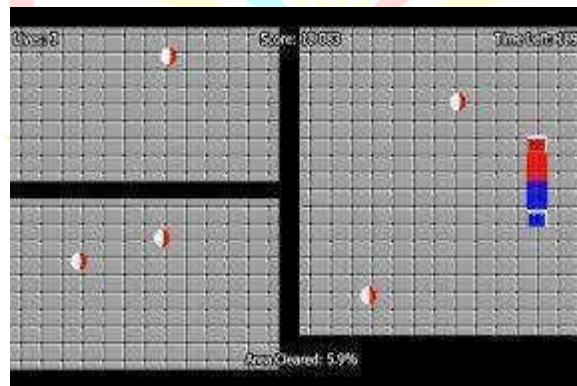


Figure 4: Jezzball Example

So, every time you split the room with a wall, you are trying to create 2 different populations within the same room. Decision trees work in very similar fashion by dividing a population in as different groups as possible.

3. SVM (Support Vector Machine)

It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**).

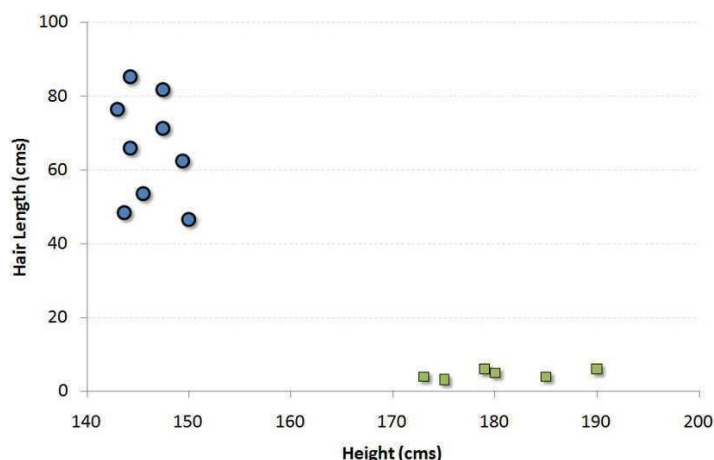


Figure 5: Support Vector Machine

Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

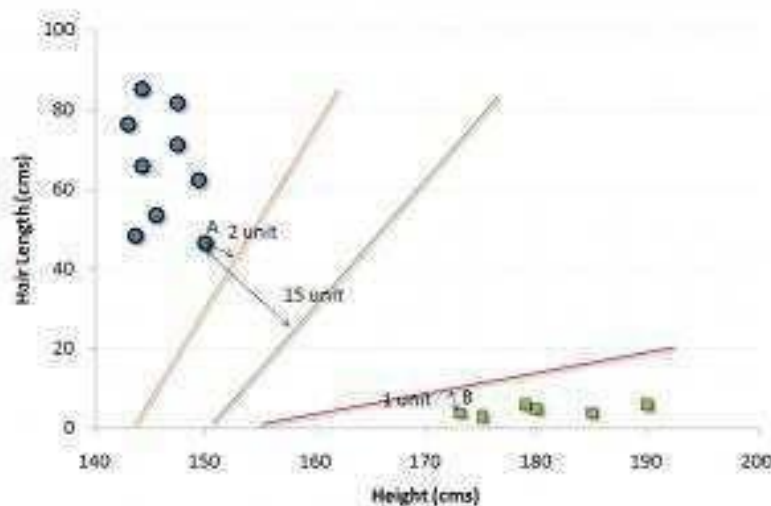


Figure 6: Support Vector Machine Classifier

In the example shown above, the line which splits the data into two differently classified groups is the *black* line, since the two closest points are the farthest apart from the line. This line is our

classifier. Then, depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

Think of this algorithm as playing JezzBall in n-dimensional space. The tweaks in the game are:

- You can draw lines / planes at any angles (rather than just horizontal or vertical as in classic game)
- The objective of the game is to segregate balls of different colors in different rooms.
- And the balls are not moving.

4. Naive Bayes

It is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier would consider all of these properties to independently contribute to the probability that this fruit is an apple.

Naive Bayesian model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(T|D)$ from $P(T)$, $P(D)$ and $P(T|D)$. Look at the equation below:

$$P(T|D) = \frac{P(T)P(D|T)}{P(D)}$$

Here,

- $P(T|D)$ is the posterior probability of *class (target)* given *predictor (attribute)*.
- $P(T)$ is the prior probability of *class*.
- $P(D|T)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(D)$ is the prior probability of *predictor*.

Example: Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play'. Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it:

Step 1: Convert the data set to frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

5. kNN (k- Nearest Neighbors)

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modeling.

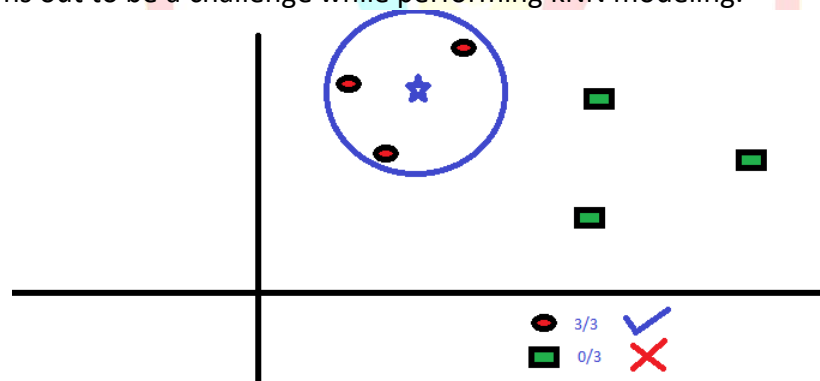


Figure 7: kNN Example

KNN can easily be mapped to our real lives. If you want to learn about a person, of whom you have no information, you might like to find out about his close friends and the circles he moves in and gain access to his/her information!

Things to consider before selecting kNN:

- KNN is computationally expensive

- Variables should be normalized else higher range variables can bias it
- Works on pre-processing stage more before going for kNN like outlier, noise removal

Application of Classification Algorithms

- Email spam classification
- Bank customer's loan pay bank willingness prediction.
- Cancer tumor cells identification.
- Sentiment analysis.
- Drugs classification
- Facial key points detection
- Pedestrians' detection in an automotive car driving.

Clustering Concepts

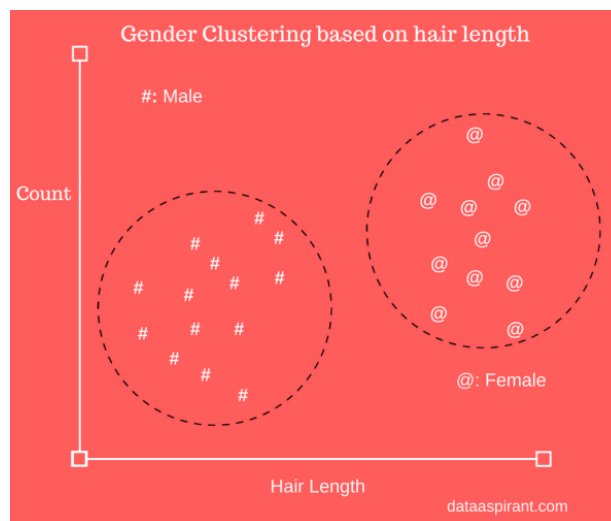


Figure 8: Clustering Example

In clustering the idea is not to predict the target class as like classification, it's more ever trying to group the similar kind of things by considering the most satisfied condition **all the items in the same group should be similar and no two different group items should not be similar**. To group the similar kind of items in clustering, different similarity measures could be used.

Group items Examples:

- While grouping similar language type documents (Same language documents are one group.)
- While categorizing the news articles (Same news category (Sport) articles are one group)

Let's understand the concept with clustering genders based on hair length example. To determine gender, different similarity measure could be used to categories male and female genders. This could be done by finding the similarity between two hair lengths and keep them in the same group if the similarity is less (Difference of hair length is less). The same process could continue until all the hair length properly grouped into two categories.

To get used to different similarity measure to perform clustering we have some popular clustering algorithms.

Clustering Algorithms

Clustering algorithms can be classified into two main categories linear clustering algorithms and Non-linear clustering algorithms.

1. K-Means

It is a type of unsupervised algorithm which solves the clustering problem. Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). Data points inside a cluster are homogeneous and heterogeneous to peer groups. Remember figuring out shapes from ink blots? k means is somewhat similar this activity. You look at the shape and spread to decipher how many different clusters / population are present!

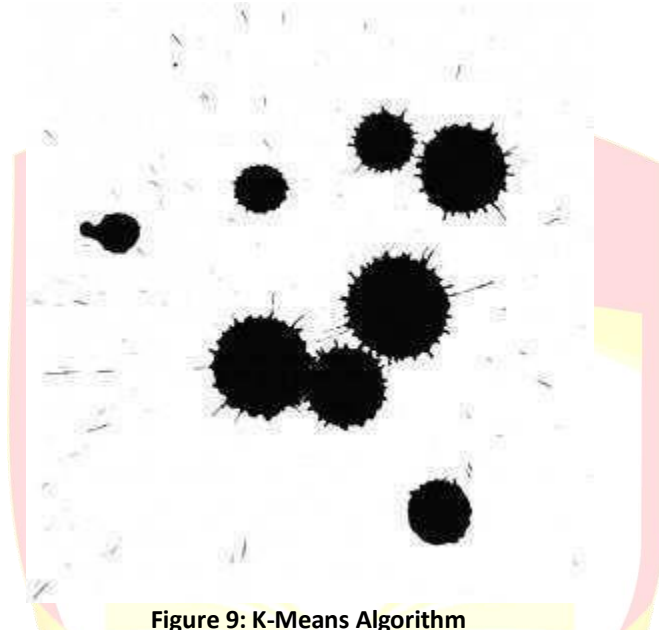


Figure 9: K-Means Algorithm

How K-means forms cluster:

1. K-means picks k number of points for each cluster known as centroids.
2. Each data point forms a cluster with the closest centroids i.e. k clusters.
3. Finds the centroid of each cluster based on existing cluster members. Here we have new centroids.
4. As we have new centroids, repeat step 2 and 3. Find the closest distance for each data point from new centroids and get associated with new k -clusters. Repeat this process until convergence occurs i.e. centroids does not change.

How to determine value of K :

In K-means, we have clusters and each cluster has its own centroid. Sum of square of difference between centroid and the data points within a cluster constitutes within sum of square value for that cluster. Also, when the sum of square values for all the clusters is added, it becomes total within sum of square value for the cluster solution.

We know that as the number of cluster increases, this value keeps on decreasing but if you plot the result you may see that the sum of squared distance decreases sharply up to some value of k , and then much more slowly after that. Here, we can find the optimum number of cluster.

2. Mean-Shift Clustering

Mean shift clustering is a sliding-window-based algorithm that attempts to find dense areas of data points. It is a centroid-based algorithm meaning that the goal is to locate the center points of each group/class, which works by updating candidates for center points to be the mean of the points within the sliding-window. These candidate windows are then filtered in a post-processing stage to eliminate near-duplicates, forming the final set of center points and their corresponding groups. Check out the graphic below for an illustration.



Figure 10: Mean-shift Clustering

Mean-Shift Clustering for a single sliding window

1. To explain mean-shift we will consider a set of points in two-dimensional space like the above illustration. We begin with a circular sliding window centered at a point C (randomly selected) and having radius r as the kernel. Mean shift is a hill climbing algorithm which involves shifting this kernel iteratively to a higher density region on each step until convergence.
2. At every iteration the sliding window is shifted towards regions of higher density by shifting the center point to the mean of the points within the window (hence the name). The density within the sliding window is proportional to the number of points inside it. Naturally, by shifting to the mean of the points in the window it will gradually move towards areas of higher point density.
3. We continue shifting the sliding window according to the mean until there is no direction at which a shift can accommodate more points inside the kernel. Check out the graphic above; we keep moving the circle until we no longer are increasing the density (i.e number of points in the window).
4. This process of steps 1 to 3 is done with many sliding windows until all points lie within a window. When multiple sliding windows overlap the window containing the most points is preserved. The data points are then clustered according to the sliding window in which they reside.

An illustration of the entire process from end-to-end with all of the sliding windows is shown below. Each black dot represents the centroid of a sliding window and each gray dot is a data point.

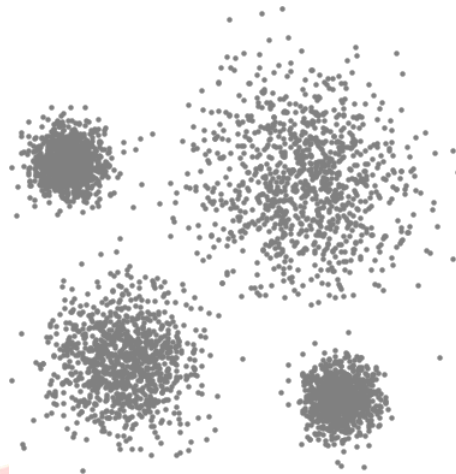


Figure 11: After Men-shift implementation

The entire process of Mean-Shift Clustering

In contrast to K-means clustering there is no need to select the number of clusters as mean-shift automatically discovers this. That's a massive advantage. The fact that the cluster centers converge towards the points of maximum density is also quite desirable as it is quite intuitive to understand and fits well in a naturally data-driven sense. The drawback is that the selection of the window size/radius " r " can be non-trivial.

3. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is a density based clustered algorithm similar to mean-shift, but with a couple of notable advantages. Check out another fancy graphic below and let's get started!

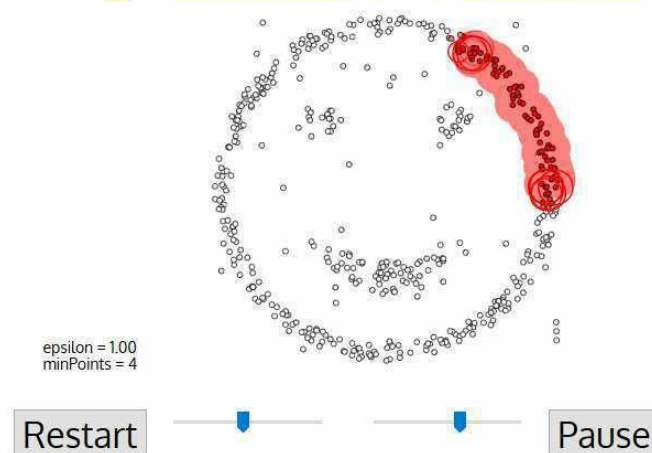


Figure 12: DBSCAN Algorithm

DBSCAN Smiley Face Clustering

1. DBSCAN begins with an arbitrary starting data point that has not been visited. The neighborhood of this point is extracted using a distance epsilon ϵ (All points which are within the ϵ distance are neighborhood points).
2. If there are a sufficient number of points (according to minPoints) within this neighborhood then the clustering process starts and the current data point becomes the first point in the new cluster. Otherwise, the point will be labeled as noise (later this noisy point might become the part of the cluster). In both cases that point is marked as "visited".
3. For this first point in the new cluster, the points within its ϵ distance neighborhood also become part of the same cluster. This procedure of making all points in the ϵ neighborhood belong to the same cluster is then repeated for all of the new points that have been just added to the cluster group.
4. This process of steps 2 and 3 is repeated until all points in the cluster are determined i.e all points within the ϵ neighborhood of the cluster have been visited and labelled.
5. Once we're done with the current cluster, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. This process repeats until all points are marked as visited. Since at the end of this all points have been visited, each point will have been marked as either belonging to a cluster or being noise.

DBSCAN poses some great advantages over other clustering algorithms. Firstly, it does not require a pre-set number of clusters at all. It also identifies outliers as noises unlike mean-shift which simply throws them into a cluster even if the data point is very different. Additionally, it is able to find arbitrarily sized and arbitrarily shaped clusters quite well.

The main drawback of DBSCAN is that it doesn't perform as well as others when the clusters are of varying density. This is because the setting of the distance threshold ϵ and minPoints for identifying the neighborhood points will vary from cluster to cluster when the density varies. This drawback also occurs with very high-dimensional data since again the distance threshold ϵ becomes challenging to estimate.

4. Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)

One of the major drawbacks of K-Means is its naive use of the mean value for the cluster center. We can see why this isn't the best way of doing things by looking at the image below. On the left hand side it looks quite obvious to the human eye that there are two circular clusters with different radius' centered at the same mean. K-Means can't handle this because the mean values of the clusters are a very close together. K-Means also fails in cases where the clusters are not circular, again as a result of using the mean as cluster center.

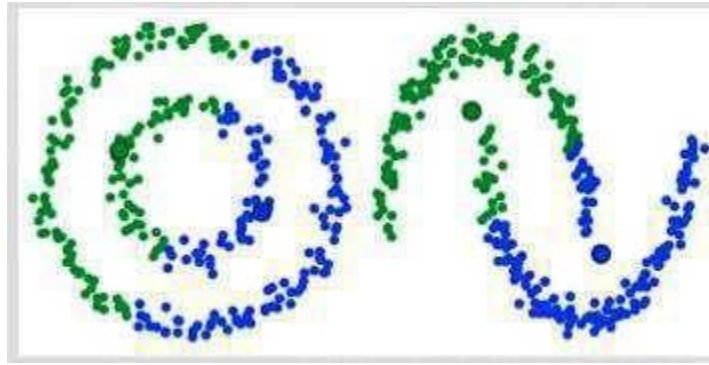


Figure 13: Expectation-Maximization

Two failure cases for K-Means:

Gaussian Mixture Models (GMMs) give us more flexibility than K-Means. With GMMs we assume that the data points are Gaussian distributed; this is a less restrictive assumption than saying they are circular by using the mean. That way, we have two parameters to describe the shape of the clusters: the mean and the standard deviation! Taking an example in two dimensions, this means that the clusters can take any kind of elliptical shape (since we have standard deviation in both the x and y directions). Thus, each Gaussian distribution is assigned to a single cluster.

In order to find the parameters of the Gaussian for each cluster (e.g the mean and standard deviation) we will use an optimization algorithm called Expectation–Maximization (EM). Take a look at the graphic below as an illustration of the Gaussians being fitted to the clusters. Then we can proceed on to the process of Expectation–Maximization clustering using GMMs.

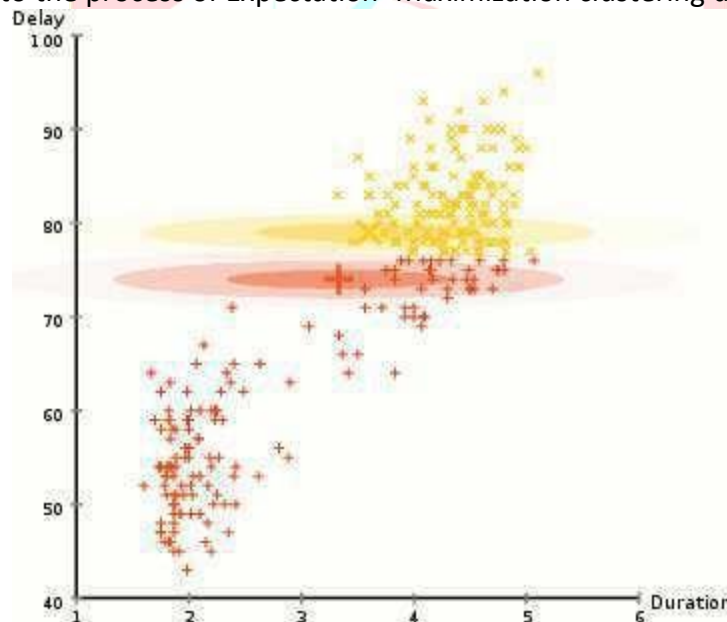


Figure 14: Expectation-Maximization Example

EM Clustering using GMMs:

1. We begin by selecting the number of clusters (like K-Means does) and randomly initializing the Gaussian distribution parameters for each cluster. One can try to provide a good guesstimate for the initial parameters by taking a quick look at the data too.

Though note, as can be seen in the graphic above, this isn't 100% necessary as the Gaussians start out as very poor but are quickly optimized.

2. Given these Gaussian distributions for each cluster, compute the probability that each data point belongs to a particular cluster. The closer a point is to the Gaussian's center, the more likely it belongs to that cluster. This should make intuitive sense since with a Gaussian distribution we are assuming that most of the data lies closer to the center of the cluster.
3. Based on these probabilities, we compute a new set of parameters for the Gaussian distributions such that we maximize the probabilities of data points within the clusters. We compute these new parameters using a weighted sum of the data point positions, where the weights are the probabilities of the data point belonging in that particular cluster. To explain this in a visual manner we can take a look at the graphic above, in particular the yellow cluster as an example. The distribution starts off randomly on the first iteration, but we can see that most of the yellow points are to the right of that distribution. When we compute a sum weighted by the probabilities, even though there are some points near the center, most of them are on the right. Thus naturally the distribution's mean is shifted closer to those set of points. We can also see that most of the points are "top-right to bottom-left". Therefore the standard deviation changes to create an ellipse that is more fitted to these points, in order to maximize the sum weighted by the probabilities.
4. Steps 2 and 3 are repeated iteratively until convergence, where the distributions don't change much from iteration to iteration.

There are really 2 key advantages to using GMMs. Firstly GMMs are a lot more **flexible** in terms of **cluster covariance** than K-Means; due to the standard deviation parameter, the clusters can take on any ellipse shape, rather than being restricted to circles. K-Means is actually a special case of GMM in which each cluster's covariance along all dimensions approaches 0. Secondly, since GMMs use probabilities, they can have multiple clusters per data point. So if a data point is in the middle of two overlapping clusters, we can simply define its class by saying it belongs X-percent to class 1 and Y-percent to class 2. I.e GMMs support **mixed membership**.

5. Agglomerative Hierarchical Clustering

Hierarchical clustering algorithms actually fall into 2 categories: top-down or bottom-up. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge (or *agglomerate*) pairs of clusters until all clusters have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is therefore called *hierarchical agglomerative clustering* or HAC. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample. Check out the graphic below for an illustration before moving on to the algorithm steps

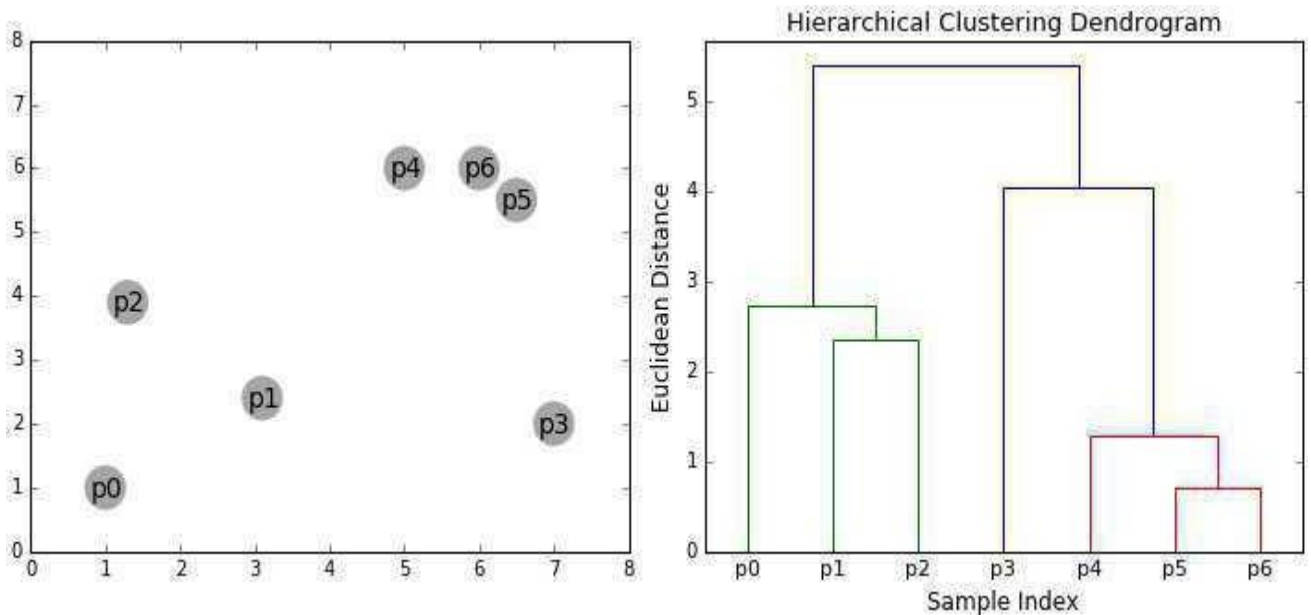


Figure 15: Agglomerative hierarchical clustering

Agglomerative Hierarchical Clustering steps:

1. We begin by treating each data point as a single cluster i.e if there are X data points in our dataset then we have X clusters. We then select a distance metric that measures the
2. distance between two clusters. As an example we will use *average linkage* which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.
3. On each iteration we combine two clusters into one. The two clusters to be combined are selected as those with the smallest average linkage. i.e according to our selected distance metric, these two clusters have the smallest distance between each other and therefore are the most similar and should be combined.
4. Step 2 is repeated until we reach the root of the tree i.e we only have one cluster which contains all data points. In this way we can select how many clusters we want in the end, simply by choosing when to stop combining the clusters i.e when we stop building the tree!

Hierarchical clustering does not require us to specify the number of clusters and we can even select which number of clusters looks best since we are building a tree. Additionally, the algorithm is not sensitive to the choice of distance metric; all of them tend to work equally well whereas with other clustering algorithms, the choice of distance metric is critical. A particularly good use case of hierarchical clustering methods is when the underlying data has a hierarchical structure and you want to recover the hierarchy; other clustering algorithms can't do this. These advantages of hierarchical clustering come at the cost of lower efficiency, as it has a time complexity of $O(n^3)$, unlike the linear complexity of K-Means and GMM.

Application of Clustering Algorithms

- Recommender systems
- Anomaly detection
- Human genetic clustering
- Genom Sequence analysis
- Analysis of antimicrobial activity
- Grouping of shopping items
- Search result grouping
- Slippy map optimization
- Crime analysis
- Climatology

Dimensionality reduction

Every day IBM creates 2.5 quintillion bytes of data and most of the data generated are high dimensional. So it is necessary to reduce the dimensions of the data to work efficiently.

One of the most common dimensionality reduction techniques is filtering, in which you leave most of the dimensions and concentrate only on certain dimensions. But that doesn't always work, when you are dealing with image data, the number of pixels represents the number of dimensions in the image. Now you have lot of dimensions and you don't want to throw out dimensions in order to make sense of your overall data set.

As the dimensionality of your data increases, the volume of the space increases, in a sense the data you have becomes more and more sparse(scattered). One way to think about it is a very high data set might live in some kind of high dimensional manifold and as you are increasing the number of dimensions, that manifold becomes bigger and bigger.

To do any statistical modelling, you have to increase the number of data points and samples you have, unfortunately that grows exponentially with the number of dimensions you have. The higher dimension you have, the more data you need to perform statistical inference. The basic idea is n (the number of data items) should be more than the number of dimensions.

The image(shown in fig below) has 64×64 pixels(4096 dimensions). To reduce the dimensions, you want to Project the high-dimensional data onto a lower dimensional subspace using linear or non-linear transformations (or projections).

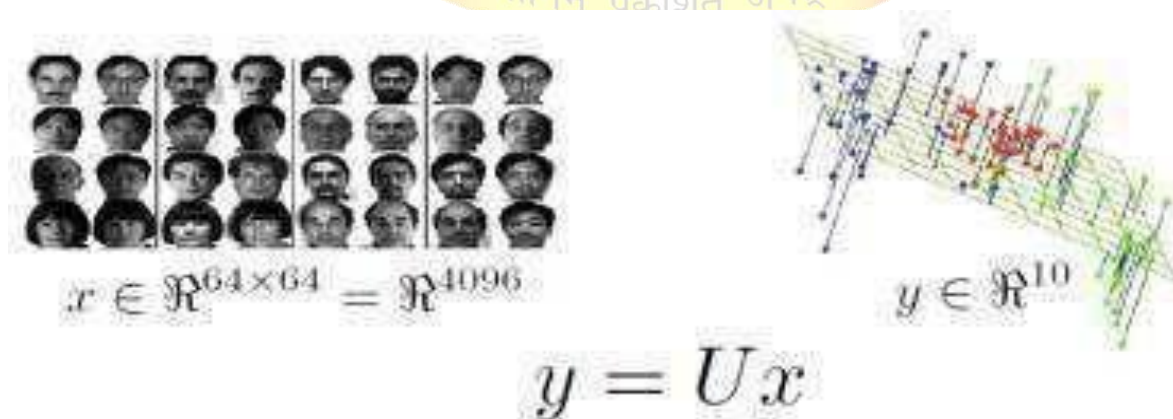


Figure 16: Dimensionality Reduction

The most widely used methods are linear projections and the main linear projection method is principal component analysis (PCA).

Principal component analysis (PCA)

You have data points that live in a 2D plane(x_1 and x_2) and you want to approximate them with a lower dimensional embedding. Here it is obvious that the vector V (as shown in the image) is a pretty good approximation of your data. Instead of storing 2 coordinates for each data points, you will store one scalar value plus the vector V , which is common across all of the data points, so you have to store it only once. So for each data point you have to store only this scalar value s , which gives the distance along this vector V .

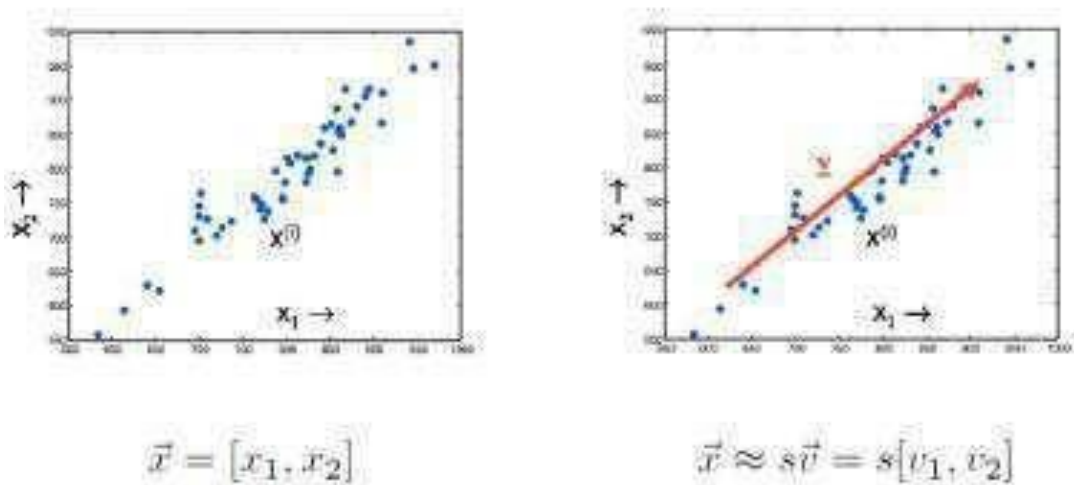
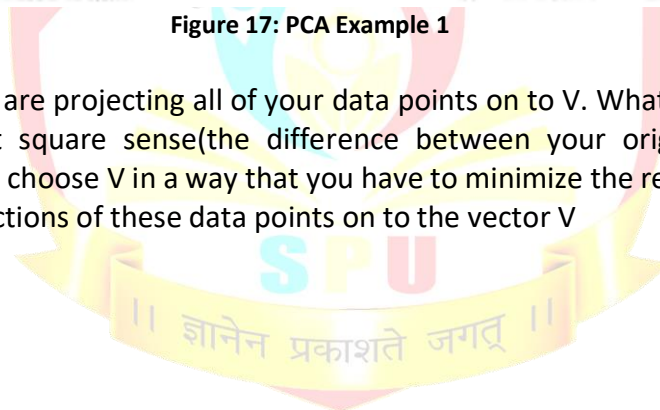
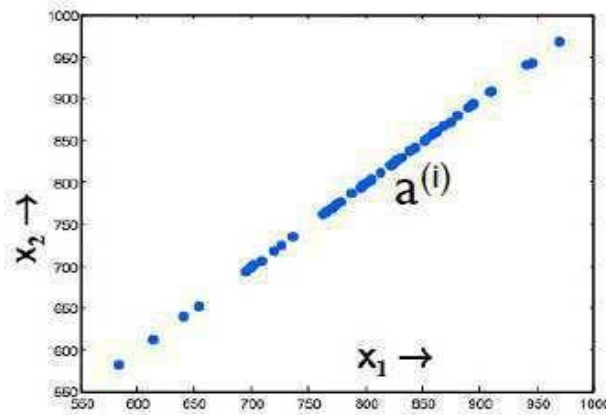


Figure 17: PCA Example 1

In the figure below, you are projecting all of your data points on to V . What you are trying to do is to minimize in least square sense(the difference between your original data and your projections). You should choose V in a way that you have to minimize the residual variance.

*residuals are the projections of these data points on to the vector V





$$\min_{a,v} \sum_i (x^{(i)} - a^{(i)}v)^2$$

Figure 18: PCA Example 2

In this case the projection is orthogonal to vector V . You have to minimize the overall sum of all of the residuals of your data points, by choosing the vector in such a way that the overall sum is minimized. It turns out that it is also the vector that closely allows you to reconstruct the original data using the least squared errors. In this case it makes intuitive sense; you are picking V in the direction of biggest spread of your data. You can extend this to multiple components. Here is the main component which we call the principal component, that's the vector V that you are using to project the data on. And then you can repeat the process and then find the second component that has second biggest variance of the data, in this case is the direction of principal comp2 (see the image below).

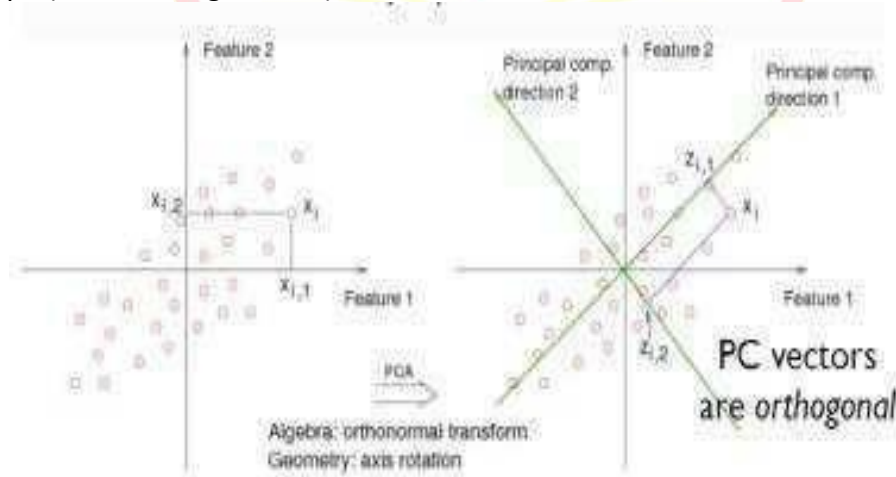


Figure 19: PCA Example 3

It is very important to understand the difference between Principal Component Analysis (PCA) and Ordinary Least Squares (OLS). I encourage you to go through this discussion on stack exchange.

To summarize, you are projecting your data on to these sub spaces (on to the principal components) to maximize the variance of the projected data, from the basic idea.

Singular Value Decomposition (SVD):

The PCA algorithm is implemented as follows:

- 1) Subtract the mean from the data.
- 2) Scale each dimensions by its variance.
- 3) Compute the covariance matrix S. Here X is a data matrix.

$$S = \left(\frac{1}{N}\right)^T$$

- 4) Compute K largest eigen vectors of S. These eigen vectors are the principal components of the data set.

Note: Eigenvectors and values exist in pairs: every eigen vector has a corresponding eigenvalue. Eigen vector is the direction of the line (vertical, horizontal, 45 degrees etc.). Aneigenvalue is a number, telling you how much variance there is in the data in that direction, eigenvalue is a number telling you how spread out the data is on the line.

But the operation (steps to implement PCA) is expensive when X is very large or when X is very small. So the best way to compute principal components is by using SVD. Singular value decomposition (SVD) is one of the best linear transformation methods out there.

You can take any matrix X, it doesn't matter if it is square, singular or diagonal, you can decompose it into a product of three matrices(as shown in the figure below); two orthogonal matrices U and V and diagonal matrix D. The orthogonal matrix has same dimensions as your data matrix and then your diagonal matrix is square and it has dimensions kxk (k is the number of variables you have), V is again a square matrix.

$$= UDV^T$$

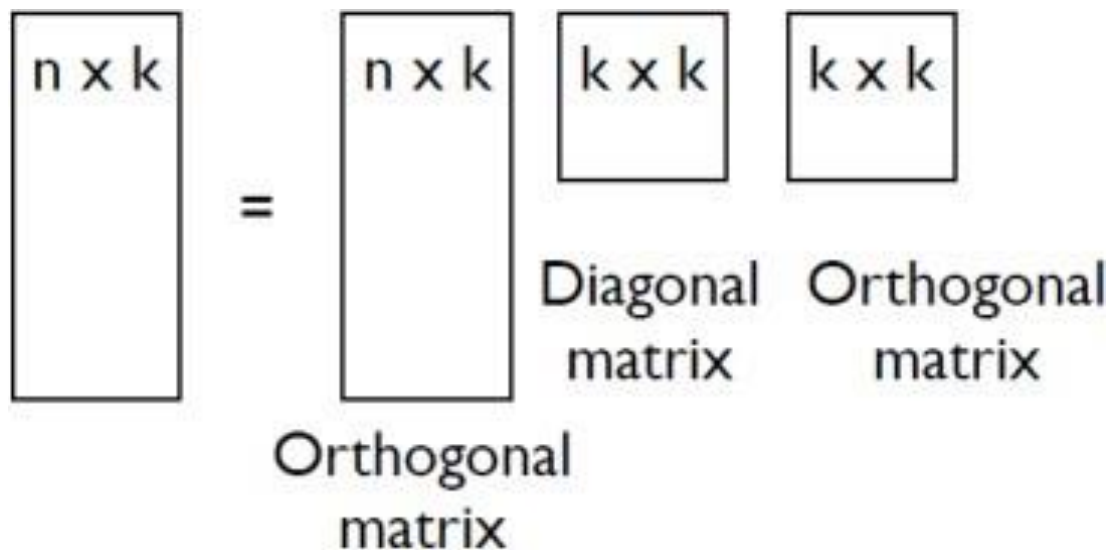


Figure 20: SVD Example

Some interesting notes on the result of factorization:

1. The columns of matrix U form the eigen vectors of S.
2. The D matrix is a diagonal matrix. The values of the diagonal are called eigen values and are in descending order.

This is much more equivalent to calculate principal components analysis, but in a much more robust way. You just need to take your matrix and feed it into svd.

Image recognition example

Let's say your task is to recognize faces of people, you take photo graph of people and you make small pictures and then you center all the faces such that they are roughly aligned.



Figure 21: SVD Example 2

Let's say the image is 64 x 64 pixels. So you have 4096 dimensions, you are rolling out each of the images into a vector, and then stack them up into your data matrix. Each pixel is dimensional and each row is a different person.

64x64 images of faces = 4096 dimensional data

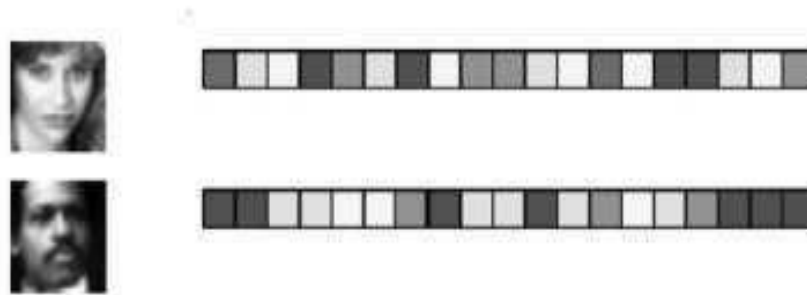


Figure 22: SVD Example 3

So you have this data matrix (as shown in the above image) and you apply pca on that one. Here is what you get (as shown in the image below), these are called eigen faces. The image on the left (this is the mean sort of average face). The first image on the right hand side sort of explains the variance in left right dimensions, the 2nd image on seems to explain the variance in front to back.

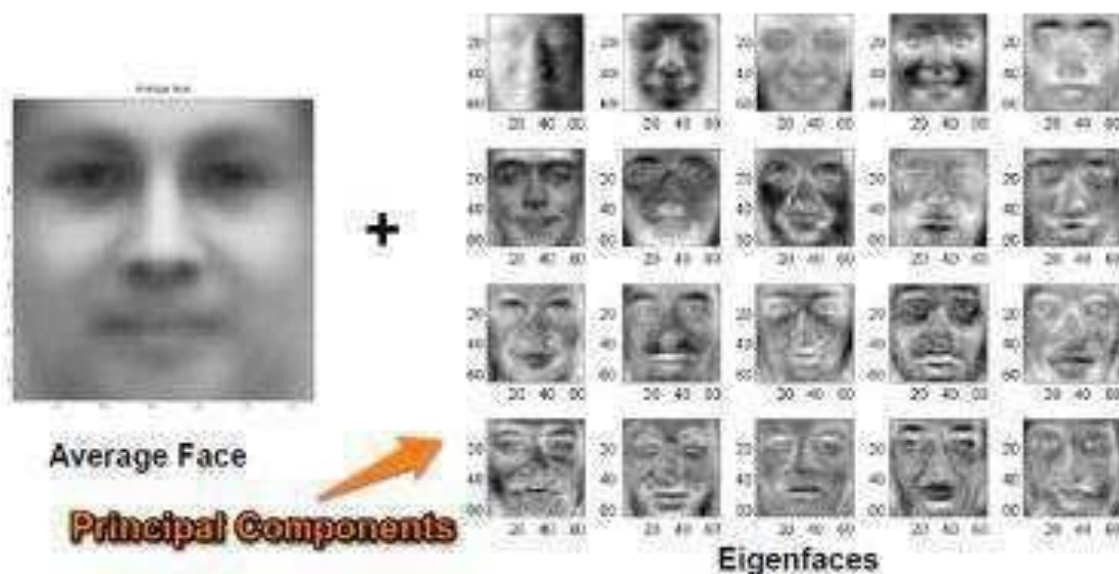


Figure 23: SVD Example 4

In the image below, you can actually look at the variance of the components, it turns out that you need 50 eigen vectors to explain 90 % of the variance of a image.

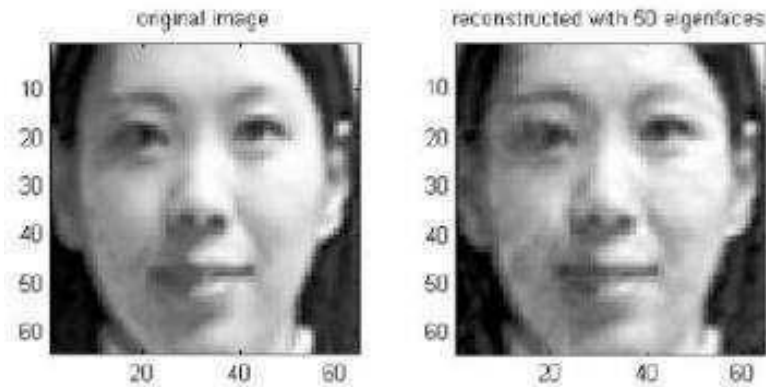


Figure 24: SVD Example 4

As you can see after 50 eigen vectors this is a pretty good reconstruction. So you went from 4096 dimensions to 50 that's a nice reduction in dimensions without too much reduction in quality. I hope I have given a broad idea of what is dimensionality reduction, Pca and Svd without getting into too much mathematical detail.

Regression and correlation analysis:

Regression analysis involves identifying the relationship between a dependent variable and one or more independent variables. A model of the relationship is hypothesized, and estimates of the parameter values are used to develop an estimated regression equation. Various tests are then employed to determine if the model is satisfactory. If the model is deemed satisfactory, the estimated regression equation can be used to predict the value of the dependent variable given values for the independent variables.

1. Regression model

In simple linear regression, the model used to describe the relationship between a single dependent variable y and a single independent variable x is $y = a_0 + a_1x + k$. a_0 and a_1 are referred to as the model parameters, and k is a probabilistic error term that accounts for the variability in y that cannot be explained by the linear relationship with x . If the error term were not present, the model would be deterministic; in that case, knowledge of the value of x would be sufficient to determine the value of y .

Least squares method.

Either a simple or multiple regression model is initially posed as a hypothesis concerning the relationship among the dependent and independent variables. The least squares method is the most widely used procedure for developing estimates of the model parameters.

As an illustration of regression analysis and the least squares method, suppose a university medical centre is investigating the relationship between stress and blood pressure. Assume that both a stress test score and a blood pressure reading have been recorded for a sample of 20 patients. The data are shown graphically in the figure below, called a scatter diagram. Values of the independent variable, stress test score, are given on the horizontal axis, and values of the dependent variable, blood pressure, are shown on the vertical axis. The line passing through

the data points is the graph of the estimated regression equation: $y = 42.3 + 0.49x$. The parameter estimates, $b_0 = 42.3$ and $b_1 = 0.49$, were obtained using the least squares method.

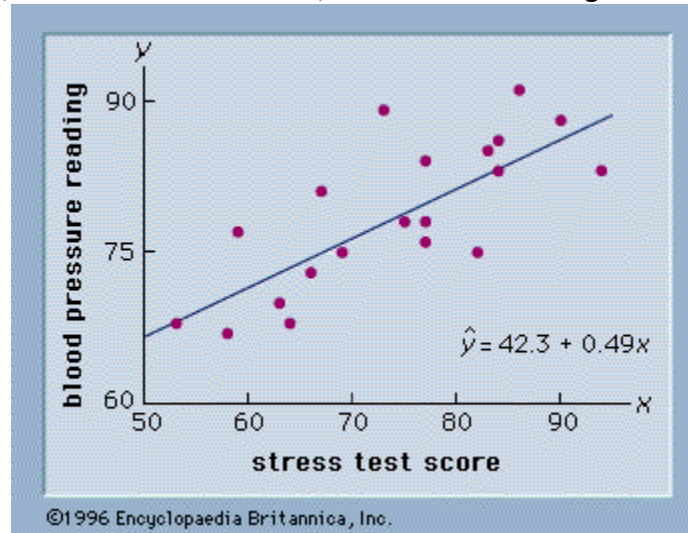


Figure 25: Regression Model Example

2. Correlation Model

Correlation and regression analysis are related in the sense that both deal with relationships among variables. The correlation coefficient is a measure of linear association between two variables. Values of the correlation coefficient are always between -1 and +1. A correlation coefficient of +1 indicates that two variables are perfectly related in a positive linear sense; a correlation coefficient of -1 indicates that two variables are perfectly related in a negative linear sense, and a correlation coefficient of 0 indicates that there is no linear relationship between the two variables. For simple linear regression, the sample correlation coefficient is the square root of the coefficient of determination, with the sign of the correlation coefficient being the same as the sign of b_1 , the coefficient of x_1 in the estimated regression equation.

Neither regression nor correlation analyses can be interpreted as establishing cause-and-effect relationships. They can indicate only how or to what extent variables are associated with each other. The correlation coefficient measures only the degree of linear association between two variables. Any conclusions about a cause-and-effect relationship must be based on the judgment of the analyst.

Training and Testing Data: An Introduction

Separating data into training and testing sets is an important part of evaluating data mining and data analysis models. Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing. Analysis Services randomly samples the data to help ensure that the testing and training sets are similar. By using similar data for training and testing, you can minimize the effects of data discrepancies and better understand the characteristics of the model.

After a model has been processed by using the training set, you test the model by making predictions against the test set. Because the data in the testing set already contains known values for the attribute that you want to predict, it is easy to determine whether the model's guesses are correct.

In order to perform supervised learning, you need two types of data sets:

1. In your **training data set**, you include both the input data together along with the corresponding expected output. This provides your model with “ground truth” data that’s usually prepared by humans or in a semi-automated way. During the testing phase, you present this data to train your model by pairing the input with its respective desired output.
2. The **test data set** contains data you are going to apply your model to. In contrast, this data doesn’t have any “expected” output. During the test phase of machine learning, this data is used to estimate how well your model has been trained and to estimate model properties.

Generally, training data is split up more or less randomly, while making sure to capture all the important classes you are aware of. For example, if you’re trying to train a sentiment analysis model using social media content, you’d want to avoid training your model using tweets from a single user. This variation will make your model more robust and help prevent it from overfitting (when the model is too closely fit to a limited set of data points).

Underfitting and Overfitting in Machine Learning

Let us consider that we are designing a machine learning model. A model is said to be a good machine learning model, if it generalizes any new input data from the problem domain in a proper way. This helps us to make predictions in the future data, that data model has never seen.

Now, suppose we want to check how well our machine learning model learns and generalizes to the new data. For that we have overfitting and underfitting, which are majorly responsible for the poor performances of the machine learning algorithms.

Underfitting:

A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. *(It’s just like trying to fit undersized pants!)* Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data. In such cases the rules of the machine learning model are too easy and flexible to be applied on such a minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.

Overfitting:

A statistical model is said to be overfitted, when we train it with a lot of data *(just like fitting ourselves in an oversized pants!)*. When a model gets trained with so much of data, it starts

learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too much of details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

Ex ample:

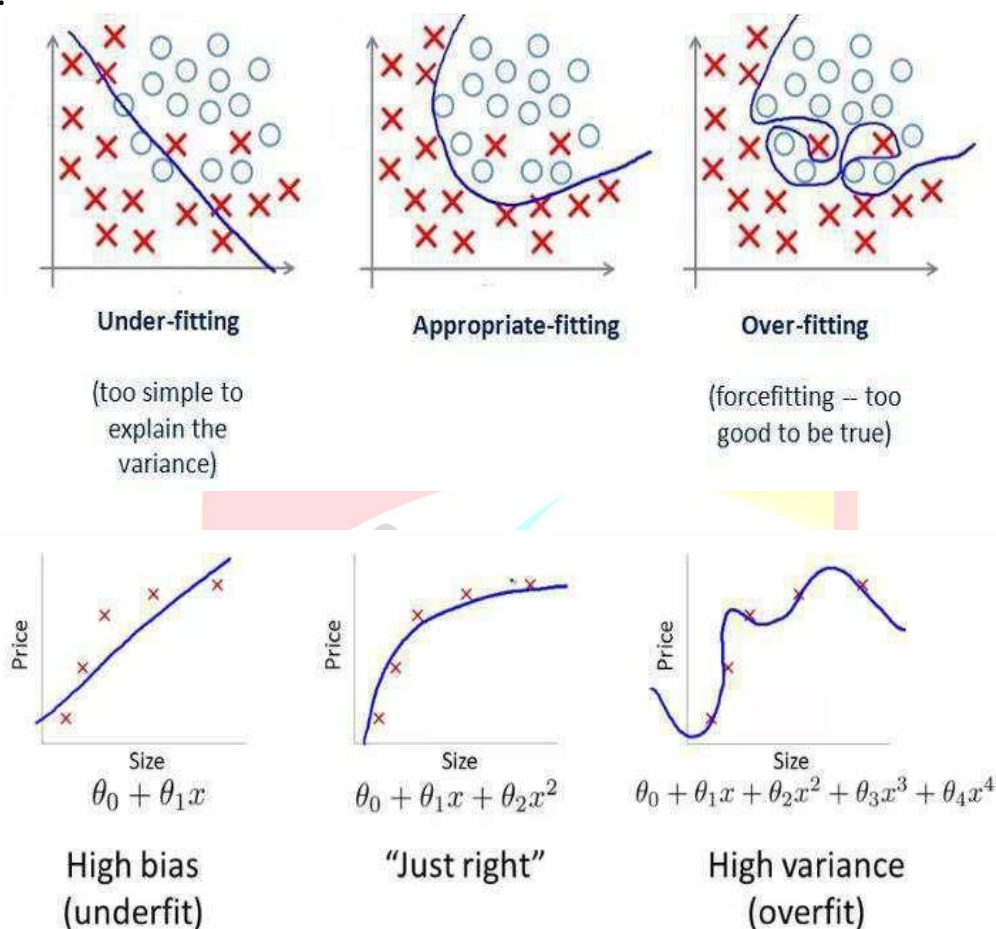


Figure 27: Overfitting & Underfitting Example

How to avoid Overfitting:

The commonly used methodologies are:

- **Cross- Validation:** A standard way to find out-of-sample prediction error is to use 5-fold cross validation.
- **Early Stopping:** Its rules provide us the guidance as to how much iteration can be run before learner begins to over-fit.
- **Pruning:** Pruning is extensively used while building related models. It simply removes the nodes which add little predictive power for the problem in hand.

- **Regularization:** It introduces a cost term for bringing in more features with the objective function. Hence it tries to push the coefficients for many variables to zero and hence reduce cost term.



Subject Name: **Data Science & Big data**

Subject Code: **CS-7005**

Semester: **7th**



Unit-4

Topics to be covered

Introduction to Information Retrieval: Boolean Model, Vector model, Probabilistic Model, Text based search: Tokenization, TF-IDF, stop words and n-grams, synonyms and parts of speech tagging.

Introduction to Information Retrieval

The goal of information retrieval (IR) is to provide users with those documents that will satisfy their information need. We use the word "document" as a general term that could also include non-textual information, such as multimedia objects. Users have to formulate their information need in a form that can be understood by the retrieval mechanism. There are several steps involved in this translation process that we will briefly discuss below. Likewise, the contents of large document collections need to be described in a form that allows the retrieval mechanism to identify the potentially relevant documents quickly. In both cases, information may be lost in the transformation process leading to a computer-usable representation. Hence, the matching process is inherently imperfect.

Information seeking is a form of problem solving. It proceeds according to the interaction among eight sub processes: problem recognition and acceptance, problem definition, search system selection, query formulation, query execution, examination of results (including relevance feedback), information extraction, and reflection/iteration/termination. To be able to perform effective searches, users have to develop the following expertise: knowledge about various sources of information, skills in defining search problems and applying search strategies, and competence in using electronic search tools.

The information need can be understood as forming a pyramid, where only its peak is made visible by users in the form of a conceptual query. The conceptual query captures the key concepts and the relationships among them. It is the result of a conceptual analysis that operates on the information need, which may be well or vaguely defined in the user's mind. This analysis can be challenging, because users are faced with the general "vocabulary problem" as they are trying to translate their information need into a conceptual query.

This problem refers to the fact that a single word can have more than one meaning, and, conversely, the same concept can be described by surprisingly many different words. Further, the concepts used to represent the documents can be different from the concepts used by the user. The conceptual query can take the form of a natural language statement, a list of concepts that can have degrees of importance assigned to them, or it can be statement that coordinates the concepts using Boolean operators. Finally, the conceptual query has to be translated into a query surrogate that can be understood by the retrieval system.

Boolean Model

A fat book which many people own is Shakespeare's Collected Works. Suppose you wanted to determine which plays of Shakespeare contain the words Brutus AND Caesar and NOT Calpurnia. One way to do that is to start at the beginning and to read through all the text, noting for each play whether it contains Brutus and Caesar and excluding it from consideration if it contains Calpurnia. The simplest form of document retrieval is for a computer to do this sort of linear scan through documents. This process is commonly referred to as *grepping* through text, after the Unix command *grep*, which performs this process. Grepping through text can be a very effective process, especially given the speed of modern computers, and often allows useful possibilities for wildcard pattern matching through the use of. With modern computers, for simple querying of modest collections (the size of Shakespeare's Collected Works is a bit under one million words of text in total), you really need nothing more.

But for many purposes, you do need more:

1. To process large document collections quickly. The amount of online data has grown at least as quickly as the speed of computers, and we would now like to be able to search collections that total in the order of billions to trillions of words.
2. To allow more flexible matching operations. For example, it is impractical to perform the query Romans NEAR countrymen with *grep*, where NEAR might be defined as "within 5 words" or "within the same sentence".
3. To allow ranked retrieval: in many cases you want the best answer to an information need among many documents that contain certain words.

The way to avoid linearly scanning the texts for each query is to *index* the documents in advance. Let us stick with Shakespeare's Collected Works, and use it to introduce the basics of the Boolean retrieval model. Suppose we record for each document - here a play of Shakespeare's - whether it contains each word out of all the words Shakespeare used (Shakespeare used about 32,000 different words). The result is a binary term-document *incidence matrix*, as in Figure. *Terms* are the indexed units; they are usually words, and for the moment you can think of them as words, but the information retrieval literature normally speaks of terms because some of them, such as perhaps I-9 or Hong Kong are not usually thought of as words. Now, depending on whether we look at the matrix rows or columns, we can have a vector for each term, which shows the documents it appears in, or a vector for each document, showing the terms that occur in it.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	

Figure 1: Boolean Model Example

To answer the query Brutus AND Caesar AND NOT Calpurnia, we take the vectors for Brutus, Caesar and Calpurnia, complement the last, and then do a bitwise AND:

110100 AND 110111 AND 101111 = 100100

The answers for this query are thus Antony and Cleopatra and Hamlet (Figure).

The *Boolean retrieval model* is a model for information retrieval in which we can pose any query which is in the form of a Boolean expression of terms, that is, in which terms are combined with the operators and, or, and not. The model views each document as just a set of words.

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to Domitius Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring; and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius Caesar: I was killed i' the
Capitol; Brutus killed me.

Figure 2: Results from Shakespeare for the query Brutus AND Caesar AND NOT Calpurnia

Vector Space Model

The vector space model represents the documents and queries as vectors in a multidimensional space, whose dimensions are the terms used to build an index to represent the documents. The creation of an index involves lexical scanning to identify the significant terms, where morphological analysis reduces different word forms to common "stems", and the occurrence of those stems is computed. Query and document surrogates are compared by comparing their vectors, using, for example, the cosine similarity measure. In this model, the terms of a query surrogate can be weighted to take into account their importance, and they are computed by using the statistical distributions of the terms in the collection and in the documents. The vector space model can assign a high ranking score to a document that contains only a few of the query terms if these terms occur infrequently in the collection but frequently in the document. The vector space model makes the following assumptions:

- 1) The more similar a document vector is to a query vector; the more likely it is that the document is relevant to that query.
- 2) The words used to define the dimensions of the space are orthogonal or independent. While it is a reasonable first approximation, the assumption that words are pair wise independent is not realistic.

Probabilistic Model

The probabilistic retrieval model is based on the Probability Ranking Principle, which states that an information retrieval system is supposed to rank the documents based on their probability of relevance to the query, given all the evidence available. The principle takes into account that there is uncertainty in the representation of the information need and the documents. There can be a variety of sources of evidence that are used by the probabilistic retrieval methods, and

the most common one is the statistical distribution of the terms in both the relevant and non-relevant documents.

- Different theoretical foundations (assumptions) for IR
 - Boolean model:
 - Used in specialized area
 - Not appropriate for general search alone – often used as a pre-filtering
 - Vector space model:
 - Robust
 - Good experimental results
 - Probabilistic models:
 - Difficulty to estimate probabilities accurately
 - Modified version (BM25) – excellent results
 - Regression models:
 - Need training data
 - Widely used (in a different form) in web search
 - Learning to rank (a later lecture)

Text based search: Tokenization

The process of segmenting running text into words and sentences.

Electronic text is a linear sequence of symbols (characters or words or phrases). Naturally, before any real text processing is to be done, text needs to be segmented into linguistic units such as words, punctuation, numbers, alpha-numeric, etc. This process is called tokenization.

In English, words are often separated from each other by blanks (white space), but not all white space is equal. Both “Los Angeles” and “rock 'n' roll” are individual thoughts despite the fact that they contain multiple words and spaces. We may also need to separate single words like “I’m” into separate words “I” and “am”.

Tokenization is a kind of pre-processing in a sense; an identification of basic units to be processed. It is conventional to concentrate on pure analysis or generation while taking basic units for granted. Yet without these basic units clearly segregated it is impossible to carry out any analysis or generation.

The identification of units that do not need to be further decomposed for subsequent processing is an extremely important one. Errors made at this stage are very likely to induce more errors at later stages of text processing and are therefore very dangerous.

What counts as a token in NLP?

The notion of a token must first be defined before computational processing can proceed. There is more to the issue than simply identifying strings delimited on both sides by spaces or punctuation.

Different notions depend on different objectives, and often different language backgrounds.

A token is

1. Linguistically significant
2. Methodologically useful

Webster and Kit suggest that finding significant tokens depends on the ability to recognize patterns displaying significant collocation. Rather than simply relying on whether a string is bounded by delimiters on either side, segmentation into significant tokens relies on a kind of pattern recognition.

Consider this hypothetical speech transcription:

Where is meadows dr who asked?

Collocation patterns could help determine if this is about meadows dr (Drive) or dr (Doctor) who.

Standard (White Space) Tokenization

Word tokenization may seem simple in a language that separates words by a special 'space' character. However, not every language does this (e.g. Chinese, Japanese, Thai), and a closer examination will make it clear that white space alone is not sufficient even for English.

Addressing Specific Challenges

Tokenization is generally considered as easy relative to other tasks in natural language, and one of the more uninteresting tasks (for English and other segmented languages). However, errors made in this phase will propagate into later phases and cause problems. To address this problem, a number of advanced methods which deal with specific challenges in tokenization have been developed to complement standard tokenizers.

Bob Carpenter states that tokenization is particularly vexing in the bio-medical text domain, where there are tons of words (or at least phrasal lexical entries) that contain parentheses, hyphens, and so on, and that this turned out to be a problem for WordNet).

Another challenge for tokenization is “dirty text”¹. Not all text has been passed through an editing and spell-check process. Text extracted automatically from PDFs, database fields, or other sources may contain inaccurately compounded tokens, spelling errors and unexpected characters. In some cases, when text is stored in a database in fixed fields, with multiple lines per object, fields sometimes need to be reassembled but the spaces have (inconsistently) been trimmed.

It is not safe to make the assumption that source text will be perfect. A tokenizer must often be customized to the data in question.

Low-Level vs High-Level Tokenization

Determining if two or more words should stand together to form a single token (like “ational Software Architect”) would be a high-level tokenization task. High-level segmentation is much more linguistically motivated than 'low-level' segmentation, and requires (at a minimum) relatively shallow linguistic processing.

Steps in Low Level Tokenization

Step 1: Segmenting Text into Words

The first step in the majority of text processing applications is to segment text into words.

In all modern languages that use a Latin-, Cyrillic-, or Greek-based writing system, such as English and other European languages, word tokens are delimited by a blank space. Thus, for

such languages, which are called segmented languages; token boundary identification is a somewhat trivial task since the majority of tokens are bound by explicit separators like spaces and punctuation. A simple program which replaces white spaces with word boundaries and cuts off leading and trailing quotation marks, parentheses and punctuation already produces a reasonable performance.

The majority of existing tokenizers signal token boundaries by white spaces. Thus, if such a tokenizer finds two tokens directly adjacent to each other, as, for instance, when a word is followed by a comma, it inserts a white space between them.

The example given in a following section will show how a standard white space tokenizer fares in a more complex example

Step 2: Handling Abbreviations

In English and other Indo-European languages although a period is directly attached to the previous word, it is usually a separate token which signals the end of the sentence. However, when a period follows an abbreviation it is an integral part of this abbreviation and should be tokenized together with it.

the dr. lives in a blue box.

Without addressing the challenge posed by abbreviation, this line would be delimited into *the dr.* *lives in a blue box.*

Unfortunately, universally accepted standards for many abbreviations and acronyms do not exist.

The most widely adopted approach to the recognition of abbreviations is to maintain a list of known abbreviations. Thus during tokenization a word with a trailing period can be looked up in such a list and, if it is found there, it is tokenized as a single token; otherwise the period is tokenized as a separate token. Naturally, the accuracy of this approach depends on how well the list of abbreviations is tailored to the text under processing. There will almost certainly be abbreviations in the text which are not included in the list. Also, abbreviations in the list can coincide with common words and trigger erroneous tokenization. For instance, 'in' can be an abbreviation for 'inches'; 'no' can be an abbreviation for 'number', 'bus' can be an abbreviation for 'business'; 'sun' can be an abbreviation for 'Sunday'; etc.

The following lists are by no means comprehensive:

Common Acronyms with Punctuation

1. I.O.U.
2. M.D.
3. N.B.
4. P.O.
5. U.K.
6. U.S.
7. U.S.A.
8. P.S.

Common Words containing Periods

1. .c
2. mr.

3. mrs.
4. .com
5. dr.
6. .sh
7. .java
8. st.

Step 3: Handling Hyphenated Words

Segmentation of hyphenated words answers a question 'One word or two?'

Hyphenated segments present a case of ambiguity for a tokenizer-sometimes a hyphen is part of a token, i.e. self-assessment, F-15, forty-two and sometimes it is not e.g. Los Angeles-based. Segmentation of hyphenated words is task dependent. For instance, part-of-speech taggers (Chapter ii) usually treat hyphenated words as a single syntactic unit and therefore prefer them to be tokenized as single tokens. On the other hand named entity recognition (NER) systems (Chapter 30) attempt to split a named entity from the rest of a hyphenated fragment; e.g. in parsing the fragment 'Moscow-based' such a system needs 'Moscow' to be tokenized separately from 'based' to be able to tag it as a location.

Types of Hyphens:

1. End-of-Line Hyphen
2. True Hyphen
 1. Lexical Hyphen
 2. Sententially Determined Hyphenation

End-of-Line Hyphen

End-of-line hyphens are used for splitting whole words into parts to perform justification of text during typesetting. Therefore they should be removed during tokenization because they are not part of the word but rather layouting instructions.

True Hyphen

True hyphens, on the other hand, are integral parts of complex tokens, e.g. forty-seven, and should therefore not be removed. Sometimes it is difficult to distinguish a true hyphen from an end-of-line hyphen when a hyphen occurs at the end of a line.

Lexical Hyphen

Hyphenated compound words which have made their way into standard language vocabulary. For instance, certain prefixes (and less commonly suffixes) are often written hyphenated, e.g. co-, pre-, meta-, multi-, etc.

Sententially Determined Hyphenation

Here hyphenated forms are created dynamically as a mechanism to prevent incorrect parsing of the phrase in which the words appear. There are several types of hyphenation in this class. One is created when a noun is modified by an 'ed'-verb to dynamically create an adjective, e.g. case-based, computer-linked, and hand-delivered. Another case involves an entire expression when it is used as a modifier in a noun group, as in a three-to-five-year direct marketing plan. In

treating these cases a lexical look-up strategy is not much help and normally such expressions are treated as a single token unless there is a need to recognize specific tokens, such as dates, measures, names, in which case they are handled by specialized sub grammars
This hypothetical sentence has many challenges:

The New York-based co-operative was fine-tuning forty-two K-9-like models.

Token	Type
New York-based	Sentential
co-operative	Lexical
fine-tuning	End-of-Line , but could also be considered a Lexical hyphen based on the author's stylistic preferences.
Forty-two	Lexical
K-9-like	Lexical and Sentential

Step 3: Numerical and special expressions

Examples:

1. Email addresses
2. URLs
3. Complex enumeration of items
4. Telephone Numbers
5. Dates
6. Time
7. Measures
8. Vehicle Licence Numbers
9. Paper and book citations
10. etc

These can produce a lot of confusion to a tokenizer because they usually involve rather complex alpha numerical and punctuation syntax.

Take phone numbers for example -

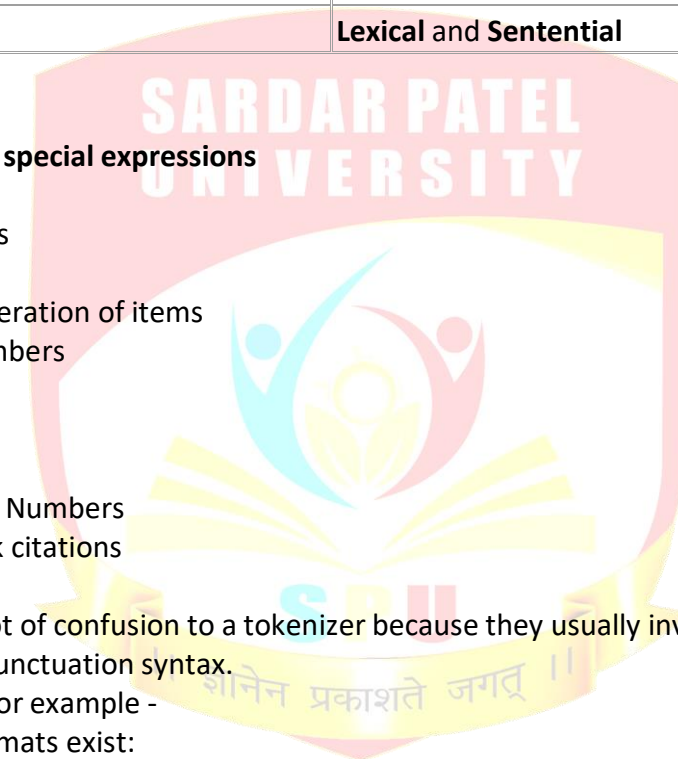
A variety of formats exist:

1. 123-456-7890
2. (123)-456-7890
3. 123.456.7890
4. (123) 456-7890
5. etc

A pre-processor should be designed to recognize phone numbers and perform normalization. All phone numbers would then be in a single format, making the job of a tokenizer easier.

Date/Time Formats:

1. 8th-Feb
2. 8-Feb-2013



	Naïve Whitespace Parser	Apache Open NLP 1.5.2 (using en-token.bin)	Stanford 2.0.3	Custom	Hypothetical Tokenizer (Ideal Tokenization)
1		"	"	"	"
2	"i	i	i	i	i
3	said,	said	said	said	said
4		,	,	,	,
5	'what're	'what	`	'	'
6			what	what're	what
7		're	're		are
8	you?	you	you	you	you
9		?	?	?	?
10	crazy?"	crazy	crazy	crazy	crazy
11		?	?	?	?
12		'	'	'	'
13	said	said	said	said	said
14	sandowsky.	sandowsky	sandowsky	sandowsky	sandowsky
15	
16		'	'	'	"
17	'i	i	i	i	i
18	can't	ca	ca	can't	can
19		n't	n't		not
20	afford	afford	afford	afford	afford
21	to	to	to	to	to
22	do	do	do	do	do
23	that.'	that	that	that	that
24	
	Naïve Whitespace Parser	Apache Open NLP 1.5.2 (using en-token.bin)	Stanford 2.0.3	Custom	Hypothetical Tokenizer (Ideal Tokenization)
1		"	"	"	"
2	"i	i	i	i	i
3	said,	said	said	said	said
4		,	,	,	,
5	'what're	'what	`	'	'
6			what	what're	what
7		're	're		are
8	you?	you	you	you	you
9		?	?	?	?
10	crazy?"	crazy	crazy	crazy	crazy

11		?	?	?	?
12		'	'	'	'
13	said	said	said	said	said
14	sandowsky.	sandowsky	sandowsky	sandowsky	sandowsky
15	
16		'	'	'	"
17	'i	i	i	i	i
18	can't	ca	ca	can't	can
19		n't	n't		not
20	afford	afford	afford	afford	afford
21	to	to	to	to	to
22	do	do	do	do	do
23	that.'	that	that	that	that
24	

3. 02/08/13

4. February 8th, 2013

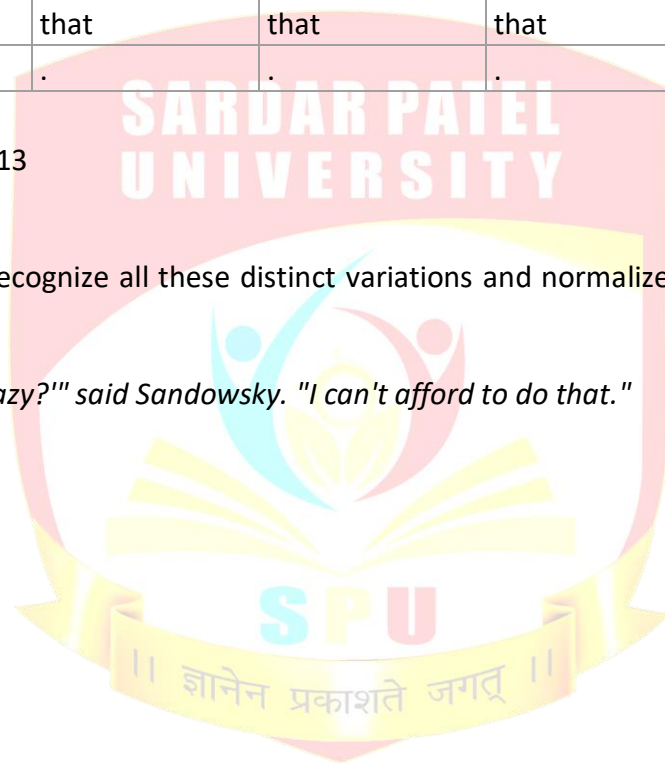
5. Feb 8th

6. etc

A pre-processor could recognize all these distinct variations and normalize into a single expression.

Tokenization Example

"I said, 'what're you? Crazy?'" said Sandowsky. "I can't afford to do that."



25		,	,	,	,
----	--	---	---	---	---

The naïve white space parser is shown to perform poorly here.

The Stanford tokenizer does somewhat better than the OpenNLP tokenizer, which is to be expected. The custom parser (included in the appendix) in the 4th column, does a nearly perfect job, though without the enclitic expansion shown in the first hypothetical pass.

The more accurate (and complex) segmentation process in the fourth and fifth columns require a morphological parsing process.

We can address some of these issues in the first three examples by treating punctuation, in addition to white space, as a word boundary. But punctuation often occurs internally, in examples like u.s.a., Ph.D., AT&T, ma'am, cap'n, 01/02/06 and stanford.edu. Similarity, assuming we want 7.1 or 82.4 as a word, we can't segment on every period, since that would segment these into "7" and "1" and "82" and "4". Should "data-base" be considered two separate tokens or a single token? The number "\$2,023.74" should be considered a single token, but in this case, the comma and period do not represent delimiters, where in other cases they might. And should the "\$" sign be considered part of that token, or a separate token in its own right?

The java.util.SimpleTokenizer class in Java is an example of a white space tokenizer, where you can define the set of characters that mark the boundaries of tokens. Another Java class, java.text.BreakIterator, can identify word or sentence boundaries, but still does not handle ambiguities.

Named Entity Extraction

It's almost impossible to separate tokenization from named entity extraction. It really isn't possible to come up with a generic set of rules that will handle all ambiguous cases within English; the easiest approach is usually just to have multi-word expression dictionaries.

Install Rational Software Architect on AIX 5.3

	Naïve Whitespace Parser	Hypothetical Tokenizer (Ideal Tokenization)
1	install	install
2	rational	rational software architect for websphere
3	software	
4	architect	
5	for	
6	websphere	
7	on	on
8	aix	aix 5.3
9	5.3	

Dictionaries will have to exist that express to the tokenization process that "Rational Software Architect for WebSphere" is a single token (a product), and "AIX 5.3" is likewise a single product.

TF-IDF

Google has already been using *TF*IDF* (or *TF-IDF*, *TFIDF*, *TF.IDF*, *Artist formerly known as Prince*) as a ranking factor for your content for a long time, as the search engine seems to focus more on *term frequency* rather than on *counting keywords*. While the visual complexity of the algorithm might turn a lot of people off, it is important to recognize that understanding TF*IDF is not as significant as knowing how it works.

TF*IDF is used by search engines to better understand content which is undervalued. For example, if you'd want to search a term "Coke" on Google, this is how Google can figure out if a page titled "COKE" is about:

- a) Coca-Cola.
- b) Cocaine.
- c) A solid, carbon-rich residue derived from the distillation of crude oil.
- d) A county in Texas.

What is TF*IDF?

TF*IDF is an information retrieval technique that weighs a term's frequency (TF) and its inverse document frequency (IDF). Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF*IDF weight of that term.

Put simply, the higher the TF*IDF score (weight), the rarer the term and vice versa.

The TF*IDF algorithm is used to weigh a keyword in any content and assign the importance to that keyword based on the number of times it appears in the document. More importantly, it checks how relevant the keyword is throughout the web, which is referred to as *corpus*. For a term *t* in a document *d*, the weight *W_{t,d}* of term *t* in document *d* is given by: $W_{t,d} = TF_{t,d} \log(N/DF_t)$

Where:

- $TF_{t,d}$ is the number of occurrences of *t* in document *d*.
- DF_t is the number of documents containing the term *t*.
- *N* is the total number of documents in the corpus.

Let's define this more concretely.

TF*IDF Defined

The TF (term frequency) of a word is the frequency of a word (i.e. number of times it appears) in a document. When you know it, you're able to see if you're using a term too much or too little.

For example, when a 100 word document contains the term "cat" 12 times, the TF for the word 'cat' is

$$TF_{cat} = 12/100 \text{ i.e. } 0.12$$

The IDF (inverse document frequency) of a word is the measure of how significant that term is in the whole corpus.

For example, say the term "cat" appears *x* amount of times in a 10,000,000 million document-sized corpus (i.e. web). Let's assume there are 0.3 million documents that contain the term "cat", then the IDF (i.e. $\log\{DF\}$) is given by the total number of documents (10,000,000) divided by the number of documents containing the term "cat" (300,000).

$$IDF(cat) = \log(10,000,000/300,000) = 1.52$$

$$\therefore W_{cat} = (TF * IDF)_{cat} = 0.12 * 1.52 = 0.182$$

Now that you have this figured out (right?), let's look at how this can benefit you.

How you can benefit from using TF*IDF

Gather words. Write your content. Run a TF*IDF report for your words and get their weights. The higher the numerical weight value, the rarer the term. The smaller the weight, the more common the term. Compare all the terms with high TF*IDF weights with respect to their search volumes on the web. Select those with higher search volumes and lower competition. Work smart.

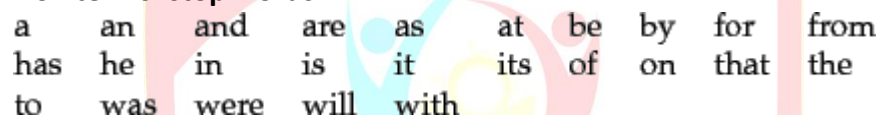
A good rule of thumb is, the more your content "makes sense" to the user, the more weight it is assigned by the search engine. With words having a high TF*IDF weight in your content, your content will always be among the top search results, so you can:

- stop worrying about using the stop-words,
- successfully hunt words with higher search volumes and lower competition,
- be sure to have words that make your content unique and relevant to the user, etc.

Stop words

What can we use n-gram models for? Given the probabilities of a sentence we can determine the likelihood of an automated machine translation being correct, we could predict the next most likely word to occur in a sentence, we could automatically generate text from speech, automate spelling correction, or determine the relative sentiment of a piece of text.

Dropping common terms: stop words



a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Figure 3: A stop list of 25 semantically non-selective words which are common in Reuters-RCV1.

Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called *stop words*. The general strategy for determining a stop list is to sort the terms by *collection frequency* (the total number of times each term appears in the document collection), and then to take the most frequent terms, often hand-filtered for their semantic content relative to the domain of the documents being indexed, as a *stop list*, the members of which are then discarded during indexing.

An example of a stop list is shown in Figure above. Using a stop list significantly reduces the number of postings that a system has to store. And a lot of the time not indexing stop words does little harm: keyword searches with terms like the and by don't seem very useful. However, this is not true for phrase searches. The phrase query "President of the United States", which contains two stop words, is more precise than President AND "United States". The meaning of flights to London is likely to be lost if the word to is stopped out.

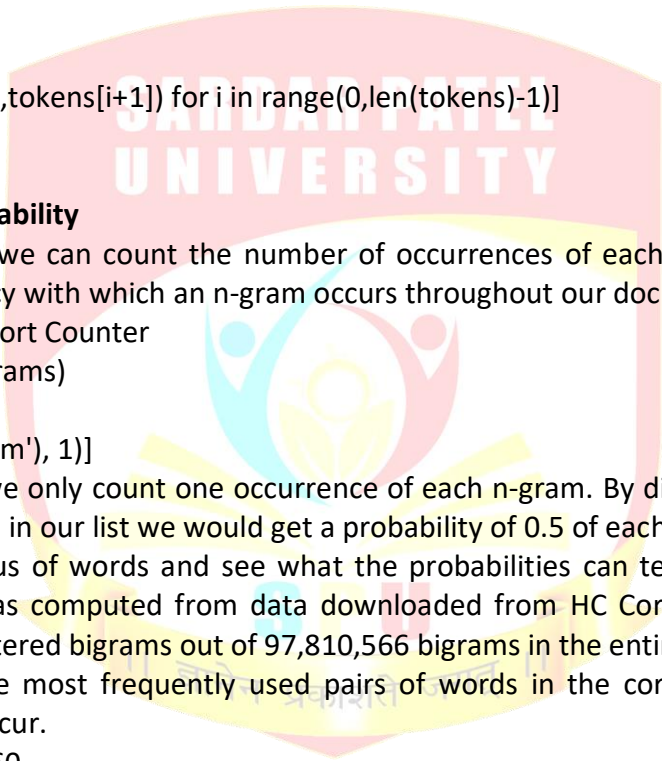
We may think will be difficult if the first three words are stopped out, and the system searches simply for documents containing the word think. Some special query types are

disproportionately affected. Some song titles and well known pieces of verse consist entirely of words that are commonly on stop lists (To be or not to be, Let It Be, I don't want to be, ...). The general trend in IR systems over time has been from standard use of quite large stop lists (200-300 terms) to very small stop lists (7-12 terms) to no stop list whatsoever. Web search engines generally do not use stop lists. Some of the design of modern IR systems has focused precisely on how we can exploit the statistics of language so as to be able to cope with common words in better ways.

N-Grams

What is an n-gram?

An n-gram is a contiguous sequence of n items from a given sequence of text. Given a sentence, s, we can construct a list of n-grams from s by finding pairs of words that occur next to each other. For example, given the sentence "I am Sam" you can construct bigrams (n-grams of length 2) by finding consecutive pairs of words.

```
>>> s = "I am Sam."
>>> tokens = s.split(" ")
>>> bigrams = [(tokens[i],tokens[i+1]) for i in range(0,len(tokens)-1)]
>>> bigrams
[('I', 'am'), ('am', 'Sam.')]

```

Calculating n-gram Probability

Given a list of n-grams we can count the number of occurrences of each n-gram; this count determines the frequency with which an n-gram occurs throughout our document.

```
>>> from collections import Counter
>>> count = Counter(bigrams)
>>> count
[('am', 'Sam.'), 1], (('I', 'am'), 1)]
```

With this small corpus we only count one occurrence of each n-gram. By dividing these counts by the size of all n-grams in our list we would get a probability of 0.5 of each n-gram occurring. Let's look a larger corpus of words and see what the probabilities can tell us. The following sequence of bigrams was computed from data downloaded from HC Corpora. It lists the 20 most frequently encountered bigrams out of 97,810,566 bigrams in the entire corpus.

This data represents the most frequently used pairs of words in the corpus along with the number of times they occur.

of	the	421560
in	the	380608
to	the	207571
for	the	190683
on	the	184430
to	be	153285
at	the	128980
and	the	114232
in	a	109527
with	the	99141
is	a	99053

for	a	90209
from	the	82223
with	a	78918
will	be	78049
of	a	78009
I	was	76788
I	have	76621
going	to	75088
is	the	70045

By consulting our frequency table of bigrams, we can tell that the sentence There was heavy rain last night is much more likely to be grammatically correct than the sentence There was large rain last night by the fact that the bigram heavy rain occurs much more frequently than large rain in our corpus. Said another way, the probability of the bigram heavy rain is larger than the probability of the bigram large rain.

Sentences as probability models

More precisely, we can use n-gram models to derive a probability of the sentence, W , as the joint probability of each individual word in the sentence, w_i .

$$P(W) = P(w_1, w_2, \dots, w_n)$$

This can be reduced to a sequence of n-grams using the Chain Rule of conditional probability.

$$P(x_1, x_2, \dots, x_n) = P(x_1)P(x_2|x_1)\dots P(x_n|x_1, \dots, x_{n-1})$$

As a concrete example, let's predict the probability of the sentence There was heavy rain.

$$P(\text{'There was heavy rain'}) = P(\text{'There'}, \text{'was'}, \text{'heavy'}, \text{'rain'})$$

$$P(\text{'There was heavy rain'}) = P(\text{'There'})P(\text{'was'}|\text{'There'})P(\text{'heavy'}|\text{'There was'})P(\text{'rain'}|\text{'There was heavy'})$$

Each of the terms on the right hand side of this equation are n-gram probabilities that we can estimate using the counts of n-grams in our corpus. To calculate the probability of the entire sentence, we just need to look up the probabilities of each component part in the conditional probability.

Unfortunately, this formula does not scale since we cannot compute n-grams of every length. For example, consider the case where we have solely bigrams in our model; we have no way of knowing the probability $P(\text{'rain'}|\text{'There was'})$ from bigrams.

By using the Markov Assumption, we can simplify our equation by assuming that future states in our model only depend upon the present state of our model. This assumption means that we can reduce our conditional probabilities to be approximately equal so that:

$$P(\text{'rain'}|\text{'There was heavy'}) \sim P(\text{'rain'}|\text{'heavy'})$$

More generally, we can estimate the probability of a sentence by the probabilities of each component part. In the equation that follows, the probability of the sentence is reduced to the probabilities of the sentence's individual bigrams.

$$P(\text{'There was heavy rain'}) \sim P(\text{'There'})P(\text{'was'}|\text{'There'})P(\text{'heavy'}|\text{'was'})P(\text{'rain'}|\text{'heavy'})$$

Synonyms

Search for the word "automobile" at Google, and the search engine might expand your search to include results for the word "car" as well, since it is a synonym of the word automobile.

Accidentally misspell the word as “automobile” and Google might automatically correct your spelling error and search for “automobile.”

Follow that up with a search for the word “driving” and Google could expand your query by using a process called stemming to look at the root of the word (driv-) and adding common endings to it, to come up with, and include in the search, such words as “driving,” and “driver.” This kind of query expansion is aimed at providing searchers with better search results. This method of expanding queries might not happen yet (though it sometimes appears to for spelling corrections at least), and it might not happen in all searches.

Typical approaches to query expansion include:

- Stemming of words,
- Correction of spelling errors, and;
- Augmentating search queries by doing things such as using synonyms of words that occur in the original query.

A couple of white papers from Google and a newly published patent application explore some of the ways that Google might use machine translation to find synonyms for words to expand the search terms that you might use.

There are a few different ways that expanding queries using synonyms can be done.

1) Synonyms for a word might be found in a thesaurus where those synonyms have been identified by experts, or a lexical ontology (an organized vocabulary of words).

2) Synonyms might be identified from other search queries that are syntactically similar (an ordering of and relationship between words in phrases that are similar) to the original query.

One challenge to those methods is when a word has multiple potential synonyms, with widely

3)

4)

5) varying meanings. For example, in the query “How to ship a box,” the word “ship” could have synonyms such as “boat” and “send.”

If that query is expanded based upon the boat meaning, it might provide very irrelevant search results to a searcher, who probably doesn’t expect to see search results related to fishing trawlers.

Methods, systems and apparatus, including computer program products, for expanding search queries. One method includes receiving a search query, selecting a synonym of a term in the search query based on a context of occurrence of the term in the received search query, the synonym having been derived from statistical machine translation of the term, and expanding the received search query with the synonym and using the expanded search query to search a collection of documents.

Alternatively, another method includes receiving a request to search a corpus of documents, the request specifying a search query, using statistical machine translation to translate the specified search query into an expanded search query, the specified search query and the expanded search query being in the same natural language, and in response to the request, using the expanded search query to search a collection of documents.

Using Statistical Machine Translation (SMT) to find Synonyms

The patent application goes into a good amount of detail on how Google might use statistical machine translation to translate a sequence of words from one language to another, to learn how words in different languages are related. If you want a detailed version of how statistical machine translation works, it’s worth looking at the patent filing for their description.

The Google Research Blog, in a post from 2006 titled **Statistical machine translation live**, provides a much simpler explanation:

Several research systems, including ours, take a different approach: we feed the computer with billions of words of text, both monolingual text in the target language, and aligned text consisting of examples of human translations between the languages. We then apply statistical learning techniques to build a translation model.

So, how does SMT help find synonyms?

The word “ship” in a particular context can be translated to another language the same way that the word “transport” can be. In that context, the word “ship” is synonymous with the word “transport”. So, our example above of a query such as “how to ship a box” might have the same translation as “how to transport a box.”

The search might be expanded to include both queries – “how to ship a box” as well as “how to transport a box.”

A machine translation system may also collect information about words in the same language, to learn about how those words might be related.

Approaches for Training a Statistical Machine Translation Model

The first step is collecting a training set of words, possibly from a number of different sources, such as the following:

1) Looking at Question-Answer Pairs

Imagine looking at as many frequently asked questions pages as possible, and comparing how the same questions are answered differently (or similarly). Taking those questions and answers pairs, and using them as a training body for statistical machine learning might be helpful.

2) Looking at Query and Snippet Pairs

Look at the search results for a query in a search engine, and the snippets of those results. Perhaps look even closer at the results that have been selected and viewed more frequently and/or longer by people who searched using those query terms (possibly indicating that those snippets are more relevant for the query term searched with).

Those query and snippet pairs might also be used as a training body for statistical machine learning. Text from the documents themselves, from anchor text in links pointing to those documents, and other information about words appearing in those results such as whether they were used in the page title, or if they are part of a string of text that is relevant to the query used may also be considered.

3) Look at phrase and paraphrase pairs

Like our examples above of “how to ship a box,” and “how to transport a box,” these phrases can be translated into the same term in another language, and that term might be reasonably translated back into either phrase.

Phrases and paraphrases might also be supplied manually by language experts. A body of synonyms and similar phrases might be collected from that approach.

A query such as “how to become a mason” might yield a translated search query of “how to be a bricklayer” using this approach.

Using Context Maps with Synonyms

Synonyms might be found during a search, or they might be prepared beforehand and used with a context map that pays attention to words that might appear to the left and right of one

of the words in a query phrase. The context map might be prepared before a search is ever conducted.

For example, with the query “how to tie a bow,” the left and right context of the word ‘tie’ in that query is “how to” and “a bow.”

In the context map, the word tie may be associated with two synonyms, ‘equal’ and ‘knot’. The word “knot” could be chosen as a synonym for “tie” since it also fits in well within the context found in the context map of “how to” and “a bow.” The query might be expanded to something like [how to (tie or knot) a bow].

Part of Speech

What is Part of Speech?

The **part of speech** explains how a word is used in a sentence. There are eight main parts of speech - **nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions** and **interjections**.

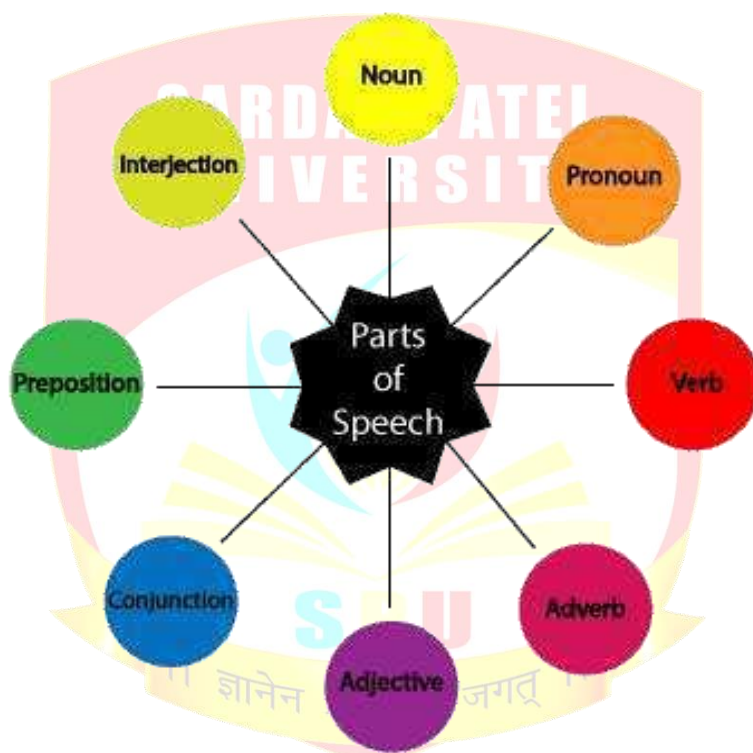


Figure 4: Part of speech components

- Noun (N)- Daniel, London, table, dog, teacher, pen, city, happiness, hope
- Verb (V)- go, speak, run, eat, play, live, walk, have, like, are, is
- Adjective(ADJ)- big, happy, green, young, fun, crazy, three
- Adverb(ADV)- slowly, quietly, very, always, never, too, well, tomorrow
- Preposition (P)- at, on, in, from, with, near, between, about, under
- Conjunction (CON)- and, or, but, because, so, yet, unless, since, if
- Pronoun(PRO)- I, you, we, they, he, she, it, me, us, them, him, her, this
- Interjection (INT)- Ouch! Wow! Great! Help! Oh! Hey! Hi!

Most **POS** are divided into sub-classes. **POS Tagging** simply means labeling words with their appropriate Part-Of-Speech.

How does POS Tagging works?

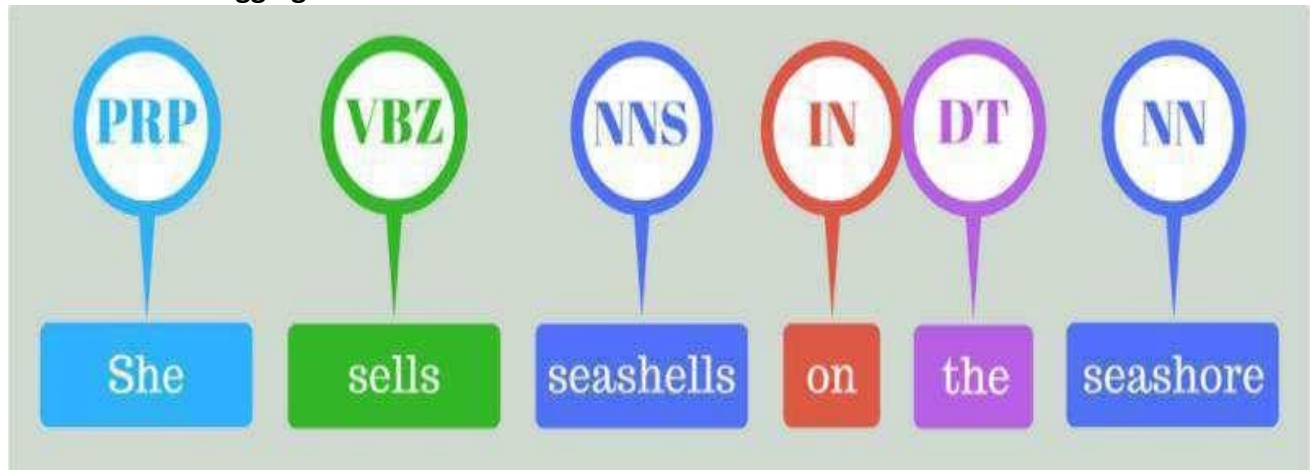


Figure 5: POS working

POS tagging is a supervised learning solution that uses features like the previous word, next word, is first letter capitalized etc. NLTK has a function to get pos tags and it works after tokenization process.

The most popular tag set is Penn Treebank tagset. Most of the already trained taggers for English are trained on this tag set.

What is Chunking?

Chunking is a process of extracting phrases from unstructured text. Instead of just simple tokens which may not represent the actual meaning of the text, it's advisable to use phrases such as "South Africa" as a single word instead of 'South' and 'Africa' separate words.

Chunking works on top of POS tagging, it uses pos-tags as input and provides chunks as output. Similar to POS tags, there are a standard set of Chunk tags like Noun Phrase(NP), Verb Phrase (VP), etc. Chunking is very important when you want to extract information from text such as Locations, Person Names etc. In NLP called Named Entity Extraction.

There are a lot of libraries which give phrases out-of-the-box such as Spacy or TextBlob. NLTK just provides a mechanism using regular expressions to generate chunks.

We will consider Noun Phrase Chunking and we search for chunks corresponding to an individual noun phrase. In order to create NP chunk, we define the chunk grammar using POS tags. We will define this using a single regular expression rule.

The rule states that whenever the chunk finds an optional determiner (DT) followed by any number of adjectives (JJ) and then a noun (NN) then the Noun Phrase(NP) chunk should be formed.

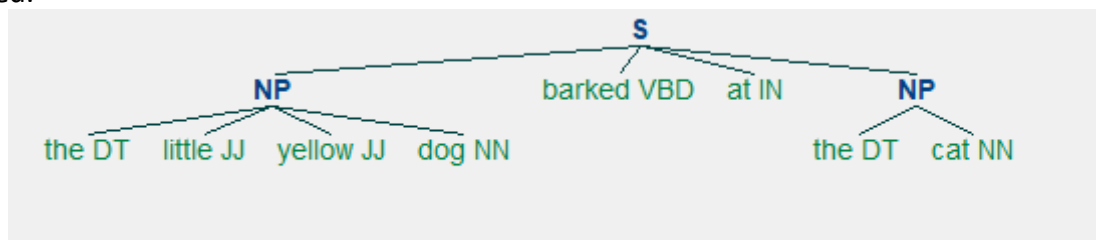


Figure 6: Chunking example

I hope you have got a gist of POS tagging and chunking in NLP. I have guided you through the basic idea of these concepts. There is much more depth to these concepts which is interesting and fun.

Architecture of POS tagger

1. Tokenization: The given text is divided into tokens so that they can be used for further analysis. The tokens may be words, punctuation marks, and utterance boundaries.

2. Ambiguity look-up: This is to use lexicon and a guesser for unknown words. While lexicon provides list of word forms and their likely parts of speech, guessers analyze unknown tokens. Compiler or interpreter, lexicon and guesser make what is known as lexical analyzer.

3. Ambiguity Resolution: This is also called disambiguation. Disambiguation is based on information about word such as the probability of the word. For example, power is more likely used as noun than as verb. Disambiguation is also based on contextual information or word/tag sequences. For example, the model might prefer noun analyses over verb analyses if the preceding word is a preposition or article. Disambiguation is the most difficult problem in tagging.

Applications of POS tagger

The POS tagger can be used as a preprocessor. Text indexing and retrieval uses POS information. Speech processing uses POS tags to decide the pronunciation. POS tagger is used for making tagged corpora.



Subject Name: **Data Science & Big data**

Subject Code: **CS-7005**

Semester: **7th**



Unit-5

Topics to be covered

Introduction to Web Search& Big data: Crawling and Indexes, Search Engine architectures, Link Analysis and ranking algorithms such as HITS and PageRank, Hadoop File system & MapReduce Paradigm

Introduction to Web Search& Big data: Crawling

Crawling refers to the ability of a search engine to traverse the billions of interlinked pages on the World Wide Web. With massive amounts of pages being generated on an hourly basis, it is impossible for us humans alone to visit, record and organize them on our own. Instead, automated search crawlers or “bots” conduct regular searches to save us the agony of finding relevant content ourselves.

Search bots wait upon signals from previously indexed pages, such as links, to be notified of new content. So if you have created a new page on your website and linked to it from an existing page or the main menu, this would be a signal to search bots that they should come visit and index it. New pages can also be introduced to bots through measures such as Sitemaps and robots.txt files. Platforms such as WordPress will automatically alert search engines that you have created a new page. Detection can be accelerated by verifying your website with search engines using Google Webmaster Tools or Bing Webmaster Tools.

Indexing

After a web page or document has been detected by crawlers, all its accessible data is stored (cached) on search engine servers so it can be retrieved when a user performs a search query. Indexing serves two purposes:

- to return results **related** to a search engine user’s query
- to rank those results **in order** of importance and relevancy

The order of ranking is dependent with each search engine’s ranking algorithm. These algorithms are highly complex formulas, made even more advanced by the relationship your website has with external sites and its on-page SEO factors.

To sum up, indexing exists to ensure that users questions are promptly answered as quickly as possible.

Ranking

As SEO’s this is the area we are most concerned with and the part that allows us to show clients tangible progress.

Once a keyword is entered into a search box, search engines will check for pages within their index that are a closest match; a score will be assigned to these pages based on an algorithm consisting of hundreds of different ranking signals.

These pages (or images & videos) will then be displayed to the user in order of score. So in order for your site to rank well in search results pages, it's important to make sure search engines can crawl and index your site correctly - otherwise they will be unable to appropriately rank your website's content in search results.

To help give you even more of a basic introduction to this process, here is a useful video from Google which explains it quite well. Each search engine follows a similar methodology to this.

Introduction to search engines

Search Engine refers to a huge database of internet resources such as web pages, newsgroups, programs, images etc. It helps to locate information on World Wide Web.

User can search for any information by passing query in form of keywords or phrase. It then searches for relevant information in its database and return to the user.



Figure 1: Google Search Engine

Search Engine Components

Generally there are three basic components of a search engine as listed below:

1. Web Crawler

2. Database
3. Search Interfaces

Web crawler

It is also known as **spider** or **bots**. It is a software component that traverses the web to gather information.

Database

All the information on the web is stored in database. It consists of huge web resources.

Search Interfaces

This component is an interface between user and the database. It helps the user to search through the database.

Search Engine Working

Web crawler, database and the search interface are the major component of a search engine that actually makes search engine to work. Search engines make use of Boolean expression AND, OR, NOT to restrict and widen the results of a search. Following are the steps that are performed by the search engine:

- The search engine looks for the keyword in the index for predefined database instead of going directly to the web to search for the keyword.
- It then uses software to search for the information in the database. This software component is known as web crawler.
- Once web crawler finds the pages, the search engine then shows the relevant web pages as a result. These retrieved web pages generally include title of page, size of text portion, first several sentences etc.

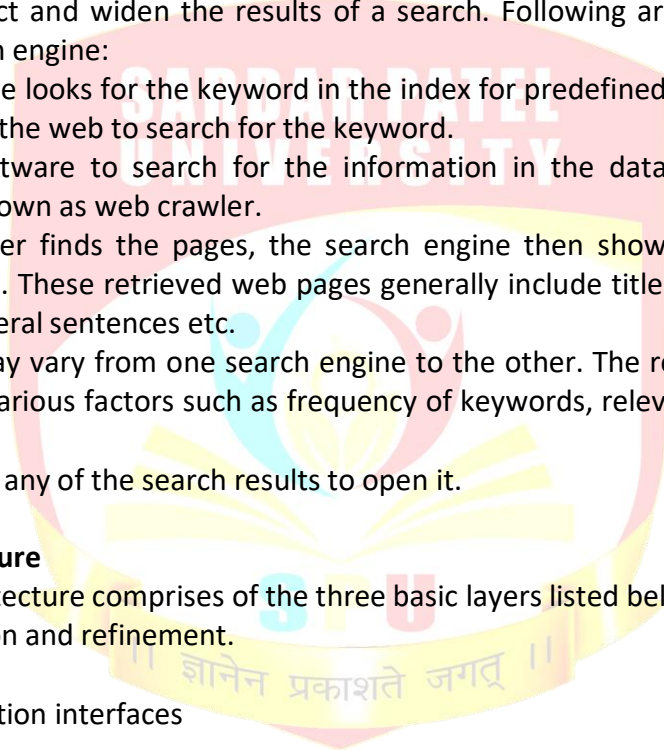
These search criteria may vary from one search engine to the other. The retrieved information is ranked according to various factors such as frequency of keywords, relevancy of information, links etc.

- User can click on any of the search results to open it.

Search Engine Architecture

The search engine architecture comprises of the three basic layers listed below:

- Content collection and refinement.
- Search core
- User and application interfaces



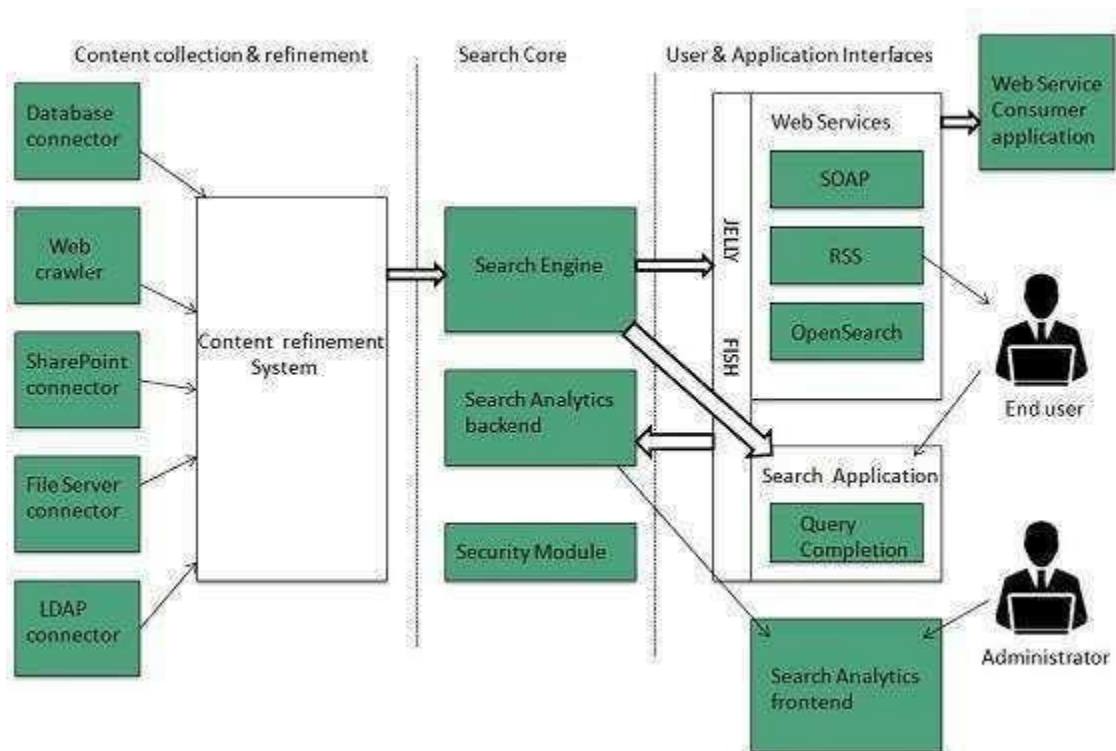


Figure 2: Search Engine Architecture

Search Engine Processing

Indexing Process

Indexing process comprises of the following three tasks:

- Text acquisition
- Text transformation
- Index creation

Text acquisition

It identifies and stores documents for indexing.

Text Transformation

It transforms document into index terms or features.

Index Creation

It takes index terms created by text transformations and create data structures to suport fast searching.

Query Process

Query process comprises of the following three tasks:

- User interaction
- Ranking
- Evaluation

User interaction

It supports creation and refinement of user query and displays the results.

Ranking

It uses query and indexes to create ranked list of documents.

Evaluation

It monitors and measures the effectiveness and efficiency. It is done offline.

Examples

Following are the several search engines available today:

Search Engine	Description
Google	It was originally called BackRub . It is the most popular search engine globally.
Bing	It was launched in 2009 by Microsoft . It is the latest web-based search engine that also delivers Yahoo's results.
Ask	It was launched in 1996 and was originally known as Ask Jeeves . It includes support for match, dictionary, and conversation question.
AltaVista	It was launched by Digital Equipment Corporation in 1995. Since 2003, it is powered by Yahoo technology.
AOL.Search	It is powered by Google.
LYCOS	It is top 5 internet portal and 13th largest online property according to Media Matrix.
Alexa	It is subsidiary of Amazon and used for providing website traffic information.

Link Analysis

Link analysis is literally about analyzing the links between objects, whether they are physical, digital or relational. This requires diligent data gathering. For example, in the case of a website where all of the links and back links that are present must be analyzed, a tool has to sift through all of the HTML codes and various scripts in the page and then follow all the links it finds in order to determine what sort of links are present and whether they are active or dead. This information can be very important for search engine optimization, as it allows the analyst to determine whether the search engine is actually able to find and index the website.

In networking, link analysis may involve determining the integrity of the connection between each network node by analyzing the data that passes through the physical or virtual links. With the data, analysts can find bottlenecks and possible fault areas and are able to patch them up more quickly or even help with network optimization.

Link analysis has three primary purposes:

- Find matches for known patterns of interests between linked objects.
- Find anomalies by detecting violated known patterns.
- Find new patterns of interest (for example, in social networking and marketing and business intelligence).

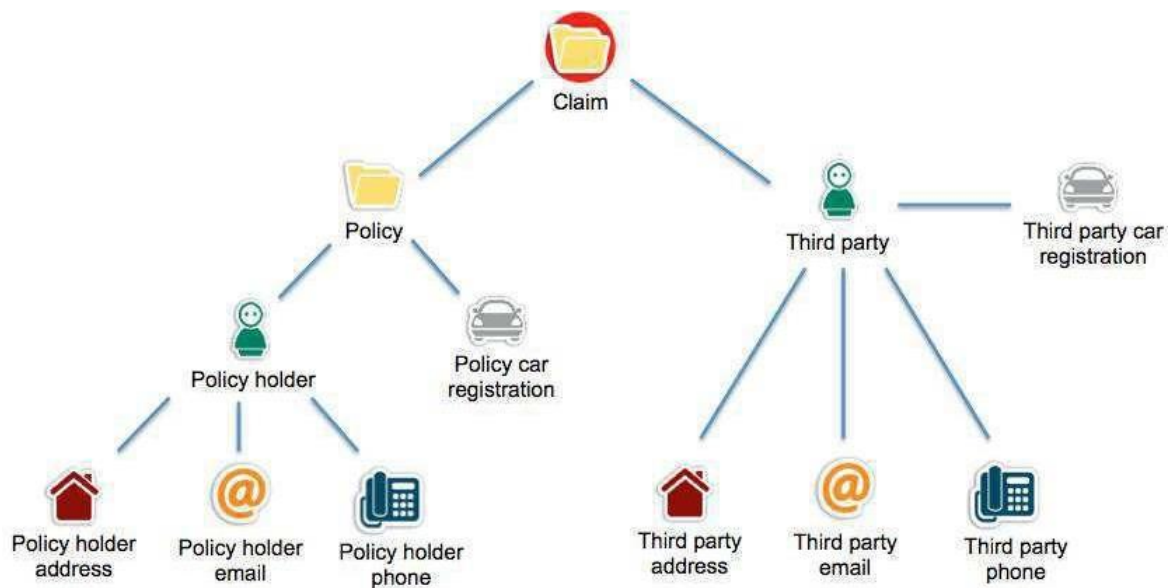


Figure 3: Link Analysis Example

Ranking Algorithm: Hits Algorithm

In the HITS algorithm, the first step is to retrieve the most relevant pages to the search query. This set is called the *root set* and can be obtained by taking the top pages returned by a text-based search algorithm. A *base set* is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The web pages in the base set and all hyperlinks among those pages form a *focused sub graph*. The HITS computation is performed only on this *focused sub graph*. According to Kleinberg the reason for constructing a base set is to ensure that most (or many) of the strongest authorities are included.

Authority and hub values are defined in terms of one another in a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to. Some implementations also consider the relevance of the linked pages.

The algorithm performs a series of iterations, each consisting of two basic steps:

- **Authority Update:** Update each node's *Authority score* to be equal to the sum of the *Hub Scores* of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.
- **Hub Update:** Update each node's *Hub Score* to be equal to the sum of the *Authority Scores* of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

The Hub score and Authority score for a node is calculated with the following algorithm:

- Start with each node having a hub score and authority score of 1.
- Run the Authority Update Rule
- Run the Hub Update Rule

- Normalize the values by dividing each Hub score by square root of the sum of the squares of all Hub scores, and dividing each Authority score by square root of the sum of the squares of all Authority scores.
- Repeat from the second step as necessary.

HITS, like Page and Brin's PageRank, is an iterative algorithm based on the linkage of the documents on the web. However it does have some major differences:

- It is query dependent, that is, the (Hubs and Authority) scores resulting from the link analysis are influenced by the search terms;
- As a corollary, it is executed at query time, not at indexing time, with the associated hit on performance that accompanies query-time processing.
- It is not commonly used by search engines. (Though a similar algorithm was said to be used by Teoma, which was acquired by Ask Jeeves/Ask.com.)
- It computes two scores per document, hub and authority, as opposed to a single score;
- It is processed on a small subset of 'relevant' documents (a 'focused sub graph' or base set), not all documents as was the case with Page Rank.

PageRank Algorithm

Introduction

PageRank is a topic much discussed by Search Engine Optimization (SEO) experts. At the heart of PageRank is a mathematical formula that seems scary to look at but is actually fairly simple to understand.

Despite this many people seem to get it wrong! In particular "Chris 'idings of www.searchenginesystems.net" has written a paper entitled "Page'ank Explained: Everything you've always wanted to know about Page'ank", pointed to by many people, that contains a fundamental mistake early on in the explanation! Unfortunately this means some of the recommendations in the paper are not quite accurate.

By showing code to correctly calculate real Page Rank I hope to achieve several things in this response:

1. Clearly explain how PageRank is calculated.
2. Go through every example in Chris' paper, and add some more of my own, showing the correct Page Rank for each diagram. By showing the code used to calculate each diagram I've opened myself up to peer review - mostly in an effort to make sure the examples are correct, but also because the code can help explain the PageRank calculations.
3. Describe some principles and observations on website design based on these correctly calculated examples.

Any good web designer should take the time to fully understand how PageRank really works - if you don't then your site's layout could be seriously hurting your Google listings!

[Note: I have nothing in particular against Chris. If I find any other papers on the subject I'll try to comment evenly]

How is PageRank Used?

Page'ank is one of the methods Google uses to determine a page's relevance or importance. It is only one part of the story when it comes to the Google listing, but the other aspects are

discussed elsewhere (and are ever changing) and PageRank is interesting enough to deserve a paper of its own.

PageRank is also displayed on the toolbar of your browser if you've installed the Google toolbar (<http://toolbar.google.com/>). But the Toolbar PageRank only goes from 0 – 10 and seems to be something like a logarithmic scale:

Toolbar (log base 10)	PageRank	Real PageRank
0		0 - 10
1		100 - 1,000
2		1,000 - 10,000
3		10,000 - 100,000
4		and so on...

We can't know the exact details of the scale because, as we'll see later, the maximum P' of all pages on the web changes every month when Google does its re-indexing! If we presume the scale is logarithmic (although there is only anecdotal evidence for this at the time of writing) then Google could simply give the highest actual PR page a toolbar PR of 10 and scale the rest appropriately.

Also the toolbar sometimes guesses! The toolbar often shows me a Toolbar P' for pages I've only just uploaded and cannot possibly be in the index yet!

What seems to be happening is that the toolbar looks at the URL of the page the browser is displaying and strips off everything down the last "/" (i.e. it goes to the "parent" page in U'L terms). If Google has a Toolbar PR for that parent then it subtracts 1 and shows that as the Toolbar P' for this page. If there's no P' for the parent it goes to the parent's parent's page, but subtracting 2, and so on all the way up to the root of your site. If it can't find a Toolbar P' to display in this way, that is if it doesn't find a page with a real calculated P', then the bar is grayed out.

Note that if the Toolbar is guessing in this way, the Actual PR of the page is 0 - though its PR will be calculated shortly after the Google spider first sees it.

PageRank says nothing about the content or size of a page, the language it's written in, or the text used in the anchor of a link!

Definitions

I've started to use some technical terms and shorthand in this paper. Now's as good a time as any to define all the terms I'll use:

PR:	Shorthand for PageRank: the actual, real, page rank for each page as calculated by Google. As we'll see later this can range from 0.15 to billions.
Toolbar PR:	The PageRank displayed in the Google toolbar in your

	browser. This ranges from 0 to 10.
Backlink:	If page A links out to page B, then page B is said to have a “backlink” from page A.

So what is PageRank?

In short PageRank is a “vote”, by all the other pages on the Web, about how important a page is. A link to a page counts as a vote of support. If there’s no link there’s no support (but it’s an abstention from voting rather than a vote against the page).

Quoting from the original Google paper, PageRank is defined like this:

We assume page A has pages $T_1 \dots T_n$ which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also $C(A)$ is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d (PR(T_1)/C(T_1) + \dots + PR(T_n)/C(T_n))$$

Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank or $PR(A)$ can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

but that’s not too helpful so let’s break it down into sections.

1. **$PR(T_n)$** - Each page has a notion of its own self-importance. That’s “ $P(T_1)$ ” for the first page in the web all the way up to “ $P(T_n)$ ” for the last page **$C(T_n)$** - Each page spreads its vote out evenly amongst all of its outgoing links. The count, or number, of outgoing links for page 1 is “ $C(T_1)$ ”, “ $C(T_n)$ ” for page n, and so on for all pages.
2. **$PR(T_n)/C(T_n)$** - so if our page (page A) has a backlink from page “n” the share of the vote page A will get is “ $P(T_n)/C(T_n)$ ”
3. **$d(\dots)$** - All these fractions of votes are added together but, to stop the other pages having too much influence, this total vote is “damped down” by multiplying it by 0.85 (the factor “ d ”)
4. **$(1 - d)$** - The $(1 - d)$ bit at the beginning is a bit of probability math magic so the “*sum of all web pages' PageRanks will be one*”: it adds in the bit lost by the **d** . It also means that if a page has no links to it (no backlinks) even then it will still get a small PR of 0.15 (i.e. $1 - 0.85$). (Aside: the Google paper says “the sum of all pages” but they mean the “the normalised sum” – otherwise known as “the average” to you and me.

How is PageRank Calculated?

This is where it gets tricky. The PR of each page depends on the PR of the pages pointing to it. But we won’t know what PR those pages have until the pages pointing to **them** have their PR calculated and so on... And when you consider that page links can form circles it seems impossible to do this calculation!

But actually it’s not that bad. ‘emember this bit of the Google paper:

PageRank or $PR(A)$ can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

What that means to us is that we can just go ahead and calculate a page's P' **without knowing the final value of the PR of the other pages**. That seems strange but, basically, each time we run the calculation we're getting a closer estimate of the final value. So all we need to do is remember the each value we calculate and repeat the calculations lots of times until the numbers stop changing much.

Lets take the simplest example network: two pages, each pointing to the other:

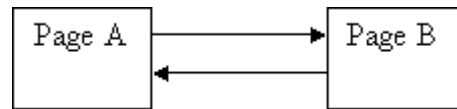


Figure 4: PageRank Example

Each page has one outgoing link (the outgoing count is 1, i.e. $C(A) = 1$ and $C(B) = 1$).

Hadoop File system & MapReduce Paradigm

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

Hadoop framework includes following four modules:

- Hadoop Common: These are Java libraries and utilities required by other Hadoop modules. These libraries provide filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- Hadoop YARN: This is a framework for job scheduling and cluster resource management.
- Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.
- Hadoop MapReduce: This is YARN-based system for parallel processing of large data sets.

We can use following diagram to depict these four components available in Hadoop framework.

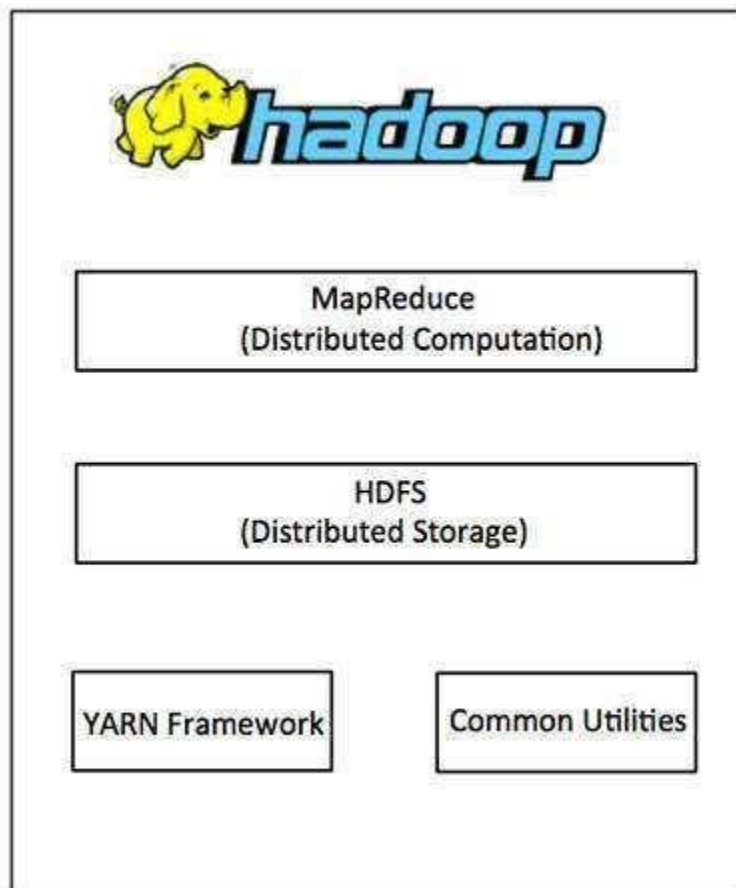


Figure 5: Hadoop Architecture

Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

MapReduce Paradigm

MapReduce Algorithm

MapReduce is a Distributed Data Processing Algorithm, introduced by Google in its MapReduce Tech Paper. MapReduce Algorithm is mainly inspired by Functional Programming model. MapReduce algorithm is mainly useful to process huge amount of data in parallel, reliable and efficient way in cluster environments. It divides input task into smaller and manageable sub-tasks (They should be executable independently) to execute them in-parallel.

MapReduce Algorithm Steps

MapReduce Algorithm uses the following three main steps:

1. Map Function
2. Shuffle Function
3. Reduce Function

Here we are going to discuss each function role and responsibility in MapReduce algorithm.

Map Function

Map Function is the first step in MapReduce Algorithm. It takes input tasks and divides them into smaller sub-tasks. Then perform required computation on each sub-task in parallel.

This step performs the following two sub-steps:

1. Splitting
2. Mapping
 - Splitting step takes input DataSet from Source and divides into smaller Sub-Datasets.
 - Mapping step takes those smaller Sub-Datasets and perform required action or computation on each Sub-Datasets.

The output of this Map Function is a set of key and value pairs as <Key, Value> as shown in the below diagram:

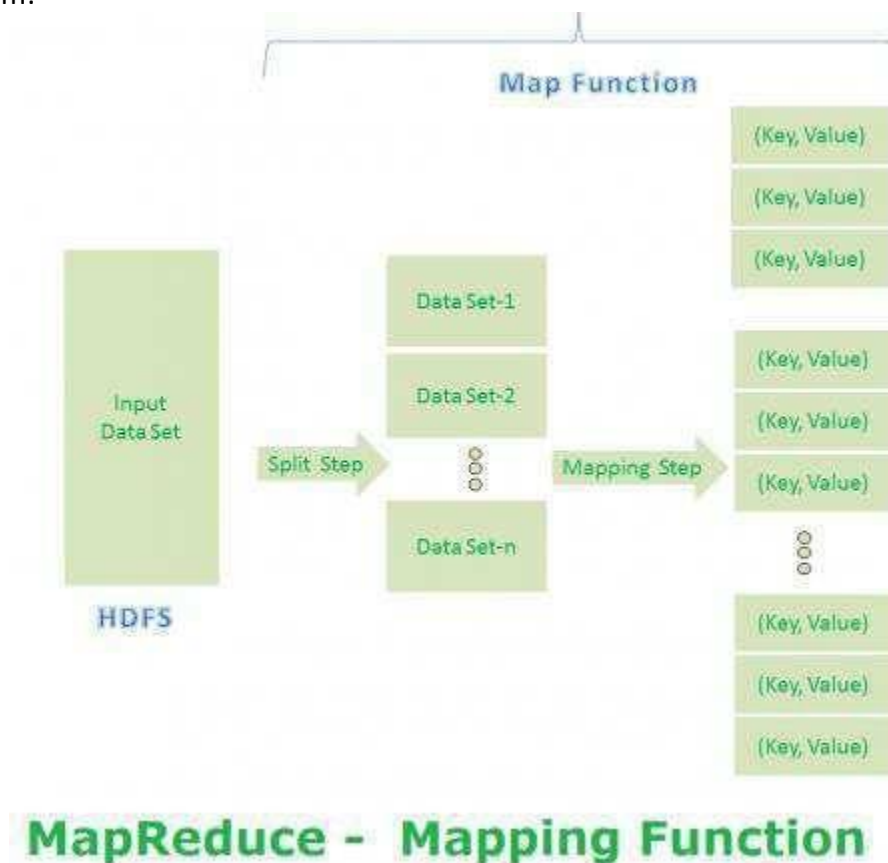


Figure 6: MapReduce – Mapping Function Example

Shuffle Function

It is the second step in MapReduce Algorithm. Shuffle Function is also known as “Combine Function”.

It performs the following two sub-steps:

1. Merging
2. Sorting

It takes a list of outputs coming from “Map Function” and performs these two sub-steps on each and every key-value pair.

- Merging step combines all key-value pairs which have same keys (that is grouping key-value pairs by comparing “Key”). This step returns <Key, List<Value>>.
 - Sorting step takes input from merging step and sorts all key-value pairs by using Keys. This step also returns <Key, List<Value>> output but with sorted key-value pairs.
- Finally, Shuffle Function returns a list of <Key, List<Value>> sorted pairs to next step.

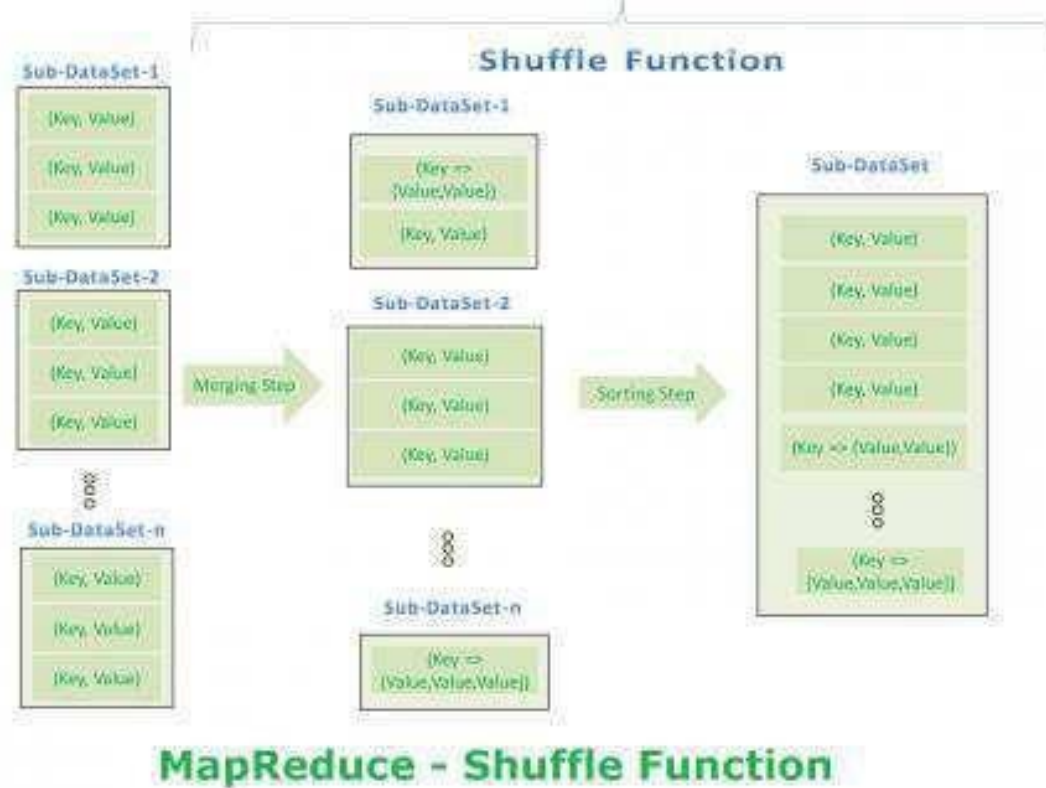
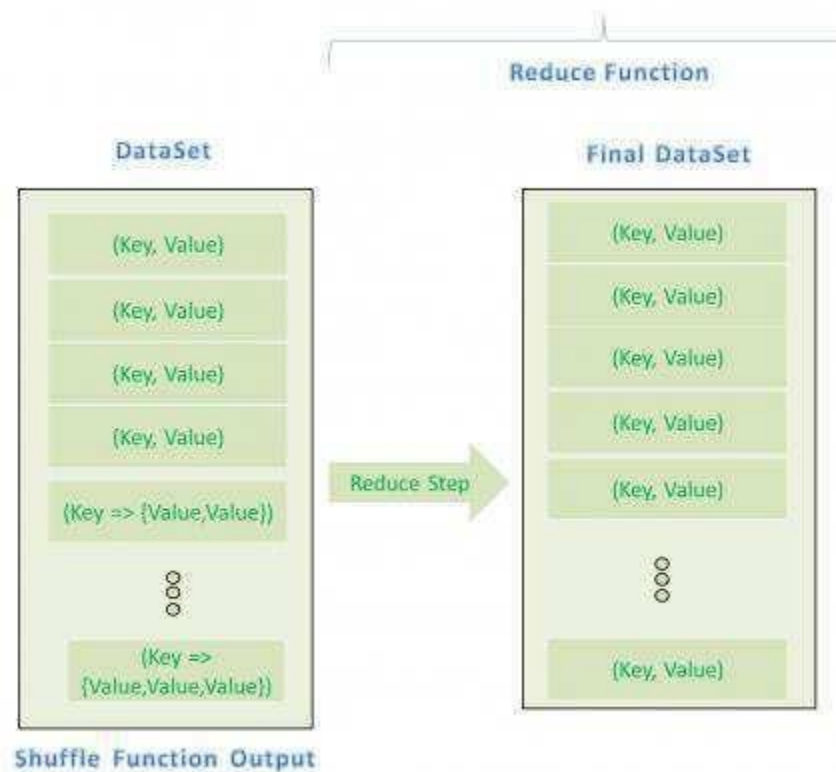


Figure 7: MapReduce – Shuffle Function Example

Reduce Function

It is the final step in MapReduce Algorithm. It performs only one step: Reduce step. It takes list of <Key, List<Value>> sorted pairs from Shuffle Function and perform reduce operation as shown below.



MapReduce - Reduce Function

Figure 8: MapReduce – Reduce Function Example

Hadoop Distributed File System

Hadoop can work directly with any mountable distributed file system such as Local FS, HFTP FS, S3 FS, and others, but the most common file system used by Hadoop is the Hadoop Distributed File System (HDFS).

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a single **NameNode** that manages the file system metadata and one or more slave **DataNodes** that store the actual data. A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes. The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

HDFS provides a shell like any other file system and a list of commands are available to interact with the file system. These shell commands will be covered in a separate chapter along with appropriate examples.

How Does Hadoop Work?

- Stage 1

A user/application can submit a job to the Hadoop (a hadoop job client) for required process by specifying the following items:

1. The location of the input and output files in the distributed file system.
 2. The java classes in the form of jar file containing the implementation of map and reduce functions.
 3. The job configuration by setting different parameters specific to the job.
- Stage 2

The Hadoop job client then submits the job (jar/executable etc) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

- Stage 3

The TaskTrackers on different nodes execute the task as per MapReduce implementation and output of the reduce function is stored into the output files on the file system.

#Advantages of Hadoop

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatically distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

