Data science -> Requires knowledge and skills in 3 major directions which is Mathematics (statistics and linear algebra), Computer Science (Machine learning, big data, data visualization), Domain-specific area (specific area the data is related to).

Main purpose of data science

* Recognize sources of data
* Extract insights from data
* Present data in an informative and accessible way
* Enable informed decisions based on data
* Predict future outcomes based on existing data
* Build models to complex systems

Who uses data science?

* Widely used in business, science and engineering
  + Marketing departments across corporate world
    - Involving customers data and metrics
  + Social media
    - Users analytics for targeted advertisement
  + News outlets
    - Studying targeted audiences
  + Technology Companies
    - Collect and analyze users data

Impact of data science in people’s life

* Using technology, we leave a digital footprint
* Computer systems have the ability to store and process huge amount of data
* Data we generate as a results of our activities
* Possible to built systems which never forget
* Benefit customers as data allow companies to improve in their products and services
* But data can affect people negatively.
* Range of data driven technologies can be found everywhere
* Ability to store and process personal information
  + Internet of things
  + Wearable devices
  + Online services (banking, shopping, services provided by local and national governments)
  + Various surveillance technologies
* Utilization of biometrics adds another level of concern(privacy perspective)

Prominent areas in data science

* Medicine, including drug discovery, medical imaging and diagnosis
* Finance and various financial technologies
* Social media
* Marketing
* Robotics and automation

**Datum**

* Entities subject to a data science study are described by a number of attributes
  + Entity: Book
    - Attributes: Author, Title, Genre, Publisher, Year, Price
  + Entity Vehicle
    - Attributes: MPG, Cyi, Engine, HP, Weight, 0to60, Year, Origin

Data points

* Data point is a single unit of observation
* Most immediate notion of a data point is numerical value
  + Can be integer, real or complex
* Or Binary
  + 0/1 or True/False
* Computer systems are digital(binary) machines, work on any type of data(such as text, sound or video) is actually done on the digital(binary) representation of that data.
* Programming languages and software systems add an abstraction layer, which allows user to process data in their original type and format.
* Allows users to directly apply domain-specific functions on data
  + Audio
  + Image processing and editing

Dataset

* Dataset is a collection of data points
* Usually two-dimensional, rows representing entities(items) and columns representing attributes(features) similar to database
* Entities, or items are part of collection such as books in a library or items in a shop
* Attributes or features describe the entities and can be of different types

Table

Description automatically generated with medium confidence

Issues with dataset

* Real world data is not always consistent
* Common issues
  + Missing attributes
  + Attributes of incorrect type or with incorrect values
* Requires pre-processing(cleaning)
  + Transforming a table into first normal form in the context of relational databases

Structure of data

* Datasets can be classified based on structure
  + Structured data
    - Organized into tables
    - Easy to search, process, study
  + Unstructured data
    - Cannot be organized into a consistent structure
    - Most real-world data are unstructured
  + Semi-structured data
    - Some structure exists but does not encompass an entire dataset
      * Emails
      * Digital images

Data types

* Concept of data types is key in statistics

Data types in statistics

* Statistics data can be
  + Categorical
    - Nominal
    - Ordinal
  + Numerical
    - Interval
    - Ratio

Data types in computer science

* Data types depend on
  + Programming paradigm
    - Can be concurrent, parallel, object-oriented, functional
  + Programming language some languages share similar set of data types
    - C/C++, C#, Java
* Specific language implementations
  + Different interpreters, compiles and IDE can offer some variations in data type range, precision and required memory
* General grouping of data types
  + Primitive (built-in)
  + Composite (records, structures, classes)
  + Abstract (including data structures)
* Programming languages offer set of primitive data types which can be accessed without the need of additional libraries
* Slight variations between languages but primitive data types are
  + Boolean
  + Integer
  + Real(float)
  + Character
  + String(text)
  + Reference(pointer)
* Datasets are stored in data structures before being processed
* In context of python, most widely used data structures include
  + Lists
  + Dictionaries
  + Arrays (Numpy)
  + DataFrames(pandas)
  + Graphs(NetworkX)

Data-related issues require pre-processing-> duplicated values, missing values

* Pandas -> Data manipulation
* Matplotlib -> Plotting numerical data
* Scikit-learn -> Machine learning
* Numpy
  + N-dmensional array processing
  + Data manipulation
* Python was designed to combine the computational capabilities and the expressive power of **ANSI C** and **shell script**
* Python considered to have a disadvantage of lower performance
* Python type system
  + Dynamically typed
  + Strongly typed
* Jupyterlab -> Cells are text boxes containing the source code of the program
* Jupyter kernel -> Language specific process within the environment of jupyter
* Jupyterlab
  + Can run code interactively
  + Possible to link a code console to jupyter kernel
  + Jupyterlab is a well based ide
* Running jupyter with –no-browser option -> launch notebook server without opening a browser
* Virtual environments
  + Assign project specific paths to external binaries
  + Isolate a project’s dependencies from other projects
  + Select different python versions for different projects

Data types

* Data science deals with a wide range of data types
  + Boolean (logical)
  + Numerical
  + Textual
  + Images
  + Sound
  + Video
  + Digital signals
* Computers are digital/binary machines, storing and processing any type of data is actually done on their corresponding binary representations
* There are a number of different binary representations
  + One’s and two’s complement
  + IEEE standardized floating point number
* Most programming languages support the following numerical data types
  + Integer
  + Real
  + Complex

Numpy library

* Aka Numerical Python
* Provides numerical functionality to process arrays (and primary data types)
* Popular for data science related tasks
* Range of mathematical functions provided
* Extensively used along with other libraries like scipy and pandas.
* Primitive type variables and arrays correspond to scalars and vectors as studied in linear algebra, a branch of mathematics with significant applications in data science

Main functionalities of Numpy

* Allocate memory for arrays (NumPy arrays)
  + Arrays may have various dimensions
* Initialize arrays with arbitrary values
  + Zero or other constants
  + Random values
  + External sources like csv, spreadsheet table and xml
* Extract statistical information from the data(aggregate)
* Perform some linear algebra tasks

Memory allocation

* One-dimensional array or integers using the ‘empty’ method
  + np.empty(shape= NumberOfElements, dtype=int)
* Does not initialize the elements of the array
* Elements have random values found on the memory locations at the time of the allocation
* Two-dimensional array or integers using the ‘empty’ method
  + np.empty(shape=(rows,columns),dtype=int)
* Will have x rows x columns un-initialized elements

Initialize arrays

* One-dimensional array or integers using a random number generator
* Specify the number of elements, their range and type:
  + np.random.seed(0) # set a seed so result will always be same every run
  + np.random.randint(min, max, number\_of\_time, dtype=10)
  + Initialize x number of element based on number\_of\_time with values from min to max
* Initialize all elements with a specific value
  + Zeros
    - np.zeros(shape=NumberOfElements, dtype=int)
  + Ones
    - np.ones(shape=NumberOfElements, dtype=int)
  + Arbitrary value
    - np.ones(shape=NumberOfElements, fill\_value=7, dtype=int)
  + Random value
    - np.randint(0,100,NumberOfElements)

Random values

* Two dimensional array
  + myArray[i,j]
  + myArray[i][j]
* Assigning values
  + myArray[index] = np.random.randint(-5,5)

Numpy arithmetic

* Common tasks
  + Minimum, maximum, sum, average
* Needed when
  + Calculating the range of the data
  + Normalizing the data
  + Summarizing and comparing datasets
* Finding min and max
  + myArray.min()
  + myArray.max()

Array arithmetic

* Sum and average
  + myArray.sum()
  + np.average(myArray)
* Finding sum per column and per row
  + np.sum(myArray, axis=axisValue)
    - 0 is by column
    - 1 is by row
* Per column and per row operations not limited to ‘sum’
* Available for minimum, maximum and average

Aggregate values => Average, sum

In 2D array, result of aggregation by column is array

Initialize an array with a constant number => myArray = np.full(shape=10, fill\_value=15, dtype=int)

Data preprocessing

* Acquired data lack the structure, consistency, and format necessary for computer algorithms to perform their tasks successfully.
* Machine learning, data visualization, computer vision and other data science-related fields require the datasets to be consistent and well-structured.
* Most common issues with data
  + Missing values
  + Type inconsistency or type mismatch
  + Duplicating values, including entire rows and columns
* Resolving this issue is called preprocessing
* Corresponds to bringing a database to 1NF.
* Common preprocessing task is to extract useful data from an existing dataset
* Amounts to generating a sub-array based on an original array
* Can be done with Numpy array by using the advanced indexing

Advanced indexing

* Advanced indexing allows us to build new arrays based on existing arrays.
* By specifying a set of elements to be included in the new array.
* Can be applied to arrays with different dimensions.
* myArray = np.array([list\_of\_element])
* Extract a sub-array
  + myArray[start\_index:end\_index]
    - start\_index => included
    - end\_index => excluded

Advanced indexing: setting the step

* Can skip element as step(stride)
  + [start\_index : end\_index : step]
  + Get all even element
    - [::2]
  + Get all odd element
    - [1::2]

Advanced indexing: moving to 2D

* For two-dimensional array
  + myArray[row\_start\_index : row\_end\_index, column\_start\_index:column\_end\_index]
  + myArray[1:4, 0:3]
    - Retrieve row 1,2,3 in a 2D array
    - Retrieve 0,1,2 elements in each row

Advanced indexing

* Data science deals with data organized in the form of a table similarly to XML and spreadsheets.
* Not uncommon for tables to contains rows and columns which prevent further processing
  + Missing values
  + Type mismatch
  + and others
* Removing, or ‘dropping’, problematic rows and columns is the first and in some cases, the only step of the preprocessing task
* Can be achieved with other libraries(pandas), which support alternative approaches with similar results.

Structured data

* Datasets can be split into two groups based on the type of their data points
  + Datasets containing data points of the same type
    - Usually numerical, either integer or real
  + Containing data points of multiple data types
    - Numerical, textual, Boolean
* Numpy library provides the functionality to store and process arrays of structured data, that is data consisting of multiple data types
* Structure needs to be consistent throughout the datasets.
* Numpy array constructed in this way correspond to tables from a relational database
* Example of defining a book
  + myArray = np.empty(shape=10, dtype=[(‘Title’,’U10’),(‘Year’,int), (‘Price’, float)])
* Structured data is usually stored as CSV,XML or a spreadsheet table
* Pandas handle this task differently.

Statistics with Numpy

* Statistics is a major branch of mathematics
* Studies the properties of numerical data which is data points organized in datasets
* Statistics can generalize the properties of datasets and draw conclusions based on data
* Two main measurement
  + Measures of central tendency
    - Evaluate a central value, around which the data cluster around
      * Mean(average)
      * Median(the middle value, splits the dataset into two halves)
      * Mode (Most repeating value)
    - Fastest way to extract this information
      * Mean (np.mean(myArray))
      * Median (np.median(myArray))
      * Mode (st.mode(myArray)) (from scipy import stats as st)
  + Measures of spread
    - Evaluate how data are spread around certain central value
    - Common measures
      * Range
        + Difference between maximum and minimum value in dataset
        + np.max(myArray)-np.min(myArray)
      * Standard deviation
        + Measures average distance of the data points from mean value
        + np.std(S)
      * Variance
    - Since variance = sd2, we only need to calculate the standard deviation
    - Dataset consists of multiple series of data organized into tables
    - These series might be organized to rows or columns
    - Bring the need to extract statistical information from the dataset per-row or per-column rather than overall
    - Numpy allows extraction by column in 2D array
      * np.mean(myArray, axis=0)
      * np.median(myArray, axis=0)
      * st.mode(myArray, axis=0)
      * np.std(myArray, axis=0)

Introduction to linear algebra

* Linear algebra is a major branch of mathematics
* Significant theoretical background and extensive applications in computer science and data science
* Volume and scope of linear algebra is considerable
* Key concepts in linear algebra
  + Linear combinations
  + Systems of linear equations
  + Linear transformations (aka linear maps)

Scalars

* In mathematical sense, scalars are values which have only magnitude
* Real numbers or any of their subsets
  + Natural, integer and rational
* Complex numbers are usually treated as scalars
* In some context they can be considered two-dimensional vectors

Vectors

* In 2D and 3D spaces, vector is a quantity represented by an arrow with both direction and magnitude
* Linear algebra is a concept of a vector is generalized for any dimension(n)
  + v=(a­­1,a2,an)
* Algebraic notion of a vector corresponds to the array data type, as supported by a number of programming languages.
* Main operations
  + Scalar multiplication
    - Scalar:a
    - Vector v = (v1,v2, …, vn)
    - av= a (v1,v2, …, vn)= (av1,av2, …, avn)
    - example at topic 2 – Numpy slide 27
  + Vector addition
    - Vector x,y,z (same dimension)
    - x = (x1,x2, …, xn), y=(y1,y2, …, yn)
    - z = x+y = (x1+ y1,x2 + y2, …, xn+ yn)
    - example at topic 2 – Numpy slide 27
  + Vector subtraction same as addition
  + Dot product
    - Vector x,y,z (same dimension)
    - x = (x1,x2, …, xn), y=(y1,y2, …, yn)
    - z = x∙y = x1y1 + x2y2 + xnyn
    - example at topic 2 – Numpy slide 28

Matrices

* A rectangular table of components (scalars or numbers)
* Scalars are arranged in rows and columns by using a two-index notation
* Generalized matrix M with mxn components
* A picture containing chart

  Description automatically generated

Matrices and Numpy

* Similarly to vectors, matrices can be defined as two-dimensional numpy arrays.
* Every square matrix has a determinant
  + Determinant is a single value which has important mathematical meanings
  + Key question is whether a determinant has zero or non zero value
  + Calculating a determinant requires multiple additions and multiplications and is done by using a software tool
  + A 3x3 matrix
    - If one of the row (or column) is a linear combination of another row (or column), the value of the determinant is 0.
    - np.linalg.det(myArray)
    - A picture containing chart

      Description automatically generated
    - Rank
      * A square matrix with non-zero determinant has rank equals to it’s dimensions
      * We can lower the rank of a matrix by defining a row as a linear combination of another row
      * np.linalg.matrix\_rank(myArray)
    - Trace
      * Trace of a matrix is the sum of all elements on the main diagonal.
      * np.trace(myArray)
      * Text

        Description automatically generated 1+3+8
* Matrix multiplication
  + Matrix multiplication is key operation in linear algebra
  + Result of multiplication is also a matrix
  + Is a dot product
  + Diagram

    Description automatically generated with medium confidence
  + np.matmul(A,B)
* Inverse matrix
  + A square matrix with non-zero determinant has an inverse
  + In the case of real numbers, where the multiplicative inverse of r is 1/r (e.g. multiplicative inverse of 2 is 0.5)
  + The inverse of matrix M is usually denoted with M-1
  + M\*M-1 = M-1\*M = I
  + Where I is the identity matrix
  + Multiplying any square matrix with I does not change the matrix
  + A close-up of a stethoscope

    Description automatically generated with low confidence
  + np.linalg.inv(M)

Linear equations

* Systems of linear equations and the approaches to solving them
* Have significant applications in CS, engineering, finance technology and many other fields
* General form of a system of linear equation
  + A picture containing text, antenna

    Description automatically generated
  + Text, letter

    Description automatically generated
  + A picture containing table

    Description automatically generated
  + Text

    Description automatically generated
  + A picture containing text

    Description automatically generated
  + Graphical user interface, text, application

    Description automatically generated

Scalar value -> Rank, Determinant, Trace

Pandas

* Provide data structures and tools for data analysis
* Open source
* Widely used in academia and industry
* Integrates with other packages to enable a complete data analysis workflow
  + Numpy
  + Scipy
  + Matplotlib
  + scikit-learn

Core data structures

* Series
  + One-dimensional labelled array
  + Data can be any type
* DateFrame
  + Two-dimensional labelled table
  + Columns can be of different types

Store data from a CSV file into pandas Dataframe

* df=pandas.read\_csv(‘Data.csv’)

Can be index in pandas series

* Character
* Integer

Removes rows with missing values

* new\_df = DataFrame.dropna(axis=0)
  + column will be axis=1
  + is not an in-place operation as it reassign the variable

Time-series

* Change over time
* Line graphs are the most common method
* A sequence of variable measurements, indexed by time like population etc..

Uses and applications

* Business and finance
  + Analysis of stock price, costs, profit, unit sold..
* Organizational
  + Monitoring process or quality, workload projections
* Government and policy making
  + Changes in economic, health, crime, statistics
* Science and engineering
  + Signal processing, energy efficiency, cell mutation
* Anything involving measurements over time

Looking for trends

* Seasonal variation
* Significant events
  + Natural disasters, changes of government, tax regulation
* Correlations
  + Do variable change in similar ways over time?
* Understanding the past
* Predicting the future

Different kinds of patterns

* Variability
* Rate of change
* Co-variance and correlation
* Cycles
* Exceptions

Representing Time

* Sampling rate
  + Some variables are constantly changing over time
    - Temperature, physiological response, stock market
      * Temperature->one measurement per hour or one measurement per second?
      * Stock market-> trend over a whole year or changes over milliseconds
  + Rate of measurement is important

Time related concepts in pandas

* Date times => A specific date and time with timezone support
* Time deltas => An absolute time duration
* Time spans => A span of time defined by a point in time and its associated frequency
* Date offsets => A relative time duration that respects calendar arithmetic
* Table

  Description automatically generated

Timestamps – point in time

* pd.Timestamp(“date\_string”)
* pd.Timestamp(year=year, month=month, day=day)
* dayfirst parameter will set it to day come first (pd.Timestamp(“date\_string”, dayfirst=True))
* yearfirst parameter will set it to year come first(pd.Timestamp(“date\_string”, yearfirst=True))
* format parameter can set the format of the date(pd.Timestamp(“date\_string”, format=’%y/%m/%d’))
* generating sequences of timestamps
  + a list will automatically be converted into a DatetimeIndex
  + date\_time\_pd = pd.to\_datetime([list of date string])
  + date\_time\_pd.inferred\_freq => show the frequency between date
  + date\_time\_pd.freq = ‘2D’ => set frequency to 2 days
* Assembling from multiple columns
  + df = pd.DataFrame({‘year’:list\_of\_year\_number, ‘month’: list\_of\_month\_number, ‘day’:list\_of\_day\_number, ‘hour’: list\_of\_hour\_number})
  + year, month, day are required
  + hour, minute, second, millisecond, microsecond, nanosecond
  + pd.to\_datetime(df[[‘year’,’month’,’day’,’hour’]])
* range timestamp
  + pd.date\_range(start\_date, end\_date, freq=’1D’) => 2 day interval
  + pd.date\_range(string\_datetime, periods=6, freq=’6H’) => 6 hour interval 6 time
* Time as index
  + pd.Series(range(3), index=pd.date\_range(‘2000’, freq=’D’, periods=3))
* Time as data
  + pd.Series(pd.date\_range(‘2000’, freq=’D’, periods=3))
* Missing time data
  + pd.Timestamp(pd.NaT)
  + acts the same as np.nan

Rolling correlation measure How similar the change between two groups over a windowed period of time.

Lag plot is a scatter plot that visualizes data values on the x-axis against the same data shifted one unit of time on the y-axis.

Both lag plots and autocorrelation plots visualize how correlated a time series is with itself

An autocorrelation plot visualizes the correlation between time-series data and itself shifted across all time intervals

Data

* Quantitative => measurements of quantity
* Qualitative => measurements of quality

DRIP syndrome => we are **data rich** but **information poor**

How to make sense of data?

* Visualization => using graphics to explore and understand data(use human visual capacities)
* Data mining => searching for patterns and trends
* Machine learning => build models, test theories, predict the future

Data != Information

Data visualization: Data -> Information

Data visualization

* Making visible the patterns and structures in data using graphs, tables, diagrams, maps etc.
* Amplify cognition
* Representation and presentation of data that exploits our visual perception abilities in order to amplify cognition
* Purposes
  + Reveal things that are
    - Important
    - Meaningful
    - Useful
  + Understanding and knowledge

Information visualization

* Is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition

Information vs scientific visualization

* Information visualization
  + Quantitative data representing abstract concepts, processes and behaviours
    - E.g. shares prices, child literacy, sales, happiness
  + Focus on understanding social, psychological, economic phenomena
* Scientific visualization
  + Quantitative data representing physical objects
    - E.g. geographic data, human body, cell structure, gravity
  + Understanding the structure of the physical world
* Topic 3 data visualization page 7 for examples
* Similarities
  + Use visual modality
  + Represent data in ways our brain can comprehend
* Shared objectives
  + Amplify cognition
  + Create and communicate knowledge

Importance of vision -> Visual displays provide the highest bandwidth channel from the computer to the human. Indeed, we acquire more information through vision than through all of the other senses combined

Semiotics of graphics

* Study of visual symbols and how they convey meaning
* Classical view (Saussure)
  + All symbols are arbitrary
  + Must be learnt
  + No system of representation can be better than any other
* Scientific view
  + The visual system has particular properties (product of evolution)
  + Different forms of representation are more closely aligned with our perception mechanisms
  + There is some grounds for determining better/worse forms of visual communication

Sensory vs arbitrary symbols

* Sensory => Aspects of visualizations that derive their expressive power from their ability to use the perceptual processing power of the brain without learning
* Arbitrary => Aspects of representation that must be learned, because the representations have no perceptual basis

Sensory symbols

* Well matched to the early stages of neural processing
* Tend to be stable across individuals, cultures, and time
* Topic 3 – visual perception page 2 for example

Arbitrary symbols

* Derive power from culture
* Dependent on individual cultural knowledge
* Topic 3 – visual perception page 3 for example

What can we do with visualizations? => Data visualization is an umbrella term to cover all types of visual representations that support the exploration, examination and communication of data

Data visualization types

* Exploratory
  + Early stages of research
  + Interactive
  + Quick and dirty
  + Helps discover trends and patterns
* Explanatory
  + Later stages of research
  + Polished, publication/presentation ready
  + Conveys a clear message
  + Makes a point or answers a question

Uses for data visualization

* Analysis
* Communication
* Monitoring
* Planning

Value of data visualization

* Greatest value of a picture is when it forces us to notice what we never expected to see
* Further values of data visualization
  + Industry demand for data analysis skills
  + Useful in many areas of work and study
  + Practical data manipulation skills
  + Experience of research
  + Visualization is a powerful means of communication and making a difference

Matplotlib

* Static, animated, and interactive visualizations
* Free and open source
* Easy to use
* Highly customizable
* Publication quality
* Extensible
  + Seaborn, holoviews, ggplot, cartopy

Different ways of working with Matplotlib

* Pyplot API
  + Quick and simple
  + Flat namespace
  + Automatic state management
  + Less fined-grained control
* Object-oriented API
  + Modular structure
  + Can feel more complex to Python beginners
  + Explicit state management
  + Full control

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Descriptive statistics

* Summary statistics
* Provide quantitative descriptions or summaries of data
* Help us understand data
* Help us decide what data to present and how
* Are the fundamental quantities visualized in many plot types

Data critical

* Vital that we know what/who any data represents
* How was it collected?
* Why was it collected?
* Who was involved?
* What valid conclusions can we draw?

Population -> the collection of all individuals or items under consideration

Sample -> the part of population from which information is obtained

Data collection

* Limitations in data collection
* What is feasible?
  + Employee salary from one particular company?
* What is difficult or impossible?
  + Household income of the whole country?
  + Height of all 8 year old boys?

Descriptive vs Inferential statistic

* Descriptive statistics -> describe, show, or summarize data in a meaningful way such that patterns might emerge from the data
* Inferential statistics -> techniques that allow us to
  + Use samples to make generalization about populations
  + Use data about the past to predict the future

Univariate and multivariate analysis

* Univariate analysis -> understand the shape, size and range of quantitative values (a single variable)
* Multivariate analysis -> explore the possible relationships between different combinations of variables and variable types (many variables)

Basic descriptive statistics

* Central tendency
* Variability and dispersion
* Correlation between variables

Measure of central tendency

* Mode
  + Most frequent score in a data set
* Median
  + Middle score for a set of data arranged in order of magnitude
* Mean (‘average’)

Table

Description automatically generated

Table

Description automatically generated

Chart, bar chart

Description automatically generated

Graphical user interface, text

Description automatically generated

Table

Description automatically generated with low confidence

Text, table

Description automatically generated

Central tendency is the most typical data values

Mean is an appropriate measure of central tendency for ratio and interval

Why median is often used rather than mean is because it is less susceptible to being skewed by outliers

Mode support categorical variables

Quartiles

* In a rank-ordered data set
  + Q1 median of the first half
  + Q2 median of the entire data set
  + Q3 median of the second half
* Different methods for computing Q1 and Q3
  + Include the Q2 data point in each “half” or not?
    - Yes for odd number of data items
    - Default in pandas and R

A picture containing table

Description automatically generated

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

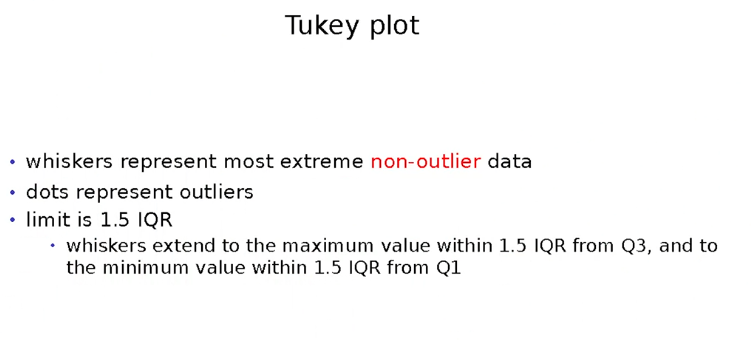
Description automatically generated

Interquartile range(IQR)

* Difference between the upper and lower quartiles
  + IQR = Q3 – Q1
  + The range of the middle 50% of your data

Graphical user interface, text

Description automatically generated



Chart, box and whisker chart

Description automatically generated

Standard deviation

* Quantifies variation or dispersion from the mean
* High values = data is more spread out
* Low values = data is close to the mean
* Graphical user interface, text, application, email

  Description automatically generated
* Graphical user interface, text, application, email

  Description automatically generated

Chart, histogram

Description automatically generated

z-score normalization

* Represent data values in terms of standard deviations from the
* Useful for comparing variables measure on different scales
* Graphical user interface, text, application

  Description automatically generated

Chart, scatter chart

Description automatically generated

Quantile plot

* f quantile of a distribution is a number, q, where approximately a fraction f of the values of the distribution is less than or equal to q
* Text

  Description automatically generated

Graphical user interface, text, application

Description automatically generated

Shape

* Skewness
  + Quantifies the asymmetry of a distribution
* Kurtosis
  + Quantifies the “tailedness”
  + Shape of a distribution’s tails in relation to its overall shape
* Code in Topic 4 – Statistics pg 26

Univariate vs multivariate analysis

* Univariate analysis (one variable)
  + Descriptive statistics of a single variable
* Multivariate analysis (two or more variables)
  + Descriptive statistics of a single variable in relation to categorical variables
  + Relationships between numerical variables
    - Correlation

Cross tabulation in topic 4 – variables pg 1

Grouped bar chart in topic 4 – variables pg 3

Stacked bar chart in topic 4 – variables pg 4

Heat map in in topic 4 – variables pg 5

Charity example in topic 4 – variables pg 7

Diverging stacked bar chart in topic 4 – variables pg 11

Cat plot in topic 4 – variables pg 12

Box plot in topic 4 – variables pg 13

Correlation

* Correlation is not causation
* Correlation coefficient
  + Measure of how two variables change in a similar way
  + Relationship is generally assumed to be linear
  + Quantified on a scale -1 < r < 1
    - r=1 => x and y are positively correlated
    - r=0 => x and y are not correlated
    - r=-1 => x and y are negatively correlated
* Examples and code in topic 4 – variables pg 17

Seaborn scatterplots in topic 4 – variables pg 20

Regression estimate confidence bands

* Conventional confidence interval is 95%
* Confidence bands are 95% sure to contain the best-fit regression line

Scatterplot matrix in topic 4 – variables pg 21

Anscombe’s quartet in topic 4 – variables pg 23

**Machine learning**

Diagram

Description automatically generated

Scikit-learn is used for machine learning

To evaluate a machine learning model are model validation and domain expertise

What is machine learning?

* A subfield of AI
  + Building models from data
    - Parameters ‘learnt’ from samples
    - Predict, explore, understand
* Diagram, venn diagram

  Description automatically generated

Machine learning approaches

* Supervised learning from data to labels
  + Classification
    - Goal: predict discrete labels
    - Find a line that separates the classes
      * Learn the parameters of the line
    - Generalize to:
      * New, unlabeled data
        + E.g. predicting topic labels for news stories
      * Higher dimensions
    - Example in topic 5 – machine learning pg 4
  + Regression
    - Goal: predict continuous labels
    - Model the relationship between variables
      * Learn the parameters from examples
    - Generalize to:
      * New, unlabeled data
      * Higher dimensions eg. Feature1=height, feature2=weight
    - Example in topic 5 – machine learning pg 5
* Unsupervised learning finding patterns in data
  + Clustering
    - Goal: infer labels on unlabeled data
    - Use intrinsic structure to find groups
      * Determine which points are related
    - Generalize to higher dimensions
      * kMeans algorithm
    - Example in topic 5 – machine learning pg 5
  + Dimensionality reduction
    - Goal: infer structure of unlabeled data
    - Transform data from high-dimensional space to low-dimensional space to low-dimensional space
      * Preserve meaningful properties of the data
    - Useful complex data sets
      * 1000s of features
      * Visualize using 2 or 3 dimensions
    - Example in topic 5 – machine learning pg 6

Introduction to scikit-learn

* Efficient versions of common ML algorithms
* Uniform API+documentation
* Easy to switch algorithms/models

Data representation in scikit-learn

* Data as tables: two-dimensional grid
  + Rows are instances (samples)
  + Columns are attributes (features)
* Features matrix: X
  + 2 dimensional: n\_samples \* n\_features
* Target array: y
  + One dimensional: n\_samples

Table

Description automatically generated with medium confidence

Scikit-learn estimator API

* Design principles
  + Consistency
  + Composition
  + Sensible defaults
* Overall process
  + Choose a class of model
  + Set model hyperparameters
  + Configure your data(X and y)
  + Fit the model to your data
  + Apply model to new (unseen) data

A target array usually have 1 dimensions

Features matrix

* Rows are used to represent samples
* Columns are used to represent features

Last step when using scikit-learn estimator API is to apply the model to unseen data

Steps to follow

1. Choose a class of model
   1. Represented by a Python class
   2. E.g. LinearRegression
2. Set model hyperparameters
   1. Instantiate and configure your model
3. Configure your data (x and y)
   1. Arrange into feature matrix and target vector
4. Fit the model to your data
   1. Calculates the slope and the intercept
5. Apply model to new(unseen) data
   1. Create a grid of x values
   2. Visualize the results
   3. Plot raw data + model
   4. Evaluate

Split your data into a test set and a training set to evaluate a model

Supervised learning: Iris classification

* Goal: classify samples based on measurements
  + Using simple classifier: Naïve Bayes
  + Fast, minimal configuration
* Need to evaluate model on unseen data
  + Configure the data (x and y)
  + Fit the model
  + Predict on test set
  + Evaluate in terms of accuracy

Naïve Bayes is a good choice for baseline classification because it has no hyperparameters and is relatively fast

Unsupervised learning: dimensionality reduction

* Type of unsupervised learning
* Iris data set has 4 columns
  + Can you visualize 4 dimensions?
  + Try to reduce the number of dimensions, but retain essential features
* Use principal components analysis
  + 4 dimensions -> 2 dimensions
* Steps to follow
  + Choose a class of model
  + Instantiate the model with hyperparameters
  + Fit the model to your data
  + Transform the data to 2 dimensions
* Example in topic-5 Machine learning pg 16

Unsupervised learning: clustering

* Unsupervised learning
* No labels
* Trying to find structure in data e.g. meaningful groups
* Various techniques
  + kMeans
  + Gaussian Mixture Models
* Steps to follow
  + Choose a class of model
  + Set model hyperparameters
  + Fit the model to your data
  + Determine the clusters(and labels)
* Topic 5 – machine learning pg 18

How do we know when our model is good enough? We need to validate our choice model and hyperparameters

Testing a k-Nearest Neighbor classifier using your training data, it will give 100% accuracy score

Model validation: using a holdout set

* Hold back some of the data for testing
* So that the model has not ‘seen’ it
* Better estimate of model’s performance
* A portion of data reserved for evaluating model performance
* But..
  + We are ‘wasting’ part of our data(50%)
  + A better solution: cross validation
    - Perform multiple splits
    - Rotate the training and test portions
    - Calculate the mean performance

Model validation -> LeaveOneOut and Cross-validation

Validation curves

* Performance is invariably better on training data
* High bias models are poor predictors for both training and test
* High variance models overfit to the training data
* Optimum may be somewhere in between
* A picture containing funnel chart

  Description automatically generated

Chart, surface chart

Description automatically generated

How do we improve our performance?

* Some options:
  + A more complex model
  + A less complex model
  + More training data
  + More/different features
* Not always obvious which approach to adopt!

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

High bias models

* Underfit the data
* Perform similarly on training data and validation data

High variance models

* Overfitting the data
* Validation data far worse than on training data
* Tend to learn ‘noise’ in the training data

But what about data?

* Optimal model complexity will depend on the size of your training data
* Validation curves depend on both
  + Model complexity
  + Size of training data
* How should model behavior change as a function of training data size?
  + Training scores should always be higher than validation scores
  + A model of given complexity will overfit a relatively small data set
  + A model of given complexity will underfit a relatively large data set
* A picture containing chart

  Description automatically generated

Degree of polynomial to use in linear regression depends on the data and the context

A high complexity model exhibit higher variance than low complexity models

How can we improve the performance of our models?

* We can analyze validation curves to find the right level of complexity
* We can analyze learning curve to explore the effect of training set size

Grid search

* Find an optimum for our model performance
* Can ask 3D grid to explore
  + Polynomial degree(int)
  + Fit to the intercept (True/False)
  + Whether to normalize (True/False)

Increasing your training set size would expect the optimum degree polynomial to increase

Feature engineering/selection

* Dealing with non-numerical data
  + Categorical data
  + Textual data
  + Image data
* Creating derived features
* Dealing with noisy data
  + Missing values

Simple numerical mapping from categories to digits is not a good approach because it implies an order that may not exist

We can vectorize categorical values to transform them to numerical data

* What about text?
  + Social media
  + Documents
  + Transcripts
* How should we encode textual data?

Main drawback using raw counts produced by scikit-learn’s CountVectorizer is they can over-emphasize frequent terms

Graphical user interface, text, application

Description automatically generated

Feature selection

* Sometimes the choice is not so simple
* Example: building a spam classifier
  + Does the email contain the word ‘Viagra’ or ‘Lottery’
  + Frequency of certain characters, e.g. ! Or $
  + Email address of the sender
* How do we choose?
  + Experience
  + Domain expertise
  + Experimentation
* Sometimes features can interact in unexpected ways
  + E.g. house prices
    - Location
    - Size
* Consider combined features or mathematically derived features

Good strategies for feature selection is by using domain expertise and experimentation

Effect of adding polynomial features is transform the input space

Dealing with missing data

* Real world data is rarely clean or homogeneous
* Missing data conventions
  + Use a mask e.g. Boolean flag
  + Use a sentinel value, e.g. -9999
* Pandas uses:
  + None (a Python object)
  + NaN (a float64)

Dealing with null values

* Detection
  + isnull()
  + notnull()
* Returns a Boolean mask over the data
* Drop null values
  + dropna()
* Removes NA values (how + thresh params control # of nulls to allow)
* Fill null values
  + fillna()
* Replaces NA values (e.g. with single value, previous value, etc..)

Common conventions for dealing with null values are using a mask and using a sentinel value

Valid reasons for using a pipline

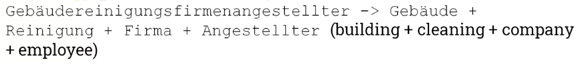
* Reduce the likelihood of bugs
* Make code easier to productionize
* Keep track of your training and validation data at each step

Sources of unstructured data

* Documents
* Emails
* Social media
* Customer feedback
* Web pages
* Open-ended survey responses
* Audio and video

Processing text data

* Polysemy
  + One word maps to many concepts e.g. bat
* Synonymy
  + One concept maps to many words e.g. smart/bright/clever
* Word order
* Language is generative
* Many different ways to express a given idea
  + Paraphrase, metaphor, idoms, etc.
* Language is changing
* Ill-formed input
* Co-ordination, negation, etc...
* Multi-linguality
* Sarcasm, irony, slang, jargon, etc..
* Language is ambiguous
  + To determine structure, we must resolve ambiguity
* Lexical analysis(tokenization)
* Stop word removal
  + No definitive list
* Stemming
  + E.g. fishing, fished, fish, fisher -> fish
  + Argue, argued, argues, arguing -> argu
* Lemmatization
  + Linguistically principled analysis
  + Text, letter

    Description automatically generated
* Morphology (prefixes, suffixes, etc..)
  + 

Graphical user interface, text, application, email

Description automatically generated

Preprocessing text data

* Syntax (part of speech tagging)
  + Noun, pronoun, adjective, determiner, verb, adverb, preposition, conjunction and interjection
    - Book -> Noun, verb
    - That -> Determiner
    - Flight -> Noun
* Ambiguity problem
* Parsing (grammar)
* Sentence boundary detection

The ambiguity in the sentence ‘Fruit flies like a banana’ is primarily at syntactic level

1. Syntactic level: The syntactic level refers to the study of the structure of sentences and phrases, including their word order, grammatical relations, and overall organization. It deals with the way in which words are combined to form meaningful sentences and how those sentences are structured in terms of clauses, phrases, and individual words.
2. Lexical level: The lexical level refers to the study of the meanings and relationships of individual words and their different forms. It deals with the way in which words are organized in a language and how they are used to convey meaning, including their various forms, such as different tenses, plurals, and inflections.
3. Morphological level: The morphological level refers to the study of the structure of words and the rules for creating new words from existing ones. It deals with the way in which words are formed, including their roots, prefixes, suffixes, and other morphemes, and how these elements combine to create new words with different meanings.

Syntactic parsing uses grammar rules

Sentence segmentation is a form of tokenization

How should an email address be tokenized? Depend on the purpose of your text processing application

Stemming and lemmatization are both forms of text normalization

Lemmatization differ from stemming

* Lemmatization is informed by the linguistic context
* Stemming is a more crude, heuristic process

Regular expression matching ‘Set’ and ‘set’

* [Ss]et
* [S-s]et
* Set|set

Regular expression match both ‘123’ and ‘321’

* [0-9][0-9][0-9]
* 123|321
* \d+

Regular expression would match any characters that are not digit

* [^0-9]

Regular expression match both ‘colour’ and ‘color’

* colou?r
* colo.\*r

Regular expression would match any non-word characters at the end of a string

* \W$

There is no universal list of stop words in English

NLP(Natural Learning Processing)

* Language can be spoken as well as written
* Text analytics ~= Natural Language Processing ~= Text Mining(?)
* Chart, line chart

  Description automatically generated

NLP perspectives

* Computer/data science
  + Theoretical foundations of computation and practical techniques for implementation
* Information science
  + Analysis, classification, manipulation, retrieval, and dissemination of information
* Computational Linguistics
  + Use of computational techniques to study linguistic phenomena
* Cognitive science
  + Study of human information processing (perception, language, reasoning, etc..)

NLP paradigms

* Statistical approaches
  + Distributional & neural approaches, supervised or unsupervised
  + Data-intensive
* Symbolic approaches
  + Rule-based, hand coded (by linguists/SMEs)
  + Knowledge-intensive

Examples of early NLP systems -> SHRDLU, ELIZA

NLP applications

* Text categorization
  + Media monitoring
    - Classify incoming news stories
  + Search engines
    - Classify query intent, e.g. search for ‘LOG313’
  + Spam detection
* Machine translation
  + Fully automatic
    - Google translate
  + Semi-automated
    - Helping human translators
* Text summarisation
  + Summarisation
    - Single-document vs multi-document
  + Search results
  + Word processing
  + Research/analysis tools
* Dialog systems
  + Chatbots
  + Smart speakers
  + Smartphone assistants
  + Call handling systems
    - Travel
    - Hospitality
    - Banking
* Sentiment Analysis
  + Identify and extract subjective information
  + Several sub-tasks
    - Subjectivity/objectivity identification
      * E.g. “fact” from opinion
    - Identify polarity, e.g. of movie reviews
      * E.g. positive, negative, or neutral
    - Identify emotional states
      * Angry, sad, happy, etc..
    - Feature/aspect-based
      * Differentiate between specific features or aspects of entities
* Text mining
  + Analogy with Data Mining
    - Discover or infer new knowledge from unstructured text resources
  + A <-> B and B <-> C
    - Infer A <-> C
    - E.g. link between migraine headaches and magnesium deficiency
  + Applications in life sciences, media/publishing, counter terrorism and competitive intelligence
* Question answering
  + Going beyond the document retrieval paradigm
    - Provide specific answers to specific questions
* Natural language generation
* Speech recognition and synthesis

Applications of NLP

* Question answering
* Sentiment analysis

NLP techniques use to infer new knowledge from unstructured text is **text mining**

Higher lexical diversity calculation -> word types/tokens

Plotting distribution of word frequencies for one of the texts in nltk.book, we would expect to see **a curve**

Sequence of words that occur together unusually are referred as **collocations**

Collection of text samples used as research data is known as **a text corpus**

Data structure use to explore the effect of compounding across multiple languages is **A conditional frequency distribution**

Bigrams -> “the cat sat on the mat”

* The cat
* Cat sat
* Sat on
* On the
* The mat

Lexical semantics: the meaning of words

* NLP tasks
  + Classification tasks (e.g. spam detection)
  + Sequence tasks (e.g. text generation)
  + Meaning tasks
* Useful for:
  + Information retrieval
  + Question answering
  + Topic modelling
* What do words mean?
  + Look in a dictionary

Lexical semantics: WordNet

* A lexical database:
  + Nodes are synsets
  + Correspond to abstract concepts
  + Polyhierarchical structure
* Using WordNet we can programmatically:
  + Identify hyponyms and hypernyms
  + Measure semantic similarity

WordNet is known as **Synsets**

Polyhierarchical structure is one that **allows multiple parents**

WordNet synsets known as **Lemmas**

Ambiguous words are likely to have **more synsets than non-ambiguous words**

Parent terms in WordNet known as **Hypernyms**

Lowest\_common\_hypernyms() in WordNet used **to locate the closest parent term that is shared by two given words.**

WordNet similarity measures is the best one to use as **it depends on the task**

Some of the shortcomings of traditional ‘one-hot’ vector encodings

* Tend to be very long
* Tend to be relatively sparse

Benefits of using word embeddings compared to traditional vector representations

* They are more compact
* Offer better generalization capabilities

Graphical user interface, text, application, email

Description automatically generated

Information extraction

* Finding structure in unstructured data
  + Identifying entities and relationships
* Structured data:
  + Pandas dataframes
  + SQL
* Applications:
  + Business Intelligence
  + Media analysis
  + Life sciences
  + Etc…
* Typical pipeline
  + Split raw text into sentences
  + Split sentences into tokens
  + Tag each token with POS tags
  + Identify interesting entities
  + Find relations between entities
* Entities can be
  + People, places, organizations, etc..
  + Temporal/numerical expressions

Graphical user interface, text, application, email

Description automatically generated

Typical information extraction pipeline, order of processes -> **Tokenisation, POS tagging, entity recognition**

POS tags represent **lexical categories**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Corpora is marked up with a variety of named entities called IEER

Computing simple statistics

* Exploring sample texts
  + From nltk
* Counting vocabulary
  + Word types and word tokens
* Frequency distributions
  + Power laws in language data
* Collocations
  + Words that stick together

Conditional frequency distributions

* Text corpora
  + The brown corpus, 500 sources categorized by genre
* Conditional frequency distributions
  + Frequency distributions conditioned on categories
  + Evidence of compounding?
* Generating fake news
  + Using bigrams to produce random text

A graph can have **0 edges** or **infinitely many edges and nodes**

**Planar graph** is a type of graph never intersect

**Undirected graph** is represented with a symmetric matrix

**Tree graph** is acyclic

Programming language support

* Many programming languages provide the necessary functionality to support graphs, in most cases as data structures. Those include C++, Java and others
* In python, the graph functionality is provided by the library NetworkX
  + Import the NetworkX module:
    - Import network as nx
  + Create an object(empty graph):
    - my\_graph = nx.Graph()
    - From this point onwards, we can start adding nodes and connect them with edges
  + Nodes and edges
    - Following lines add two nodes named A and BL
      * my\_graph.add\_node(“A”)) my\_graph.add\_node(“B”))
    - Following line connects nodes A and B with an edge:
      * my\_graph.add\_edge(‘A’, ‘B’)
    - Result is an undirected graph with two connected nodes
  + NetworkX allows us to create various types of graphs
* Allows us to process graphs as data structures and to visualize them as diagrams
* With this library we can:
  + Build and visualize networks
  + Analyse their properties
  + Utilise graph algorithms

Types of graphs

* Undirected graph
  + All edges are bi-directional(symmetrical)
  + Edges represent a two-way relation
  + Edges can be traversed in both directions
  + E.g. facebook
  + Topic 8 – Graphs and Networks pg 5
* Directed graph
  + All edges have directions
  + Usually represented with arrows
  + Edges can be represented as ordered pairs (A,B)
  + Traversing the edges in the directed graph is restricted to one direction only
  + E.g. Twitter, X follows Y, but Y does not follow X
  + Topic 8 – Graphs and Networks pg 6
* Complete graph
  + In this type of graph, every possible pair of nodes is connected to a unique edge
  + As a data structure, it is not different from an undirected graph
  + Except all possible edges are connected
  + Complete graph with N nodes has N\*(N-1)/2 edges
* Weighted graph
  + For this type of graph, a value is assigned to every edge
  + Value might represent:
    - Distance
    - Time
    - Cost
    - Any numerical value which characterises the transition from one node to another
  + Topic 8 – Graphs and Networks pg 8
* Tree
  + In this case, the graph has a designated root node and every edge points towards the root node (in-tree) or away from the root node (out-tree)
* Directed Acyclic Graphs (DAG)
  + These are directed graphs without cycles. All trees are DAGs but not all DAGs are trees

Graph attributes

* We use attributes to add properties to the nodes and edges
  + These properties help to represent, or to model, a real- world network
* Examples
  + Weight of edges (distance, time, cost, others)
  + Colour of nodes/edges
  + Co-ordinates of nodes
* Attributes are stored as dictionaries, where:
  + Key is the node name, or the (ordered) pair of the edge
  + Value is the attribute
* Attributes can be specified when adding new node/edge:
  + my\_graph.add\_node(‘A’, A1=1)
  + Or at a later stage
    - nx.set\_node\_attributes(my\_graph, values={‘A’:100, ‘B’: 35, ‘C’:2, ‘D’:4}, name=’A1’)
  + Note the syntax for working with dictionaries
* If needed, attributes can be extracted from the graph data structure and stored in a dictionary:
  + node\_dictionary = nx.get\_node\_attributes(MyG, ‘A1’)
* And if necessary in a list:
  + node\_list = node\_dictionary.values()
* Printing the contents of the dictionary and the list highlights how these two data structures organize and store data

Isomorphic graphs

* Nodes in a typical graph in most cases do not have information about their co-ordinates
* Even if this information exists, it is usually “abstracted out”
* As a result, a graph can be visualized differently, based on the position of its nodes
* These different “versions”, or layouts, of the same graph are called Isomorphic Graphs
* In some cases in practice there is a requirement for the edges not to cross each other
  + For example designing a single layer circuit board
* Those type of graphs are called planar
* It is possible to instruct NetworkX to follow this requirement:
  + my\_graph = nx.Graph()
  + position = nx.planar\_layout(my\_graph)
* In the new layout edges will not intersect
* Original graph and its Planar version are both isomorphic graphs
* It is not always possible to obtain a Planar Graph

Chart, line chart

Description automatically generated

Graph layouts

* A graph can have multiple layouts
  + All layouts retain the same connectivity
* These layouts can be based on
  + Aesthetics
  + Need to emphasize connectivity of specific nodes
  + Need to emphasize specific connections
* NetworkX library provides a number of different layouts to choose from Topic 8 -Graphs and Networks pg 12

Two graphs are isomorphic if **they have the same nodes and the same connectivity**.

Python programming and NetworkX library provide support for graphs as **Both data structure and diagram**

NetworkX library uses **Class** data structure to represent graphs

Main operations

* Traversing a network
  + Visiting all network nodes once
* Finding optimal (shortest) path
  + Using algorithms

Traversing a network

* Graphs (and networks) are non-linear data structures
  + Unlike arrays and lists
* As a result, traversing them is not a trivial task
* Also, there are more than one way to conduct the traversal
* Most popular traversal algorithms are
  + Breadth-first Search (BFS)
    - Visit all neighbouring nodes connected to the root node, then select one of the nodes already visited and visit all nodes connected to it, repeat until all nodes are visited
  + Depth-first Search (DFS)
    - Select one node connected to the ‘root’ node and then visit all nodes in that direction until the last node at the lowest level is reached. Select the next node connected to the root node and repeat the steps
* Both BFS and DFS treat the nodes as organised in levels
* Highest level being a ‘root’ node, which acts as a starting point for the algorithms
* Both the BFS and DFS algorithms repeat their steps until all nodes are visited
* To avoid visiting a node more than once, visited node are marked
  + Is crucial when working with cyclic graphs
* Result of both BFS and DFS algorithms is a sequence of nodes arranged in the order they are visited
* Sequences can be graphically presented
* Topic 8 – Graphs and Networks pg 17

Finding the optimal (shortest) path

* Finding the shortest, or most optimal, path in a graph or network has applications in many areas in practice (Google Maps)
* Several algorithms exist, with Dijkstra algorithm being one of the most popular
* The optimal path between two nodes can be based on the number of nodes, or the weights assigned to the edges
* Real world example Topic 8 – Graphs and Networks pg 19 onwards

Challenges in network visualisation

* Large-scale networks visualised in a limited space
  + Examples include: population networks, social media, biological networks, chemical structures and others
* Complex networks (with multiple attributes)
  + Need to ‘abstract out’ unnecessary data
  + If not ‘removed’ , unnecessary details may obscure important information
* Nodes and edges with different levels of importance
  + Need to emphasise the important elements, which help to covey the main message, for example, most important/connected node

Traversal algorithms can identify shortest path between two nodes based on **Number of the intermediate nodes** and **Sum of the weights of the intermediate edges**

In context of graphs, a clique is **A subgraph where every two nodes are connected with a distinct edge**

In the context of developing graphs and networks, abstraction involves **Avoiding items which are not relevant and focusing on those which are important**

Dimensions in structured data

* Multidimensional data combine data points from a wide range of data types organized in a single entity, called dataset
* Multidimensional datasets are expected to be ‘information rich’, but this might not always be true
* In most cases, datasets with multiple dimensions need pre-processing before visualizing
* Multidimensional datasets are not necessarily large, although this is usually the case
* Structured data (datasets) are organized into rows and columns
* A row represents the characteristics of a single object
* A column represent a single characteristic of the objects
* The dimensions of the datasets are represented by the number of columns
* Multidimensional dataset may contain different types of data
* Example in Topic 8 – visualizing multidimensional data

Heat maps

* A graphical representation of multidimensional data
* Use two-dimensional plane for visualization
* Usually used for summarised information
* Data is presented colour-coded
* Rely solely on colour scheme to distinguish different data points or groups of data points
* Choosing correct colour scheme is critical for accurate and informative visualisation
* Common case for using heatmap is for ‘mapping’ numerical data on geographical map
* Dataset needs to be accompanied with information about geographical location, for example longitude and latitude or other coordinate system used by the map
* Usually used to indicate areas of concern
  + For example: Air pollution, Extreme temperatures, property prices, others…
* Example in topic 8 – visualising multidimensional data pg 3
* Advantages
  + Multidimensional numerical data can be visually communicated instantly
  + Rarely needs pre-processing of the datasets
  + Can integrate into geographical maps
* Disadvantages
  + Not able to visualize different types of data in a single diagram, for example **ratios and intervals**
  + Over-rely on colour-codes to distinguish between data points or groups of data points
* Example in topic 8 – visualising multidimensional data pg 10

Setting a threshold value for a heat map, it needs to be **any values within the range of the data points**

Multidimensional datasets are not necessarily large

A dataset with multiple dimensions can store values of different data types

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Structure of the diagram

* Parallel coordinates present data in a specific style diagram
* The diagram consists of:
  + A number of parallel lines (axis), usually vertically oriented
    - The number of lines correspond to the dimension of the dataset
  + Multiple ‘polylines’, which cross each axis at points corresponding to the actual data points from the dataset
* Can represent both positive and negative values
* Are there diagrams which cannot meaningfully represent negative values? (pie chart, radar?)
* Multidimensional datasets might contain widely different data types
* In the case of numerical values(ratios), the ranges of the data points can vary significantly
* Visualising dimensions with different ranges requires the data points to be normalised to a certain scale
* The scale should be suitable for all dimensions from the dataset

Random values => Topic 8 – Visualising multidimensional data pg 14

Iris dataset => Visualising multidimensional data pg 18

Vehicle dataset => Visualising multidimensional data pg 19

Diagrams for both positive and negative numbers on same plot

* Heat map
* Parallel coordinates
* Bar chart

Diagrams that cannot display negative value

* Pie chart

In the context of parallel coordinates a poly-line represents **the actual values of the data points**

Axis in a parallel coordinates diagram is also known as **Spine**

**The plotting area of the diagram, since dimensions are part of the diagram structure** is limiting the number of dimensions a parallel coordinates diagram can visualize

Graphical user interface, text, application

Description automatically generated

In Bayes theorem the baseline likelihood of a hypothesis is know as **the Prior**

If the likelihood of an outcome was increased by Evidence(E) how would this affect the Prior(H) – **It would remain unchanged**

Text

Description automatically generated

Table

Description automatically generated with medium confidence

Bayes Theorem for classification

* We need to calculate P(features | Li) for each label
* Assume a generative model for each label -> features
  + Specifying this model is the main part of training in Bayesian classifier
* Apply some simplifying (‘naïve’) assumptions, e.g.
  + Independence
  + Equality

Naive Bayes commonly used as a baseline for a classification problem because

* **It has few tunable parameters**
* **It is well-suited to high dimensional datasets**

In Bayes Theorem the Posterior probability is the product of **The Prior and the likelihood**

Assumptions naïve bayes typically rely on

* **Features are independent**
* **Features are equally weighted**

Multinomial Naïve Bayes

* Features are generated from a multinomial distribution
  + Probability of observing counts across various categories
  + Find a best-fit multinomial distribution
* Example: text classification
  + Categories are word types
* Convert from strings to numbers
  + Feature engineering: TfidfVectorizer() -> convert strings to numbers

We use a multinomial distribution to represent text, the categories corresponded to is **word types**

Confusion matrix typically used to show **the overlap between true labels and predicted labels**

Simple linear regression

* Good starting point for regression tasks
  + Can be fit very **quickly**
  + Very **interpretable**
* Commonly used to fit a **straight line** to 2D data, e.g. y=ax+b
  + Can be extended to accommodate more **complex patterns**

If the data was 3 dimensional, the linear model will represent **a plane** form

Basis function regression

* Can we use linear regression to model nonlinear relationship?
  + Transform the data using basis function
    - E.g. polynomial features
* Text, letter

  Description automatically generated
* Is still a linear model
  + Coefficients are linearly independent and uncorrelated Projected one-dimensional x values into higher dimension
  + Linear fit can accommodate more complex relationships
* Polynomial basis functions:
  + PolynomialFeatures()
* Gaussian basis functions:
  + GaussianFeatures()

In polynomial regression, **exponentiation** is applied to x

In gaussian basis function regression, the model represent **a sum of Gaussian bases**

Regularization

* Introduction of basis functions makes model more flexible
  + But can lead to overfitting
  + E.g. 30 polynomial features
* Plot the **coefficients** of the Gaussian bases against their **locations**
  + Amplitude **oscillates** where basis functions overlap
* Need to **limit** such **spikes**
  + **Penalize large value** of the model parameters
* A picture containing diagram

  Description automatically generated

Purpose of regularization is **to penalize high coefficients**

To control the degree of regularization is to **use a parameter to control the strength of the penalty**

Support Vector Machines

* A type of discriminative classifier:
  + Rather than modelling each class, we find a line to separate the classes
  + Can be used for classification and regression

Maximising the margin

* Rather than simply drawing a line
  + Think in terms of maximizing the margin between the classes
* Some separation lines have a wider margin than others

Key idea behind support vector machine is maximising **margin**

**Discriminative** classifier is a support vector machine

**Support vectors** are data points that touch the margin

Kernel SVMs

* SVMs become more powerful when combined with kernels:
  + Project data into a higher dimensional space
* No linear discrimination can separate the classes but:
  + Recall basis function regression: polynomial + Gaussian basis functions to model non-linear relationships with linear classifier
* Could we project this data into a higher dimension
  + E.g. a radial basis function
* How do we choose the right projection?
  + Compute a basis function at every point, let SVM sift through results? Computationally intensive, N points \* N dimensions
* Kernel trick: allow us to fit on transformed data without computing coordinates in higher dimensional space

**A higher dimensional space**  is the key idea behind kernel SVMs when projecting our data

Increasing the C parameter in kernel SVMs will **harden the margin**

Decision Trees

* Another type of discriminative classifier:
  + Ask questions to ‘zero in’ on a class
  + A bit like “20 questions(board game)”
* Many benefits:
  + Easy to understand and interpret
  + The classification process is transparent
  + Handles numeric & categorical data (and missing data)
  + Can be used for both classifications and regression
* Drawbacks:
  + How do you decide which questions to ask?
  + Easy to overfit
  + Hard to find an optimal decision tree
* The process
  + Try partitioning the data by each of the attributes
  + Choose the partition with the lowest entropy
  + Add a decision node for that attribute and repeat on each subset
  + Lowest entropy => lowest uncertainty

Random forests

* Using two decision trees is better than one
  + How about many decision trees?
* Ensemble methods
  + Bagging: use multiple parallel estimators to return an average result
  + An ensemble of decision trees is called a random forest
* Scikit-learn offers a BaggingClassifier and a RandomForestClassifier

**Classification** and **Regression** can be used by random forest

k-Means Clustering

* Seeks to learn an optimal division or labelling of a data set
  + K-Means is one such popular algorithm
  + Searches for a predetermined number of clusters within an unlabelled dataset
* An optimal clustering is one where
  + A cluster centre is arithmetic mean of all its members
  + Each point is close to its own cluster centre than to any others
* Is a unsupervised learning

Optimal clustering

* Each point is closer to its own cluster centre than to other cluster centres
* A cluster centre is the arithmetic mean of all its members

Expectation-Maximization

* Finding an optimal division of a data set is difficult
  + Number of possible combinations is exponential in number of points
* EM algorithm:
  + Assign some random cluster centres
  + Repeat until converged:
    - E-step: Assign points to nearest cluster centre
    - M-step: Set the cluster centres to the mean
  + Each repetition will result in a better estimate

Extending kMeans to deal with non-linear boundaries through **project the data into a higher dimensional space**