

Data Science from scratch

Table of Contents

I.	Fundamentals.....	3
1.	Matrices & Linear Algebra Fundamentals.....	3
2.	Hash Functions, Binary Trees	3
3.	Relational Algebra and DB basics.....	4
4.	Inner, Outer, Cross and Theta Join.....	4
5.	CAP Theorem	4
6.	Sharding.....	5
7.	OLAP.....	5
8.	Multidimensional Data Model.....	6
9.	ETL.....	7
10.	JSON & XML	7
11.	NoSQL.....	8
12.	Regex (Regular Expressions)	9
13.	Virtual Environments Setup (ISN'T COMPLETED).....	9
II.	Programming (Python and R).....	10
1.	Install Packages.....	10
2.	Factor Analysis	10
3.	Functions	11
4.	Read Data.	12
5.	Manipulate Data Frames	13
6.	Subsetting Data Frames.....	14
III.	Statistics (and how to calculate them in Python)	14
1.	Descriptive Statistics.....	14
2.	Histograms.....	16
3.	Percentiles and Outliers.....	17
4.	Probability Theory	17
5.	Bayes Theorem	19
6.	Random Variables.....	22

7.	Probability Distribution Function	23
8.	Cumulative Distribution Function	24
9.	Central Limit Theorem.....	25
10.	Continuous Distributions.....	25
11.	Skewness.....	26
12.	Monte Carlo Method	27
13.	Hypothesis Testing.....	28
14.	Normality Tests.....	30
15.	Correlation Tests	31
16.	Stationary Tests.....	33
17.	Parametric Statistical Hypothesis Tests	34
18.	Nonparametric Statistical Hypothesis Tests.....	35
19.	Analysis of Variance (ANOVA).....	38
20.	Estimation	41
21.	Confidence intervals.....	42
22.	Maximum Likelihood Estimation (MLE)	45
23.	Kernel Density Estimate.....	48
24.	Regression.....	49
25.	Covariance and Correlation	50
26.	Causation	50
27.	Distance Measures	51
IV.	Machine Learning	52
1.	What is Machine Learning?	52
2.	Numerical, Categorical and Ordinal Variables	52
3.	Supervised Learning	53
4.	Unsupervised Learning	53
5.	Reinforcement Learning.....	54
6.	Training and Test Data.....	55
7.	Prediction.....	55
8.	Lift.....	56
9.	Bias & Variance.....	56
10.	Overfitting & Underfitting	56

11.	Trees & Classifications	59
12.	Classification Rating	60
13.	Classification Algorithms	62
a.	Logistic Regression with Example (Binary)	62
b.	K-Nearest Neighbor with Example (Binary).....	65
c.	Naïve Bayes Classifier	68
d.	Stochastic Gradient Descent.....	69
e.	Decision Trees	69
f.	Random Forests.....	70
g.	Support Vector Machine	70
14.	Binary Classification	71
	Neuronal Networks	Error! Bookmark not defined.
15.	Multi-Class Classification	72
a.	Quick Example Using Decision Tree Classifier.....	73
16.	Multi-Label Classification.....	73
17.	Imbalanced Classification	74
a.	Synthetic Minority Oversampling Technique (SMOTE) Example.....	74

I. Fundamentals

1. Matrices & Linear Algebra Fundamentals

Linear algebra topics:

- Vectors
- Matrices
- Transpose of a matrix
- Inverse of a matrix
- Determinant of a matrix
- Trace of a matrix
- Covariance Matrix
- Dot product
- Eigenvalues
- Eigenvectors

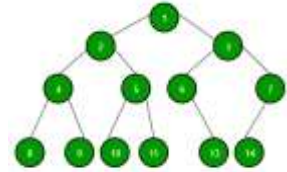
2. Hash Functions, Binary Trees

Hashing is a technique that is used to uniquely identify a specific object from a group of similar objects. Some examples of how hashing is used in our lives include:

- In universities, each student is assigned a unique roll number that can be used to retrieve information about them.
- In libraries, each book is assigned a unique number that can be used to determine information about the book, such as its exact position in the library or the users it has been issued to etc.

In both these examples the students and books were hashed to a unique number.

With binary trees you can do a lot of operations like transversals, summations, construction, conversions, etc.



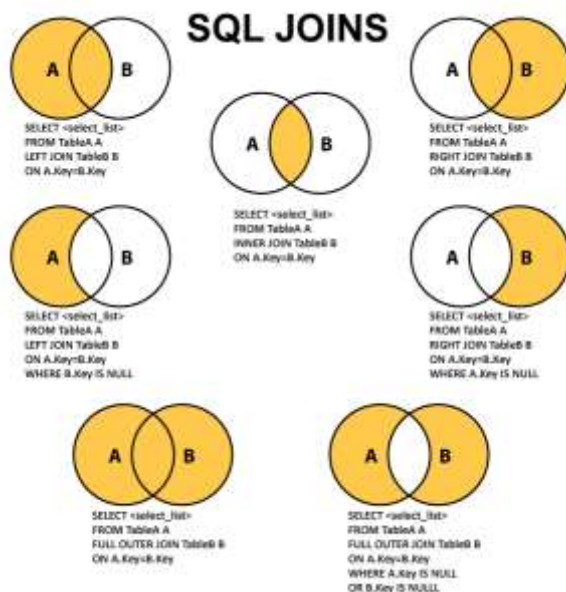
3. Relational Algebra and DB basics

Relational algebra query operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations.

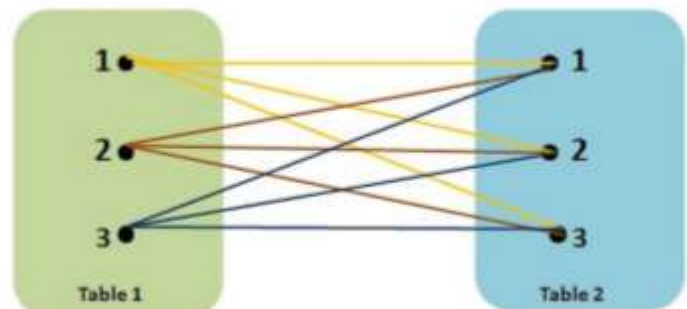
The relational databases are tabular data that are interrelated.



4. Inner, Outer, Cross and Theta Join.



SELECT * FROM Table1 CROSS JOIN Table2;



Example-Theta Join

Beers

name	manf
Beer 1	XYZ
Beer 2	ABC
Beer 3	ABC

Likes

drinker	beer
Aryo	Beer 1
Lamo	Beer 1
Amir	Beer 3

**SELECT * FROM Beers B JOIN
Likes L ON B.name = L.beer**

name	manf	drinker	beer
Beer 1	XYZ	Aryo	Beer 1
Beer 1	XYZ	Lamo	Beer 1
Beer 3	ABC	Amir	Beer 3

5. CAP Theorem

- Consistency
- Availability
- Partition Tolerance

There's a trade off between consistency and availability when the system has partitions. Partitioning is when the system can't communicate each part with each other. If the system never has partitions, you can make the system consistent and available.

In real world you have degrees of consistency and degrees of availability and make trade offs between those two.

6. Sharding

Sharding is when you have an enormous amount of data and you want to access them in the fastest way. This process is by fragmentating the data into small pieces and storing them in different servers to speed up the performance.

One way to access to every piece is by hierarchical sharding. Which means that you cut your piece into another small pieces and so on.

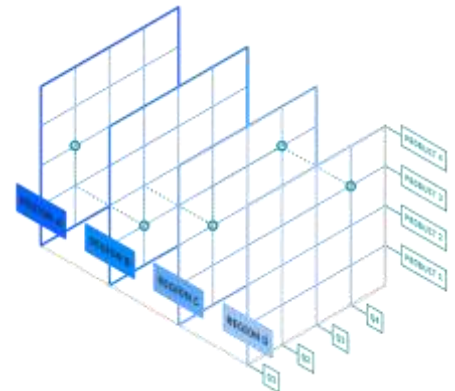
NoSQL uses sharding internally.

7. OLAP

OLAP enables fast, flexible multidimensional data analysis for business intelligence (BI) and decision support applications.

OLAP (for online analytical processing) is software for performing multidimensional analysis at high speeds on large volumes of data from a data warehouse, data mart, or some other unified, centralized data store.

In theory, a cube can contain an infinite number of layers. (An OLAP cube representing more than three dimensions is sometimes called a hypercube.) And smaller cubes can exist within layers—for example, each store layer could contain cubes arranging sales by salesperson and product. In practice, data analysts will create OLAP cubes containing just the layers they need, for optimal analysis and performance.



OLAP cubes enable four basic types of multidimensional data analysis:

Drill-down

The drill-down operation converts less-detailed data into more-detailed data through one of two methods—moving down in the concept hierarchy or adding a new dimension to the cube. For example, if you view sales data for an organization's calendar or fiscal quarter, you can drill-down to see sales for each month, **moving down in the concept hierarchy of the "time" dimension.**

Roll up

Roll up is the opposite of the drill-down function—it aggregates data on an OLAP cube by moving up in the concept hierarchy or by reducing the number of dimensions. For example, you could move up in the concept hierarchy of the "location" dimension by viewing each country's data, rather than each city.

Slice and dice

The slice operation **creates a sub-cube by selecting a single dimension from the main OLAP cube.** For example, you can perform a slice by highlighting all data for the organization's first fiscal or calendar quarter (time dimension).

The dice operation isolates a **sub-cube by selecting several dimensions within the main OLAP cube.** For example, you could perform a dice operation by highlighting all data by an organization's calendar or fiscal quarters (time dimension) and within the U.S. and Canada (location dimension).

Pivot

The pivot function rotates the current cube view to display a new representation of the data—enabling dynamic multidimensional views of data. The OLAP pivot function is comparable to the pivot table feature in spreadsheet software, such as Microsoft Excel, but while pivot tables in Excel can be challenging, OLAP pivots are relatively easier to use (less expertise is required) and have a faster response time and query performance.

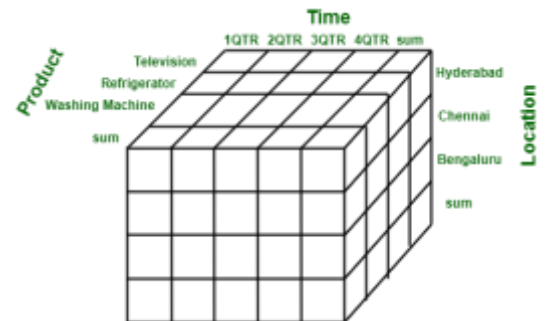
8. Multidimensional Data Model

The multi-Dimensional Data Model is a method which is used for ordering data in the database along with good arrangement and assembling of the contents in the database.

The Multi-Dimensional Data Model allows customers to interrogate analytical questions associated with market or business trends, unlike relational databases which allow customers to access data in the form of queries. They allow users to rapidly receive answers to the requests which they made by creating and examining the data comparatively fast.

OLAP (online analytical processing) and data warehousing uses multi-dimensional databases. It is used to show multiple dimensions of the data to users.

It represents data in the form of data cubes. Data cubes allow to model and view the data from many dimensions and perspectives. It is defined by dimensions and facts and is represented by a fact table. Facts are numerical measures and fact tables contain measures of the related dimensional tables or names of the facts.



The following stages should be followed by every project for building a Multi-Dimensional Data Model:

Stage 1: Assembling data from the client: In first stage, a Multi-Dimensional Data Model collects correct data from the client. Mostly, software professionals provide simplicity to the client about the range of data which can be gained with the selected technology and collect the complete data in detail.

Stage 2: Grouping different segments of the system: In the second stage, the Multi-Dimensional Data Model recognizes and classifies all the data to the respective section they belong to and builds it problem-free to apply step by step.

Stage 3: Noticing the different proportions: In the third stage, it is the basis on which the design of the system is based. In this stage, the main factors are recognized according to the user's point of view. These factors are also known as "Dimensions".

Stage 4: Preparing the actual-time factors and their respective qualities: In the fourth stage, the factors which are recognized in the previous step are used further for identifying the related qualities. These qualities are also known as "attributes" in the database.

Stage 5: Finding the actuality of factors which are listed previously and their qualities: In the fifth stage, A Multi-Dimensional Data Model separates and differentiates the actuality from the factors which are collected by it. These play a significant role in the arrangement of a Multi-Dimensional Data Model.

Stage 6: Building the Schema to place the data, with respect to the information collected from the steps above: In the sixth stage, on the basis of the data, which was collected previously, a Schema is built.

Advantages of Multi-Dimensional Data Model

- It is easy to maintain.

- Its performance is better than that of normal databases (e.g. relational databases).
- The representation of data is better than traditional databases. That is because the multi-dimensional databases are multi-viewed and carry different types of factors.
- It is workable on complex systems and applications, contrary to the simple one-dimensional database systems.
- The compatibility in this type of database is an upliftment for projects having lower bandwidth for maintenance staff.

Disadvantages of Multi-Dimensional Data Model

- The multi-dimensional Data Model is slightly complicated in nature, and it requires professionals to recognize and examine the data in the database.
- During the work of a Multi-Dimensional Data Model, when the system caches, there is a great effect on the working of the system.
- It is complicated in nature due to which the databases are generally dynamic in design.
- The path to achieving the end product is complicated most of the time.
- As the Multi-Dimensional Data Model has complicated systems, databases have many databases due to which the system is very insecure when there is a security break.

9. ETL

ETL is a type of data integration that refers to the three steps (extract, transform, load) used to blend data from multiple sources. **It's often used to build a data warehouse.** During this process, data is taken (extracted) from a source system, converted (transformed) into a format that can be analyzed, and stored (loaded) into a data warehouse or other system. Extract, load, transform (ELT) is an alternate but related approach designed to push processing down to the database for improved performance.

10. JSON & XML

JSON:

- Stands for JavaScript Object Notation
- Is a lightweight format for storing and transporting data
- Is often used when data is sent from a server to a web page
- Is "self-describing" and easy to understand

JSON example:

```
{
  "employees": [
    { "firstName": "John", "lastName": "Doe" },
    { "firstName": "Anna", "lastName": "Smith" },
    { "firstName": "Peter", "lastName": "Jones" }
  ]
}
```

XML:

- stands for eXtensible Markup Language
- Is a markup language much like HTML
- Was designed to store and transport data

- Was designed to be self-descriptive

XML Example:

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

is displayed as →



11. NoSQL

NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications. NoSQL databases are widely recognized for their ease of development, functionality, and performance at scale. These types of databases are optimized specifically for applications that require large data volume, low latency, and flexible data models, which are achieved by relaxing some of the data consistency restrictions of other databases.

In a NoSQL database, a book record is usually stored as a JSON document. For each book, the item, ISBN, Book Title, Edition Number, Author Name, and AuthorID are stored as attributes in a single document. In this model, data is optimized for intuitive development and horizontal scalability.

NoSQL databases are a great fit for many modern applications such as mobile, web, and gaming that require flexible, scalable, high-performance, and highly functional databases to provide great user experiences.

- Flexibility: NoSQL databases generally provide flexible schemas that enable faster and more iterative development. The flexible data model makes **NoSQL databases ideal for semi-structured and unstructured data.**
- Scalability: NoSQL databases are generally designed to scale out by using distributed clusters of hardware instead of scaling up by adding expensive and robust servers. Some cloud providers handle these operations behind-the-scenes as a fully managed service.
- High-performance: NoSQL database are optimized for specific data models and access patterns that enable higher performance than trying to accomplish similar functionality with relational databases.
- Highly functional: NoSQL databases provide highly functional APIs and data types that are purpose built for each of their respective data models.

	Relational databases	NoSQL databases
Optimal workloads	Relational databases are designed for transactional and strongly consistent online transaction processing (OLTP) applications and are good for online analytical processing (OLAP).	NoSQL databases are designed for a number of data-access patterns that include low-latency applications. NoSQL search databases are designed for analytics over semi-structured data.
Data model	The relational model normalizes data into tables that are composed of rows and columns. A schema strictly defines the tables, rows, columns, indexes, relationships between tables, and other database elements. The database enforces the referential integrity in relationships between tables.	NoSQL databases provide a variety of data models such as key-value, document, and graph, which are optimized for performance and scale.
ACID properties	<p>Relational databases provide atomicity, consistency, isolation, and durability (ACID) properties:</p> <ul style="list-style-type: none"> • Atomicity requires a transaction to execute completely or not at all. • Consistency requires that when a transaction has been committed, the data must conform to the database schema. • Isolation requires that concurrent transactions execute separately from each other. • Durability requires the ability to recover from an unexpected system failure or power outage to the last known state. 	NoSQL databases often make tradeoffs by relaxing some of the ACID properties of relational databases for a more flexible data model that can scale horizontally. This makes NoSQL databases an excellent choice for high throughput, low-latency use cases that need to scale horizontally beyond the limitations of a single instance.
Performance	Performance is generally dependent on the disk subsystem. The optimization of queries, indexes, and table structure is often required to achieve peak performance.	Performance is generally a function of the underlying hardware cluster size, network latency, and the calling application.
Scale	Relational databases typically scale up by increasing the compute capabilities of the hardware or scale-out by adding replicas for read-only workloads.	NoSQL databases typically are partitionable because access patterns are able to scale out by using distributed architecture to increase throughput that provides consistent performance at near boundless scale.
APIs	Requests to store and retrieve data are communicated using queries that conform to a structured query language (SQL). These queries are parsed and executed by the relational database.	Object-based APIs allow app developers to easily store and retrieve data structures. Partition keys let apps look up key-value pairs, column sets, or semistructured documents that contain serialized app objects and attributes.

12. Regex (Regular Expressions)

/d: digits /:D non-digits UPPERCASE [A-Z] lowercase [a-z] numbers [0-9]

Combinations

- Upper, lower and numbers = [a-zA-Z0-9]
- Upper and lower=[a-zA-Z]

Python example of regular expressions

There's a lot of regular expressions to study!!!!!!!!!!!!!!!!!!!!!!

```
import re
pattern='[a-zA-Z0-9]+@[a-zA-Z]+\.(com|edu|net)'
user_input=input()
if (re.search(pattern, user_input)):
    print('valid email')
else:
    print('Invalid email')
```



```
C:\Users\kwd\Documents\Regex>python verify_user_email.py
bill@gmail.com
valid email

C:\Users\kwd\Documents\Regex>python verify_user_email.py
bill@gmail.com
Invalid email
```

13. Virtual Environments Setup (ISN'T COMPLETED)

First thing you need to do is execute the venv module, which is part of the Python standard library.

```
PS C:\Users\S6945\Desktop\test> python3 -m venv venv/
c:\n de aplicaciones.
PS C:\Users\S6945\Desktop\test> py -m venv venv/
```

The only thing left to do is to “activate” our environment by running the following scripts

```
PS C:\Users\S6945\Desktop\test> cd venv
PS C:\Users\S6945\Desktop\test\venv> cd bin
```

II. Programming (Python and R)

1. Install Packages

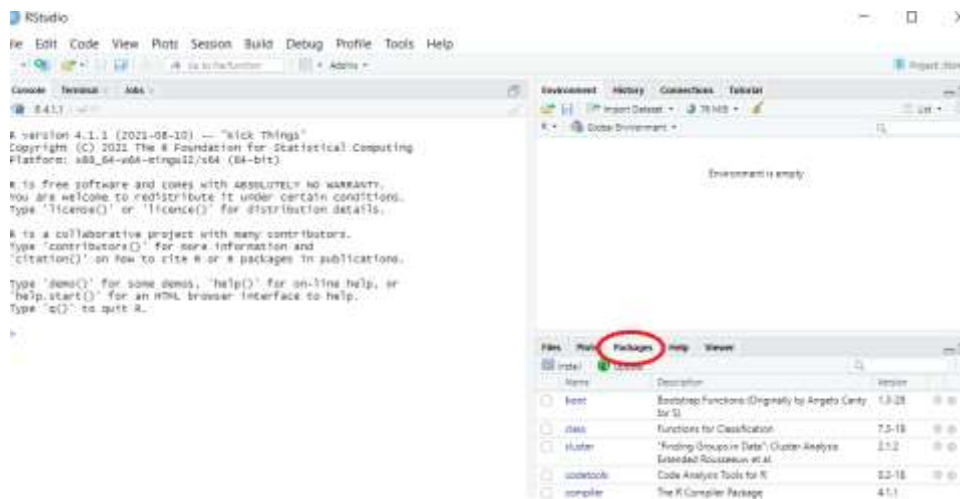
To Install packages in Python you need to open the Windows Terminal or Windows PowerShell and write the following command:

```
pip install package name or py -m pip install package name
```

```
pip install scikit-learn

Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/1f/af/e3c3cd6f610938300591386...
    |#####| 6.6MB 32.5MB/s
Collecting scipy>=0.17.0 (from scikit-learn)
  Downloading https://files.pythonhosted.org/packages/14/49/8f13fa215e10a7ab0731cc9...
    |#####| 25.1MB 35.5MB/s
```

To use packages in Rstudio you must activate them in the Packages window



And if the packages aren't in that window, you have to open Tools → Install Packages and write the package name

2. Factor Analysis

Factor analysis is a technique that is used to reduce a large number of variables into fewer numbers of factors. This technique extracts maximum common variance from all variables and puts them into a common score. As an index of all variables, we can use this score for further analysis. Factor analysis is part of general linear model (GLM) and this method also assumes several assumptions: **there is linear relationship, there is no multicollinearity, it includes relevant variables into analysis, and there is true correlation between variables and factors.** Several methods are available, but principal component analysis (PCA) is used the most.

Assumptions:

No outlier: Assume that there are no outliers in data.

Adequate sample size: The case must be greater than the factor.

No perfect multicollinearity: Factor analysis is an interdependency technique. There should not be perfect multicollinearity between the variables.

Homoscedasticity: Since factor analysis is a linear function of measured variables, it does not require homoscedasticity between the variables.

Linearity: Factor analysis is also based on linearity assumption. Non-linear variables can also be used. After transfer, however, it changes into linear variable.

Interval Data: Interval data are assumed.

There are different types of methods used to extract the factor from the data set:

1. **Principal component analysis:** This is the most common method used by researchers. PCA starts extracting the maximum variance and puts them into the first factor. After that, it removes that variance explained by the first factors and then starts extracting maximum variance for the second factor. This process goes to the last factor.
2. **Common factor analysis:** The second most preferred method by researchers, it extracts the common variance and puts them into factors. This method does not include the unique variance of all variables. This method is used in SEM.
3. **Image factoring:** This method is based on correlation matrix. OLS Regression method is used to predict the factor in image factoring.
4. **Maximum likelihood method:** This method also works on correlation metric, but it uses maximum likelihood method to factor.
5. **Other methods of factor analysis:** Alfa factoring outweighs least squares. Weight square is another regression-based method which is used for factoring.

3. Functions

An R function is created by using the keyword **function**. The basic syntax of an R function definition is as follows:

```
function_name <- function(arg_1, arg_2, ...) {  
  Function body  
}
```

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

```
# Create a function to print squares of numbers in sequence.  
new.function <- function(a) {  
  for(i in 1:a) {  
    b <- i^2  
    print(b)  
  }  
}  
  
# Call the function new.function supplying 6 as an argument.  
new.function(6)
```

→

```
[1] 1  
[1] 4  
[1] 9  
[1] 16  
[1] 25  
[1] 36
```

In Python functions work the same as in R, but have a different code

```
def my_function(fname):  
    print(fname + " Refsnes")  
  
my_function("Emil")
```

→

```
Emil Refsnes
```

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition. In the following example the functions will print out the length of the list that will be created by iterating each value passed to the function

```
def new_function(*args):
    lst=[]
    for i in args:
        lst.append(i)
    print(len(lst))

print(new_function('swed','ko')) → 2
```

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: ** before the parameter name in the function definition.

```
def my_function(**kid):
    print("his last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes") → His last name is Refsnes
```

Also, If we call the function without argument, it uses the default value:

```
def my_function2(country = "Norway"):
    print("I am from " + country)

my_function2("Sweden")
my_function2("India")
my_function2()
my_function2("Brazil") → I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

4. Read Data.

In Python to read data for data science you'll use the pandas package as pd and call each function for each data type.

```
import pandas as pd

pd.read_csv('data.csv')
pd.read_json('data.json')
pd.read_excel('data.xlsx')

from sqlalchemy import create_engine
engine = create_engine("sqlite:///memory:")
with engine.connect() as conn, conn.begin():
    data = pd.read_sql_table("data", conn)
```

Each function has its own arguments, but I'll show you only the most used ones.

```
# If your data does not contain a header, you can use the <prefix>
parameter to include a test.

df = pd.read_csv('content/sample.csv', header=None, prefix='col_')
```

```
df = pd.read_csv('content/sample.csv',
                header=None,
                prefix='col_')

df.head(5)
```

	col_0	col_1	col_2	col_3	col_4	col_5
0	1	2	1	-4	5	6
1	21	2	334	554	54	55
2	23	333	445	6	65	56

In Rstudio its much easier to read data. Just click the import dataset option and select your data type



And if you're working with CSV files you need to call them following code: `read.csv("data.csv")`

5. Manipulate Data Frames

In this chapter I'm going to introduce you to pandas functions to manipulate data frames in python

```
df.rename(columns={'Name': 'CountryName',
                  'Code': 'CountryCode'},
          inplace=False)
```

```
pd.merge(df, df1, on='Name')
```

```
df.drop(columns=['B', 'C'])
```

```
pd.DataFrame({'Name': ['Rohan', 'Rahul', 'Gaurav',
                      'Ananya', 'Vinay', 'Rohan',
                      'Vivek', 'Vinay'],
              'Score': [76, 69, 70, 88, 79, 64, 62, 57]})
```

→

	Name	Score
0	Rohan	76
1	Rahul	69
2	Gaurav	70
3	Ananya	88
4	Vinay	79
5	Rohan	64
6	Vivek	62
7	Vinay	57

```
student.sort_values(by=['Score'], ascending=True)
```

```
student.sort_values(by=['Name', 'Score'],
                    ascending=[True, False])
```

→

```
new_row = {'A':4, 'B':2, 'C':6, 'D':4, 'city':'Berlin'}
df = df.append(new_row, ignore_index=True)
```

	A	B	C	D	city
0	8	7	0	1	Rome
1	5	7	5	8	Madrid
2	1	9	1	6	Houston
3	4	2	5	4	Berlin

```
df.pivot_table(index='cat', columns='city', aggfunc='mean')
```

	A			B		
	city	Berlin	Houston	Rome	Berlin	Houston
cat						
cat1		4.5	5.666667	6.333333	5.5	7.333333
cat2		4.5	2.666667	7.333333	7.0	5.333333
cat3		6.5	3.250000	4.500000	3.5	4.000000

And now I'm going to show you some R code to manipulate data frames

This code shows us the data type for each column →

```
data
#> data: 200 obs. of 4 variables
#> $ Gender : chr "Male" "Male" "Female" "Female" ...
#> $ Age : int 19 21 20 23 31 22 35 23 64 30 ...
#> $ Annual.Income..k.. : int 15 15 16 16 17 17 18 18 19 19 ...
#> $ Spending.Score..1.100.: int 39 81 6 77 40 76 6 94 3 72 ...
```

Data types →

	logical	TRUE	FALSE
	integer	1L	0L
	double	1.5	0.0
	character	"a"	"b"
	complex	1+1i	1-1i
	raw	raw(100)	raw(100)
	list	list(1)	list(1)

```
> library(plyr)
> rename(data, c("CustomerID"="ID"))
#> ID Gender Age Annual.Income..k.. Spending.Score..1.100.
#> 1 1 Male 19 15 39
#> 2 2 Male 21 15 81
#> 3 3 Female 20 16 6
#> 4 4 Female 23 16 77
#> 5 5 Female 31 17 40
#> 6 6 Female 22 17 76
#> 7 7 Female 35 18 6
#> 8 8 Female 23 18 94
```

```
data[["ID"]]<-NULL
```

```
# Permanently modifying column order
# in a Data frame
df <- df[c(2, 1, 3)]
```

6. Subsetting Data Frames

In this chapter I will show you only how to slice your data in R

```
# select 1st and 5th thru 10th variables
newdata <- mydata[c(1,5:10)]
```

```
# exclude variables v1, v2, v3
myvars <- names(mydata) %in% c("v1", "v2", "v3")
newdata <- mydata[!myvars]
```

```
# exclude 3rd and 5th variable
newdata <- mydata[c(-3,-5)]
```

```
# delete variables v3 and v5
mydata$v3 <- mydata$v5 <- NULL
```

```
# using subset function
newdata <- subset(mydata, age >= 20 | age < 10,
select=c(ID, Weight))
```

```
# first 5 observations
newdata <- mydata[1:5,]
```

```
# based on variable values
newdata <- mydata[ which(mydata$gender=='F'
& mydata$age > 65), ]
```

```
# or
attach(mydata)
newdata <- mydata[ which(gender=='F' & age > 65),]
detach(mydata)
```

```
# using subset function (part 2)
newdata <- subset(mydata, sex=="m" & age > 25,
select=weight:income)
```

Now the code to subset data in Python

```
# Selecting rows where score is
# greater than 70
print(student[student.Score>70])

# Selecting rows where score is greater than 60
# OR less than 70
print(student[(student.Score>60) | (student.Score<70)])
```

```
# Printing five rows with name column only
# i.e. printing first 5 student names.
print(student.iloc[:5, 'Name'])

# Printing all the rows with score column
# only i.e. printing score of all the
# students
print(student.iloc[:, 'Score'])

# Printing only first rows having name,
# score columns i.e. print first student
# name & their score.
print(student.iloc[0, 0:2])

# Printing first 3 rows having name, score &
# percentage columns i.e. printing first three
# student name, score & percentage.
print(student.iloc[0:3, 0:3])

# Printing all rows having name & score
# columns i.e. printing all student
# name & their score.
print(student.iloc[:, 0:2])
```

III. Statistics (and how to calculate them in Python)

For this chapter we are going to work with the same dataset to perform all the statistics in python.

1. Descriptive Statistics

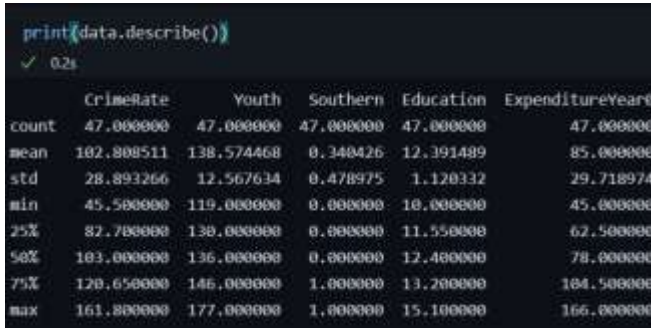
The descriptive statistics are a group of different metrics that we can extract from a data set.

- Mean: Average value
- Median: Midpoint between the highest and the lowest values

- Range: Span of values over which your data occurs
- Standard Deviation of the Mean (σ): On average, how much each measurement deviates from the mean
- Variance (σ^2): Is a measure of how spread out a data set is. It is calculated as the average squared deviation of each number from the mean of a data set

In just one line of code, we have the number of values, mean, standard deviation, minimum value, maximum value, the values positioned in 25%, 50% (median) and 75%. For each column.

```
print(data.describe())
```



	CrimeRate	Youth	Southern	Education	ExpenditureYear0
count	47.000000	47.000000	47.000000	47.000000	47.000000
mean	102.808511	138.574468	0.340426	12.391489	85.000000
std	28.893266	12.567634	0.478975	1.120332	29.718974
min	45.500000	119.000000	0.000000	10.000000	45.000000
25%	82.700000	130.000000	0.000000	11.550000	62.500000
50%	103.000000	136.000000	0.000000	12.400000	78.000000
75%	120.650000	146.000000	1.000000	13.200000	104.500000
max	161.800000	177.000000	1.000000	15.100000	166.000000

A small standard deviation can be a goal in certain situations where the results are restricted, for example, in product manufacturing and quality control. A particular type of car part that has to be 2 centimeters in diameter to fit properly had better not have a very big standard deviation during the manufacturing process! A big standard deviation in this case would mean that lots of parts end up in the trash because they don't fit right; either that, or the cars will have major problems down the road. So the more spread out the group of numbers are, the higher the standard deviation.

But in situations where you just observe and record data, **a large standard deviation isn't necessarily a bad thing**; it just reflects a large amount of variation in the group that is being studied.

For example, if you look at salaries for everyone in a certain company, including everyone from the student intern to the CEO, the standard deviation may be very large. On the other hand, if you narrow the group down by looking only at the student interns, the standard deviation is smaller, because the individuals within this group have salaries that are similar and less variable. The second data set isn't better, it's just less variable.

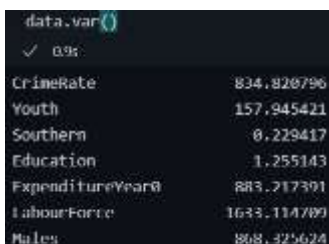
Similar to the mean, **outliers affect the standard deviation** (after all, the formula for standard deviation includes the mean). Here's an example: the salaries of the L.A. Lakers in the 2009–2010 season range from the highest, \$23,034,375 (Kobe Bryant) down to \$959,111 (Didier Ilunga-Mbenga and Josh Powell). Lots of variation, to be sure!

The standard deviation of the salaries for this team turns out to be \$6,567,405; it's almost as large as the average. However, as you may guess, if you remove Kobe Bryant's salary from the data set, the standard deviation decreases because the remaining salaries are more concentrated around the mean. The standard deviation becomes \$4,671,508

The standard deviation has the same units of measure as the original data. If you're talking about inches, the standard deviation will be in inches.

Variance is the square of the standard deviation and is the last metric:

```
data.var()
```



	CrimeRate	Youth	Southern	Education	ExpenditureYear0
CrimeRate	834.826796				
Youth		157.945421			
Southern			0.229417		
Education				1.255143	
ExpenditureYear0					883.717391
LabourForce					1633.114709
Males					868.325624

The more spread out the values are in a dataset, the higher the variance. Therefore, the variance gives more importance to outliers than the standard deviation.

2. Histograms

All histograms have the same components but because they have different values they differ from each other's shape.

- Range
 - How widely dispersed are the frequencies of each bin? Extremely large frequency ranges (particularly as a percentage) may indicate data that is fundamentally unreliable.
 - How wide are the bins themselves? Specifically, how broad are the intervals or how descriptive are the classes? Unusually large or small intervals, or unusually broad or narrow categories may indicate important observations about the data as a whole.
- Frequency Density
 - Frequency is measured by the area of the bar. What that means it that you can use a histogram with different interval or class widths to represent data with varying densities.
- Shape
 - The shape of a histogram can lead to valuable conclusions about the trend(s) of the data

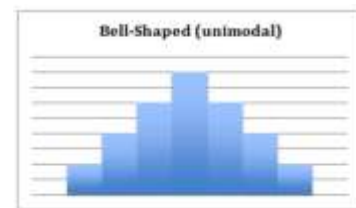
As you can see in the code (→), we obtained the histogram for the CrimeRate and Education columns. The shape of the data isn't a bell because they have some outliers.

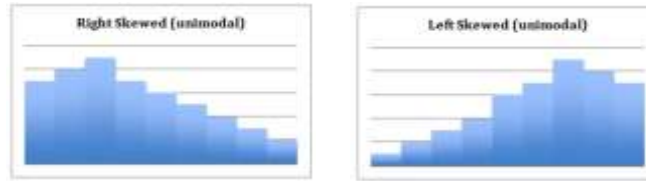
We can say that the crime rates between 80 and 120 have more frequency than the others. And Education has more values between 12 and 13.



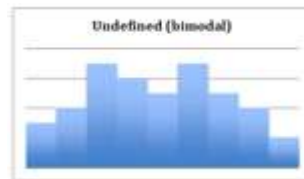
Other types of histograms:

- Bell-shaped: A histogram with a prominent 'mound' in the center and similar tapering to the left and right. One indication of this shape is that the data is unimodal – meaning that the data has a single mode, identified by the 'peak' of the curve. **If the shape is symmetrical, then the mean, median, and mode are all the same value.** Note that a normally distributed data set creates a symmetric histogram that looks like a bell, leading to the common term for a normal distribution: a bell curve
- Right or Left Skewed: A skewed histogram has a peak that is left (or right) of center and a more gradual tapering to the other side of the graph. This is a unimodal data set, with the mode closer to the left (or right) of the graph and smaller than either the mean or the median. The mean of right-skewed data will be located to the right side of the graph and will be a greater value than either the median or the mode. This shape indicates that there are a number of data points, perhaps outliers, that are greater than the mode for the left skewed and lesser than the mode for the right





- Undefined Bimodal: This shape is not specifically defined, but we can note regardless that it is bimodal, having two separated classes or intervals equally representing the maximum frequency of the distribution.



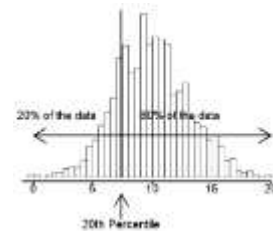
3. Percentiles and Outliers

Percentiles is in everyday use, but there is no universal definition for it. The most common definition of a percentile is a number where a certain percentage of scores fall below that a number.

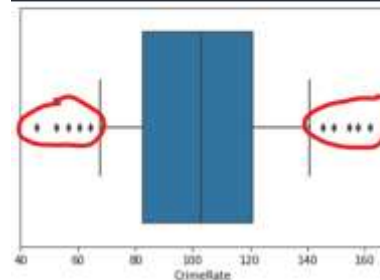
An outlier is an observation that lies an abnormal distance from other values in a random sample.

One way to visualize the percentiles rank and the outliers is by plotting a boxplot.

The box limits the 25th and 75th percentiles. The two lines outside the box limit the 10th and 90th percentile and the circles in red show the outliers.



```
sns.boxplot(data=data, x='CrimeRate', whis=[10,90])
plt.plot()
✓ 0.2s
[]
```



4. Probability Theory

The probability theory is a branch of mathematics concerned with the analysis of random phenomena. The outcome of a random event cannot be determined before it occurs, but it may be any one of several possible outcomes. The actual outcome is considered to be determined by chance. This theory has great importance for data science. One needs to possess a comprehensive understanding of the probability theory to be a well-performing data scientist. For instance, probability distributions play a key role in predictive analytics.

Conditional Probability

We defined the conditional probability as the probability of event A given that event B has occurred, is denoted as $p(A|B)$.

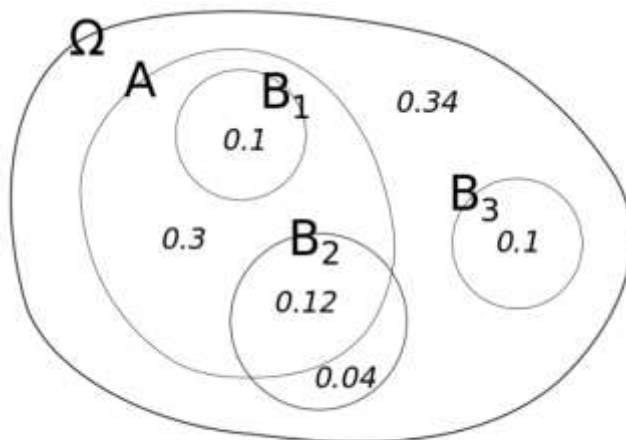
“Conditional probability is the likelihood of an event A to occur given that another event that has a relation with event A has already occurred.”

The formula of the conditional probability:

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

$P(A \cap B)$ is the probability that both events A and B occur. $P(B)$ is the probability that event B occurs. The conditional probability is typically not commutative which means that $P(A | B)$ is not equal to $P(B | A)$.

In the following example we see a probability space (Ω) which indicates all the probabilities add up to 1.



(image source: Wikipedia)

The unconditional probability of event A ,

$$P(A) = 0.1 + 0.3 + 0.12 = 0.52$$

The conditional probability of event A given that B_2 occurred, $P(A | B_2)$,

$$P(A | B_2) = P(A \cap B_2) / P(B_2) = 0.12 / 0.16 = 0.75$$

Conditional probability is a fundamental concept in probability theory and statistics. For instance, Bayesian statistics arises from an interpretation of the conditional probability.

A given condition might not have any effect on an event so the conditional and unconditional probabilities are equal (i.e. $P(A | B) = P(A)$). In such cases, the events **A and B are said to be independent.**

Joint Probability

Joint probability is the probability of two events occurring together. If two events are independent, the joint probability is calculated by multiplying the probabilities of each event.

$$P(A \cap B) = P(A) * P(B) \quad \text{If we put that in the equation of the conditional probability:} \quad P(A | B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) * P(B)}{P(B)} = P(A)$$

Joint and conditional probability example:

The following table shows the number of female and male students enrolled in the sociology and music classes.

Count	Sociology	Music	Total
Female	24	28	52
Male	32	19	51
Total	56	47	103

There are 103 students. We will first calculate the unconditional probabilities.

$$P(\text{Female}) = 52 / 103 = 0.505$$

$$P(\text{Male}) = 51 / 103 = 0.495$$

$$P(\text{Music}) = 47 / 103 = 0.456$$

$$P(\text{Sociology}) = 56 / 103 = 0.544$$

The joint probabilities can be calculated by dividing the number in a cell by the total number of students. For instance, the probability that a student is female and enrolled in the sociology class:

$$P(\text{Female} \cap \text{Sociology}) = (P(\text{Female}) * P(\text{Sociology})) = 24 / 103 = 0.233$$

We can calculate the other joint probabilities similarly. The following table contains all the probabilities for these events.

Probability	Sociology	Music	Total
Female	0.233	0.272	0.505
Male	0.311	0.184	0.495
Total	0.544	0.456	1

We will calculate the conditional probabilities now.

What is the probability that a student is female given that the student is enrolled in the music class?

$$P(\text{Female} | \text{Music}) = P(\text{Female} \cap \text{Music}) / P(\text{Music}) = 0.272 / 0.456 = 0.596$$

What is the probability that a student is male given that the student is enrolled in the sociology class?

$$P(\text{Male} | \text{Sociology}) = P(\text{Male} \cap \text{Sociology}) / P(\text{Sociology}) = 0.311 / 0.544 = 0.572$$

5. Bayes Theorem

Principled way of calculating a conditional probability without the joint probability.

Specifically, one conditional probability can be calculated using the other conditional probability; for example:

Firstly, in general, the result $P(A|B)$ is referred to as the posterior probability and $P(A)$ is referred to as the prior probability and sometimes $P(B|A)$ is referred to as the likelihood and $P(B)$ is referred to as the evidence.

$$P(A|B) = P(B|A) * P(A) / P(B)$$

The reverse is also true; for example:

$$P(B|A) = P(A|B) * P(B) / P(A)$$

It is often the case that we do not have access to the denominator directly, e.g. $P(B)$.

We can calculate it an alternative way; for example:

$$P(B) = P(B|A) * P(A) + P(B|\text{not } A) * P(\text{not } A)$$

This gives a formulation of Bayes Theorem that we can use that uses the alternate calculation of $P(B)$, described below::

$$P(A|B) = P(B|A) * P(A) / (P(B|A) * P(A) + P(B|\text{not } A) * P(\text{not } A))$$

Diagnostic Test Scenario

Consider a human population that may or may not have cancer (Cancer is True or False) and a medical test that returns positive or negative for detecting cancer (Test is Positive or Negative), e.g. like a mammogram for detecting breast cancer.

Problem: If a randomly selected patient has the test and it comes back positive, what is the probability that the patient has cancer?

Manual Calculation

Medical diagnostic tests are not perfect; they have error.

Sometimes a patient will have cancer, but the test will not detect it. This capability of the test to detect cancer is referred to as the sensitivity, or the true positive rate.

In this case, we will contrive a sensitivity value for the test. The test is good, but not great, with a true positive rate or sensitivity of 85%. That is, of all the people who have cancer and are tested, 85% of them will get a positive result from the test.

$$P(\text{Test=Positive} | \text{Cancer=True}) = 0.85$$

Given this information, our intuition would suggest that there is an 85% probability that the patient has cancer.

Our intuitions of probability are wrong.

This type of error in interpreting probabilities is so common that it has its own name; it is referred to as the **base rate fallacy**.

It has this name because the error in estimating the probability of an event is caused by ignoring the base rate. That is, it **ignores the probability of a randomly selected person having cancer**, regardless of the results of a diagnostic test.

In this case, we can assume the probability of breast cancer is low and use a contrived base rate value of one person in 5,000, or (0.0002) 0.02%.

$$P(\text{Cancer=True}) = 0.02\%.$$

We can correctly calculate the probability of a patient having cancer given a positive test result using Bayes Theorem.

Let's map our scenario onto the equation:

$$P(A|B) = P(B|A) * P(A) / P(B)$$

$$P(\text{Cancer=True} | \text{Test=Positive}) = P(\text{Test=Positive} | \text{Cancer=True}) * P(\text{Cancer=True}) / P(\text{Test=Positive})$$

We know the probability of the test being positive given that the patient has cancer is 85%, and we know the base rate or the prior probability of a given patient having cancer is 0.02%; we can plug these values in:

$$P(\text{Cancer=True} | \text{Test=Positive}) = 0.85 * 0.0002 / P(\text{Test=Positive})$$

We don't know $P(\text{Test}=\text{Positive})$, it's not given directly.

Instead, we can estimate it using:

$$P(B) = P(B|A) * P(A) + P(B|\text{not } A) * P(\text{not } A)$$

$$P(\text{Test}=\text{Positive}) = P(\text{Test}=\text{Positive}|\text{Cancer}=\text{True}) * P(\text{Cancer}=\text{True}) + P(\text{Test}=\text{Positive}|\text{Cancer}=\text{False}) * P(\text{Cancer}=\text{False})$$

Firstly, we can calculate $P(\text{Cancer}=\text{False})$ as the complement of $P(\text{Cancer}=\text{True})$, which we already know

$$P(\text{Cancer}=\text{False}) = 1 - P(\text{Cancer}=\text{True})$$

$$= 1 - 0.0002$$

$$= 0.9998$$

We can plug in our known values as follows:

$$P(\text{Test}=\text{Positive}) = 0.85 * 0.0002 + P(\text{Test}=\text{Positive}|\text{Cancer}=\text{False}) * 0.9998$$

We still do not know the probability of a positive test result given no cancer.

This requires additional information.

Specifically, we need to know how good the test is at correctly identifying people that do not have cancer. That is, testing negative result ($\text{Test}=\text{Negative}$) when the patient does not have cancer ($\text{Cancer}=\text{False}$), called the true negative rate or the specificity.

We will use a contrived specificity value of 95%.

$$P(\text{Test}=\text{Negative} | \text{Cancer}=\text{False}) = 0.95$$

With this final piece of information, we can calculate the false positive or false alarm rate as the complement of the true negative rate.

$$P(\text{Test}=\text{Positive}|\text{Cancer}=\text{False}) = 1 - P(\text{Test}=\text{Negative} | \text{Cancer}=\text{False})$$

$$= 1 - 0.95$$

$$= 0.05$$

We can plug this false alarm rate into our calculation of $P(\text{Test}=\text{Positive})$ as follows:

$$P(\text{Test}=\text{Positive}) = 0.85 * 0.0002 + 0.05 * 0.9998$$

$$P(\text{Test}=\text{Positive}) = 0.00017 + 0.04999$$

$$P(\text{Test}=\text{Positive}) = 0.05016$$

Excellent, so the probability of the test returning a positive result, regardless of whether the person has cancer or not is about 5%.

We now have enough information to calculate Bayes Theorem and estimate the probability of a randomly selected person having cancer if they get a positive test result.

$$P(\text{Cancer}=\text{True} \mid \text{Test}=\text{Positive}) = P(\text{Test}=\text{Positive} \mid \text{Cancer}=\text{True}) * P(\text{Cancer}=\text{True}) / P(\text{Test}=\text{Positive})$$

$$P(\text{Cancer}=\text{True} \mid \text{Test}=\text{Positive}) = 0.85 * 0.0002 / 0.05016$$

$$P(\text{Cancer}=\text{True} \mid \text{Test}=\text{Positive}) = 0.00017 / 0.05016$$

$$P(\text{Cancer}=\text{True} \mid \text{Test}=\text{Positive}) = 0.003389154704944$$

The calculation suggests that if the patient is informed they have cancer with this test, then there is only 0.33% chance that they have cancer.

And we code all of this in a few lines of code, as it follows:

```
# calculate the probability of cancer patient and diagnostic test
# calculate P(A|B) given P(A), P(B|A), P(B|not A)
def bayes_theorem(p_a, p_b_given_a, p_b_given_not_a):
    # calculate P(not A)
    not_a = 1 - p_a
    # calculate P(B)
    p_b = p_b_given_a * p_a + p_b_given_not_a * not_a
    # calculate P(A|B)
    p_a_given_b = (p_b_given_a * p_a) / p_b
    return p_a_given_b

# P(A)
p_a = 0.0002
# P(B|A)
p_b_given_a = 0.85
# P(B|not A)
p_b_given_not_a = 0.85
# calculate P(A|B)
result = bayes_theorem(p_a, p_b_given_a, p_b_given_not_a)
# summarize
print("P(A|B) = %.3f%%" % (result * 100))
✓ 0.3%
P(A|B) = 0.339%
```

6. Random Variables

A random variable, usually written X , is a variable whose possible values are numerical outcomes of a random phenomenon. There are two types of random variables, discrete and continuous.

Discrete

A discrete random variable is one which may take on only a countable number of distinct values such as 0,1,2,3,4,..... Discrete random variables are usually (but not necessarily) counts. If a random variable can take only a finite number of distinct values, then it must be discrete. Examples of discrete random variables include the number of children in a family, the Friday night attendance at a cinema, the number of patients in a doctor's surgery, the number of defective light bulbs in a box of ten.

The probability distribution of a discrete random variable is a list of probabilities associated with each of its possible values. It is also sometimes called the probability function or the probability mass function.

Suppose a random variable X may take k different values, with the probability that $X = x_i$ defined to be $P(X = x_i) = p_i$. The probabilities p_i must satisfy the following:

- $0 < p_i < 1$ for each i
- $p_1 + p_2 + \dots + p_k = 1$.

Example: Suppose a variable X can take the values 1, 2, 3, or 4.

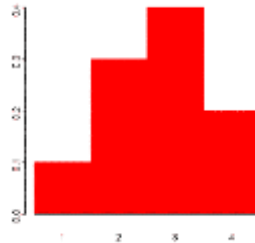
The probabilities associated with each outcome are described by the following table:

Outcome	1	2	3	4
---------	---	---	---	---

Probability	0.1	0.3	0.4	0.2
-------------	-----	-----	-----	-----

The probability that X is equal to 2 or 3 is the sum of the two probabilities: $P(X = 2 \text{ or } X = 3) = P(X = 2) + P(X = 3) = 0.3 + 0.4 = 0.7$. Similarly, the probability that X is greater than 1 is equal to $1 - P(X = 1) = 1 - 0.1 = 0.9$, by the complement rule.

This distribution may also be described by the probability histogram shown below:



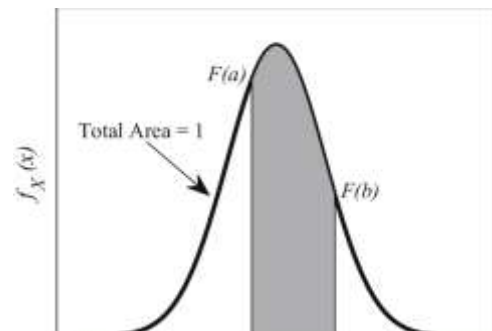
Continuous

A continuous random variable is one which takes an infinite number of possible values. Continuous random variables are usually measurements. Examples include height, weight, the amount of sugar in an orange, the time required to run a mile.

A continuous random variable is not defined at specific values. Instead, it is defined over an interval of values, and is represented by the area under a curve (in advanced mathematics, this is known as an integral). The probability of observing any single value is equal to 0, since the number of values which may be assumed by the random variable is infinite.

Suppose a random variable X may take all values over an interval of real numbers. Then the probability that X is in the set of outcomes A , $P(A)$, is defined to be the area above A and under a curve. The curve, which represents a function $p(x)$, must satisfy the following:

- 1: The curve has no negative values ($p(x) > 0$ for all x)
- 2: The total area under the curve is equal to 1.



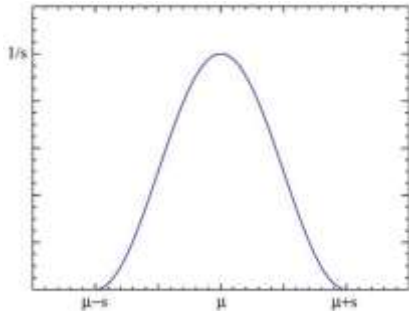
7. Probability Distribution Function

A probability distribution can be described in various forms, such as by a probability density function or a cumulative distribution function. Probability density functions, or PDFs, are mathematical functions that usually apply to continuous and discrete values. Now that we have a basic idea of what a PDF is, how are they used in statistics? More importantly, how are they applied to Data Science, and what do they look like on paper?

PDFs are very commonly used in statistical analysis, and thus are quite commonly used for Data Science. Generally, PDFs are a necessary tool when studying data with applied science using statistics

The normal distribution, that is quite commonly used in machine-learning. Standard scaling of data is quite a popular way to normalize continuous values whose data has high variance. This PDF transforms continuous samples into standard deviations from the population's mean.

Looking at a PDF, we see that it has a parabolic curve, where the center is where most of our data lies:



The normal distribution is a great example of a PDF. This is partially because it is very commonly used and familiar to most Scientists, but also partially because it has prominent applications across the entire spectrum of applied statistics. For many of the other distributions, the normal distribution acts as a base to be built upon for probability theory.

8. Cumulative Distribution Function

All random variables (**discrete and continuous**) have a cumulative distribution function. It is a function giving the probability that the random variable X is less than or equal to x , for every value x . For a discrete random variable, the cumulative distribution function is found by summing up the probabilities

The cumulative distribution function for the above probability distribution is calculated as follows:

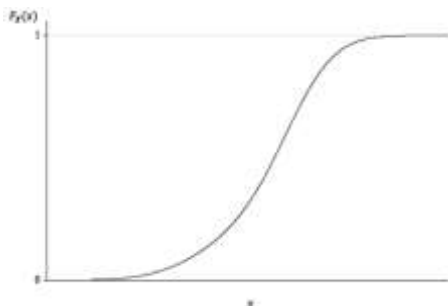
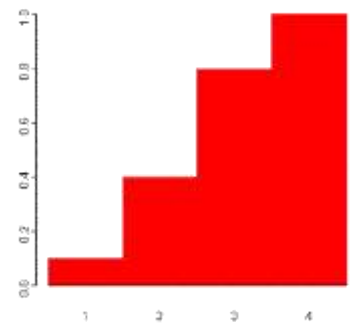
The probability that X is less than or equal to 1 is 0.1,

the probability that X is less than or equal to 2 is $0.1+0.3 = 0.4$,

the probability that X is less than or equal to 3 is $0.1+0.3+0.4 = 0.8$, and

the probability that X is less than or equal to 4 is $0.1+0.3+0.4+0.2 = 1$.

The probability histogram for the cumulative distribution of this random variable is shown to the right:



Comparison between data and CDF

Let's work with the 'Education' column from our dataset and compare it with the probability density function and cumulative density function.

First, prepare the data and the normal theoretical data

```
data_ed=data['education']
mean=np.mean(data_ed)
std=np.std(data_ed)
samples=np.random.normal(mean, std, size=10000)
```

steps to create the normal data

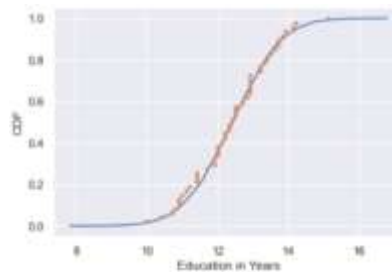
Second, define a function to compute the CDF for one-dimensional data

```
def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""
    # number of data points: n
    n = len(data)
    # x-data for the ECDF: x
    x = np.sort(data)
    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n
    return x, y

x, y = ecdf(data_ed)
x_norm, y_norm = ecdf(samples)
```

And the last step is plot everything together to check normality thru cumulative density function

```
sns.set()
plt.plot(x_norm, y_norm)
plt.plot(x, y, marker='.', linestyle='none')
plt.xlabel('Education in Years')
plt.ylabel('CDF')
```



As we can see, our data has similar shape to the cumulative distribution function. Thanks to this we can say that the education column is nearly normally distributed.

9. Central Limit Theorem

In probability theory, the central limit theorem (CLT) states that the distribution of a sample variable approximates a normal distribution (i.e., a “bell curve”) as the sample size becomes larger, assuming that all samples are identical in size, and regardless of the population's actual distribution shape.

Put another way, CLT is a statistical premise that, given a sufficiently large sample size from a population with a finite level of variance, **the mean of all sampled variables from the same population will be approximately equal to the mean of the whole population**. Furthermore, these samples approximate a normal distribution, with their variances being approximately equal to the variance of the population as the sample size gets larger, according to the law of large numbers.

10. Continuous Distributions

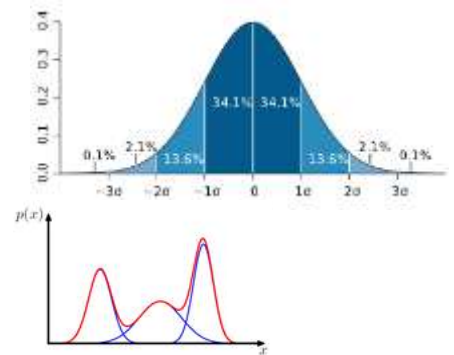
a. Normal Distribution (or Gaussian Distribution)

- Gaussian distribution is a continuous probability distribution with symmetrical sides around its center.
- Its mean, median and mode are equal.

- Its shape looks like below with most of the data points clustered around the mean with asymptotic tails.

84% of the values drawn from normal distribution lie within 1σ

99.7% of the values drawn from normal distribution lie within 3σ



Simple Gaussian distribution fails to capture the following structure →

b. Poisson Distribution

Poisson Distributions are commonly used to find the probability that an event might happen or not knowing how often it usually occurs. Additionally, Poisson Distributions can also be used to predict how many times an event might occur in a given time period.

Poisson Distributions are for example frequently used by insurance companies to conduct risk analysis (eg. predict the number of car crash accidents within a predefined time span) to decide car insurance pricing.

When working with Poisson Distributions, we can be confident of the average time between the occurrence of different events, but the precise moment an event might take place is randomly spaced in time.

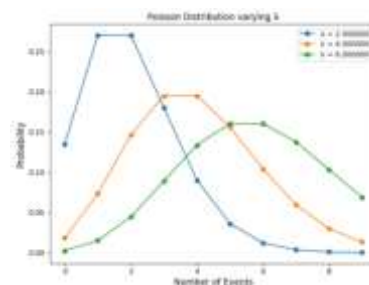
The main characteristics which describe Poisson Processes are:

The events are independent of each other (if an event happens, this does not alter the probability that another event can take place).

- An event can take place any number of times (within the defined time period).
- Two events can't take place simultaneously.
- The average rate between events occurrence is constant.

Poisson Distribution can be modelled using the following formula, where lambda represents the expected number of events which can take place in a period. Sometimes called the event rate or rate parameter.

$$P(X) = \frac{\lambda^x e^{-\lambda}}{X!}$$



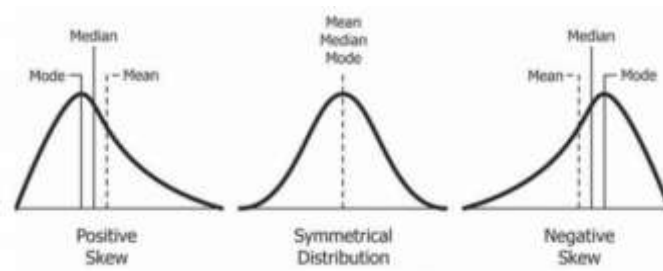
11. Skewness

Skewness is the measure of the asymmetry of an ideally symmetric probability distribution and is given by the third standardized moment. If that sounds way too complex, don't worry! Let me break it down for you.

In simple words, skewness is the measure of how much the probability distribution of a random variable deviates from the normal distribution. Now, you might be thinking – why am I talking about normal distribution here?

Well, the normal distribution is the probability distribution without any skewness. You can look at the image below which shows symmetrical distribution that's basically a normal distribution and you can see that it is symmetrical on both sides of the dashed line. Apart from this, there are two types of skewness:

- Positive Skewness
- Negative Skewness

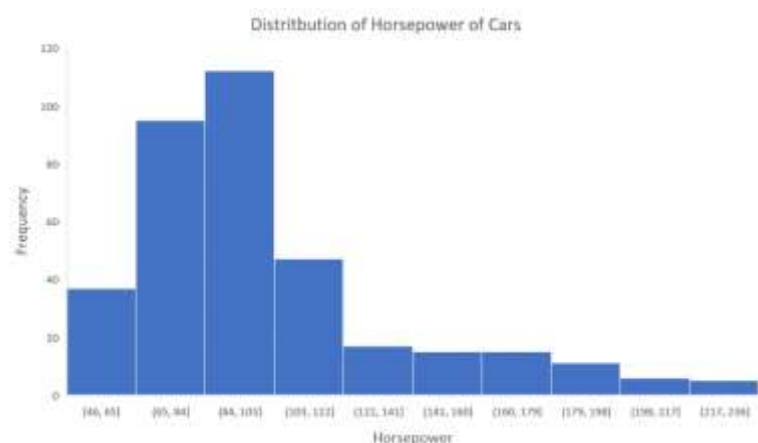


Let's take a look to the following distribution. It is the distribution of horsepower of cars:

You can clearly see that the above distribution is positively skewed. Now, let's say you want to use this as a feature for the model which will predict the mpg (miles per gallon) of a car.

Since our data is positively skewed here, it means that it has a higher number of data points having low values, i.e., cars with less horsepower. So when we train our model on this data, it will perform better at predicting the mpg of cars with lower horsepower as compared to those with higher horsepower.

Also, skewness tells us about the direction of outliers. You can see that our distribution is positively skewed and most of the outliers are present on the right side of the distribution.



Note: The skewness does not tell us about the number of outliers. It only tells us the direction.

To statistically determine the skewness, we have the Pearson's coefficient of skewness. This coefficient tells us the direction of the skewness (positive or negative) and we can obtain it in python as follows:

```
from scipy.stats import skew
sk1=[1,1,1,2,4,1,1,1,1,1,1,1,1,2,3,4,2,1,9,8,9]    more lower values
sk2=[1,2,1,1,9,8,9,8,7,8,9,8,7,14,9,9,8,9,7,9]    more higher values
print('The coefficients of skewness are ',skew(sk1),'(lower values)' and', skew(sk2), '(higher values)')
✓ 0.3s
The coefficients of skewness are  1.5657272468749186 (lower values) and -0.6625905684897368 (higher values)
```

The direction of skewness is given by the sign.

- The coefficient compares the sample distribution with a normal distribution. The larger the value, the larger the distribution differs from a normal distribution.
- A value of zero means no skewness at all.
- A **large negative** value means the distribution is **negatively skewed**.
- A **large positive** value means the distribution is **positively skewed**.

12. Monte Carlo Method

Monte Carlo simulation (also called the Monte Carlo Method or Monte Carlo sampling) is a way to account for risk in decision making and quantitative analysis. The method finds all possible outcomes of your decisions and assesses the impact of risk. The technique uses intensive statistical sampling methods that are so complex they are usually only performed with the aid of a computer.

The Monte Carlo method tells you:

- **All of the possible events that could or will happen,**
- **The probability of each possible outcome.**

The Monte Carlo simulation returns a quantified probability, which means that it gives you scenarios with numbers you can use. Let's say your company wants to know if local bird life will be adversely affected by the construction of a new factory close to wetlands. A quantified probability would be "If we build the factory, there is a 30% chance the nesting bird population will be adversely affected." This is more useful than a more general, qualified statement like "If we build the factory, the nesting bird population will be affected".

Monte Carlo simulations are used in many areas of industry and science, including:

- Analyzing radiative heat transfer problems (Wang et.al),
- Estimating the transmission of particles through matter (Biersack & Haggmark),
- Calculating the probability of cost overruns in large projects (McCabe),
- Foreseeing where prices of securities are likely to move (Boyle et. al),
- Analyzing how a network or electric grid will perform in different scenarios. For example, Sortomme et. al ran simulations for how electric vehicle charging will affect the electric grid in the future.
- Assessing risk for credit or insurance (Gordy).
- Simulating proteins in biology (Earl et. al)

Accuracy

While a Monte Carlo simulation provides some good accuracy, it is unlikely to hit the "exact" mark for several reasons:

- Vast amounts of data are usually involved.
- There are usually several unknowns in the system.
- As it is probabilistic (i.e. randomness plays a role in predicting future events), there will always be a margin of error related to the results.

In fact, it can be quite easy to run a "bad" Monte Carlo simulation (Brandimarte, 2014). This can happen for a variety of reasons, including:

- Use of an incorrect model or an unrealistic probability distribution,
- The underlying risk factors aren't complete (i.e. you haven't specified them well enough),
- The choice of Monte Carlo (which uses a stochastic model) isn't suited to your data,
- The random number generator chosen for the method isn't good enough,
- Computer bugs, which you may not be aware of if your area of expertise is statistics (as opposed to programming)

13. Hypothesis Testing

First, you might need to know what is a hypothesis, well, a hypothesis is an educated guess about something in the world around you. It should be testable, either by experiment or observation. For example:

- A new medicine you think might work.
- A way of teaching you think might be better.
- A possible location of new species.
- A fairer way to administer standardized tests.

It can really be anything at all as long as you can put it to the test.

If you are going to propose a hypothesis, it's customary to write a statement. Your statement will look like this:

“If I...(do this to an independent variable)....then (this will happen to the dependent variable).”

For example:

- If I (decrease the amount of water given to herbs) then (the herbs will increase in size).
- If I (give patients counseling in addition to medication) then (their overall depression scale will decrease).

A good hypothesis statement should:

- Include an “if” and “then” statement (according to the University of California).
- Include both the independent and dependent variables.
- Be testable by experiment, survey or other scientifically sound technique.
- Be based on information in prior research (either yours or someone else's).
- Have design criteria (for engineering or programming projects).

What is Hypothesis Testing?

Hypothesis testing in statistics is a way for you to test the results of a survey or experiment to see if you have meaningful results. You're basically testing whether your results are valid by figuring out the odds that your results have happened by chance. If your results may have happened by chance, the experiment won't be repeatable and so has little use.

Hypothesis testing can be one of the most confusing aspects for students, mostly because before you can even perform a test, you must know what your null hypothesis is. Often, those tricky word problems that you are faced with can be difficult to decipher. But it's easier than you think; all you need to do is:

1. Figure out your null hypothesis,
2. State your null hypothesis,
3. Choose what kind of test you need to perform,
4. Either support or reject the null hypothesis.

What is the Null Hypothesis?

If you trace back the history of science, the null hypothesis is always the accepted fact. Simple examples of null hypotheses that are generally accepted as being true are:

- DNA is shaped like a double helix.
- There are 8 planets in the solar system (excluding Pluto).
- Taking Vioxx can increase your risk of heart problems (a drug now taken off the market).

The null hypothesis is the default claim, premise, value or parameter that is generally accepted. The alternative hypothesis is a research or investigative hypothesis that includes the claim to be tested which could hopefully replace the null hypothesis, in case it is proved to be true. The various tests below help to figure out to what extent information from the sample data can be extrapolated to the population from which it was drawn.

One tailed test: A test of a statistical hypothesis, where the region of rejection is on only one side of the sampling distribution, is called a one-tailed test.

Example: a college has ≥ 4000 student or data science $\leq 80\%$ org adopted.

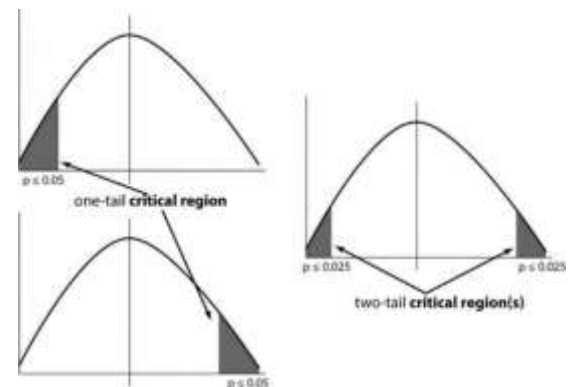
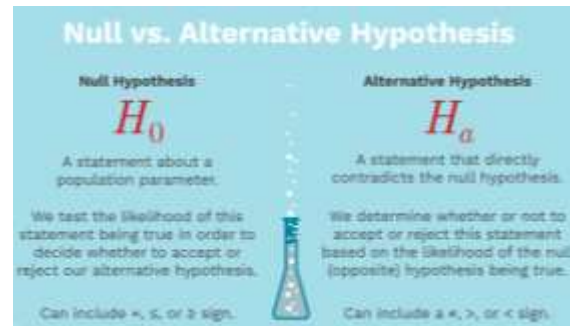
Two-tailed test: A two-tailed test is a statistical test in which the critical area of a distribution is two-sided and tests whether a sample is greater than or less than a certain range of values. If the sample being tested falls into either of the critical areas, the alternative hypothesis is accepted instead of the null hypothesis.

Example: a college $\neq 4000$ student or data science $\neq 80\%$ org adopted

P-Value

The P value, or calculated probability, is the probability of finding the observed, or more extreme, results when the null hypothesis (H_0) of a study question is true — the definition of ‘extreme’ depends on how the hypothesis is being tested.

If your P value is less than the chosen significance level, then you reject the null hypothesis i.e. accept that your sample gives reasonable evidence to support the alternative hypothesis. It does NOT imply a “meaningful” or “important” difference; that is for you to decide when considering the real-world relevance of your result.



14. Normality Tests

This section lists statistical tests that you can use to check if your data has a Gaussian distribution.

- **Shapiro-Wilk Test:** You use it when you want to tell if a random sample comes from a normal distribution or not. The smaller the W value returned by the test, the more confident you can be that the sample is not normally distributed. This result can lead to the rejection of the null hypothesis you started with in the first place. You can also think about this in terms of a p-value: when it is less than or equal to a certain threshold (usually 0.05), then you can reject the null hypothesis.

```
# Example of the Shapiro-Wilk Normality test for a sample of weights
from scipy.stats import shapiro
data2 = data1.pweight
stat, p = shapiro(data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably Gaussian')
else:
    print('Probably not Gaussian')
✓ 0.36
stat=0.969, p=0.055
probably Gaussian
```

- D'Agostino's K^2 Test: It first computes the skewness and kurtosis to quantify how far the distribution is from Gaussian in terms of asymmetry and shape. It then calculates how far each of these values differs from the value expected with a Gaussian distribution and computes a single P value from the sum of these discrepancies. It is a versatile and powerful normality test, and is recommended.

```
# Example of the D'Agostino's K^2 Normality Test
from scipy.stats import normaltest
data2 = data1.pweight
stat, p = normaltest(data2)
print('stat-%.3f, p-%.3f' % (stat, p))
if p > 0.05:
    print('Probably Gaussian')
else:
    print('Probably not Gaussian')
✓ 53s
stat=4.843, p=0.889
Probably Gaussian
```

- Anderson-Darling Test: It computes the P value by comparing the cumulative distribution of your data set against the ideal cumulative distribution of a Gaussian distribution. It takes into account the discrepancies at all parts of the cumulative distribution curve

```
# Example of the Anderson-Darling Normality Test
from scipy.stats import anderson
data2 = data1.pweight
result = anderson(data2)
print('stat-%.3f' % (result.statistic))
for i in range(len(result.critical_values)):
    sl, cv = result.significance_level[i], result.critical_values[i]
    if result.statistic < cv:
        print('Probably Gaussian at the %.1f%% level' % (sl))
    else:
        print('Probably not Gaussian at the %.1f%% level' % (sl))
✓ 11s
stat=0.368
Probably Gaussian at the 15.0% level
Probably Gaussian at the 10.0% level
Probably Gaussian at the 5.0% level
Probably Gaussian at the 2.5% level
Probably Gaussian at the 1.0% level
```

15. Correlation Tests

Correlation is a bivariate analysis that measures the strength of association between two variables and the direction of the relationship. In terms of the strength of relationship, the value of the correlation coefficient varies between +1 and -1. A value of ± 1 indicates a perfect degree of association between the two variables. As the correlation coefficient value goes towards 0, the relationship between the two variables will be weaker. The direction of the relationship is indicated by the sign of the coefficient; a + sign indicates a positive relationship and a - sign indicates a negative relationship

- Pearson's Correlation Coefficient: Pearson r correlation is the most widely used correlation statistic to measure the degree of the relationship between linearly related variables (Is there a relationship between temperature, measured in degrees Fahrenheit, and ice cream sales, measured by income?)
 - Observations in each sample are independent and identically distributed (iid).
 - Observations in each sample are normally distributed.
 - Observations in each sample have the same variance

```
# Example of the Pearson's Correlation test
from scipy.stats import pearsonr
data1 = [0.872, 2.817, 0.123, -0.945, -0.055, -1.436, 0.300, -1.478, -1.637, -1.869]
data2 = [0.353, 1.517, 0.125, -7.545, -0.555, -1.536, 1.350, -1.578, -1.537, -1.579]
stat, p = pearsonr(data1, data2)
print('stat-%.3f, p-%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
✓ 0.6s
stat=-0.688, p=0.028
Probably dependent
```

- Spearman's Rank Correlation: Spearman rank correlation is a non-parametric test that is used to measure the degree of association between two variables. The Spearman **rank correlation** test does not carry any assumptions about

the distribution of the data and is the appropriate correlation analysis when the variables are measured on a scale that is at least ordinal. (Is there a statistically significant relationship between participants' level of education (high school, bachelor's, or graduate degree) and their starting salary?)

- Observations in each sample are independent and identically distributed (iid).
- Observations in each sample can be ranked.

```
# Example of the Spearman's Rank Correlation Test
from scipy.stats import spearmanr
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [0.353, 3.517, 0.125, -7.545, -0.555, -1.536, 3.350, -1.578, -3.537, -1.579]
stat, p = spearmanr(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
✓ 0.7s

stat=0.855, p=0.002
Probably dependent
```

- Kendall's Rank Correlation: Kendall rank correlation is a non-parametric test that measures the strength of dependence between two variables. If we consider two samples, a and b, where each sample size is n, we know that the total number of pairings with a b is $n(n-1)/2$.
 - Observations in each sample are independent and identically distributed (iid).
 - Observations in each sample can be ranked.

```
# Example of the Kendall's Rank Correlation Test
from scipy.stats import kendalltau
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [0.353, 3.517, 0.125, -7.545, -0.555, -1.536, 3.350, -1.578, -3.537, -1.579]
stat, p = kendalltau(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
✓ 0.6s

stat=0.733, p=0.002
Probably dependent
```

- Chi-Squared Test: The test is applied when you have **two categorical variables** from a single population. It is used to determine **whether there is a significant association between the two variables**. This is useful when dealing with categorical variables, to see how likely it would be to observe your data's distribution if those variables were independent. For instance, let's say you want to know if being a man or a woman makes you more or less likely to be a smoker. If those two variables were independent, you could expect the same percentage of men and women among both smokers and non-smokers. Pearson's chi-squared test will tell you how confident you can be that there is such a difference, or not


```

import scipy.stats as stats

# create sample data according to survey
data = [['18-29', 'Conservative'] for i in range(141)] + \
        [['18-29', 'Socialist'] for i in range(66)] + \
        [['18-29', 'Other'] for i in range(4)] + \
        [['30-44', 'Conservative'] for i in range(129)] + \
        [['30-44', 'Socialist'] for i in range(159)] + \
        [['30-44', 'Other'] for i in range(7)] + \
        [['45-65', 'Conservative'] for i in range(280)] + \
        [['45-65', 'Socialist'] for i in range(276)] + \
        [['45-65', 'Other'] for i in range(4)] + \
        [['65 & older', 'Conservative'] for i in range(86)] + \
        [['65 & older', 'Socialist'] for i in range(101)] + \
        [['65 & older', 'Other'] for i in range(4)]
df = pd.DataFrame(data, columns = ['Age group', 'Political Affiliation'])

# create contingency table
data_crosstab = pd.crosstab([df['Age group'],
                             df['Political Affiliation'],
                             ],
                             margins=True, margins_name="Total")

# significance level
alpha = 0.05

# calculation of Chi-square test statistics
chi_square = 0
rows = df['Age group'].unique()
columns = df['Political Affiliation'].unique()
for i in columns:
    for j in rows:
        o = data_crosstab[i][j]
        e = data_crosstab[i]['total'] * data_crosstab['total'][j] / data_crosstab['total']['total']
        chi_square += (o - e)**2/e

# significance level
alpha = 0.05

# calculation of Chi-square test statistics
chi_square = 0
rows = df['Age group'].unique()
columns = df['Political Affiliation'].unique()
for i in columns:
    for j in rows:
        o = data_crosstab[i][j]
        e = data_crosstab[i]['total'] * data_crosstab['total'][j] / data_crosstab['total']['total']
        chi_square += (o - e)**2/e

# the p-value approach
print("Approach 1: The p-value approach to hypothesis testing in the decision rule")
p_value = 1 - stats.norm.cdf(chi_square, (len(rows)-1)*(len(columns)-1))
conclusion = "Failed to reject the null hypothesis."
if p_value <= alpha:
    conclusion = "Null hypothesis is rejected."

print("chisquare score is:", chi_square, " and p value is:", p_value)
print(conclusion)

# The critical value approach
print("Approach 2: The critical value approach to hypothesis testing in the decision rule")
critical_value = stats.chi2.ppf(1-alpha, (len(rows)-1)*(len(columns)-1))
conclusion = "Failed to reject the null hypothesis."
if chi_square > critical_value:
    conclusion = "Null hypothesis is rejected."

print("chisquare score is:", chi_square, " and p value is:", p_value)
print(conclusion)

Approach 1: The p-value approach to hypothesis testing in the decision rule
chisquare score is: 24.367421717385282 and p value is: 0.0
Null hypothesis is rejected.

-----
Approach 2: The critical value approach to hypothesis testing in the decision rule
chisquare score is: 24.367421717385282 and p value is: 12.591587243743977
Null hypothesis is rejected.

```

16. Stationary Tests

Stationarity is an important concept in time series analysis because this means that the statistical properties of a time series (or rather the process generating it) do not change over time. Stationarity is important because many useful analytical tools and statistical tests and models rely on it. **Time series are stationary if they do not have trend or seasonal effects**

One way to detecting stationarity in time series data is using statistical tests developed to detect specific types of stationarity, namely those brought about by simple parametric models of the generating stochastic process

- Augmented Dickey-Fuller: Tests whether a time series has a unit root, e.g. has a trend or more generally is autoregressive.

- Observations in are temporally ordered

```
# Example of the Augmented Dickey-Fuller unit root test
from statsmodels.tsa.stattools import adfuller
data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
stat, p, lags, obs, crit, t = adfuller(data)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably not Stationary')
else:
    print('Probably Stationary')
✓ 0.4s

stat=-0.517, p=0.985
Probably not Stationary
```

- Kwiatkowski-Phillips-Schmidt-Shin: **Tests whether a time series is trend stationary or not.**

- Observations in are temporally ordered

```
# Example of the Kwiatkowski-Phillips-Schmidt-Shin test
from statsmodels.tsa.stattools import kpss
data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
stat, p, lags, crit = kpss(data)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably not Stationary')
else:
    print('Probably Stationary')
✓ 0.6s

stat=0.410, p=0.073
Probably not Stationary
```

17. Parametric Statistical Hypothesis Tests

- Parametric tests can provide trustworthy results with distributions that are skewed and nonnormal
- Parametric tests can provide trustworthy results when the groups have different amounts of variability
- Parametric tests have greater statistical power

Parametric analyses	Sample size requirements for nonnormal data
1-sample t-test	Greater than 20
2-sample t-test	Each group should have more than 15 observations
One-Way ANOVA	<ul style="list-style-type: none"> + For 2-9 groups, each group should have more than 15 observations + For 10-12 groups, each group should have more than 20 observations

- If the mean is a better measure and you have a sufficiently large sample size, a parametric test usually is the better, more powerful choice than the nonparametric

Types of tests:

- Student's t-test: It is essentially, testing the significance of the difference of the mean values when the sample size is small (i.e, less than 30) and when the population standard deviation is not available.
 - Population distribution is normal, and
 - Samples are random and independent

- The sample size is small.
- Population standard deviation is not known.
- A T-test can be a:
 - One Sample T-test: To compare a sample mean with that of the population mean.
 - Two-Sample T-test: To compare the means of two different samples.

```
* Example of the Student's T-test
from scipy.stats import ttest_ind
data1 = [0.871, 1.817, 0.121, -0.945, -0.050, -1.436, 0.360, -1.478, -1.637, 1.880]
data2 = [1.542, -0.432, -0.938, -0.729, 0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
stat, p = ttest_ind(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
✓ 0.7x
stat=-0.126, p=0.748
probably the same distribution
```

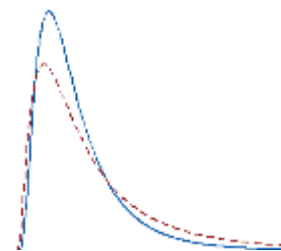
- F-Test: It is a test for the null hypothesis that two normal populations have the same variance.
 - An F-test is regarded as a comparison of equality of sample variances.
 - F-statistic is simply a ratio of two variances.
 - By changing the variance in the ratio, F-test has become a very flexible test. It can then be used to:
 - Test the overall significance for a regression model.
 - To compare the fits of different models and
 - To test the equality of means.
 - Population distribution is normal, and samples are drawn randomly and independently.
- Analysis of Variance Test (ANOVA): Next chapter
- Repeated Measures ANOVA Test: Next chapter

18. Nonparametric Statistical Hypothesis Tests

Nonparametric tests assess the median which can be better for some study areas. Nonparametric analyses provide an advantage because they assess the median rather than the mean. The mean is not always the better measure of central tendency for a sample. Even though you can perform a valid parametric analysis on skewed data, that doesn't necessarily equate to being the better method.

Example: Salaries tend to be a right-skewed distribution. The majority of wages cluster around the median, which is the point where half are above, and half are below. However, there is a long tail that stretches into the higher salary ranges. This long tail pulls the mean far away from the central median value. The two distributions are typical for salary distributions.

In these distributions, if several very high-income individuals join the sample, the mean increases by a significant amount even though incomes for most people don't change. They still cluster around the median.



In this situation, parametric and nonparametric test results can give you different results, and they both can be correct! For the two distributions, if you draw a large random sample from each population, the difference between the means is statistically significant. Despite this, the difference between the medians is not statistically significant. Here's how this works.

For skewed distributions, changes in the tail affect the mean substantially. Parametric tests can detect this mean change. Conversely, the median is relatively unaffected, and a nonparametric analysis can legitimately indicate that the median has not changed significantly.

You need to decide whether the mean or median is best for your study and which type of difference is more important to detect.

Nonparametric tests are valid when our sample size is small, and your data are potentially nonnormal. Use a nonparametric test when your sample size isn't large enough to satisfy the requirements in the table above and you're not sure that your data follow the normal distribution. With small sample sizes, be aware that normality tests can have insufficient power to produce useful results.

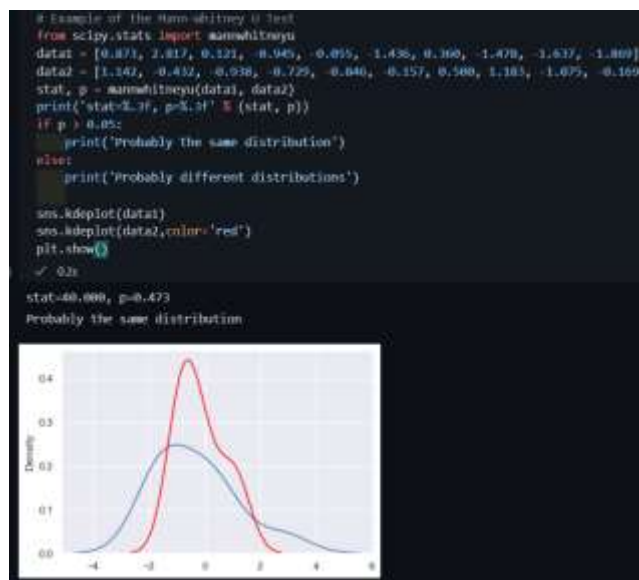
This situation is difficult. Nonparametric analyses tend to have lower power at the outset, and a small sample size only exacerbates that problem.

Nonparametric tests can analyze ordinal data, ranked data, and outliers

If the median is a better measure, consider a nonparametric test regardless of your sample size.

Types of tests:

- **Mann-Whitney U Test:** This is also called the Mann-Whitney U test in statistics. This test becomes useful when you want to **look at the difference between two independent groups**, especially when the dependent variable is either ordinal or continuous, but not normally distributed.
 - Observations in each sample are independent and identically distributed (iid).
 - Observations in each sample can be ranked.



- **Wilcoxon Signed-Rank Test:**
 - Wilcoxon signed-rank test does not assume normality in the data, it can be used when this assumption has been violated and the use of the dependent t-test is inappropriate. It is used to compare two sets of scores that come from the same participants. This can occur when we wish to investigate any change in scores from one time point to another, or when individuals are subjected to more than one condition.
 - For example, you could use a Wilcoxon signed-rank test to understand whether there was a difference in smokers' daily cigarette consumption before and after a 6 week hypnotherapy programme (i.e., your dependent variable would be "daily cigarette consumption", and your two related groups would be the cigarette consumption values "before" and "after" the hypnotherapy programme).

- Also use a Wilcoxon signed-rank test to understand whether there was a difference in reaction times under two different lighting conditions (i.e., your dependent variable would be "reaction time", measured in milliseconds, and your two related groups would be reaction times in a room using "blue light" versus "red light").
- Observations in each sample are independent and identically distributed (iid).
- Observations in each sample can be ranked.
- Observations across each sample are paired.

```
# Example of the Wilcoxon Signed-Rank Test
from scipy.stats import wilcoxon
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.368, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
stat, p = wilcoxon(data1, data2)
print('stat=%3f, p=%3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
✓ 0.4s

stat=21.000, p=0.557
Probably the same distribution
```

- Kruskal-Wallis H Test:

- This test is used for comparing two or more independent samples of equal or different sample sizes. It extends the Mann-Whitney-U-Test which is used to comparing only two groups.
- One-Way ANOVA is the parametric equivalent of this test. And that's why it is also known as 'One-Way ANOVA on ranks'.
- It uses ranks instead of actual data.
- It does not assume the population to be normally distributed.
- The test statistic used here is "H".



- Friedman Test

- The Friedman test is the non-parametric alternative to the one-way ANOVA with repeated measures. It is used to test for differences between groups when the dependent variable being measured is ordinal. It can also be used for continuous data that has violated the assumptions necessary to run the one-way ANOVA with repeated measures (e.g., data that has marked deviations from normality).
- Observations in each sample are independent and identically distributed (iid).
- Observations in each sample can be ranked.
- Observations across each sample are paired.

```
# Example of the Friedman Test
from scipy.stats import friedmanchisquare
data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]
data2 = [1.142, -0.432, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.075, -0.169]
data3 = [-0.208, 0.696, 0.928, -1.148, -0.213, 0.229, 0.137, 0.269, -0.870, -1.204]
stat, p = friedmanchisquare(data1, data2, data3)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
✓ 0.5s

stat=0.800, p=0.670
Probably the same distribution
```

19. Analysis of Variance (ANOVA)

An ANOVA test is a way to find out if survey or experiment results are significant. In other words, they help you to figure out if you need to reject the null hypothesis or accept the alternate hypothesis.

Basically, you're testing groups to see if there's a difference between them. Examples of when you might want to test different groups:

- A group of psychiatric patients are trying three different therapies: counseling, medication and biofeedback. You want to see if one therapy is better than the others.
- A manufacturer has two different processes to make light bulbs. They want to know if one process is better than the other.
- Students from different colleges take the same exam. You want to see if one college outperforms the other.

There are two different types of independent variables to perform an ANOVA, which are:

- One-way has one independent variable (with 2 levels). For example: brand of cereal,
- Two-way has two independent variables (it can have multiple levels). For example: brand of cereal, calories.

Groups or levels are different groups within the same independent variable. In the above example, your levels for “brand of cereal” might be Lucky Charms, Raisin Bran, Cornflakes — a total of three levels. Your levels for “Calories” might be: sweetened, unsweetened — a total of two levels.

Let’s say you are studying if an alcoholic support group and individual counseling combined is the most effective treatment for lowering alcohol consumption. You might split the study participants into three groups or levels: Medication only, Medication and counseling, Counseling only. Your dependent variable would be the number of alcoholic beverages consumed per day.

If your groups or levels have a hierarchical structure (each level has unique subgroups), then use a nested ANOVA for the analysis.

One Way ANOVA

A one-way ANOVA is used to compare two means from two independent (unrelated) groups using the F-distribution. The null hypothesis for the test is that the two means are equal. Therefore, a significant result means that the two means are unequal.

Examples of when to use a one-way ANOVA

Situation 1: You have a group of individuals randomly split into smaller groups and completing different tasks. For example, you might be studying the effects of tea on weight loss and form three groups: green tea, black tea, and no tea.

Situation 2: Similar to situation 1, but in this case the individuals are split into groups based on an attribute they possess. For example, you might be studying leg strength of people according to weight. You could split participants into weight categories (obese, overweight and normal) and measure their leg strength on a weight machine.

Example in Python

For this new data we have a column named Diet which tells us the different types of diets assigned to a person. We have 3 different values →

```
data=pd.read_csv('Diet_8.csv')
print(data.head())
```

	Person	gender	Age	height	pre_weight	Diet	weight6weeks
0	25	41	171	68	2	68.8	
1	26	32	174	103	2	103.0	
2	1	0	22	159	58	1	54.2

Now let’s perform an ANOVA test comparing gender and pre weight →

The p-value (2.16×10^{-11}) for gender is much lower than the level of significance (0.05), therefore, we can say that there’s an association between the gender and the pre weight.

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model=ols('weight~gender', data=data).fit()
anova_lm=sm.stats.anova_lm(model, type=2)
print(anova_lm)
```

	df	sum_sq	mean_sq	F	PR(>F)
gender	2.0	2835.568616	1417.784308	34.687858	2.157987e-11
Residual	79.0	3643.880302	46.125177		

Two Way ANOVA

A Two-Way ANOVA is an extension of the One Way ANOVA. With a One Way, you have one independent variable affecting a dependent variable. With a Two-Way ANOVA, there are two independents. Use a two-way ANOVA when you have one measurement variable (i.e. a quantitative variable) and two nominal variables. In other words, **if your experiment has a quantitative outcome and you have two categorical explanatory variables**, a two-way ANOVA is appropriate.

For example, you might want to find out if there is an interaction between income and gender for anxiety level at job interviews. The anxiety level is the outcome, or the variable that can be measured. Gender and Income are the two categorical variables. These categorical variables are also the independent variables, which are called factors in a Two Way ANOVA.

Two-way example

For this example, we'll use the same dataset as the one-way analysis.

As we can see the p-value for gender is much lower than the level of significance but the p-value for Diet is higher. Therefore, there is no association between the pre weight and the diet but there is an association between pre weight and gender.

We have one last value which is the interaction between gender and Diet and their p value is lower than the level of significance. The interaction between them is statistically significant.

```
model2=ols('preweight~gender+Diet+gender*Diet', data=data1).fit()
anova2way= sm.stats.anova_lm(model2, type=2)
print(anova2way)
```

	df	sum_sq	mean_sq	F	PR(>F)
gender	2.0	2815.560416	1407.780208	33.887387	4.056530e-11
Diet	1.0	3.757281	3.757281	0.090230	7.647389e-01
gender:Diet	2.0	981.101229	490.550615	11.780414	3.681700e-05
Residual	73.0	3039.807739	41.641202	NaN	NaN

MANOVA

MANOVA is just an ANOVA with several dependent variables. It's similar to many other tests and experiments in that its purpose is to find out if the response variable (i.e. your dependent variable) is changed by manipulating the independent variable. The test helps to answer many research questions, including:

Do changes to the independent variables have statistically significant effects on dependent variables?

What are the interactions among dependent variables?

What are the interactions among independent variables?

MANOVA Example

Suppose you wanted to find out if a difference in textbooks affected students' scores in math and science. Improvements in math and science means that there are two dependent variables, so a MANOVA is appropriate.

An ANOVA will give you a single (univariate) f-value while a MANOVA will give you a multivariate F value. MANOVA tests the multiple dependent variables by creating new, artificial, dependent variables that maximize group differences. These new dependent variables are linear combinations of the measured dependent variables.

Interpreting the MANOVA results

If the multivariate F value indicates the test is statistically significant, this means that something is significant. In the above example, you would not know if math scores have improved, science scores have improved (or both). Once you have a significant result, you would then have to look at each individual component (the univariate F tests) to see which dependent variable(s) contributed to the statistically significant result.

Advantages

MANOVA enables you to test multiple dependent variables.

MANOVA can protect against Type I errors.

Disadvantages

MANOVA is many times more complicated than ANOVA, making it a challenge to see which independent variables are affecting dependent variables.

One degree of freedom is lost with the addition of each new variable.

The dependent variables should be uncorrelated as much as possible. If they are correlated, the loss in degrees of freedom means that there isn't much advantages in including more than one dependent variable on the test.

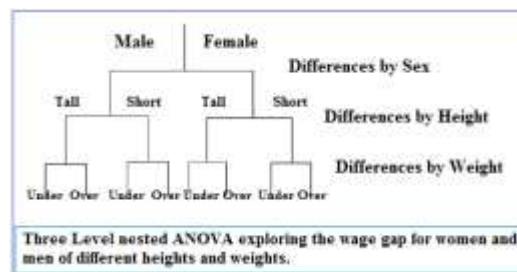
Nested ANOVA

A nested ANOVA (also called a hierarchical ANOVA) is an extension of a simple ANOVA for experiments where each group is divided into two or more random subgroups. It tests to see if there is variation between groups, or within nested subgroups of the attribute variable. You should use nested ANOVA when you have:

- One measurement variable,
- Two or more nested nominal variables (factors).

Example

Let's say you wanted to investigate the wage gap between men and women. You also think that height affects wages (which is true — see The Atlantic's story on Why Tall People Make More) as does obesity (also true: see Forbes' story The Price of Obesity). Your factors or levels (sex, height, weight) are nested within each other. For example, "weight" is not a standalone factor — it's nested under male/female. The following image shows the hierarchical model:



20. Estimation

The process of extrapolating information regarding the population from a sample is called Estimation. As it's impossible to collect information from every single member of the population we instead gather information from a sample and work our way to estimate the information about the population. And in this process, we use something called an "Estimator" to generate the estimation.

When a value is calculated for the entire population, it's called a parameter and the corresponding term for a subset of the population also known as the sample is called a statistic.

Is my estimator biased or unbiased?

Variance is a commonly used estimator which is used to determine the spread in the data usually given by the following formula:

$$S^2 = \frac{1}{n} \sum (x_i - \bar{x})^2$$

But this formula of variance tends to underestimate the value of variance when we move to a larger sample (or population). In other words, it tends to be biased. Bias is nothing but the difference between the expected value and actual value. When this bias equals 0 we say that the estimator is unbiased. The above formula yields a non-zero bias

$$S_{n-1}^2 = \frac{1}{n-1} \sum (x_i - \bar{x})^2$$

This formula yields a bias that is equal to zero. It also doesn't underestimate population variance. Intuitively, since the denominator is a smaller number now, the value of variance is larger, and hence for a larger sample (or the population), we naturally expect a larger variance.

The sample mean is always an unbiased estimator of the population mean. That is because the expected value of mean equals the actual value or true mean of the population. Some samples might have a larger mean than the population mean and some might have a sample mean lower than the population mean. However, when this process is repeated over many iterations and the average of the estimates is calculated over these iterations, the mean of these sampling experiments will eventually equal the population mean

How do we determine the best estimator?

That really depends on whether we are trying to minimize the error or maximize the chance of getting the right answer.

If we are trying to minimize the error, we use something called Mean Squared Error or Root Mean Squared Error. In a real experiment, the process of calculating the estimator is iterated many times over numerous different samples. In the absence of an outlier, the sample mean, 'xbar', minimizes the Mean Squared Error (MSE) :

$$MSE = \frac{1}{m} \sum (\bar{x} - \mu)^2$$

where 'm' is the number of iterations. RMSE is nothing but the Square root of MSE.

21. Confidence intervals

As it sounds, the confidence interval **is a range of values**. In the ideal condition, it should contain the best estimate of a statistical parameter. It is expressed as a percentage. 95% confidence interval is the most common. You can use other values like 97%, 90%, 75%, or even 99% confidence interval if your research demands. Let's understand it by an example:

Here is a statement:

"In a sample of 659 parents with toddlers, about 85%, stated they use a car seat for all travel with their toddler. From these results, a 95% confidence interval was provided, going from about 82.3% up to 87.7%."

This statement means, we are 95% certain that the population proportion who use a car seat for all travel with their toddler will fall between 82.3% and 87.7%. If we take a different sample or a subsample of these 659 people, 95% of the time, the percentage of the population who use a car seat in all travel with their toddlers will be in between 82.3% and 87.7%.

Remember, 95% confidence interval does not mean 95% probability

The conditions to calculate a valid confidence interval for a proportion are the following:

- **The sample must be random.**
- A normal distribution can approximate the sampling distribution of the sample proportions. The rule of thumb is that you need to have at least 10 successes and 10 failures.
- The samples are required to be independent. The rule of thumb is that if you are sampling without replacement, your sample size should be less than 10% of the population size.

- But if the sample size is large enough (30 or more) normal distribution is not necessary.

How to Calculate the Confidence Interval

The calculation of the confidence interval involves the best estimate which is obtained by the sample and a margin of error. So, we take the best estimate and add a margin of error to it. Here is the formula for the confidence interval and the margin of error →

$$\text{Best Estimate} \pm \text{Margin of Error}$$

$$\text{Margin of Error} = z * \text{Estimated SE}$$

Combining these two formulas above, we can elaborate the **formula for CI** as follows:

$$\text{Population Proportion or Mean} \pm z - \text{score} * \text{Standard Error}$$

Population proportion or the mean is calculated from the sample. In the example of “the parents with toddlers”, the best estimate or the population proportion of parents that uses car seats in all travel with their toddlers is 85%. So, the best estimate (population proportion) is 85. z-score is fixed for the confidence level (CL).

A z-score for a 95% confidence interval for a large enough sample size(30 or more) is 1.96.

Here are the z-scores for some commonly used confidence levels:

Confidence %	z
75%	1.15
90%	1.64
95%	1.96
97%	2.17
99%	2.57
99.90%	3.29

The method to calculate the standard error is different for population proportion and mean. The formula to calculate standard error of population proportion is:

$$\text{Standard Error For Population Proportion} = \sqrt{(\text{Population Proportion} * \frac{(1 - \text{Population Proportion})}{\text{Number Of Observations}})}$$

The formula to calculate the standard error of the sample mean is:

$$\text{Standard Error For Mean} = \frac{\text{Standard Deviation}}{\sqrt{\text{Number Of Observations}}}$$

Confidence interval in Python

We are going to construct a CI for the female population proportion that has heart disease. We do not need all the columns in the dataset. We will only use the ‘target’ column as that contains if a person has heart disease or not and the Sex1 column we just created. Make a DataFrame with only these two columns and drop all the null values

We need the number of females who have heart disease. The line of code below will give the number of males and females with heart disease and with no heart disease.

```
df = pd.read_csv('heart.csv')
df['Sex1'] = df.sex.replace({1: "Male", 0: "female"})
dx = df[['target', 'Sex1']].dropna()
pd.crosstab([dx.target, dx.Sex1])
```

✓ 0.8s

Sex1	Female	Male
target		
0	86	413
1	226	300

Now let's calculate the confident intervals

```
p_fm = 25/(72+25) # a female population proportion with heart disease.
n = 72+25 # size of the female population
se_female = np.sqrt(p_fm * (1 - p_fm) / n) # standard error
z_score = 1.96 # z score for 95% confident interval
lcb = p_fm - z_score * se_female # lower limit of the CI
ucb = p_fm + z_score * se_female # upper limit of the CI
print('Lower limit confident interval:', lcb)
print('Upper limit confident interval:', ucb)
import statsmodels.api as sm # with stats model
print(sm.stats.proportion_confint(n * p_fm, n))
```

✓ 0.4s

Lower limit confident interval: 0.17068878406861235
Upper limit confident interval: 0.3447751334571608
(0.17069038350737342, 0.3447753340183997)

CI for the Difference in Population Proportion

Is the population proportion of females with heart disease the same as the population proportion of males with heart disease? If they are the same, then the difference in both the population proportions will be zero.

We will calculate a confidence interval of the difference in the population proportion of females and males with heart disease.

Calculate the male population proportion with heart disease and standard error using the same procedure. I'm going to jump all the steps and show you the standard error for the male population which is 0.034. Now calculate the difference in standard error. →

$$\sqrt{SE_1^2 + SE_2^2}$$

Let's use this formula to calculate the difference in the standard error of male and female population with heart disease and then use this standard error to **calculate the difference in the population proportion of males and females with heart disease** and construct the CI of the difference.

```
se_male=0.034
se_diff = np.sqrt(se_female**2 + se_male**2)

d = 0.55 - 0.26 #The male population proportion with heart disease is 0.55 and female is 0.26
z_score = 1.96
lcb = d - z_score * se_diff #lower limit of the CI
ucb = d + z_score * se_diff #upper limit of the CI
print('Lower limit confident interval:', lcb)
print('Upper limit confident interval:', ucb)
```

✓ 0.4s

Lower limit confident interval: 0.18037607988738072
Upper limit confident interval: 0.3996239201126194

The CI is 0.18 and 0.4. This range does not have 0 in it. Both the numbers are above zero. So, We cannot make any conclusion that the population proportion of females with heart disease is the same as the population proportion of males with heart disease. If the CI would be -0.12 and 0.1, we could say that the male and female population proportion with heart disease is the same.

If you want to learn more about confidence interval you can look to the calculation of CI of mean or the calculation of CI of The Difference in Mean.

22. Maximum Likelihood Estimation (MLE)

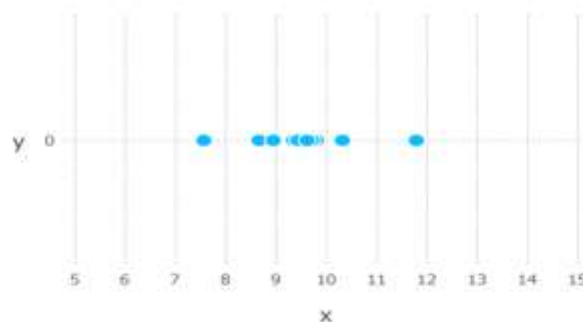
Often in machine learning we use a model to describe the process that results in the data that are observed. For example, we may use a random forest model to classify whether customers may cancel a subscription from a service (known as churn modelling) or we may use a linear model to predict the revenue that will be generated for a company depending on how much they may spend on advertising (this would be an example of linear regression). Each model contains its own set of parameters that ultimately defines what the model looks like.

For a linear model we can write this as $y = mx + c$. In this example x could represent the advertising spend and y might be the revenue generated. m and c are parameters for this model. Different values for these parameters will give different lines (see the next figure).

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximize the likelihood that the process described by the model produced the data that were actually observed.

The above definition may still sound a little cryptic so let's go through an example to help understand this.

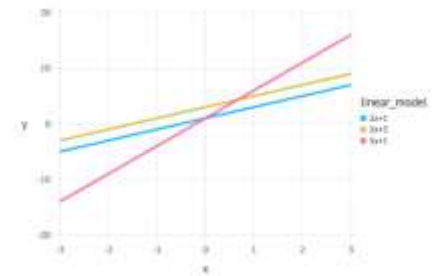
Let's suppose we have observed 10 data points from some process. For example, each data point could represent the length of time in seconds that it takes a student to answer a specific exam question. These 10 data points are shown in the figure below

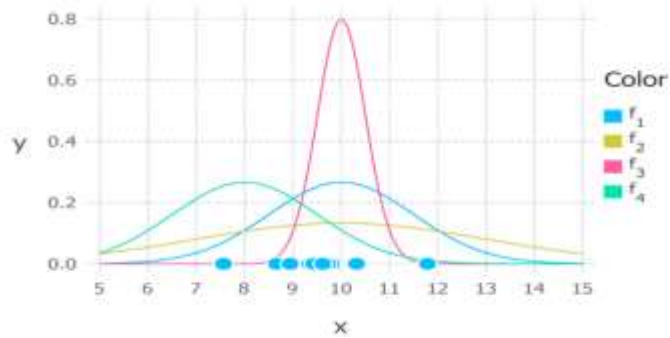


We first have to decide which model we think best describes the process of generating the data. This part is very important. At the very least, we should have a good idea about which model to use. This usually comes from having some domain expertise but we won't discuss this here.

For these data we'll assume that the data generation process can be adequately described by a Gaussian (normal) distribution. Visual inspection of the figure above suggests that a Gaussian distribution is plausible because most of the 10 points are clustered in the middle with few points scattered to the left and the right. (Making this sort of decision on the fly with only 10 data points is ill-advised but given that I generated these data points we'll go with it).

Recall that the Gaussian distribution has 2 parameters. The mean, μ , and the standard deviation, σ . Different values of these parameters result in different curves (just like with the straight lines above). We want to know which curve was most likely responsible for creating the data points that we observed? (See figure below). **Maximum likelihood estimation is a method that will find the values of μ and σ that result in the curve that best fits the data. For which parameter value does the observed data have the biggest probability?**





The true distribution from which the data were generated was $f_1 \sim N(10, 2.25)$, which is the blue curve in the figure above.

Example

Regression on Normally Distributed Data

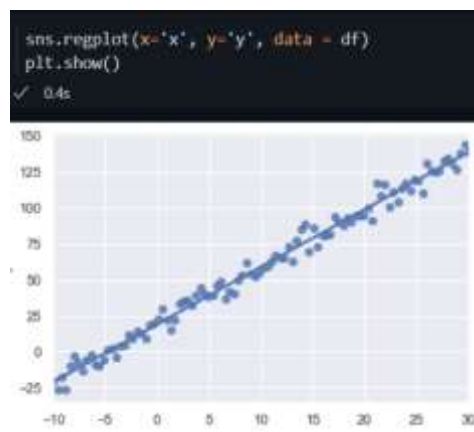
Here, we perform simple linear regression on synthetic data. The data is ensured to be normally distributed by incorporating some random Gaussian noises. Data can be said to be normally distributed if its residual follows the normal distribution.

Generate some synthetic data based on the assumption of Normal Distribution.

```
# generate an independent variable
x = np.linspace(-10, 30, 100)
# generate a normally distributed residual
e = np.random.normal(10, 5, 100)
# generate ground truth
y = 10 + 4*x + e
df = pd.DataFrame({'x':x, 'y':y})
df.head()
```

	x	y
0	-10.000000	-20.423714
1	-9.595960	-26.559443
2	-9.191919	-17.950011
3	-8.787879	-26.688599
4	-8.383838	-9.890875

Visualize the synthetic data on Seaborn's regression plot.



The data is normally distributed, and the output variable is a continuously varying number. Hence, we can use the Ordinary Least Squares (OLS) method to determine the model parameters and use them as a benchmark to evaluate the Maximum Likelihood Estimation approach. Apply the OLS algorithm to the synthetic data and find the model parameters

```
features = api.add_constant(df.x)
model = api.OLS(y, features).fit()
model.summary()
```

Dep. Variable:	y	R-squared:	0.991
Model:	OLS	Adj. R-squared:	0.991
Method:	Least Squares	F-statistic:	1.118e+04
Date:	Fri, 16 Apr 2021	Prob (F-statistic):	8.44e-103
Time:	12:02:58	Log-Likelihood:	-289.15
No. Observations:	100	AIC:	582.3
Df Residuals:	98	BIC:	587.5
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	19.8108	0.580	34.144	0.000	18.659	20.962
x	3.9924	0.038	106.714	0.000	3.917	4.067

Omnibus:	0.031	Durbin-Watson:	2.045
Prob(Omnibus):	0.985	Jarque-Bera (JB):	0.066
Skew:	0.033	Prob(JB):	0.972
Kurtosis:	2.904	Cond. No.	20.3

We get the intercept and regression coefficient values of the simple linear regression model. Further, we can derive the standard deviation of the normal distribution with the following codes.

```
res = model.resid
standard_dev = np.std(res)
standard_dev
✓ 0.3s
5.289562802122958
```

As we have solved the simple linear regression problem with an OLS model, it is time to solve the same problem by formulating it with Maximum Likelihood Estimation.

Define a user-defined Python function that can be iteratively called to determine the negative log-likelihood value. The key idea of formulating this function is that it must contain two elements: the first is the model building equation (here, the simple linear regression). The second is the logarithmic value of the probability density function (here, the log PDF of normal distribution). Since we need negative log-likelihood, it is obtained just by negating the log-likelihood.

Minimize the negative log-likelihood of the generated data using the minimize method available with SciPy's optimize module.

```

# MLE function
# ml modeling and neg ll calculation
def MLE_Norm(parameters):
    # extract parameters
    const, beta, std_dev = parameters
    # predict the output
    pred = const + beta*x
    # Calculate the log-likelihood for normal distribution
    ll = np.sum(stats.norm.logpdf(y, pred, std_dev))
    # Calculate the negative log-likelihood
    neg_ll = -1*ll
    return neg_ll

# minimize arguments: function, initial_guess_of_parameters, method
mle_model = minimize(MLE_Norm, np.array([2,2,2]), method='L-BFGS-B')
mle_model

✓ 0.1s
fun: 308.46741295595257
hess_inv: <3x3 lbfgsInvHessProduct with dtype=float64>
jac: array([-2.27373657e-05, -2.55795386e-04,  2.27373677e-05])
message: 'CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH'
nfev: 220
nit: 34
njev: 55
status: 0
success: True
x: array([19.0073068 ,  3.99519348,  5.28956606])

```

The MLE approach arrives at the final optimal solution after 35 iterations. The model's parameters, the intercept, the regression coefficient and the standard deviation are well matching to those obtained using the OLS approach.

Here comes the big question. If the OLS approach provides the same results without any tedious function formulation, why do we go for the MLE approach? The answer is that the OLS approach is completely problem-specific and data-oriented. It cannot be used for a different kind of problem or a different data distribution. **On the other hand, the MLE approach is a general template for any kind of problem.** With expertise in Maximum Likelihood Estimation, users can formulate and solve their own machine learning problems with raw data in hand.

23. Kernel Density Estimate

Density Estimation¶

Density estimation walks the line between unsupervised learning, feature engineering, and data modeling. Some of the most popular and useful density estimation techniques are mixture models such as Gaussian Mixtures (`sklearn.mixture.GaussianMixture`), and neighbor-based approaches such as the kernel density estimate (`sklearn.neighbors.KernelDensity`). Gaussian Mixtures are discussed more fully in the context of clustering, because the technique is also useful as an unsupervised clustering scheme.

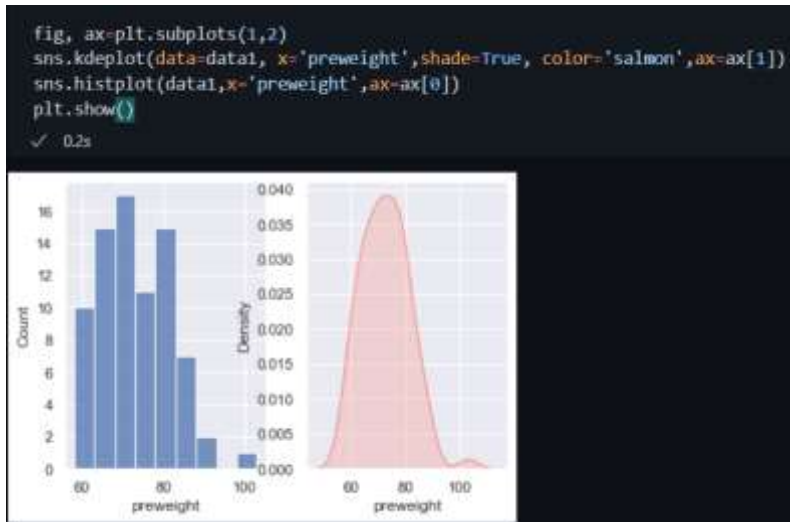
Density estimation is a very simple concept, and most people are already familiar with one common density estimation technique: the histogram.

Kernel Density Estimation

Kernel density estimation in scikit-learn is implemented in the `sklearn.neighbors.KernelDensity` estimator, which uses the Ball Tree or KD Tree for efficient queries (see Nearest Neighbors for a discussion of these). Though the above example uses

a 1D data set for simplicity, kernel density estimation can be performed in any number of dimensions, though in practice the curse of dimensionality causes its performance to degrade in high dimensions.

The kernel density estimator can be used with any of the valid distance metrics (see `sklearn.neighbors.DistanceMetric` for a list of available metrics), though the results are properly normalized only for the Euclidean metric. One particularly useful metric is the Haversine distance which measures the angular distance between points on a sphere.



24. Regression

What is regression?

Regression analysis is defined in Wikipedia as:

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features').

The terminology you will often listen related to regression analysis is:

- Dependent variable or target variable: Variable to predict.
- Independent variable or predictor variable: Variables to estimate the dependent variable.
- Outlier: Observation that differs significantly from other observations. It should be avoided since it may hamper the result.
- Multicollinearity: Situation in which two or more independent variables are highly linearly related.
- Homoscedasticity or homogeneity of variance: Situation in which the error term is the same across all values of the independent variables.

Regression analysis is primarily used for two distinct purposes. First, it is widely used for prediction and forecasting, which overlaps with the field of machine learning. Second, it is also used to infer causal relationships between independent and dependent variables.

I'm going to explain it deep in the machine learning chapter

25. Covariance and Correlation

Covariance and correlation are two key concepts in statistics and probability theory, but what do they actually mean in practice? More importantly, when do we use them?

Put simply: we use both concepts to understand relationships between data variables and values. We can quantify the relationship between variables and then use these learnings to either select, add or alter variables for predictive modeling, insight generation or even storytelling using data. Thus, both correlation and covariance have high utility in machine learning and data analysis.

Put simply, both covariance and correlation measure the relationship and the dependency between two variables. **Covariance indicates the direction of the linear relationship between variables while correlation measures both the strength and direction of the linear relationship between two variables.** Correlation is a function of the covariance.

What sets these two concepts apart is the fact that correlation values are standardized whereas covariance values are not.

Types of correlations

- Pearson's correlation:

```
from scipy.stats import pearsonr
print(data1[['preweight', 'weight6weeks']].head(3))
#pearson correlation
corr, _ = pearsonr(data1['weight6weeks'], data1['preweight'])
print('Pearsons correlation: %.3f' % corr)
✓ 0.4s
```

	preweight	weight6weeks
0	60	60.0
1	103	103.0
2	58	54.2

Pearsons correlation: 0.958

- Spearman's correlation:

```
from scipy.stats import spearmanr
#spearman's correlation
corr, _ = spearmanr(data1['weight6weeks'], data1['preweight'])
print('Pearsons correlation: %.3f' % corr)
✓ 0.2s
```

Pearsons correlation: 0.957

26. Causation

While causation and correlation can exist at the same time, correlation does not imply causation. **Causation explicitly applies to cases where action A causes outcome B.** On the other hand, correlation is simply a relationship. Action A relates to Action B—but one event doesn't necessarily cause the other event to happen.

Correlation and causation are often confused because the human mind likes to find patterns even when they do not exist. We often fabricate these patterns when two variables appear to be so closely associated that one is dependent on the other. That would imply a cause and effect relationship where the dependent event is the result of an independent event.

However, we cannot simply assume causation even if we see two events happening, seemingly together, before our eyes. One, our observations are purely anecdotal. Two, there are so many other possibilities for an association, including:

- The opposite is true: B actually causes A.
- The two are correlated, but there's more to it: A and B are correlated, but they're actually caused by C.

- There's another variable involved: A does cause B—as long as D happens.
- There is a chain reaction: A causes E, which leads E to cause B (but you only saw that A causes B from your own eyes).

27. Distance Measures

a. Euclidean distance

it is a distance measure that best can be explained as the length of a segment connecting two points.

- Disadvantages

Although it is a common distance measure, Euclidean distance is not scale in-variant which means that distances computed might be skewed depending on the units of the features. Typically, one needs to normalize the data before using this distance measure.

Moreover, as the dimensionality increases of your data, the less useful Euclidean distance becomes. This has to do with the curse of dimensionality which relates to the notion that higher-dimensional space does not act as we would, intuitively, expect from 2- or 3-dimensional space. For a good summary, see this post.

- Use Cases

Euclidean distance works great when you have low-dimensional data and the magnitude of the vectors is important to be measured. Methods like kNN and HDBSCAN show great results out of the box if Euclidean distance is used on low-dimensional data.

Although many other measures have been developed to account for the disadvantages of Euclidean distance, it is still one of the most used distance measures for good reasons. It is incredibly intuitive to use, simple to implement and shows great results in many use-cases.

b. Cosine Similarity

Cosine similarity has often been used as a way to counteract Euclidean distance's problem with high dimensionality. The cosine similarity is simply the cosine of the angle between two vectors. It also has the same inner product of the vectors if they were normalized to both have length one.

Two vectors with exactly the same orientation have a cosine similarity of 1, whereas two vectors diametrically opposed to each other have a similarity of -1. Note that their magnitude is not of importance as this is a measure of orientation.

- Disadvantages

One main disadvantage of cosine similarity is that the magnitude of vectors is not taken into account, merely their direction. In practice, this means that the differences in values are not fully taken into account. If you take a recommender system, for example, then the cosine similarity does not take into account the difference in rating scale between different users.

- Use Cases

We use cosine similarity often when we have high-dimensional data and when the magnitude of the vectors is not of importance. For text analyses, this measure is quite frequently used when the data is represented by word counts. For example, when a word occurs more frequently in one document over another this does not necessarily mean that one document is more related to that word. It could be the case that documents have uneven lengths and the magnitude of the count is of less importance. Then, we can best be using cosine similarity which disregards magnitude

IV. Machine Learning

1. What is Machine Learning?

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

UC Berkeley breaks out the learning system of a machine learning algorithm into three main parts.

- A Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.
- An Error Function: An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
- An Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

2. Numerical, Categorical and Ordinal Variables

Numerical data

These data have meaning as a measurement, such as a person's height, weight, IQ, or blood pressure; or they're a count, such as the number of stock shares a person owns, how many teeth a dog has, or how many pages you can read of your favorite book before you fall asleep. (Statisticians also call numerical data quantitative data.)

Numerical data can be further broken into two types: discrete and continuous.

- Discrete data represent items that can be counted; they take on possible values that can be listed out. The list of possible values may be fixed (also called finite); or it may go from 0, 1, 2, on to infinity (making it countably infinite). For example, the number of heads in 100 coin flips takes on values from 0 through 100 (finite case), but the number of flips needed to get 100 heads takes on values from 100 (the fastest scenario) on up to infinity (if you never get to that 100th heads). Its possible values are listed as 100, 101, 102, 103 . . . (representing the countably infinite case).
- Continuous data represent measurements; their possible values cannot be counted and can only be described using intervals on the real number line. For example, the exact amount of gas purchased at the pump for cars with 20-gallon tanks would be continuous data from 0 gallons to 20 gallons, represented by the interval $[0, 20]$, inclusive. You might pump 8.40 gallons, or 8.41, or 8.414863 gallons, or any possible number from 0 to 20. In this way, continuous data can be thought of as being uncountably infinite. For ease of recordkeeping, statisticians usually pick some point in the number to round off. Another example would be that the lifetime of a C battery can be anywhere from 0 hours to an infinite number of hours (if it lasts forever), technically, with all possible values in

between. Granted, you don't expect a battery to last more than a few hundred hours, but no one can put a cap on how long it can go (remember the Energizer Bunny?).

Categorical data

Categorical data represent characteristics such as a person's gender, marital status, hometown, or the types of movies they like. Categorical data can take on numerical values (such as "1" indicating male and "2" indicating female), but those numbers don't have mathematical meaning. You couldn't add them together, for example. (Other names for categorical data are qualitative data, or Yes/No data.)

Ordinal data

Ordinal data mixes numerical and categorical data. The data fall into categories, but the numbers placed on the categories have meaning. For example, rating a restaurant on a scale from 0 (lowest) to 4 (highest) stars gives ordinal data.

Ordinal data are often treated as categorical, where the groups are ordered when graphs and charts are made. However, unlike categorical data, the numbers do have mathematical meaning. For example, if you survey 100 people and ask them to rate a restaurant on a scale from 0 to 4, taking the average of the 100 responses will have meaning. This would not be the case with categorical data.

3. Supervised Learning

What is supervised learning?

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its **use of labeled datasets to train algorithms that to classify data or predict outcomes accurately**. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately, which occurs as part of the cross validation process. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox.

How supervised learning works

Supervised learning uses a training set to teach models to yield the desired output. This training dataset includes inputs and correct outputs, which allow the model to learn over time. The algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized.

Supervised learning can be separated into two types of problems when data mining—classification and regression:

- **Classification** uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbor, and random forest, which are described in more detail below.
- **Regression** is used to understand the relationship between dependent and independent variables. It is commonly used to make projections, such as for sales revenue for a given business. Linear regression, logistical regression, and polynomial regression are popular regression algorithms.

4. Unsupervised Learning

What is unsupervised learning?

Unsupervised learning, also known as unsupervised machine learning, uses **machine learning algorithms to analyze and cluster unlabeled datasets**. These algorithms **discover hidden patterns or data groupings** without the need for human

intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

Common unsupervised learning approaches

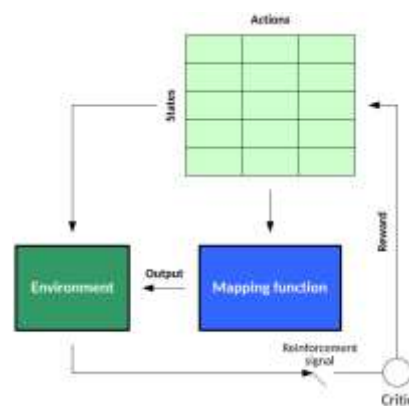
Unsupervised learning models are utilized for **three main tasks—clustering, association, and dimensionality reduction**. Below we'll define each learning method and highlight common algorithms and approaches to conduct them effectively.

- **Clustering:** This is a data mining technique which groups unlabeled data based on their similarities or differences. Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information. Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic
- **Association Rules:** An association rule is a rule-based method for finding relationships between variables in a given dataset. These methods are frequently used for market basket analysis, allowing companies to better understand relationships between different products. Understanding consumption habits of customers enables businesses to develop better cross-selling strategies and recommendation engines. Examples of this can be seen in Amazon's "Customers Who Bought This Item Also Bought" or Spotify's "Discover Weekly" playlist. While there are a few different algorithms used to generate association rules, such as Apriori, Eclat, and FP-Growth, the Apriori algorithm is most widely used.
- **Dimensionality reduction:** While more data generally yields more accurate results, it can also impact the performance of machine learning algorithms (e.g. overfitting) and it can also make it difficult to visualize datasets. Dimensionality reduction is a technique used when the number of features, or dimensions, in a given dataset is too high. It reduces the number of data inputs to a manageable size while also preserving the integrity of the dataset as much as possible. It is commonly used in the preprocessing data stage, and there are a few different dimensionality reduction methods that can be used, such as:

5. Reinforcement Learning

Reinforcement learning is an interesting learning model, with the ability not just to learn how to map an input to an output but to map a series of inputs to outputs with dependencies (Markov decision processes, for example). Reinforcement learning exists in the context of states in an environment and the actions possible at a given state. During the learning process, the algorithm randomly explores the state–action pairs within some environment (to build a state–action pair table), then in practice of the learned information exploits the state–action pair rewards to choose the best action for a given state that lead to some goal state

Consider a simple agent that plays blackjack. The states represent the sum of the cards for the player. The actions represent what a blackjack-playing agent may do — in this case, hit or stand. Training an agent to play blackjack would involve many hands of poker, where reward for a given state–action nexus is given for winning or losing. For example, the value for a state of 10 would be 1.0 for hit and 0.0 for stand (indicating that hit is the optimal choice). For state 20, the learned reward would likely be 0.0 for hit and 1.0 for stand. For a less-straightforward hand, a state of 17 may have action values of 0.95 stand and 0.05 hit. This agent would then probabilistically stand 95 percent of the time and hit 5 percent of the time. These rewards would be leaned over many hands of poker, indicating the best choice for a given state (or hand).



6. Training and Test Data

To introduce this idea, you can think of dividing your data set into two subsets:

- training set—a subset to train a model.
- test set—a subset to test the trained model.

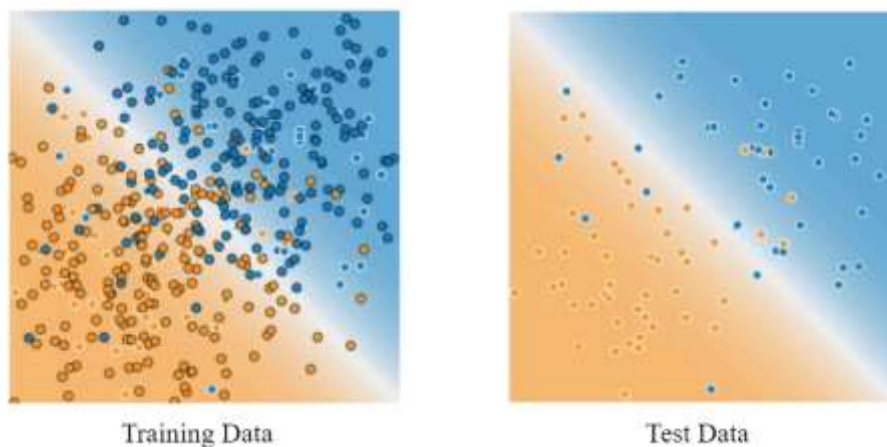
You could imagine slicing the single data set as follows:



Make sure that your test set meets the following two conditions:

- Is large enough to yield statistically meaningful results.
- Is representative of the data set as a whole. In other words, don't pick a test set with different characteristics than the training set.

Assuming that your test set meets the preceding two conditions, your goal is to create a model that generalizes well to new data. Our test set serves as a proxy for new data. For example, consider the following figure. Notice that the model learned for the training data is very simple. This model doesn't do a perfect job—a few predictions are wrong. However, this model does about as well on the test data as it does on the training data. In other words, this simple model does not overfit the training data.



7. Prediction

Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

The word “prediction” can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you’re using machine learning to determine the next best action in a marketing campaign. Other times, though, the “prediction” has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you’re making an educated guess about whether it was legitimate, allowing you to take the appropriate action.

Why are Predictions Important?

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more. These provide the business with insights that result in tangible business value. For example, if a model predicts a customer is likely to churn, the business can target them with specific communications and outreach that will prevent the loss of that customer.

8. Lift

9. Bias & Variance

What is bias?

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

What is variance?

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data.

10. Overfitting & Underfitting

Overfitting and underfitting are the two biggest causes for poor performance of machine learning algorithms.

Statistical Fit

In statistics, a fit refers to how well you approximate a target function.

This is good terminology to use in machine learning, because supervised machine learning algorithms seek to approximate the unknown underlying mapping function for the output variables given the input variables.

Statistics often describe the goodness of fit which refers to measures used to estimate how well the approximation of the function matches the target function.

Some of these methods are useful in machine learning (e.g. calculating the residual errors), but some of these techniques assume we know the form of the target function we are approximating, which is not the case in machine learning.

If we knew the form of the target function, we would use it directly to make predictions, rather than trying to learn an approximation from samples of noisy training data.

Overfitting in Machine Learning

Overfitting refers to a model that models the training data too well.

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model's ability to generalize.

Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns.

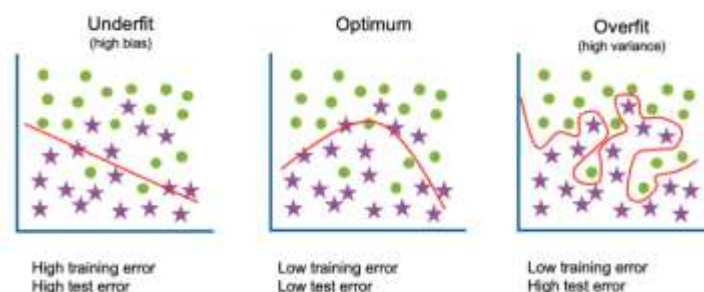
For example, decision trees are a nonparametric machine learning algorithm that is very flexible and is subject to overfitting training data. This problem can be addressed by pruning a tree after it has learned in order to remove some of the detail it has picked up.

Underfitting in Machine Learning

Underfitting refers to a model that can neither model the training data nor generalize to new data.

An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting.

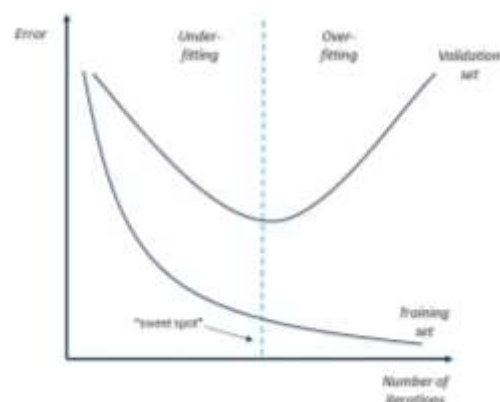


In both scenarios, the model cannot establish the dominant trend within the training dataset. As a result, underfitting also generalizes poorly to unseen data. However, unlike overfitting, underfitted models experience high bias and less variance within their predictions. This illustrates the bias-variance tradeoff, which occurs when as an underfitted model shifted to an overfitted state. As the model learns, its bias reduces, but it can increase in variance as it becomes overfitted. When fitting a model, the goal is to find the “sweet spot” in between underfitting and overfitting, so that it can establish a dominant trend and apply it broadly to new datasets.

How to detect overfit models

To understand the accuracy of machine learning models, it's important to test for model fitness. K-fold cross-validation is one of the most popular techniques to assess accuracy of the model.

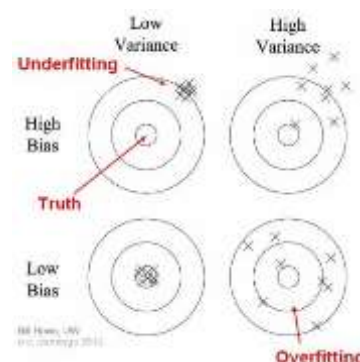
In k-folds cross-validation, data is split into k equally sized subsets, which are also called “folds.” One of the k-folds will act as the test set, also known as the holdout set or validation set, and the remaining folds will train the model. This process



repeats until each of the fold has acted as a holdout fold. After each evaluation, a score is retained and when all iterations have completed, the scores are averaged to assess the performance of the overall model.

In supervised learning, underfitting happens when a model is unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kinds of models are very simple to capture the complex patterns in data like Linear and logistic regression.

In supervised learning, overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over a noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.



Why is Bias Variance Tradeoff?

If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

How to avoid overfitting

While using a linear model helps us avoid overfitting, many real-world problems are nonlinear ones. In addition to understanding how to detect overfitting, it is important to understand how to avoid overfitting altogether. Below are a number of techniques that you can use to prevent overfitting:

- **Early stopping:** As we mentioned earlier, this method seeks to pause training before the model starts learning the noise within the model. This approach risks halting the training process too soon, leading to the opposite problem of underfitting. **Finding the “sweet spot” between underfitting and overfitting is the goal here.**
- **Train with more data:** Expanding the training set to include more data can increase the accuracy of the model by providing more opportunities to parse out the dominant relationship among the input and output variables. That said, this is a more effective method when clean, relevant data is injected into the model. Otherwise, you could just continue to add more complexity to the model, causing it to overfit.
- **Data augmentation:** While it is better to inject clean, relevant data into your training data, sometimes noisy data is added to make a model more stable. However, this method should be done sparingly.
- **Feature selection:** When you build a model, you'll have a number of parameters or features that are used to predict a given outcome, but many times, these features can be redundant to others. Feature selection is the process of identifying the most important ones within the training data and then eliminating the irrelevant or redundant ones. This is commonly mistaken for dimensionality reduction, but it is different. However, both methods help to simplify your model to establish the dominant trend in the data.
- **Regularization:** If overfitting occurs when a model is too complex, it makes sense for us to reduce the number of features. But what if we don't know which inputs to eliminate during the feature selection process? If we don't know which features to remove from our model, regularization methods can be particularly helpful. Regularization

applies a “penalty” to the input parameters with the larger coefficients, which subsequently limits the amount of variance in the model. While there are a number of regularization methods, such as L1 regularization, Lasso regularization, and dropout, they all seek to identify and reduce the noise within the data. Only for regressions.

- Ensemble methods: Ensemble learning methods are made up of a set of classifiers—e.g. decision trees—and their predictions are aggregated to identify the most popular result. The most well-known ensemble methods are bagging and boosting. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. After several data samples are generated, these models are then trained independently, and depending on the type of task—i.e. regression or classification—the average or majority of those predictions yield a more accurate estimate. This is commonly used to reduce variance within a noisy dataset.

11. Trees & Classifications

Decision Trees are a class of very powerful Machine Learning model capable of achieving high accuracy in many tasks while being highly interpretable. What makes decision trees special in the realm of ML models is really their clarity of information representation. The “knowledge” learned by a decision tree through training is directly formulated into a hierarchical structure. This structure holds and displays the knowledge in such a way that it can easily be understood, even by non-experts.

Decision Tree models are created using 2 steps: Induction and Pruning. Induction is where we actually build the tree i.e set all of the hierarchical decision boundaries based on our data. Because of the nature of training decision trees they can be prone to major overfitting. Pruning is the process of removing the unnecessary structure from a decision tree, effectively reducing the complexity to combat overfitting with the added bonus of making it even easier to interpret.

Induction

From a high level, decision tree induction goes through 4 main steps to build the tree:

1. Begin with your training dataset, which should have some feature variables and classification or regression output.
2. Determine the “best feature” in the dataset to split the data on; more on how we define “best feature” later
3. Split the data into subsets that contain the possible values for this best feature. This splitting basically defines a node on the tree i.e each node is a splitting point based on a certain feature from our data.
4. Recursively generate new tree nodes by using the subset of data created from step 3. We keep splitting until we reach a point where we have optimised, by some measure, maximum accuracy while minimising the number of splits / nodes.

Pruning

Because of the nature of training decision trees they can be prone to major overfitting. Setting the correct value for minimum number of instances per node can be challenging. Most of the time, we might just go with a safe bet and make that minimum quite small, resulting in there being many splits and a very large, complex tree. The key is that many of these splits will end up being redundant and unnecessary to increasing the accuracy of our model.

Tree pruning is a technique that leverages this splitting redundancy to remove i.e prune the unnecessary splits in our tree. From a high-level, pruning compresses part of the tree from strict and rigid decision boundaries into ones that are more smooth and generalise better, effectively reducing the tree complexity. The complexity of a decision tree is defined as the number of splits in the tree.

A simple yet highly effective pruning method is to go through each node in the tree and evaluate the effect of removing it on the cost function. If it doesn’t change much, then prune away!

12. Classification Rating

Evaluating a model is a major part of building an effective machine learning model. The most frequent classification evaluation metric that we use should be 'Accuracy'. You might believe that the model is good when the accuracy rate is 99%! However, it is not always true and can be misleading in some situations.

Confusion Matrix

Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. The confusion matrix provides a more insightful picture which is not only the performance of a predictive model, but also which classes are being predicted correctly and incorrectly, and what type of errors are being made. To illustrate, we can see how the 4 classification metrics are calculated (TP, FP, FN, TN), and our predicted value compared to the actual value in a confusion matrix is clearly presented in the below confusion matrix table.

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

The equations of 4 key classification metrics:

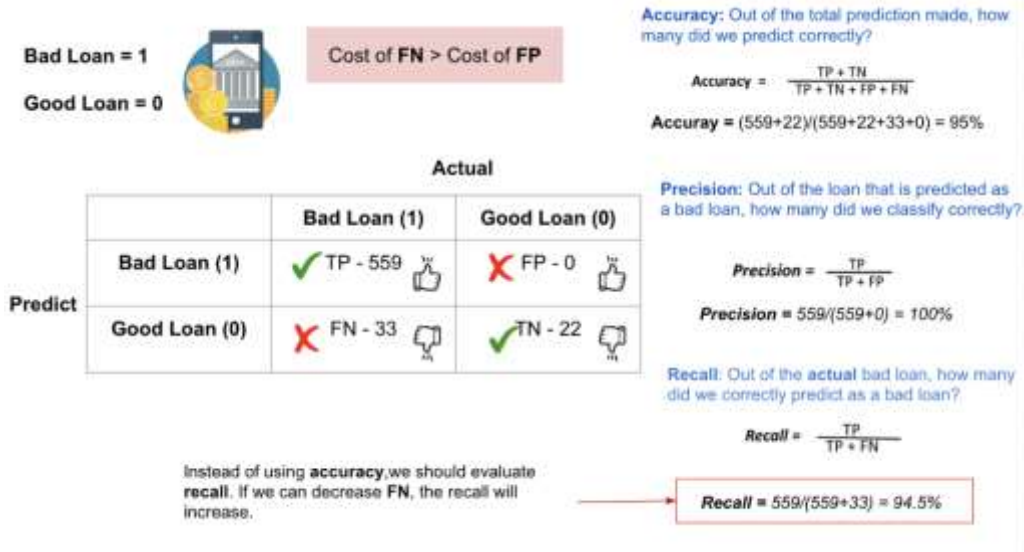
Accuracy: $ACC = \frac{TP + TN}{TP + TN + FP + FN}$	Recall: $Recall = \frac{TP}{TP + FN}$
Precision: $Precision = \frac{TP}{TP + FP}$	F₁ score: $F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$

Precision is the ratio of True Positives to all the positives predicted by the model.

- Low precision: the more False positives the model predicts, the lower the precision.

Recall (Sensitivity) is the ratio of True Positives to all the positives in your Dataset.

- Low recall: the more False Negatives the model predicts, the lower the recall.



F-Measure provides a single score that balances both the concerns of precision and recall in one number. A good F1 score means that you have low false positives and low false negatives, so you're correctly identifying real threats, and you are not disturbed by false alarms. An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0.

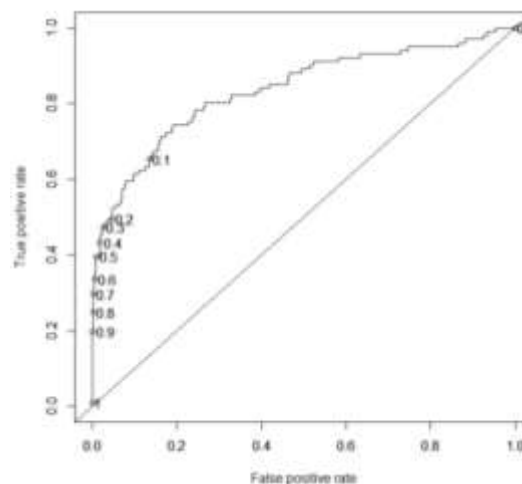
$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Decision Threshold & Receiver Operating Characteristic (ROC) curve

ROC is a major visualization technique for presenting the performance of a classification model. It summarizes the trade-off between the true positive rate (tpr) and false positive rate (fpr) for a predictive model using different probability thresholds.

$$\text{true positive rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad \text{false positive rate} = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}}$$

A ROC curve plots the true positive rate (tpr) versus the false positive rate (fpr) as a function of the model's threshold for classifying a positive. Given that c is a constant known as decision threshold, the below ROC curve suggests that by default $c=0.5$, when $c=0.2$, both tpr and fpr increase. When $c=0.8$, both tpr and fpr decrease. In general, tpr and fpr increase as c decrease. In the extreme case when $c=1$, all cases are predicted as negative; $\text{tpr}=\text{fpr}=0$. On the other hand, when $c=0$, all cases are predicted as positive; $\text{tpr}=\text{fpr}=1$.



We can assess the performance of the model by the area under the ROC curve (AUC). As a rule of thumb,

0.9–1=excellent; 0.8–0.9=good; 0.7–0.8=fair; 0.6–0.7=poor; 0.50–0.6=fail.

The relation between Sensitivity, Specificity, FPR, and Threshold.

Sensitivity and Specificity are inversely proportional to each other. So when we increase Sensitivity, Specificity decreases, and vice versa. When we decrease the threshold, we get more positive values thus it increases the sensitivity and decreasing the specificity.

Similarly, when we increase the threshold, we get more negative values thus we get higher specificity and lower sensitivity.

As we know FPR is $1 - \text{specificity}$. So when we increase TPR, FPR also increases and vice versa.

13. Classification Algorithms

a. Logistic Regression with Example (Binary)

It is a classification algorithm in machine learning that uses one or more independent variables to determine an outcome. The outcome is measured with a dichotomous variable meaning it will have only two possible outcomes.

The goal of logistic regression is to find a best-fitting relationship between the dependent variable and a set of independent variables. It is better than other binary classification algorithms like nearest neighbor since it quantitatively explains the factors leading to classification.

Advantages and Disadvantages

- Logistic regression is specifically meant for classification, it is useful in understanding how a set of independent variables affect the outcome of the dependent variable.
- The main disadvantage of the logistic regression algorithm is that it only works when the predicted variable is binary, it assumes that the data is free of missing values and assumes that the predictors are independent of each other.

Use Cases

- Identifying risk factors for diseases
- Word classification
- Weather Prediction
- Voting Applications

Example:

First, we have our dataset, and we are going to split it following the pareto principle. We use StandardScaler to scale the data and work more efficiently.

```

#pareto principle
x_train,x_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0.2)

pipeline=Pipeline(steps=[('scaler', StandardScaler()),('logreg',LogisticRegression())])
logreg_scaled=pipeline.fit(X_train,y_train)

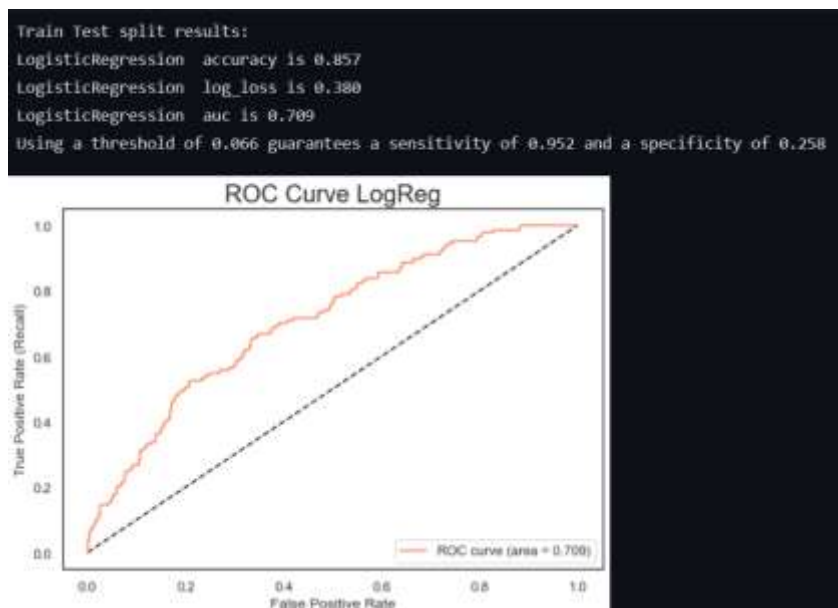
y_pred=logreg_scaled.predict(X_test)
y_pred_proba = logreg_scaled.predict_proba(X_test)[:, 1]
fpr, tpr, thr = roc_curve(y_test, y_pred_proba)
print('Train Test split results:')
print('LogisticRegression', " accuracy is %2.3f" % accuracy_score(y_test, y_pred))
print('LogisticRegression', " log_loss is %2.3f" % log_loss(y_test, y_pred_proba))
print('LogisticRegression', " auc is %2.3f" % auc(fpr, tpr))

idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which the sensibility > 0.95
print("Using a threshold of %3f" % thr[idx] + " guarantees a sensitivity of %3f" % tpr[idx] +
      "and a specificity of %3f" % (1-fpr[idx]) )

plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr, tpr, colour='coral', label='ROC curve (area = %0.3f)' % auc(fpr, tpr))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate (Recall)')
plt.title('ROC Curve LogReg', fontsize=20)
plt.legend(loc="lower right")
plt.show()

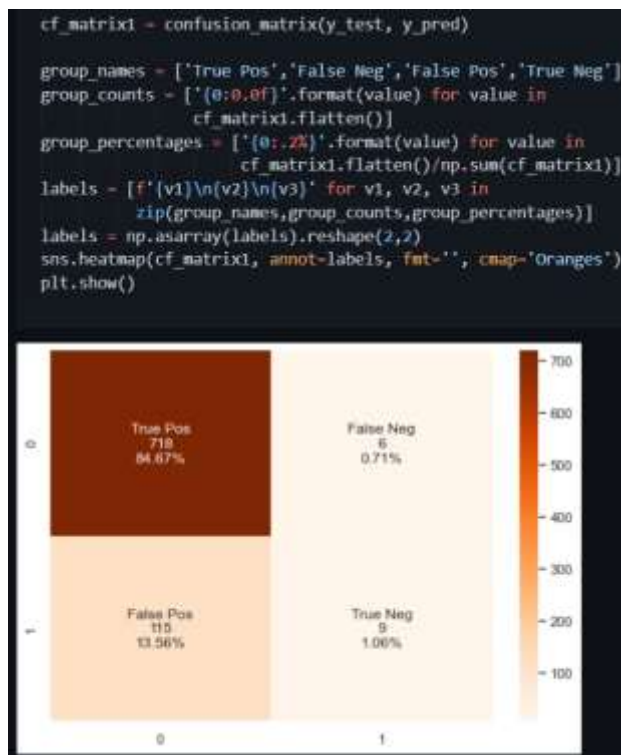
```

Now, we are going to observe all the results from the model



We have a fair AUC and a good accuracy. When AUC is 0.709, it means there is a 70,9% chance that the model will be able to distinguish between positive class and negative class.

Now, lets visualize our confusion matrix



Now, model evaluation with K-fold cross validation using `cross_val_score()` function

```

logreg = LogisticRegression(max_iter=2000)
# We are passing the entirety of X and y, not X_train or y_train, it takes care of splitting the data
# 10 folds
scores_accuracy = cross_val_score(logreg, X, y, cv=10, scoring='accuracy')
scores_log_loss = cross_val_score(logreg, X, y, cv=10, scoring='neg_log_loss')
scores_auc = cross_val_score(logreg, X, y, cv=10, scoring='roc_auc')
print('k-fold cross-validation results:')
print('LogisticRegression'+" average accuracy is %2.3f" % scores_accuracy.mean())
print('LogisticRegression'+" average log_loss is %2.3f" % scores_log_loss.mean())
print('LogisticRegression'+" average auc is %2.3f" % scores_auc.mean())

k-fold cross-validation results:
LogisticRegression average accuracy is 0.856
LogisticRegression average log_loss is 0.382
LogisticRegression average auc is 0.726

```

As we can see, we have similar values to the first inspection

In the following step, we'll be searching for the optimum parameters for the logistic regression


```

C = np.arange(1e-05, 5.5, 0.1)
scoring = {'accuracy': 'accuracy', 'AUC': 'roc_auc', 'log_loss': 'neg_log_loss'}

rskfold = RepeatedStratifiedKFold(n_splits=5, n_repeats=5, random_state=42)

steps=[('scale',StandardScaler(with_mean=False,with_std=False)),('clf',LogisticRegression(max_iter=3000))]
log_pipe = Pipeline(steps=steps)

log_clf = GridSearchCV(estimator=log_pipe, cv=rskfold,
                      scoring=scoring, return_train_score=True,
                      param_grid=dict(clf__C=C), refit='Accuracy')

log_clf.fit(X_train, y_train)
results = log_clf.cv_results_

print("best params: " + str(log_clf.best_estimator_))
print("best params: " + str(log_clf.best_params_))
print('best score:', log_clf.best_score_)

best params: Pipeline(steps=[('scale', StandardScaler(with_mean=False, with_std=False)),
                             ('clf',
                              LogisticRegression(C=1.1000100000000002, max_iter=3000))])
best params: {'clf__C': 1.1000100000000002}
best score: 0.8530973451327435

```

And the last step it's the visualization of the key indicators.

```

best params: Pipeline(steps=[('scale', StandardScaler(with_mean=False, with_std=False)),
                             ('clf',
                              LogisticRegression(C=1.1000100000000002, max_iter=3000))])
best params: {'clf__C': 1.1000100000000002}
best score: 0.8530973451327435

print(classification_report(y_test,y_pred))

```

	precision	recall	f1-score	support
0	0.86	0.99	0.92	724
1	0.60	0.07	0.13	124
accuracy			0.86	848
macro avg	0.73	0.53	0.53	848
weighted avg	0.82	0.86	0.81	848

b. K-Nearest Neighbor with Example (Binary)

It is a lazy learning algorithm that stores all instances corresponding to training data in n-dimensional space. It is a lazy learning algorithm as it does not focus on constructing a general internal model, instead, it works on storing instances of training data.

Classification is computed from a simple majority vote of the k nearest neighbors of each point. It is supervised and takes a bunch of labeled points and uses them to label other points. To label a new point, it looks at the labeled points closest to that new point also known as its nearest neighbors. It has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point. The “k” is the number of neighbors it checks.

Advantages And Disadvantages

- This algorithm is quite simple in its implementation and is robust to noisy training data. Even if the training data is large, it is quite efficient.

- The only disadvantage with the KNN algorithm is that there is no need to determine the value of K and computation cost is pretty high compared to other algorithms.

Use Cases

- Industrial applications to look for similar tasks in comparison to others
- Handwriting detection applications
- Image recognition
- Video recognition
- Stock analysis

Example

First, we need to know the best K (neighbors) for this model

```
x_train,x_test1,y_train1,y_test1 = train_test_split(X,y,test_size=0.2,random_state=42, stratify=y)

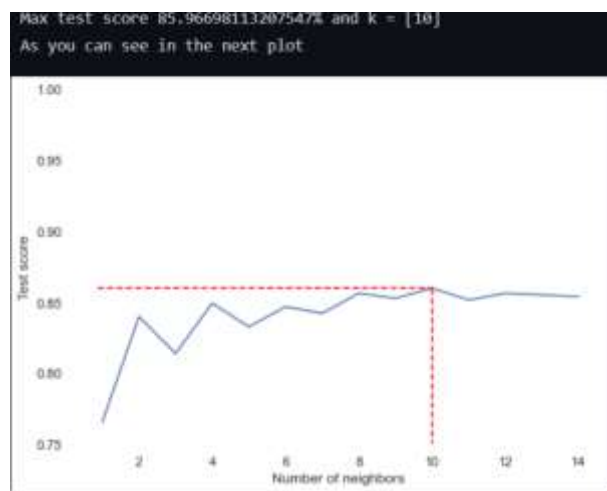
test_scores = []

for i in range(1,15):
    knn = KNeighborsClassifier(i)
    knn.fit(X_train,y_train)

    test_scores.append(knn.score(X_test,y_test))

max_test_score = max(test_scores) #busca el score maximo
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_test_score] #busca el mejor k
print('Max test score {}% and k = {}'.format(max_test_score*100,list(map(lambda x: x+1, test_scores_ind))))
print('As you can see in the next plot')

sns.lineplot(x=range(1,15),y=test_scores)
plt.plot([10,10],[0,max_test_score], '--',color='red')
plt.plot([max_test_score,10],[max_test_score,max_test_score], '--',color='red')
plt.ylim([0.75,1])
plt.xlabel('Number of neighbors')
plt.ylabel('Test score')
sns.despine(left=True,bottom=True)
plt.show()
```



As we can see, we are going to use 10 neighbors because it has the maximum test score

Now, lets train our model and show the performance.

```

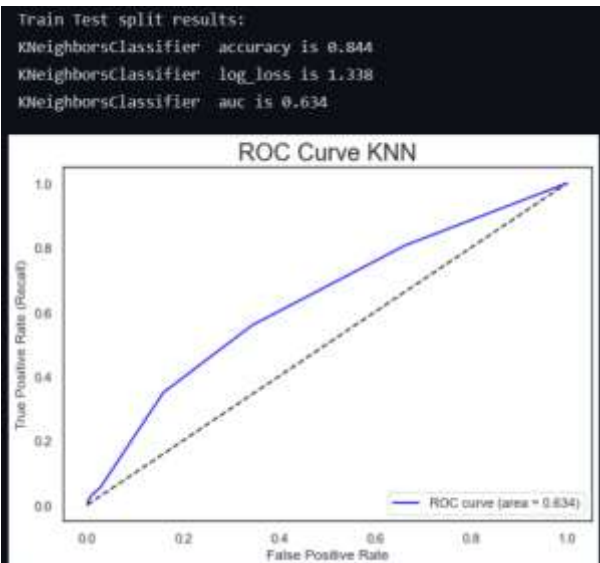
steps1=[('scaler',StandardScaler()),('knn',KNeighborsClassifier(n_neighbors=10))]
knn_scaled=Pipeline(steps1)
knn_scaled.fit(X_train1,y_train1)

y_pred1=knn_scaled.predict(X_test1)
y_pred_proba1 = knn_scaled.predict_proba(X_test1)[:, 1]
fpr1, tpr1, thr1 = roc_curve(y_test1, y_pred_proba1)
print('Train Test split results:')
print('KNeighborsClassifier', " accuracy is %2.3f" % accuracy_score(y_test1, y_pred1))
print('KNeighborsClassifier', " log_loss is %2.3f" % log_loss(y_test1, y_pred_proba1))
print('KNeighborsClassifier', " auc is %2.3f" % auc(fpr1, tpr1))

idx1 = np.min(np.where(tpr1 > 0.95)) # index of the first threshold for which the sensibility > 0.95

plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr1, tpr1, color='blue', label='ROC curve (area = %0.3f)' % auc(fpr1, tpr1))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate (Recall)')
plt.title('ROC Curve KNN', fontsize=20)
plt.legend(loc='lower right')
plt.show()

```



With KNN we have less accuracy and AUC than the logistic regression model.



```
print(classification_report(y_test1,y_pred1))
```

	precision	recall	f1-score	support
0	0.85	0.99	0.92	719
1	0.36	0.03	0.06	129
accuracy	0.84			848
macro avg	0.61	0.51	0.49	848
weighted avg	0.78	0.84	0.78	848

And for the key indicator, the logistic regression is better.

c. Naïve Bayes Classifier

It is a classification algorithm based on Bayes's theorem which gives an assumption of independence among 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.

Even if the features depend on each other, all of these properties contribute to the probability independently. Naive Bayes model is easy to make and is particularly useful for comparatively large data sets. Even with a simplistic approach, Naive Bayes is known to outperform most of the classification methods in machine learning. Following is the Bayes theorem to implement the Naive Bayes Theorem.

Advantages and Disadvantages

- The Naive Bayes classifier requires a small amount of training data to estimate the necessary parameters to get the results.

- ## Use Cases

- #### d. Stochastic Gradient Descent

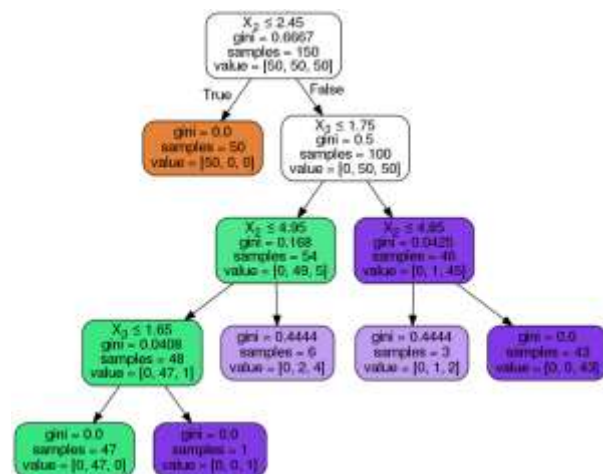
Advantages and Disadvantages

- ## Use Cases

- ### e. Decision Trees

The tree is constructed in a top-down recursive divide and conquer approach. A decision node will have two or more branches and a leaf represents a classification or decision. The topmost node in the decision tree that corresponds to the best predictor is called the root node, and the best thing about a decision tree is that it can handle both categorical and numerical data.

Advantages and Disadvantages



- A decision tree gives an advantage of simplicity to understand and visualize, it requires very little data preparation as well.
- The disadvantage that follows with the decision tree is that it can create complex trees that may not categorize efficiently. They can be quite unstable because even a simplistic change in the data can hinder the whole structure of the decision tree.

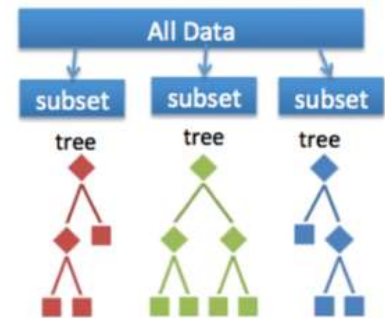
Use Cases

- Data exploration
- Pattern Recognition
- Option pricing in finances
- Identifying disease and risk threats

f. Random Forests

Random decision trees or random forest are an ensemble learning method for classification, regression, etc. It operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes or classification or mean prediction(regression) of the individual trees.

A random forest is a meta-estimator that fits a number of trees on various subsamples of data sets and then uses an average to improve the accuracy in the model's predictive nature. The sub-sample size is always the same as that of the original input size but the samples are often drawn with replacements.



Advantages and Disadvantages

- The advantage of the random forest is that it is more accurate than the decision trees due to the reduction in the over-fitting.
- The only disadvantage with the random forest classifiers is that it is quite complex in implementation and gets pretty slow in real-time prediction.

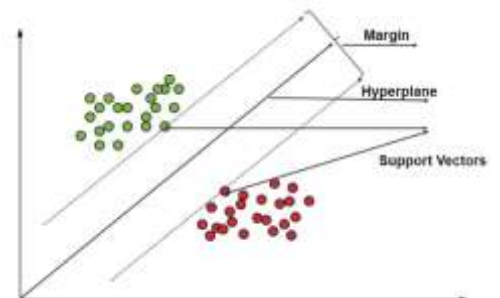
Use Cases

- Industrial applications such as finding if a loan applicant is high-risk or low-risk
- For Predicting the failure of mechanical parts in automobile engines
- Predicting social media share scores
- Performance scores

g. Support Vector Machine

The support vector machine is a classifier that represents the training data as points in space separated into categories by a gap as wide as possible. New points are then added to space by predicting which category they fall into and which space they will belong to.

Advantages and Disadvantages



- It uses a subset of training points in the decision function which makes it memory efficient and **is highly effective in high dimensional spaces.**
- The only disadvantage with the support vector machine is that the algorithm does not directly provide probability estimates.

Use cases

- Business applications for comparing the performance of a stock over a period of time
- Investment suggestions
- Classification of applications requiring accuracy and efficiency

14. Binary Classification

A binary classification refers to those tasks which can give either of any two class labels as the output. Generally, one is considered as the normal state and the other is considered to be the abnormal state. The following examples will help you to understand them better.

- Email Spam detection: Normal State – Not Spam, Abnormal State – Spam
- Conversion prediction: Normal State – Not churned, Abnormal State – Churn
- Conversion Prediction: Normal State – Bought an item, Abnormal State – Not bought an item

You can also add the example of that “No cancer detected” to be a normal state and “Cancer detected” to be the abnormal state. The notation mostly followed is that the normal state gets assigned the value of 0 and the class with the abnormal state gets assigned the value of 1. For each example, one can also create a model which predicts the Bernoulli probability for the output. You can read more about the probability here. In short, it returns a discrete value that covers all cases and will give the output as either the outcome will have a value of 1 or 0. Hence after the association to two different states, the model can give an output for either of the values present.

The most popular algorithms which are used for binary classification are:

- K-Nearest Neighbors (example)
- Logistic Regression (example)
- Support Vector Machine
- Decision Trees
- Naive Bayes

Out of the mentioned algorithms, some algorithms were specifically designed for the purpose of binary classification and natively do not support more than two types of class. Some examples of such algorithms are Support Vector Machines and Logistic Regression.

15. Multi-Class Classification

Multi-class classification refers to those classification tasks that have more than two class labels.

Examples include:

- Face classification.
- Plant species classification.
- Optical character recognition.

Unlike binary classification, multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes.

The number of class labels may be very large on some problems. For example, a **model may predict a photo as belonging to one among thousands or tens of thousands of faces in a face recognition system.**

Problems that involve predicting a sequence of words, such as text translation models, may also be considered a special type of multi-class classification. Each word in the sequence of words to be predicted involves a multi-class classification where the size of the vocabulary defines the number of possible classes that may be predicted and could be tens or hundreds of thousands of words in size.

It is common to model a multi-class classification task with a model that predicts a Multinoulli probability distribution for each example.

The Multinoulli distribution is a discrete probability distribution that covers a case where an event will have a categorical outcome, e.g. K in $\{1, 2, 3, \dots, K\}$. For classification, this means that the model predicts the probability of an example belonging to each class label.

Many algorithms used for binary classification can be used for multi-class classification.

Popular algorithms that can be used for multi-class classification include:

- k-Nearest Neighbors.
- Decision Trees.
- Naive Bayes.
- Random Forest.
- Gradient Boosting.

Algorithms that are designed for binary classification can be adapted for use for multi-class problems.

This involves using a strategy of fitting multiple binary classification models for each class vs. all other classes (called one-vs-rest) or one model for each pair of classes (called one-vs-one).

- One-vs-Rest: Fit one binary classification model for each class vs. all other classes.
- One-vs-One: Fit one binary classification model for each pair of classes.

Binary classification algorithms that can use these strategies for multi-class classification include:

- Logistic Regression.
- Support Vector Machine.

a. Quick Example Using Decision Tree Classifier

```
# importing necessary libraries
from sklearn import datasets
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

# loading the iris dataset
iris = datasets.load_iris()

# X -> features, y -> label
X = iris.data
y = iris.target

# dividing X, y into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)

# training a DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
dtree_model = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)
dtree_predictions = dtree_model.predict(X_test)

print(classification_report(y_test, dtree_predictions))
```

✓ 0.4s

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.83	0.94	0.88	16
2	0.86	0.67	0.75	9
accuracy			0.89	38
macro avg	0.90	0.87	0.88	38
weighted avg	0.90	0.89	0.89	38

16. Multi-Label Classification

Multi-label classification refers to those **classification tasks that have two or more class labels, where one or more class labels may be predicted for each example.**

Consider the example of photo classification, where a given photo may have multiple objects in the scene and a model may **predict the presence of multiple known objects in the photo, such as “bicycle,” “apple,” “person,” etc.**

This is unlike binary classification and multi-class classification, where a single class label is predicted for each example.

It is common to model multi-label classification tasks with a model that predicts multiple outputs, with each output taking predicted as a Bernoulli probability distribution. This is essentially a model that makes multiple binary classification predictions for each example.

Classification algorithms used for binary or multi-class classification cannot be used directly for multi-label classification. Specialized versions of standard classification algorithms can be used, so-called multi-label versions of the algorithms, including:

- Multi-label Decision Trees
- Multi-label Random Forests
- Multi-label Gradient Boosting

Another approach is to use a separate classification algorithm to predict the labels for each class.

One way to work with multi-label classification is using neuronal networks and deep learning. Both concepts will be explained later.

17. Imbalanced Classification

Imbalanced classification refers to classification tasks where the number of examples in each class is unequally distributed.

Typically, imbalanced classification tasks are binary classification tasks where the majority of examples in the training dataset belong to the normal class and a minority of examples belong to the abnormal class.

Examples include:

- Fraud detection.
- Outlier detection.
- Medical diagnostic tests.

These problems are modeled as binary classification tasks, although may require specialized techniques.

Specialized techniques may be used to change the composition of samples in the training dataset by **undersampling the majority class or oversampling the minority class**.

Examples include:

- Random Undersampling.
- SMOTE Oversampling.

Specialized modeling algorithms may be used that pay more attention to the minority class when fitting the model on the training dataset, such as cost-sensitive machine learning algorithms.

Examples include:

- Cost-sensitive Logistic Regression.
- Cost-sensitive Decision Trees.
- Cost-sensitive Support Vector Machines.

a. Synthetic Minority Oversampling Technique (SMOTE) Example

SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically $k=5$). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

```

from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)

X_sm, y_sm = sm.fit_resample(X, y)

print(f'''Shape of X before SMOTE: {X.shape}
Shape of X after SMOTE: {X_sm.shape}''')

print('\nBalance of positive and negative classes (%):')
y_sm.value_counts(normalize=True) * 100

Shape of X before SMOTE: (25134, 29)
Shape of X after SMOTE: (49424, 29)

Balance of positive and negative classes (%):
1    50.0
0    50.0
Name: TARGET, dtype: float64

```

```

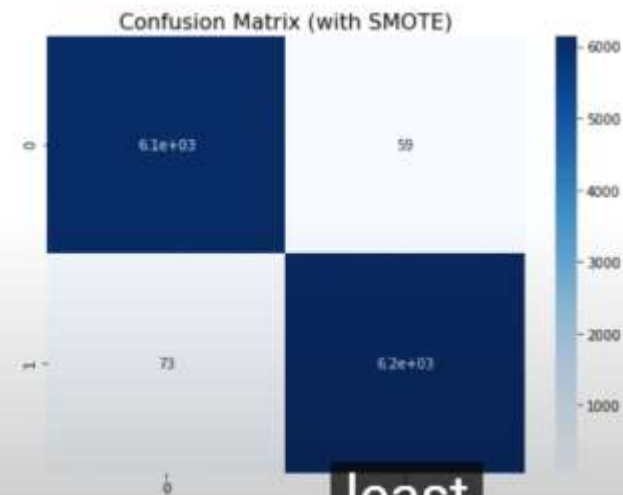
X_train, X_test, y_train, y_test = train_test_split(
    X_sm, y_sm, test_size=0.25, random_state=42
)

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
preds = model.predict(X_test)

print(f'Accuracy = {accuracy_score(y_test, preds):.2f}\nRecall = {recall_score(y_test, preds):.2f}\n')
cm = confusion_matrix(y_test, preds)
plt.figure(figsize=(8, 6))
plt.title('Confusion Matrix (with SMOTE)', size=16)
sns.heatmap(cm, annot=True, cmap='Blues');

Accuracy = 0.99
Recall = 0.99

```



18. Regression

19. Ensemble Methods

Neuronal Networks

A neural network consists of neurons that are arranged in layers, they take some input vector and convert it into an output. The process involves each neuron taking input and applying a function which is often a non-linear function to it and then passes the output to the next layer.

In general, the network is supposed to be feed-forward meaning that the unit or neuron feeds the output to the next layer but there is no involvement of any feedback to the previous layer.

Weighings are applied to the signals passing from one layer to the other, and these are the weighings that are tuned in the training phase to adapt a neural network for any problem statement.

Advantages and Disadvantages

- It has a high tolerance to noisy data and able to classify untrained patterns, it performs better with continuous-valued inputs and outputs.
- The disadvantage with the artificial neural networks is that it has poor interpretation compared to other models.

Use Cases

- Handwriting analysis
- Colorization of black and white images
- Computer vision processes
- Captioning photos based on facial features

ADD MORE ALGORITHMS CLASIFICATION,
REGRESSIOM CLUSTERING