Artificial Intelligence

Machine Learning / Python / Deep Learning

Version 1.0

Why Quality Thought ?

Get your career started in AI, ML in 3 steps

1
- **Register for 2 hours FREE seminar**
- Register and attend our 2 hours FREE seminar on Introduction to Machine Learning to get a preview of our two months internship program on Machine learning

2
- **Register for 2 months Training program**
- Register and start the two months training and internship program on machine learning with Python.

3
- **Get job placement!**
- With QT HR team's placement assistance attend the job interviews and get career opportunities with great companies in the AI, ML field

Contents

Artificial Intelligence History **10**
Why is artificial intelligence important? 11
How Artificial Intelligence Is Being Used **12**
Health Care 12
Retail 12
Manufacturing 12
Sports 13
What are the challenges of using artificial intelligence? 15
How Artificial Intelligence Works **15**
Why Algebra Matters 18
3 Reasons Why We Learn Algebra **19**
Algebra in Everyday Life 19
Examples of using algebra in everyday life **19**
Going shopping **19**
Calculating grocery expense **20**
Filling up the gas tank **21**
Basic Terms **22**
Coefficient of Algebraic Expressions Properties **26**
Frequency Histogram: **30**
Relative Frequency Histograms: **31**
When to Use a Histogram: **32**
Where Do Outliers Come From? **45**
Detecting Outliers **45**
Table of Contents **56**
Common Data Types **57**
Types of Distributions **57**
Bernoulli Distribution 57
Uniform Distribution 58
Binomial Distribution 59
Normal Distribution 61
Poisson Distribution 62

Exponential Distribution 64
Hypothesis Testing **65**
What is a Hypothesis? **65**
What is a Hypothesis Statement? **66**
What is Hypothesis Testing? **67**
What is the Null Hypothesis? 67
How do I State the Null Hypothesis? 67
Hypothesis Testing Examples #1: Basic Example **67**
Rejecting the null hypothesis **68**
Hypothesis Testing Examples (One Sample Z Test) **68**
One Sample Hypothesis Testing Examples: #2 ..... **68**
One Sample Hypothesis Testing Examples: #3 ..... **69**
Hypothesis Testing Examples: Mean (Using TI 83) **70**
Bayesian Hypothesis Testing: What is it? **71**
P Values. **71**
Bayesian Hypothesis Testing **71**
Differences Between Traditional and Bayesian Hypothesis Testing **72**
Arguments for Bayesian Testing **72**
Arguments against. **72**
BREAKING DOWN 'Z-Test' **73**
Hypothesis Test **73**
One-Sample Z-Test Example **73**
The ANOVA Test: **79**
Types of ANOVA **79**
What Does "One-Way" or "Two-Way Mean? **80**
What are "Groups" or "Levels"? **80**
Notation **85**
What is a 'Derivative': **86**
Interesting Functions **88**
Accuracy **89**
1. Add Δx **89**
2. Subtract the Two Formulas **89**

3. Rate of Change **90**
4. Reduce Δx close to 0 ..... **90**
Try It On A Function **90**
Derivative Rules: 91
Power Rule 91
Example: What is the derivative of x^3 ? 91
Defining the Partial Derivative **92**
Taking the Partial Derivative of a Partial Derivative **93**
Extending the Idea of Partial Derivatives **96**
Example: The surface area of a square prism. 96
Chain Rule **97**
Why use neural networks? 143
How the Human Brain Learns? 143
2.2 From Human Neurones to Artificial Neurones 144
An engineering approach **144**
1. A simple neuron 144
2 Firing rules 145
3 Pattern Recognition - an example 146
4 A more complicated neuron 148
Architecture of neural networks **149**
1. Feed-forward networks 149
2 Feedback networks 149
3 Network layers 150
4 Perceptrons 151
Why Call it "Deep Learning"? **153**
Gather data 154
Clean the data 154
Split the data 155
Engineer features 155
Preprocessing data **157**
Summary of data preparation and preprocessing **158**

How Do You Start Machine Learning in Python? **159**
Python Can Be Intimidating When Getting Started 159
Beginners Need A Small End-to-End Project 159
Hello World of Machine Learning 160
Machine Learning in Python: Step-By-Step **160**
In this section, we are going to work through a small machine learning project end-to-end. **160**
1. Downloading, Installing and Starting Python SciPy **161**
1.1 Install SciPy Libraries 161
1.2 Start Python and Check Versions 161
2. Load The Data **163**
2.1 Import libraries 163
2.2 Load Dataset 164
3. Summarize the Dataset **164**
3.1 Dimensions of Dataset 164
3.2 Peek at the Data 165
3.3 Statistical Summary 165
3.4 Class Distribution 166
4. Data Visualization **166**
4.1 Univariate Plots 166
4.2 Multivariate Plots 168
5. Evaluate Some Algorithms **168**
5.1 Create a Validation Dataset 169
5.2 Test Harness 169
5.3 Build Models 170
5.4 Select Best Model 171
6. Make Predictions **172**
1. Business Understanding **173**
2. Data Acquisition and Understanding **175**
3. Modeling **176**
4. Deployment **178**
5. Customer Acceptance **179**

Summary **179**
Linear Regression Machine Learning Algorithm 183
What is Lasso Regression? **187**
L1 Regularization **188**
Performing the Regression **188**
Shrinkage **189**
Regularization **189**
On Mathematics **190**
Gradient Descent **192**
1. What is Gradient Descent? **192**
Logistic Regression **196**
When to Use Logistic Regression Machine Learning Algorithm 197
Advantages of Using Logistic Regression 197
Drawbacks of Using Logistic Regression 197
Applications of Logistic Regression 198
Naïve Bayes Classifier Algorithm 198
Decision Tree Machine Learning Algorithm 200
Why should you use Decision Tree Machine Learning algorithm? 201
When to use Decision Tree Machine Learning Algorithm 202
Advantages of Using Decision Tree Machine Learning Algorithms 202
Drawbacks of Using Decision Tree Machine Learning Algorithms 202
Applications of Decision Tree Machine Learning Algorithm 203
Random Forest Machine Learning Algorithm **203**
Why use Random Forest Machine Learning Algorithm? 204
Advantages of Using Random Forest Machine Learning Algorithms 205
Support Vector Machine Learning Algorithm 206
Soft Margin Classification 208
Nonlinear SVM **208**
Kernel Trick **209**
Hyperparameters **210**
C Parameter 210
γ Parameter 211

Implementation using scikit-learn **211**
Polynomial Kernel 215
Gaussian Kernel 216
Soft Margin Classification 224
Nonlinear SVM **225**
Kernel Trick **226**
Polynomial Kernel 226
Gaussian RBF Kernel 226
Hyperparameters **227**
C Parameter 227
Parameter 227
Implementation using scikit-learn **227**
Linear Kernel 227
Polynomial Kernel 231
Gaussian Kernel 233
How Gradient Boosting Works **242**
Improvements to Basic Gradient Boosting **244**
K-Means Clustering **246**
Use Cases 247
Algorithm 247
Choosing the Value of K 249
Implementation using Python 250
The scikit-learn approach 256
K Means Clustering Algorithm 258
TensorFlow **263**
The Iris classification problem **265**
Import and parse the training dataset **266**
Download the dataset 266
Inspect the data 267
Parse the dataset 268
Create the training tf.data.Dataset 268
Select the type of model **269**

Why model? 269
Select the model 269
Create a model using Keras 270
Train the model 271
Define the loss and gradient function 271
Create an optimizer 272
Training loop 273
Visualize the loss function over time 274
Evaluate the model's effectiveness 275
Setup the test dataset 276
Evaluate the model on the test dataset 276
Use the trained model to make predictions 277
Setup 278
Reading The Census Data 278
Converting Data into Tensors 279
Selecting and Engineering Features for the Model 280
Base Categorical Feature Columns 281
Base Continuous Feature Columns 282
Making Continuous Features Categorical through Bucketization 282
Intersecting Multiple Columns with CrossedColumn 283
Training and Evaluating Our Model 284
Adding Regularization to Prevent Overfitting 285
How Logistic Regression Works 286

Artificial Intelligence

What is Artificial Intelligence (AI)?

According to the father of Artificial Intelligence, **John McCarthy**, it is "The science and engineering of making intelligent machines, especially intelligent computer programs".

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

- Artificial intelligence (AI, also machine intelligence, MI) is intelligence demonstrated by machines, in contrast to the natural intelligence (NI) displayed by humans and other animals.
- Artificial intelligence (AI) makes it possible for machines to learn from experience, adjust to new inputs and perform human-like tasks.

Goals of AI :

- **To Create Expert Systems** – The systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users.
- **To Implement Human Intelligence in Machines** – Creating systems that understand, think, learn, and behave like humans.

Some of the activities computers with artificial intelligence are designed for include:

- Speech recognition
- Learning
- Planning
- Problem solving

Ex:-chess-playing computers to self-driving cars – rely heavily on deep learning and natural language processing. Using these technologies, computers can be trained to accomplish specific tasks by processing large amounts of data and recognizing patterns in the data.

- AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.
- Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding.
- Attendees Allen Newell (CMU), Herbert Simon (CMU), John McCarthy (MIT), Marvin Minsky (MIT) and Arthur Samuel (IBM) became the founders and leaders of AI research.

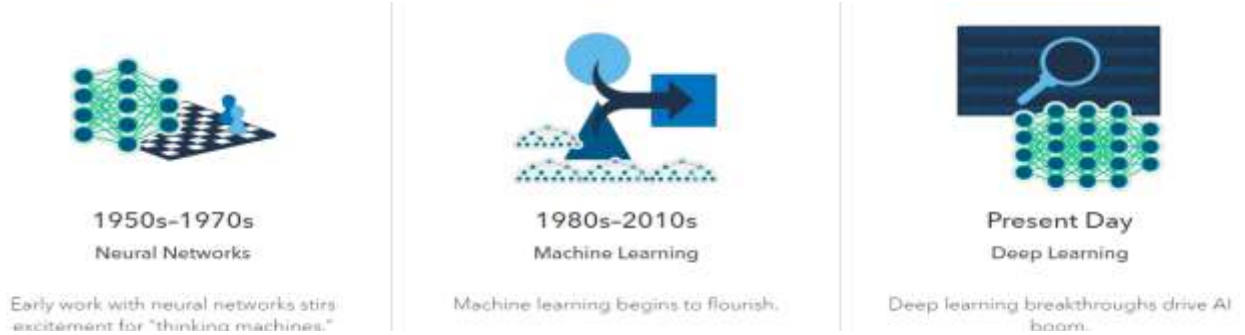
Research associated with artificial intelligence is highly technical and specialized. The core problems of artificial intelligence include programming computers for certain traits such as:

- ✓ Knowledge
- ✓ Reasoning

- ✓ Problem solving
- ✓ Perception
- ✓ Learning
- ✓ Planning
- ✓ Ability to manipulate and move objects

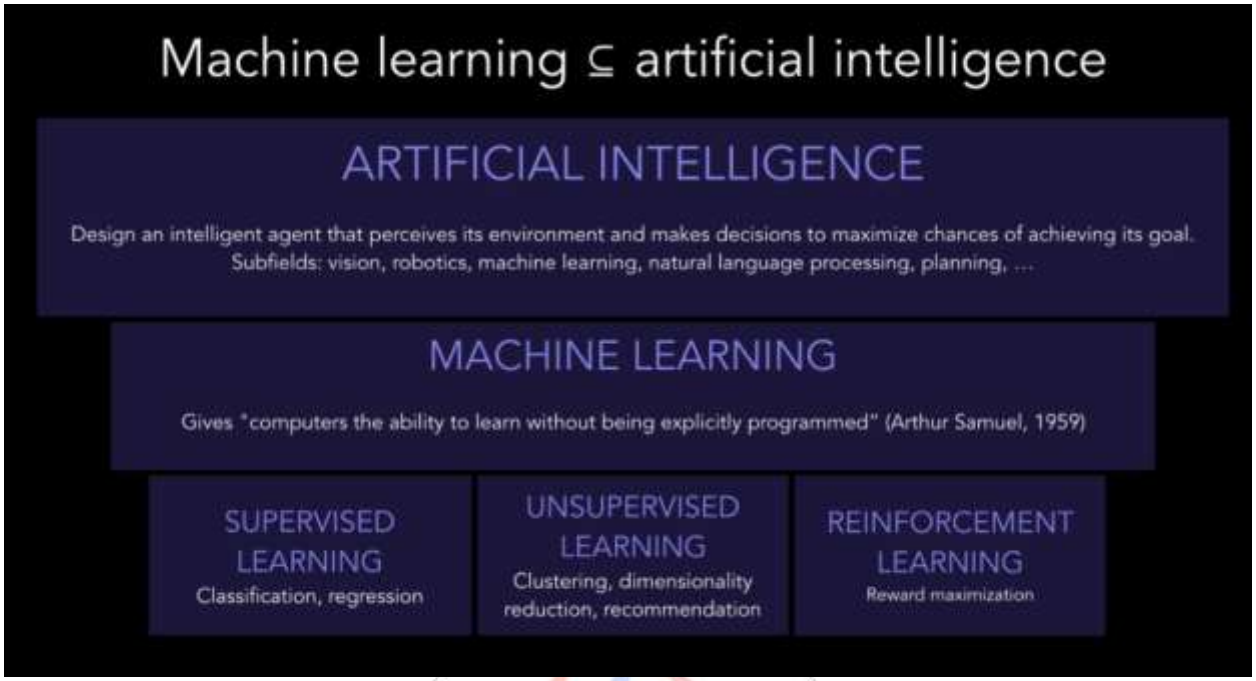
Artificial Intelligence History

- ➢ The term artificial intelligence was coined in 1956, but AI has become more popular today thanks to increased data volumes, advanced algorithms, and improvements in computing power and storage.
- ➢ Early AI research in the 1950s explored topics like problem solving and symbolic methods.
- ➢ In the 1960s, the US Department of Defense took interest in this type of work and began training computers to mimic basic human reasoning.
- ➢ For example, Hollywood movies and science fiction novels depict AI as human-like robots that take over the world, the current evolution of AI technologies isn't that scary – or quite that smart. Instead, AI has evolved to provide many specific benefits in every industry.



Why is artificial intelligence important?

- **AI automates repetitive learning and discovery through data.** But AI is different from hardware-driven, robotic automation. Instead of automating manual tasks, AI performs frequent, high-volume, computerized tasks reliably and without fatigue. For this type of automation, human inquiry is still essential to set up the system and ask the right questions.
- **AI adds intelligence** to existing products. In most cases, AI will not be sold as an individual application. Rather, products you already use will be improved with AI capabilities, much like Siri was added as a feature to a new generation of Apple products. Automation, conversational platforms, bots and smart machines can be combined with large amounts of data to improve many technologies at home and in the workplace, from security intelligence to investment analysis.
- **AI adapts through progressive learning algorithms** to let the data do the programming. AI finds structure and regularities in data so that the algorithm acquires a skill: The algorithm becomes a classifier or a predicator. So, just as the algorithm can teach itself how to play chess, it can teach itself what product to recommend next online. And the models adapt when given new data. Back propagation is an AI technique that allows the model to adjust, through training and added data, when the first answer is not quite right
- **AI analyzes more and deeper data** using neural networks that have many hidden layers. Building a fraud detection system with five hidden layers was almost impossible a few years ago. All that has changed with incredible computer power and big data. You need lots of data to train deep learning models because they learn directly from the data. The more data you can feed them, the more accurate they become.
- **AI achieves incredible accuracy** though deep neural networks – which was previously impossible. For example, your interactions with Alexa, Google Search and Google Photos are all based on deep learning – and they keep getting more accurate the more we use them. In the medical field, AI techniques from deep learning, image classification and object recognition can now be used to find cancer on MRIs with the same accuracy as highly trained radiologists.
- **AI gets the most out of data.** When algorithms are self-learning, the data itself can become intellectual property. The answers are in the data; you just have to apply AI to get them out. Since the role of the data is now more important than ever before, it can create a competitive advantage. If you have the best data in a competitive industry, even if everyone is applying similar techniques, the best data will win.



How Artificial Intelligence Is Being Used

Every industry has a high demand for AI capabilities – especially question answering systems that can be used for legal assistance, patent searches, risk notification and medical research.

Other uses of AI include:

Health Care

- AI applications can provide personalized medicine and X-ray readings.
- Personal health care assistants can act as life coaches, reminding you to take your pills, exercise or eat healthier.

Retail

- AI provides virtual shopping capabilities that offer personalized recommendations and discuss purchase options with the consumer.
- Stock management and site layout technologies will also be improved with AI.

Manufacturing

- AI can analyze factory IoT data as it streams from connected equipment to forecast expected load and demand using recurrent networks, a specific type of deep learning network used with sequence data.

Sports

- AI is used to capture images of game play and provide coaches with reports on how to better organize the game, including optimizing field positions and strategy.

AI at Recent News:

Google's New AI Algorithm Can Predict Heart Disease By Scanning Your Eyes

JEEVAN BISWAS – 1 WEEK AGO

19 SHARES

Remember that song by Bryan Adams that said "*Look into my eyes... And when you find me there, you'll search no more*" ? Google's new AI algorithm can do one better — it can look into your eyes, search and find signs of cardiovascular risks.

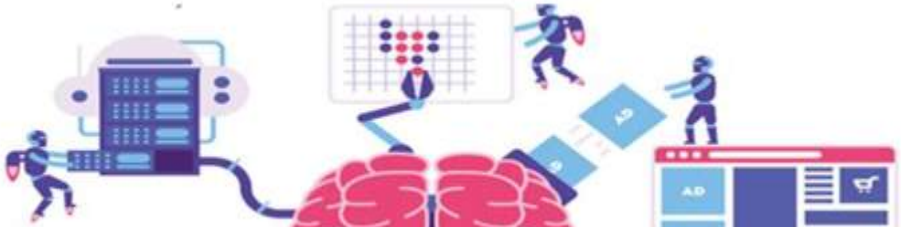
Ameerpet / Kondapur

Artificial Intelligence has the potential to add 1 trillion to India's economy in 2035

A new NITI Aayog working paper outlines a comprehensive strategy for AI adoption in India. Given many countries, including China and France, have formalised strategies even as the US powers through with AI research, Indian policymakers will do well to pay heed to NITI's recommendations.

By: The Financial Express | Published: June 9, 2018 1:36 AM

2 SHARES SHARE



Artificial Intelligence facing large skills shortage, says Microsoft

The fast-emerging field of Artificial Intelligence, which has suddenly caught the attention of the IT industry and the governments across the world, is facing a large skills shortage, a top Microsoft official has said.

By: PTI | Washington | Updated: May 13, 2018 10:30 PM

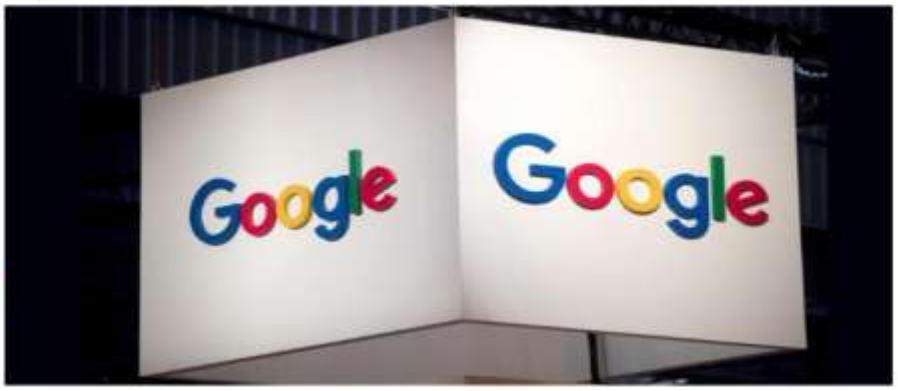
86 SHARES SHARE

News / Technology / News / Google can now predict when you will die, with 95 per cent accuracy

Google can now predict when you will die, with 95 per cent accuracy

India Today Tech
New Delhi
June 20, 2018
UPDATED: June 20, 2018 14:30 IST

Follow @IndiaTodayTech
FOLLOW
EMAIL AUTHOR
READ LATER



Google has seeped into our lives and it is no surprise that it knows a lot about us. From our contacts to our address, there is hardly anything hidden from Google and now it can even predict the survival chances of a hospital patient.

What are the challenges of using artificial intelligence?

- Artificial intelligence is going to change every industry, but we have to understand its limits.
- The principle limitation of AI is that it learns from the data. There is no other way in which knowledge can be incorporated. That means any inaccuracies in the data will be reflected in the results. And any additional layers of prediction or analysis have to be added separately.
- Today's AI systems are trained to do a clearly defined task. The system that plays poker cannot play solitaire or chess.
- The system that detects fraud cannot drive a car or give you legal advice. In fact, an AI system that detects health care fraud cannot accurately detect tax fraud or warranty claims fraud.
- In other words, these systems are very, very specialized. They are focused on a single task and are far from behaving like humans.
- self-learning systems are not autonomous systems. The imagined AI technologies that you see in movies and TV are still science fiction. But computers that can probe complex data to learn and perfect specific tasks are becoming quite common.

How Artificial Intelligence Works

AI works by combining large amounts of data with fast, iterative processing and intelligent algorithms, allowing the software to learn automatically from patterns or features in the data. AI is a broad field of study that includes many theories, methods and technologies, as well as the following major subfields:

- **Machine learning** automates analytical model building. It uses methods from neural networks, statistics, operations research and physics to find hidden insights in data without explicitly being programmed for where to look or what to conclude.

  Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

- **A neural network** is a type of machine learning that is made up of interconnected units (like neurons) that processes information by responding to external inputs, relaying information between each unit. The process requires multiple passes at the data to find connections and derive meaning from undefined data.
- **Deep learning** uses huge neural networks with many layers of processing units, taking advantage of advances in computing power and improved training techniques to learn complex patterns in large amounts of data. Common applications include image and speech recognition.

- **Cognitive computing** is a subfield of AI that strives for a natural, human-like interaction with machines. Using AI and cognitive computing, the ultimate goal is for a machine to simulate human processes through the ability to interpret images and speech – and then speak coherently in response.
- **Computer vision** relies on pattern recognition and deep learning to recognize what's in a picture or video. When machines can process, analyze and understand images, they can capture images or videos in real time and interpret their surroundings.
- **Natural language processing** (NLP) is the ability of computers to analyze, understand and generate human language, including speech. The next stage of NLP is natural language interaction, which allows humans to communicate with computers using normal, everyday language to perform tasks.

Additionally, several technologies enable and support AI:

- **Graphical processing units** are key to AI because they provide the heavy compute power that's required for iterative processing. Training neural networks requires big data plus compute power.
- **The Internet of Things** generates massive amounts of data from connected devices, most of it unanalyzed. Automating models with AI will allow us to use more of it.
- **Advanced algorithms** are being developed and combined in new ways to analyze more data faster and at multiple levels. This intelligent processing is key to identifying and predicting rare events, understanding complex systems and optimizing unique scenarios.
- **APIs, or application processing interfaces,** are portable packages of code that make it possible to add AI functionality to existing products and software packages. They can add image recognition capabilities to home security systems and Q&A capabilities that describe data, create captions and headlines, or call out interesting patterns and insights in data.

Algebra

What is Algebra: -

Algebra from Arabic "al-Jabr", literally meaning "reunion of broken parts as per Wikipedia.

Algebra: -Whole branch of math called "Algebra".

Algebra Definition:

- Algebra can be defined as the representation of numbers and quantities in equations and formulae in the form of letters.
- The topic of algebra is broadly divided into 2 parts namely, elementary algebra (or basic algebra) and abstract algebra (or modern algebra).
- The modern algebra is mainly studied by professional mathematicians.
- The basic algebra is involved in almost most of the mathematical and scientific subjects.
- The application of algebra is not only limited to science and engineering but is also used in medicine and economics.

Various types of definitions for Algebra:

1) Algebra is a branch of mathematics that deals with properties of operations and the structures these operations are defined on.
2) Algebra is a branch of mathematics that substitutes letters for numbers, and an algebraic equation represents a scale where what is done on one side of the scale is also done to the other side of the scale and the numbers act as constants.
3) Algebra is a branch of pure mathematics that deals with the rules of operations and solving equations.

* Algebra is lot like Arithmetic it follows all the rules of arithmetic and it uses the same four main operations that arithmetic is built on ------"Operations"

-->Algebra introduces a new element of unknown "?"

Ex: - 1+2=?

The answer isn't known until you go ahead and do the arithmetic. Coming to algebra we use a symbol in its place. The Symbol is usually just any letter either of the alphabet A to Z.

Really popular letter to choose is the letter "X". In Algebra we had to write it like this 1+2=X

Arithmetic 1+2=____

Algebra 1+2=X

"This is the very basic algebraic equation."

=>An equation is just a mathematical statement that two things are equal.

Algebraic Equation 1+2=X

where 1+2 = x Known Unknown

GOALS: Figure out what the unknown values in equations are and when you do that, it called solving the equation

So 1+2= X

X=3

Definition: Algebra solving equation is all lot like again where you are given mixed up complicated equations and its your job to simplify them and rearrange them until it is nice a simple equation where it's easy to tell what the unknown values are

2X= (30-2X)/4

8X=30-2X

8x+2X=30

10X=30

X=3

Concepts Associated with Algebra

Elementary Algebra involves simple rules and operations on numbers such as:

- Addition
- Subtraction
- Multiplication
- Division
- Equation solving techniques
- Variables
- Functions
- Polynomials
- Algebraic Expressions

There is no limit on the complexity of all these concepts as far as "Algebra" is concerned

Why Algebra Matters

Even if you don't think you'll need algebra outside of the hallowed halls of your average high school, managing budgets, paying bills, and even determining health care costs and planning for future investments will require a basic understanding of algebra.

3 Reasons Why We Learn Algebra

1. Speed: We can directly use algebra to solve problems more quickly and easily than we could do otherwise. For example, a simple algebraic equation can help you make a recipe smaller or when you're doing home repairs.

2. A Building Block: Algebra can serve as a building block that we can use to learn more advanced math like statistics, calculus, etc. and learn more advanced subjects, like physics, chemistry, etc. Experts say that members of the next generation could change jobs 30-40 times throughout their career, so you don't know if you will need some of the algebra you learn or not. Learning algebra is a life skill beneficial for upward mobility.

3. So that we can understand and critically evaluate the math done by others such as reporters, political candidates, insurance salesmen, bank loan officers, etc..

Algebra in Everyday Life

Examples of using algebra in everyday life

Here are some simple examples that demonstrate the relevance of algebra in the real world.

You purchased 10 items from a shopping plaza, and now you need plastic bags to carry them home. If each bag can hold only 3 items, how many plastic bags you will need to accommodate 10 items?

10 items(3 items/bag) = 3.333 bags ≈ 4bags

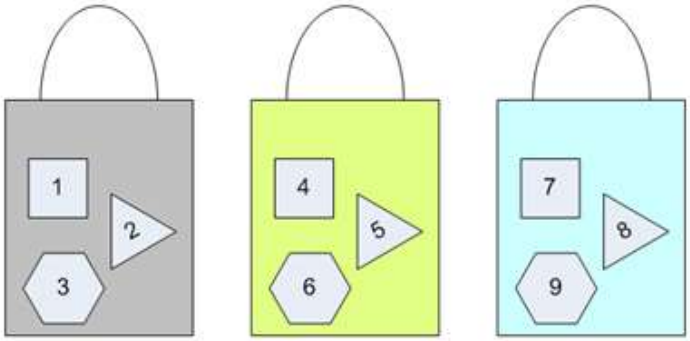
Explanation :

Example 1:

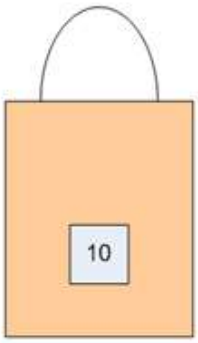
Going shopping

Explanation :

The figure below illustrates the problem:
The different shapes inside the bags denote different items purchased. The number depicts the item number.



Bag 1 Bag 2 Bag 3



Bag 4

We use simple algebraic formula xy to calculate the number of bags.

x = Number of items purchased = 10

y = Capacity of 1 bag = 3

Hence,

10/3 = 3.333 bags ≈ 4 bags

So,we need 4 shopping bags to put 10 items.

Example 2:

Calculating grocery expense

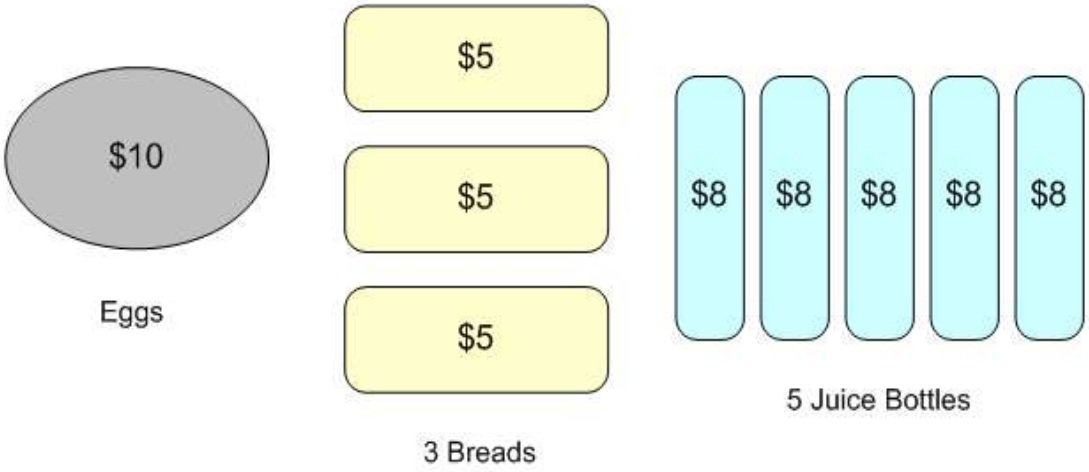
You have to buy two dozen eggs priced at $10, three breads (each bread is $5), and five bottles of juice (each bottle is $8). How much money you will need to take to the grocery store?

$65

Explanation :

The figure below shows the three items in different shapes and colors.

This will help your mind to calculate faster.



We will use algebra to solve the problem easily and quickly.

The prices are

a = Price of two dozen eggs = $10

b = Price of one bread = $5

c = Price of one bottle of juice= $8

=> Money needed = a + 3b + 5c

=> Money needed = $10 + 3($5) + 5($8) = $10 + $15 + $40 = $65

Example 3:

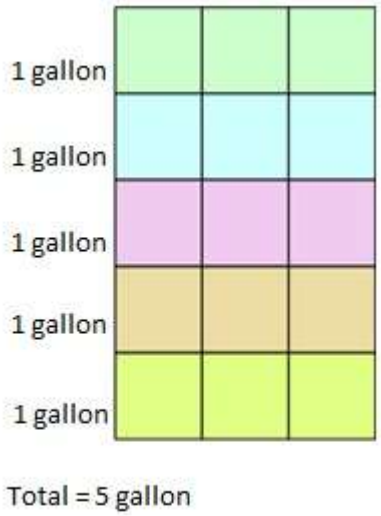
Filling up the gas tank

You need to fill the gas tank but you have only $15 in your pocket. If the price of the gas is $3 a gallon, how many gallons can you buy?

55 gallons

Explanation :

In the below diagram, each block represents $1, and each row is a bundle of $3, which is used to buy 1 gallon of gas.



We use simple algebraic formula, xy to calculate the total gallons that can be bought.

x = Money in your pocket= $15

y = Price of 1 gallon of gas= $3

Hence,

$15/$3 = 5 gallon

So, with $15 we can buy 5 gallons of gas.

Basic Terms

Equation : An equation can be defined as a statement involving symbols (variables), numbers (constants) and mathematical operators (Addition, Subtraction, Multiplication, Division etc) that asserts the equality of two mathematical expressions. The equality of the two expressions is shown by using a symbol "=" read as "**is equal to**".

For example: 3X + 7 = 16 is an equation in the variable X.

==========

Variable: A variable is a symbol that represents a quantity in an algebraic expression. It is a value that may change with time and scope of the concerned problem.

For example: in the equation 3X + 7 = 16, X is the variable.

Also in the polynomial $X^2 + 5XY - 3Y^2$,

both X and Y are variables.

A variable is a letter or symbol that represents an unknown value.

- When variables are used with other numbers, parentheses, or operations, they create an algebraic expression.

a + 2

(a) (b)

3m + 6n - 6

A variable can use any letter of the alphabet.

- n + 5
- x – 7
- w - 25

In algebra, variables are symbols used to represent unspecified numbers or values. Any letter may be used as a variable.

- An expression that represents a particular number is called a numerical expression.

  Example: 3 + 2

- An algebraic expression consists of one or more constants and variables along with one or more arithmetic operationss.

  constants - numbers variables - letters

  operations - addition, subtraction, multiplication and division

  Example of an algebraic expression: 3x + 2

- In algebraic expressions, a raised dot or parentheses are often used to indicate multiplication as the symbol x can be easily mistaken for the variable x.

  Here are some ways to represent the product of x and y.

  x.y , x(y) , (x)y , (x)(y) , xy

In each expression, the quantities being multiplied are called factors, and the result is called the product.

- Variables are used to change verbal expressions into algebraic expressions, that is, expressions that are composed of letters that stand for numbers.

Key words that can help you translate words into letters and numbers include:

- For addition: sum, more than, greater than, increase

- For subtraction: minus, less than, smaller than, decrease
- For multiplication: times, product, multiplied by, of
- For division: halve, divided by, ratio.

One Variable Equation:

An equation that involves only one variable is knows as a One Variable Equation.

$3X + 7 = 16$ is an example of it.

Two Variable Equation: An equation that involves two variables is knows as a Two Variable Equation.

$2X + Y = 10$ is a Two Variable Equation of where X and Y are variables.

Please note that here both X and Y have a power or exponent of 1.

Hence it is an equation with degree 1. The degree is equal to the highest power of the variable(s) involved.

$X^2 + 5XY - 3Y^2 = 25$ is an example of a Two Variable Equation of degree 2.

Three Variable Equation: An equation that comprises three variables / symbols is called a Three Variable Equation

$x + y - Z = 1$ ----------(1)
$8x + 3y - 6z = 1$ ----------(2)
$-4x - y + 3z = 1$ ----------(3)

The above three equations form a system of 3 equations in 3 variables X, Y and Z. Each of these equations is a Three Variable Equation of degree 1. also, these equations are called **Linear equations** in three variables.

Monomial: A monomial is a product of powers of variables. A monomial in a single variable is of the form $\mathbf{x}^n$ where X is a variable and n is a positive integer.

- There can also be monomials in more than one variable. For example, $\mathbf{x}^m \mathbf{y}^n$ is a monomial in two variables where m, n are any positive integers. Monomials can also be multiplied by nonzero constant values.
- $24x^2 y^5 z^3$ is a monomial in three variables x,y,z with exponents 2,5 and 3 respectively.

Polynomial: A polynomial is formed by a finite set of monomials that relate with each other through the operators of addition and subtraction.

- The order of the polynomial is defined as the order of the highest degree monomial present in the mathematical statement.
- $2x^3 + 4x^2 + 3x - 7$ is a polynomial of order 3 in a single variable.
- Polynomials also exist in multiple variables.
- $x^3 + 4x^2y + xy^5 + y^2 - 2$ is a polynomial in variables x and y.

COEFFICIENT

"A coefficient is the number multiplied by the variable in an algebraic expression".

Algebraic Expression	Coefficient
6m + 5	6
8r + 7m + 4	8, 7
14b - 8	14

In the expression 7x + 9y + 15, 7x, 9y, and 15 are called terms.

- A term can be a number, a variable, or a product of numbers and variables.
- Terms in an expression are separated by + and –.

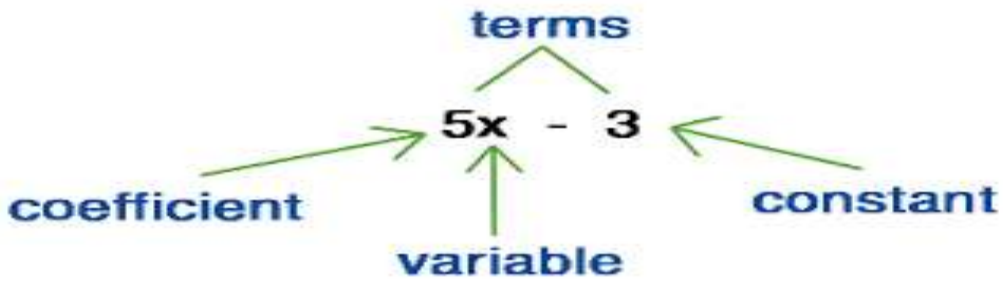
7x + 5 – 3y2 + y + x/3

term term term term term

In the term 7x, 7 is called the coefficient. A coefficient is a number that is multiplied by a variable in an algebraic expression.

A variable by itself, like y, has a coefficient of 1. So y = 1y.

7x => coefficient = 7
variable = x



Ex: Find the coefficient of p in p3 + 2p2 - 5p - 1.

Sol: The coefficient of p in the expression p3 + 2p2 - 5p - 1 is -5.

Coefficient of Algebraic Expressions Properties

1. the coefficient may be positive or negative
2. When we are performing, addition or subtraction operation in polynomials, we need to add or subtract the coefficients of the like terms.
3. Depending up on the coefficient, the term can be named as positive term or negative term.

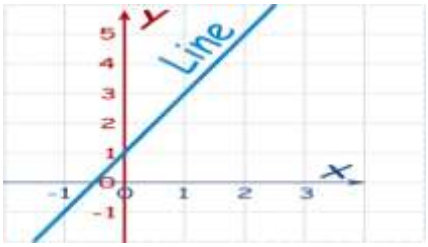
For example -3 x --- negative term

3x ---- positive term

LINEAR EQUATIONS

A **linear equation in one variable** is an equation which can be written in the form:

$ax + b = c$



Why are the following Not linear equations in one variable.

$2x + 3y = 11$	$(x - 1)^2 = 8$	$\frac{2}{3x} + 5 = x - 7$
Two variables	x is squared.	variable in the denominator

Types of linear equations:

There are three types of linear equations.

- The most common type is a conditional it has one solution
- A linear equation whose solution is all real numbers is called an identity
- Linear equations that have no solution are called contradictions

A **solution** of a linear equation in one variable is a real number which, when substituted for the variable in the equation, makes the equation true.

Example: Is 3 a solution of 2x + 3 = 11?

2x + 3 = 11

2(3) + 3 = 11

6 + 3 = 11 false equation

=> 3 is not a solution of 2x + 3 = 11.

Example: Is 4 a solution of 2x + 3 = 11?

2x + 3 = 11

2(4) + 3 = 11

8 + 3 = 11 true equation

=> 4 is a solution of 2x + 3 = 11.

To solve linear equations, we will make heavy use of the following facts.

1. If $a=b$ then $a+c=b+c$ for any c. All this is saying is that we can add a number, c, to both sides of the equation and not change the equation.
2. If $a=b$ then $a-c=b-c$ for any c. As with the last property we can subtract a number, c, from both sides of an equation.
3. If $a=b$ then $ac=bc$ for any c. Like addition and subtraction. we can multiply both sides of an equation by a number, c, without changing the equation.
4. If $a=b$ then $\frac{a}{c}=\frac{b}{c}$ for any non-zero c. We can divide both sides of an equation by a non-zero number, c, without changing the equation.

Graphs of functions:

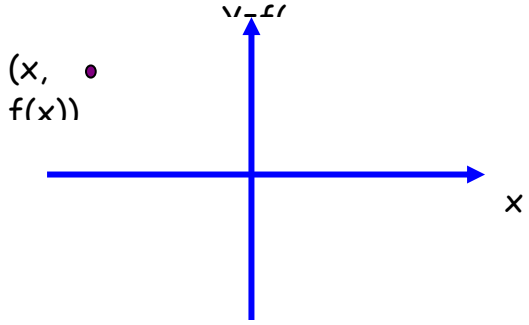
Graphs of functions: A graph of a function is a visual representation of a function's behavior on an x-y plane.

- Graphs help us understand different aspects of the function, which would be difficult to understand by just looking at the function itself.
- You can graph thousands of equations, and there are different formulas for each one. That said, there are always ways to graph a function if you forget the exact steps for the specific type of function.

The graph of a function f is the set of all points in the plane of the form (x, f(x)). We could also define the graph of f to be the graph of the equation y = f(x). So, the graph of a function if a special case of the graph of an equation.

- Variable x is called independent variable

- Variable y is called dependent variable
- For convenience, we use f(x) instead of y.
- The ordered pair in new notation becomes:
- (x, y) = (x, f(x))

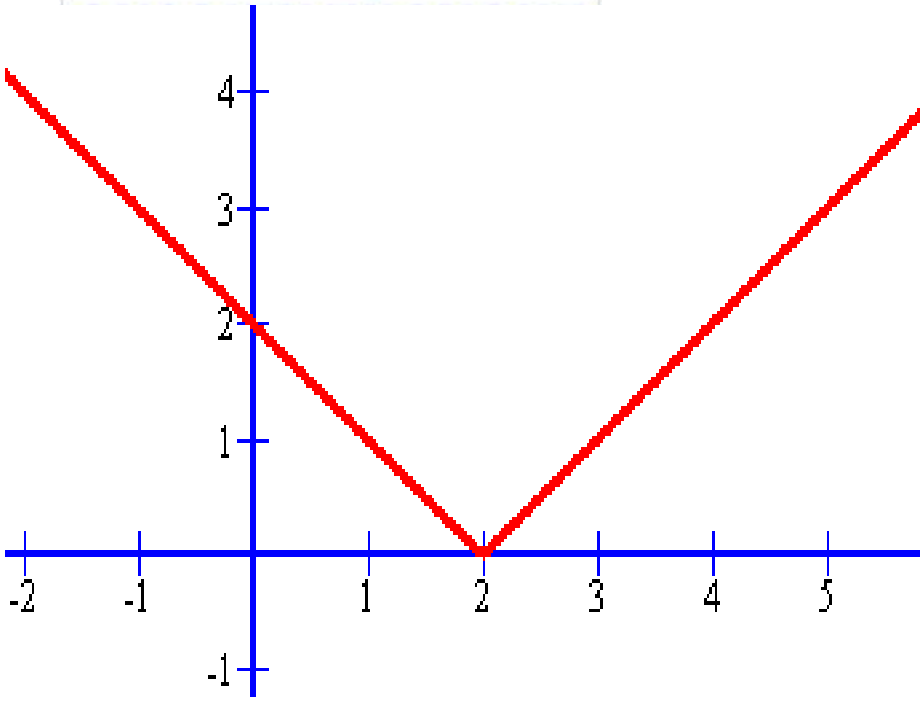


Graphing steps:

1) Isolate the variable (solve for y).
2) Make a t-table. If the domain is not given, pick your own values.
3) Plot the points on a graph.
4) Connect the points.

A **function** is a relation in which each element of the domain is paired with exactly one element of the range. Another way of saying it is that there is one and only one output (y) with each input (x).

X →f(x)→Y



Example 1.

Let **f(x) = x² - 3**.

The graph of f(x) in this example is the graph of $y = x^2 - 3$. It is easy to generate points on the graph. Choose a value for the first coordinate, then evaluate f at that number to find the second coordinate. The following table shows several values for x and the function f evaluated at those numbers.

Each column of numbers in the table holds the coordinates of a point on the graph of f.

Summary of Graphs of Common Functions:

Histograms:

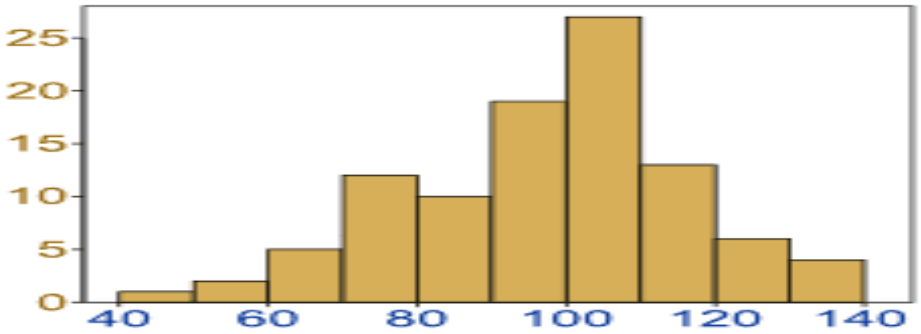
Histograms: - A graphical display of data using bars of different heights.

Def 1:- A **histogram** is a visual way to display frequency data using bars. A feature of **histograms** is that they show the frequency of continuous data, such as the number of trees at various heights from 3 feet to 8 feet.

General Characteristics:

- Column label is quantitative variable (ages).
- Column label is a range of values (or single value).
- Column height is size of the group.
- Columns NOT separated by space.
- Calculate mean, median, quartiles, standard deviation, and so on.

It is similar to a Bar Chart, but a histogram groups numbers into **ranges.**



Histograms are a great way to show results of continuous data, such as:

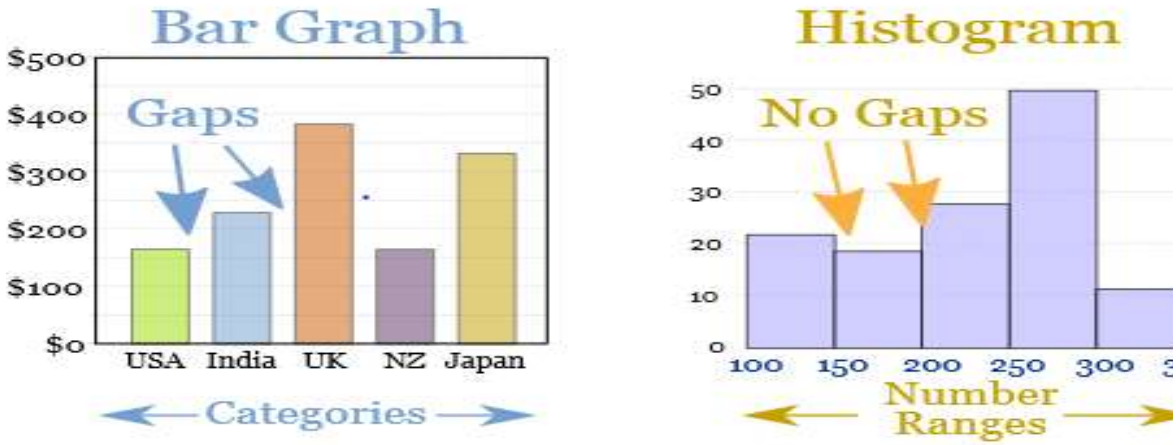
- weight
- height
- how much time. etc.

But when the data is in **categories** (such as Country or Favorite Movie), we should use a Bar Chart.

Histograms versus Bar Graphs:

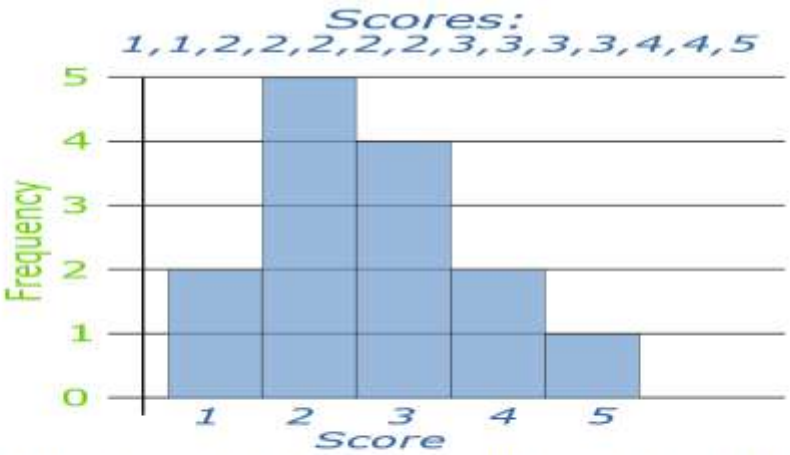
1. Histograms are used to show distributions of variables, while bar graphs are used

to compare variables.
2. Histograms plot quantitative (numerical) data with ranges of the data grouped into intervals, while bar graphs plot categorical data.
3. The bars in a bar graph can be rearranged, but it does not make sense to rearrange the bars in a histogram.
4. It is possible to speak of the skewness of a histogram, but not of a bar graph.
5. Bar graphs have space between the columns, while histograms do not.



Frequency Histogram:

A Frequency Histogram is a special histogram that uses vertical columns to show frequencies (how many times each score occurs):



Here I have added up how often 1 occurs (2 times), how often 2 occurs (5 times), etc, and shown them as a histogram.

Cumulative Frequency Histograms:

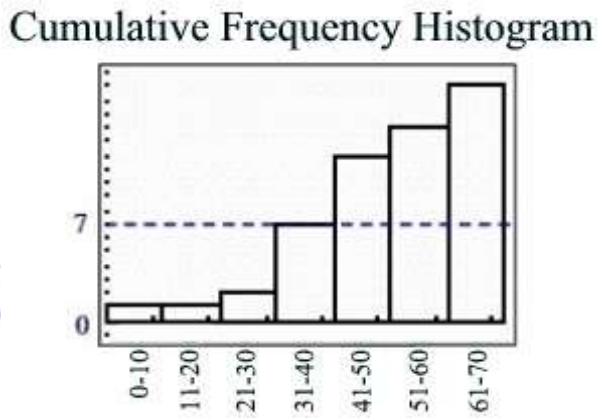
- The term "cumulative frequency" refers to the running total of the frequencies.

- Each interval now contains a frequency number which represents that specific interval's count added to the sum of all the previous intervals' counts.
- A cumulative frequency histogram will always contain column bars that get increasingly taller (or stay the same height) as you move to the right.

Data set: {9, 25, 30, 31, 34, 36, 37, 42, 45, 47, 49, 43, 55, 58, 61, 63, 67}

Cumulative Frequency Table

Interval	Count (frequency)	Cumulative frequency
0-10	1	1
11-20	0	1 + 0 = 1
21-30	1	1 + 1 = 2
31-40	5	2 + 5 = 7
41-50	5	7 + 5 = 12
51-60	2	12 + 2 = 14
61-70	3	14 + 3 = 17



Notice that the columns increase in height, o stay the same, as you move to the right.

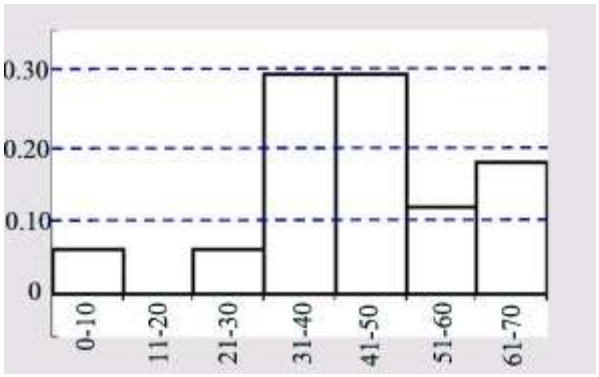
Relative Frequency Histograms:

- While the term "frequency" refers to the number of observations (or counts) of a given piece of data, the term "relative frequency" refers to the number of observations (or counts) expressed as a "part of the whole" or percentage.
- Each of the number of observations is divided by the total number of observations from the entire data set.
- In the data set we have been examining, there are 2 pieces of data in the interval "51-60", but there are 17 total pieces of data.
- The relative frequency of the 2 pieces of data in the interval "51-60" is 2/17 or 0.12 (to the nearest hundredth) or 12%. Notice that when you add all of the relative frequencies in the chart below, you get a value of one (or 100%).
- Relative frequency may be expressed as a fraction (ratio), a decimal, or a percentage.

Data set: {9, 25, 30, 31, 34, 36, 37, 42, 45, 47, 49, 43, 55, 58, 61, 63, 67}

Relative Frequency Table Relative Frequency Histogram

Interval	Count (frequency)	Relative frequency
0-10	1	1/17 = 0.06
11-20	0	0/17 = 0
21-30	1	1/17 = 0.06
31-40	5	5/17 = 0.29
41-50	5	5/17 = 0.29
51-60	2	2/17 = 0.12
61-70	3	3/17 = 0.18
	Total = 17	Total = 1



The intervals appear on the horizontal axis. The relative frequency (as a percentage in decimal form) appears on the vertical axis. The graph is the same shape as the "frequency" histogram, but with a different y-scale.

Histograms: Pros and Cons

- While a box plot emphasizes center, and spread of a distribution, a histogram emphasizes the distribution of values.
- Histograms clearly show the shape of the data.
- In addition, values occurring with high frequency are easier to identify in a histogram than in a box plot.
- With histograms, however, the center and spread are harder to identify, exact values cannot be read due to the grouping of the data, and it is difficult to compare two data set.

When to Use a Histogram:

- How do you know when to use a histogram? You can decide this by looking at your data. Ask yourself, 'Is the data continuous, or can I group the data into ranges?'
- What you are looking for is a group of data that is **continuous**. What this means is that the data covers a range of values that does not jump.
- For example, the range of tree heights is 3 to 8 feet. The data does not jump from 4 to 6 feet. There is no gap. The countries of Norway, Finland, and Sweden, though, do jump.
- There is no continuity between the countries. They are separate entities. For histograms, you need continuous information and not just categories that jump.

Logarithms:

- In mathematics, the logarithm is the inverse function to exponentiation. ...
- The logarithm to base 10 (that is b = 10) is called the common logarithm and has many applications in science and engineering. ...
- Two kinds of logarithms are often used : common (or Briggian) logarithms and natural (or Napierian) logarithms.
- The power to which a base of 10 must be raised to obtain a number is called the common **logarithm** (log) of the number.
- The power to which the base e (e = 2.718281828.....) must be raised to obtain a number is called the **natural logarithm** (ln) of the number.
- Using $\log_{10}$ ("log to the base 10"): $\log_{10}100 = 2$ is equivalent to $10^2 = 100$ where 10 is the base, 2 is the logarithm (i.e., the exponent or power) and 100 is the number.
- Using natural logs ($\log_e$ or ln):
  Carrying all numbers to 5 significant figures,
  ln 30 = 3.4012 is equivalent to $e^{3.4012} = 30$ or $2.7183^{3.4012} = 30$

A logarithm is just an exponent.

1) A logarithm is the opposite of a power. In other words, if we take a logarithm of a number, we undo an exponentiation.

To be specific, the logarithm of a number x to a base b is just the exponent you put onto b to make the result equal x.

generically, if $x = b^y$, then we say that y is "the logarithm of x to the base b" or "the base-b logarithm of x".

For instance, since $5^2 = 25$, we know that 2 (the power) is the logarithm of 25 to base 5. Symbolically, $\log_5(25) = 2$.

exponential equation is equivalent to a logarithmic equation:

$x = b^y$ is the same as $y = \log_b x$

Log Rules:

1) $\log_b(mn) = \log_b(m) + \log_b(n)$

2) $\log_b(^m/_n) = \log_b(m) - \log_b(n)$

3) $\log_b(m^n) = n \cdot \log_b(m)$

- Definition of **Common Logarithm**: Logarithms with a base of 10 are called common logarithms. It is customary to write $\log_{10}x$ as logx
- Definition of **Natural Logarithm**: Logarithms with the base of e are called natural logarithms. It is customary to write $\log_e x$ as ln

Derivatives

The **derivative** of a function of a real variable measures the sensitivity to change of the function value (output value) with respect to a change in its argument (input value).

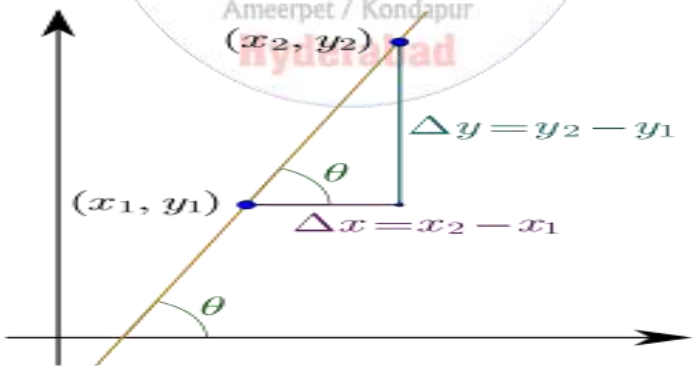
- Derivatives are a fundamental tool of calculus.
- For example, the derivative of the position of a moving object with respect to time is the object's velocity: this measures how quickly the position of the object changes when time advances.

Differentiation is the action of computing a derivative.

- The derivative of a function y = f(x) of a variable x is a measure of the rate at which the value y of the function changes with respect to the change of the variable x.
- It is called the derivative of f with respect to x. If x and y are real numbers, and if the graph of f is plotted against x, the derivative is the slope of this graph at each point.

$$m = f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

Called the derivative of f(x) ,Computing Called differentiation $f'(x)$



Slope of a linear function:

A derivative is a contract between two parties which derives its value/price from an underlying asset.

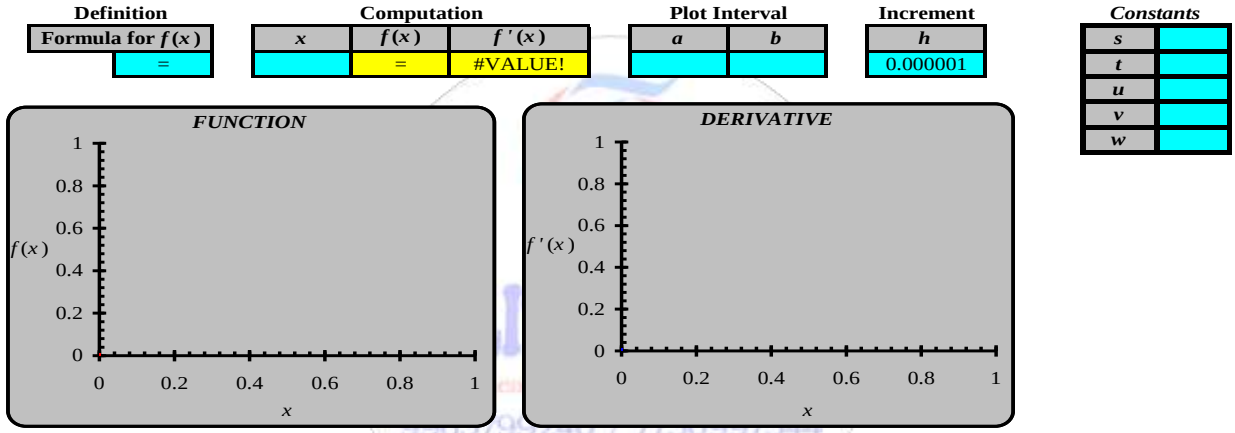
The Derivative is the exact rate at which one quantity changes with respect to another.

2. Geometrically, the derivative is the slope of curve at the point on the curve.

3. The derivative is often called the “instantaneous “ rate of change.

4. The derivative of a function represents an infinitely small change the function with respect to one of its variables. • The Process of finding the derivative is called “differentiation.”

- Numerical differentiation is used to avoid tedious difference quotient calculations
- Differentiating.xls file (Numerical differentiation utility)
- Graphs both function and derivative
- Can evaluate function and derivative



- Tangent line approximations
- Useful for easy approximations to complicated functions
- Need a point and slope (derivative)
- Use y = mx +b

Uses of Algebra in Machine Learning:

- Algebras have played an important role in logic and top-down approaches in Artificial Intelligence (AI).
- We show how to express learning problems as elements and relationships in an extended semi lattice algebra.
- We give a concrete algebraic algorithm, the Sparse Crossing, that finds solutions as sets of \atomic" elements, or atoms.
- The algebraic approach has important differences to more standard approaches. It does not use function minimization. Minimizing functions has proven very useful in ML.

- algebraic algorithm uses cardinal minimization. i.e. minimization of the number of atoms.
- Algebraic Learning is designed to \compress" training examples into atoms and not directly aimed at reducing the error.
- We found no evidence of over fitting using algebraic learning, so we do not use a validation dataset. Also, it is parameter-free, so there is no need to reallocate parameter values like a network architecture, with the algebra growing by itself using the training data.

Statistics

- Many studies generate large numbers of data points, and to make sense of all that data, researchers use statistics that summarize the data, providing a better understanding of overall tendencies within the distributions of scores.
- Statistics is a branch of mathematics dealing with the collection, analysis, interpretation, presentation, and organization of data.
- Statistics deals with all aspects of data including the planning of data collection in terms of the design of surveys and experiments.
- Two main statistical methods are used in data analysis: descriptive statistics, which summarize data from a sample using indexes such as the mean or standard deviation, and inferential statistics, which draw conclusions from data that are subject to random variation (e.g., observational errors, sampling variation).
- Descriptive statistics are most often concerned with two sets of properties of a distribution (sample or population): central tendency (or location) seeks to characterize the distribution's central or typical value,while dispersion (or variability) characterizes the extent to which members of the distribution depart from its center and each other.
- Inferences on mathematical statistics are made under the framework of probability theory, which deals with the analysis of random phenomena.

Statistics: The science of collecting, describing, and interpreting data.

We use statistics for many reasons:

- To mathematically describe/depict our findings
- To draw conclusions from our results
- To test hypotheses
- To test for relationships among variables
- aids in summarizing the results
- helps us recognize underlying trends and tendencies in the data
- aids in communicating the results to others

Two areas of statistics:

Descriptive Statistics: collection, presentation, and description of sample data.

Inferential Statistics: making decisions and drawing conclusions about populations.

Types of statistics:

1. descriptive (which summarize some characteristic of a sample)
    - Measures of central tendency
    - Measures of dispersion

- Measures of skewness

2. inferential (which test for significant differences between groups and/or significant relationships among variables within the sample
    - t-ratio, chi-square, beta-value

Descriptive statistics:

- A descriptive statistic (in the count noun sense) is a summary statistic that quantitatively describes or summarizes features of a collection of information, while descriptive statistics in the mass noun sense is the process of using and analyzing those statistics.
- Descriptive statistics is distinguished from inferential statistics (or inductive statistics), in that descriptive statistics aims to summarize a sample, rather than use the data to learn about the population that the sample of data is thought to represent.
- This generally means that descriptive statistics, unlike inferential statistics, is not developed on the basis of probability theory, and are frequently nonparametric statistics.
- Even when a data analysis draws its main conclusions using inferential statistics, descriptive statistics are generally also presented.
- For example, in papers reporting on human subjects, typically a table is included giving the overall sample size, sample sizes in important subgroups (e.g., for each treatment or exposure group), and demographic or clinical characteristics such as the average age, the proportion of subjects of each sex, the proportion of subjects with related comorbidities, etc.

Descriptive Statistics:

- Frequencies
- Basic measurements

Descriptive statistics can be used to summarize and describe a single variable (aka, UNI variate)

->Frequencies (counts) & Percentages
-Use with categorical (nominal) data
Levels, types, groupings, yes/no, Drug A vs. Drug B
->Means & Standard Deviations
-Use with continuous (interval/ratio) data
Height, weight, cholesterol, scores on a test

Can be applied to any measurements (quantitative or qualitative)

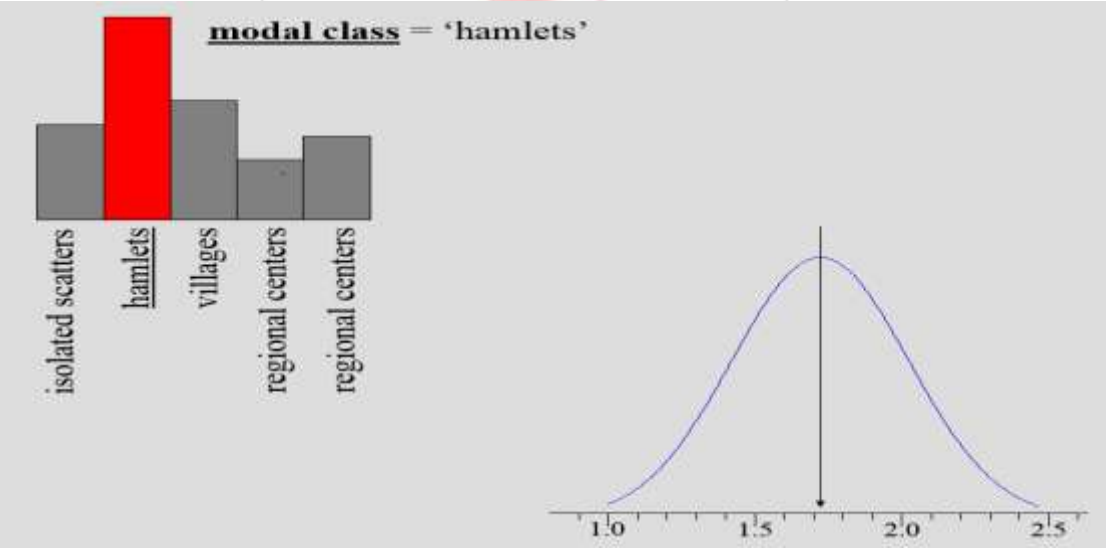
Offers a summary/ overview/ description of data. Does not explain or interpret.

- Number
- Frequency Count
- Percentage
- Deciles and quartiles
- Measures of Central Tendency (Mean, Midpoint, Mode)
- Variability
- Variance and standard deviation
- Graphs
- Normal Curve

Averages:

Mode: most frequently occurring value in a distribution (any scale, most unstable).

- the most numerous category
- for ratio data, often implies that data have been grouped in some way
- can be more or less created by the grouping procedure
- for theoretical distributions—simply the location of the peak on the frequency distribution



Advantages:	Disadvantages:
• Quick and easy to compute	Terminal Statistic
• Unaffected by extreme scores	A given sub-group could make this measure unrepresentative.
• Can be used at any level of measurement.	

Mean: arithmetic average- the sum of all values in a distribution divided by the number of

cases (interval or ratio)

- center of gravity
- evenly partitions the sum of all measurement among all cases; average of all measures $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

mean pros and cons:

- crucial for inferential statistics
- mean is not very resistant to outliers
- a "trimmed mean" may be better for descriptive purposes

rim diameter (cm)

	unit 1	unit 2
	12.6	16.2
	11.6	16.4
	16.3	13.8
	13.1	13.2
	12.1	11.3
	26.9	14.0
	9.7	9.0
	11.5	12.5
	14.8	15.6
	13.5	11.2
	12.4	12.2
	13.6	15.5
		11.7
n	12	13
total	168.1	172.6
total/n	14.0	13.3

	unit 1		unit 2	
	9	26		
		25		
		24		
		23		
		22		
		21		
		20		
		19		
		18		
		17		
	3	16	24	
		15	56	
14.0==	8	14	0	
	651	13	28	==13.3
	641	12	25	
	65	11	237	
		10		
	7	9	0	

R: mean(x)

- Arithmetic Mean

-Harmonic Mean

-Geometric Mean

also.. -f mean

-Truncated mean ,-Power mean,-Weighted arithmetic mean ,-Chisini mean -

Identic mean, etc,

Median: midpoint in the distribution below which half of the cases reside (ordinal and above)

- 50th percentile…
- less useful for inferential purposes
- more resistant to effects of outliers…

rim diameter (cm)

unit 1	unit 2	
9.7	9.0	
11.5	11.2	
11.6	11.3	
12.1	11.7	
12.4	12.2	
12.6	12.5	
12.9 <--	13.2	13.2
13.1	13.8	
13.5	14.0	
13.6	15.5	
14.8	15.6	
16.3	16.2	
26.9	16.4	

unit 1		unit 2	
9	26		
	25		
	24		
	23		
	22		
	21		
	20		
	19		
	18		
	17		
3	16	24	
	15	56	
8	14	0	
651	13	28	==13.20
12.85== 641	12	25	
65	11	237	
	10		
7	9	0	

The **mean** or average of the data values is

$$mean = \frac{sum\ of\ all\ data\ values}{number\ of\ data\ values} \qquad mean = \frac{x_1+x_2+\cdots+x_n}{n} = \frac{\sum x}{n}$$

- Sample mean: $\bar{x}$
- Population mean: μ (“mu”)

outlier

outliers : An **outlier** is any value that is numerically distant from most of the other data points in a set of data. It is not uncommon to find an outlier in a data set.

- In statistics, an outlier is an observation point that is distant from other observations.
- An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set.
- An outlier can cause serious problems in statistical analyses.
- Outliers can occur by chance in any distribution, but they often indicate

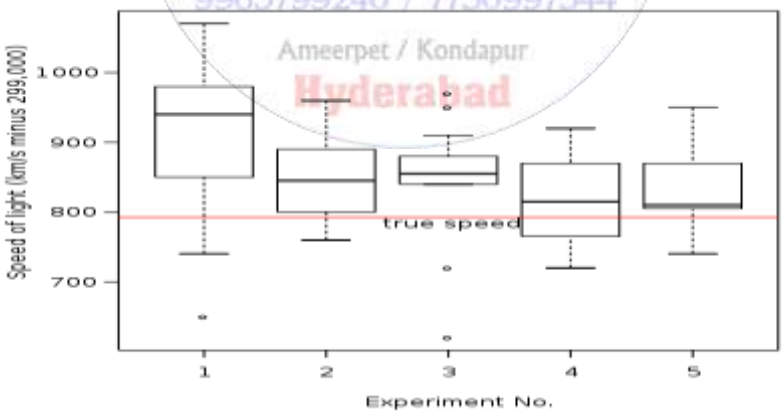
either measurement error or that the population has a heavy-tailed distribution.

An outlier is an observation that is unlike the other observations.

→It is rare, or distinct, or does not fit in some way.

Outliers can have many causes, such as:

- Measurement or input error.
- Data corruption.
- True outlier observation (e.g. Michael Jordan in basketball).
- There is no precise way to define and identify outliers in general because of the specifics of each dataset. Instead, you, or a domain expert, must interpret the raw observations and decide whether a value is an outlier or not.
- Nevertheless, we can use statistical methods to identify observations that appear to be rare or unlikely given the available data.
- This does not mean that the values identified are outliers and should be removed. But, the tools described in this tutorial can be helpful in shedding light on rare events that may require a second look.
- A good tip is to consider plotting the identified outlier values, perhaps in the context of non-outlier values to see if there are any systematic relationship or pattern to the outliers. If there is, perhaps they are not outliers and can be explained, or perhaps the outliers themselves can be identified more systematically.



Ways to describe data

Two activities are essential for characterizing a set of data:

1. Examination of the overall shape of the graphed data for important features, including symmetry and departures from assumptions. The chapter on Exploratory Data Analysis (EDA) discusses

assumptions and summarization of data in detail.

2. Examination of the data for unusual observations that are far removed from the mass of data. These points are often referred to as outliers. Two graphical techniques for identifying outliers, scatter plots and box plots, along with an analytic procedure for detecting outliers when the distribution is normal (Grubbs' Test), are also discussed in detail in the EDA chapter.

Box plot construction

The box plot is a useful graphical display for describing the behavior of the data in the middle as well as at the ends of the distributions. The box plot uses the median and the lower and upper quartiles (defined as the 25th and 75th percentiles). If the lower quartile is Q1 and the upper quartile is Q3, then the difference (Q3 - Q1) is called the interquartile range or IQ.

Box plots with fences

A box plot is constructed by drawing a box between the upper and lower quartiles with a solid line drawn across the box to locate the median. The following quantities (called fences) are needed for identifying extreme values in the tails of the distribution:

1. lower inner fence: Q1 - 1.5*IQ
2. upper inner fence: Q3 + 1.5*IQ
3. lower outer fence: Q1 - 3*IQ
4. upper outer fence: Q3 + 3*IQ

Outlier detection criteria

A point beyond an inner fence on either side is considered a **mild outlier**. A point beyond an outer fence is considered an **extreme outlier**.

Example of an outlier box plot

The data set of N = 90 ordered observations as shown below is examined for outliers:

30, 171, 184, 201, 212, 250, 265, 270, 272, 289, 305, 306, 322, 322, 336, 346, 351, 370, 390, 404, 409, 411, 436, 437, 439, 441, 444, 448, 451, 453, 470, 480, 482, 487, 494, 495, 499, 503, 514, 521, 522, 527, 548, 550, 559, 560, 570, 572, 574, 578, 585, 592, 592, 607, 616, 618, 621, 629, 637, 638, 640, 656, 668, 707, 709, 719, 737, 739, 752, 758, 766, 792, 792, 794, 802, 818, 830, 832, 843, 858, 860, 869, 918, 925, 953, 991, 1000, 1005, 1068, 1441

The above data is available as a text file.

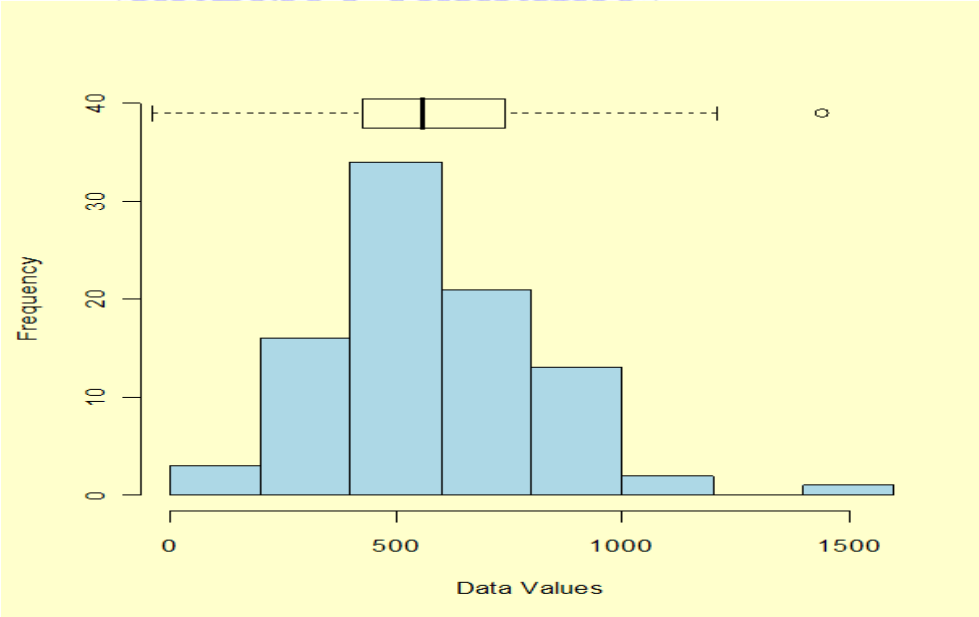
The computations are as follows:

- Median = (n+1)/2 largest data point = the average of the 45th and 46th ordered points = (559 + 560)/2 = 559.5
- Lower quartile = .25(N+1)th ordered point = 22.75th ordered point = 411 + .75(436-411) = 429.75
- Upper quartile = .75(N+1)th ordered point = 68.25th ordered point = 739 +.25(752-739) = 742.25
- Interquartile range = 742.25 - 429.75 = 312.5
- Lower inner fence = 429.75 - 1.5 (312.5) = -39.0
- Upper inner fence = 742.25 + 1.5 (312.5) = 1211.0
- Lower outer fence = 429.75 - 3.0 (312.5) = -507.75
- Upper outer fence = 742.25 + 3.0 (312.5) = 1679.75

From an examination of the fence points and the data, one point (1441) exceeds the upper inner fence and stands out as a mild outlier; there are no extreme outliers.

Histogram with box plot

A histogram with an overlaid box plot are shown below.



The outlier is identified as the largest value in the data set, 1441, and appears as the circle to the right of the box plot.

Outliers

Outliers should be investigated carefully. Often, they contain valuable

may contain important information	information about the process under investigation or the data gathering and recording process. Before considering the possible elimination of these points from the data, one should try to understand why they appeared and whether it is likely similar values will continue to appear. Of course, outliers are often bad data points.

Where Do Outliers Come From?

- The most common source of outliers is measurement error.

For example, it could be that there were battery problems with the timer that caused the alarm to go off before the runner's 60 seconds were up.

- Another cause of outliers is experimental error.

For example, it could be that the running signal was not loud enough for all of the athletes to hear, resulting in one runner having a late start. This would put the runner's time far below that of the other runners.

Other sources of outliers include:

- Human error (i.e. errors in data entry or data collection)
- Participants intentionally reporting incorrect data (This is most common in self-reported measures and measures that involve sensitive data, i.e. teens underreporting the amount of alcohol that they use on a survey)
- Sampling error (i.e. including high school basketball players in the sample even though the research study was only supposed to be about high school track runners)

If it is determined that an outlier is due to some type of error (i.e. measurement or experimental error), then it is okay to exclude the data point from the analysis.

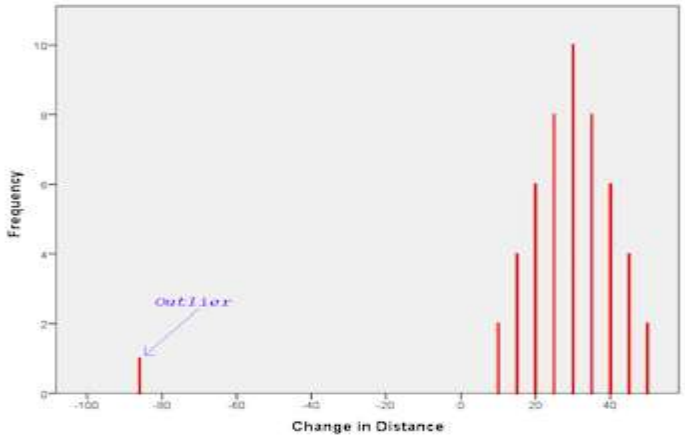
However, if the outlier was due to chance or some natural process of the construct that is being measured, it should not be removed.

Detecting Outliers

The easiest way to detect an outlier is by creating a graph. We can spot outliers by using histograms, scatterplots, number lines, and the interquartile range.

Histogram :

Suppose that we were asked to create a histogram using the data that we collected from the high school track runners. The following is the histogram of the change in distance for each of the track runners.



N percentiles

A "**percentile"** is a comparison score between a particular score and the scores of the rest of a group. It shows the percentage of scores that a particular score surpassed.

The percentile rank is calculated using the formula :

R=P/100(N)

where P is the desired percentile and N is the number of data points.

Where we use percentile: -

Percentiles are commonly used to report scores in tests, like the SAT, GRE and LSAT.

Definition 1: The nth percentile is the lowest score that is greater than a certain percentage ("n") of the scores. In this example, our n is 25, so we're looking for the lowest score that is greater than 25%.

Definition 2: The nth percentile is the smallest score that is greater than **or equal to** a certain percentage of the scores. To rephrase this, it's the percentage of data that falls at or below a certain observation. **This is the definition used in AP statistics.**

- Are standard scores and may be used to compare scores of different measurements.
- Change at different rates (remember comparison of low and and high percentile scores with middle percentiles), so they should not be used to determine one score for several different tests.
- May prefer to use T-scale when converting raw scores to standard scores.

Example 1:

If the scores of a set of students in a math test are 20 , 30 , 15 and 75 what is the percentile rank of the score 30 ?

Arrange the numbers in ascending order and give the rank ranging from 1 to the lowest to 4 to the highest.

Number	15	20	30	
Rank	1	2	3	4

Use the formula:

$3 = \frac{P}{100}(4)$

$3 = \frac{P}{25}$

$75 = P$

Therefore, the score 30 has the 75 th percentile.

Variance:

- analogous to average deviation of cases from mean
- in fact, based on sum of squared deviations from the mean—"sum-of-squares"

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

R = VAR(X)

- computational form:

$$s^2 = \frac{\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 / n}{n-1}$$

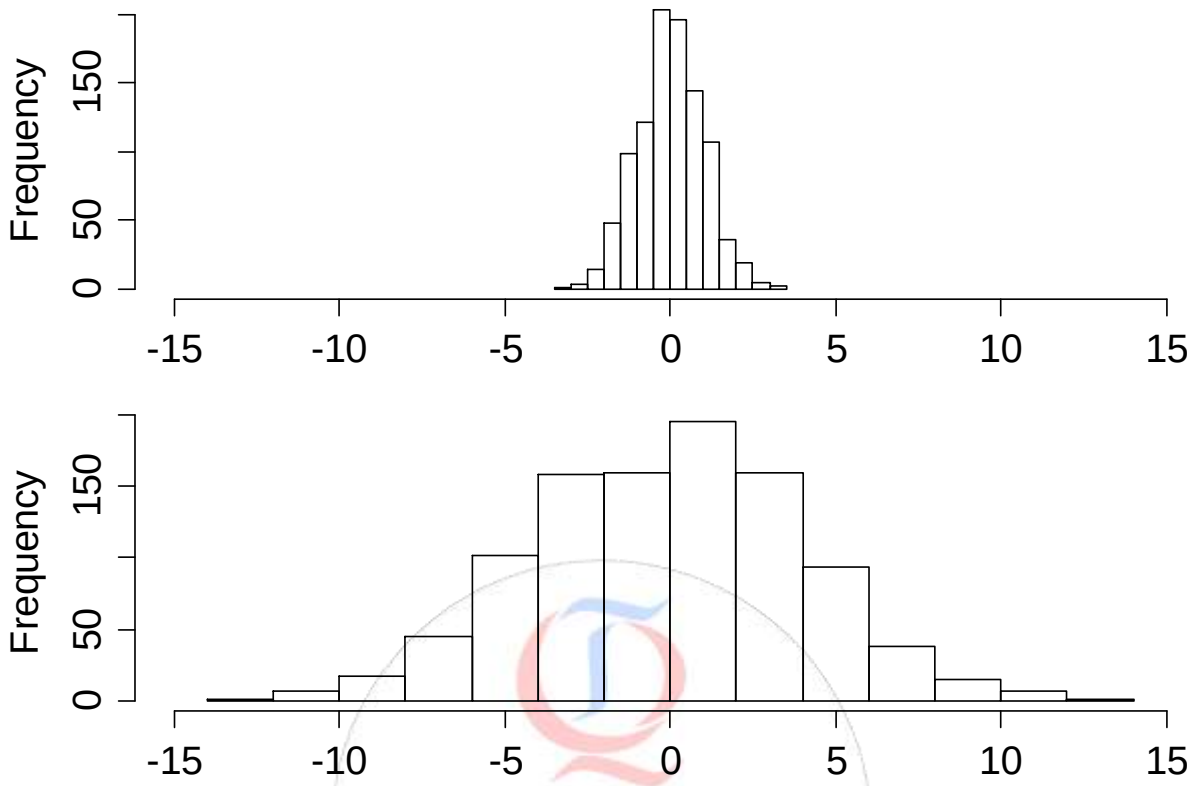
- note: units of variance are squared…
- this makes variance hard to interpret
- ex.: projectile point sample:

mean = 22.6 mm

variance = 38 mm^2

standard deviation:

- The standard deviation gives a rough estimate of the typical distance of a data values from the mean
- The larger the standard deviation, the more variability there is in the data and the more spread out the data are



Both of these distributions are **bell-shaped**

Square root of the Variance:

- expressed in the original units of measurement
- Represents the average amount of dispersion in a sample
- Used in a number of inferential statistics

- square root of variance:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \qquad s = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 / n}{n-1}}$$

- units are in same units as base measurements
- ex.: projectile point sample:

  mean = 22.6 mm

  standard deviation = 6.2 mm
- mean +/- sd (16.4—28.8 mm)
  - should give at least some intuitive sense of where most of the cases lie, barring major effects of outliers

rim diameter (cm)

unit 1	unit 2	unit 1	unit 2	unit 1	unit 2
12.6	16.2	-1.4	2.9	1.98	8.54
11.6	16.4	-2.4	3.1	5.80	9.75
16.3	13.8	2.3	0.5	5.25	0.27
13.1	13.2	-0.9	-0.1	0.83	0.01
12.1	11.3	-1.9	-2.0	3.64	3.91
26.9	14.0	12.9	0.7	166.20	0.52
9.7	9.0	-4.3	-4.3	18.56	18.29
11.5	12.5	-2.5	-0.8	6.29	0.60
14.8	15.6	0.8	2.3	0.63	5.40
13.5	11.2	-0.5	-2.1	0.26	4.31
12.4	12.2	-1.6	-1.1	2.59	1.16
13.6	15.5	-0.4	2.2	0.17	4.94
	11.7		-1.6		2.49
mean:	14.0	13.3		sum of sq.:	212.19	60.20
n:	12	13		variance:	19.29	5.02
				stand. dev.:	4.39	2.24

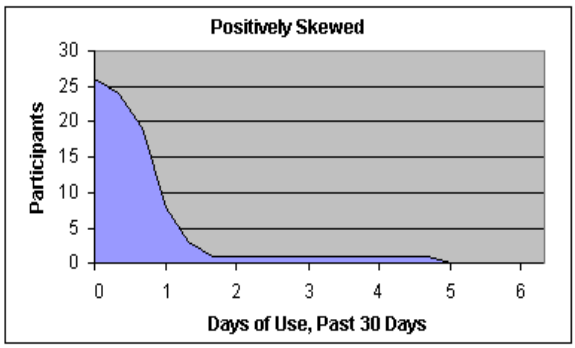
Shape of data measured by

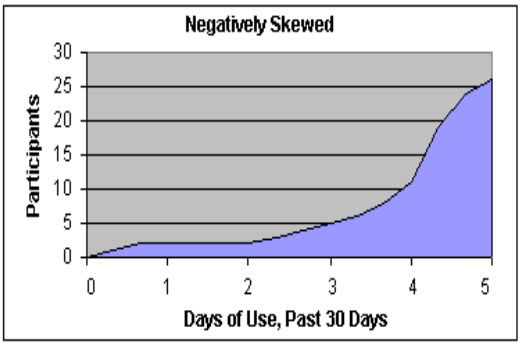
Skewness

Kurtosis

Skewness(symmetry):

- Measures look at how lopsided distributions are—how far from the ideal of the normal curve they are
- When the median and the mean are different, the distribution is skewed. The greater the difference, the greater the skew.
- Distributions that trail away to the left are negatively skewed and those that trail away to the right are positively skewed
- If the skewness is extreme, the researcher should either transform the data to make them better resemble a normal curve or else use a different set of statistics—nonparametric statistics—to carry out the analysis





Skewness

- Measures of asymmetry of data
  - Positive or right skewed: Longer right tail
  - Negative or left skewed: Longer left tail

Let $x_1, x_2, \ldots x_n$ be n observations. Then,

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

Indicators of Skewness:

- Frequency curve is not Symmetrical bell shaped.
- Values of Mean, Median, and Mode do not coincide.
- Sum of positive deviation is not equal to sum of negative deviation.

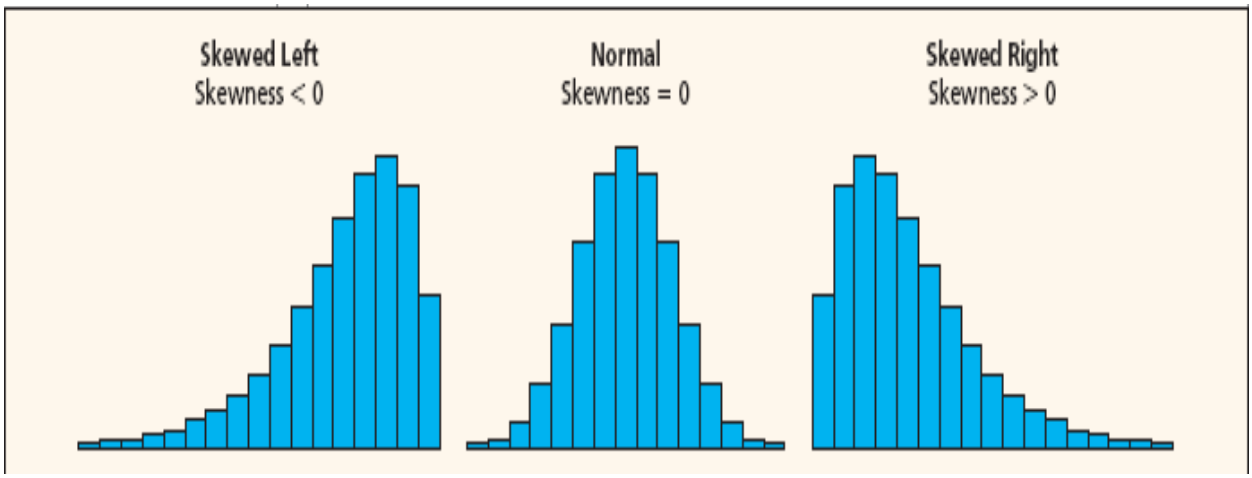
Measures of Skewness:

Karl Pearson's coefficient of skewness,

Sk = Mean - Mode

Standard Deviation

- Generally, skewness may be indicated by looking at the sample histogram or by comparing the mean and median.

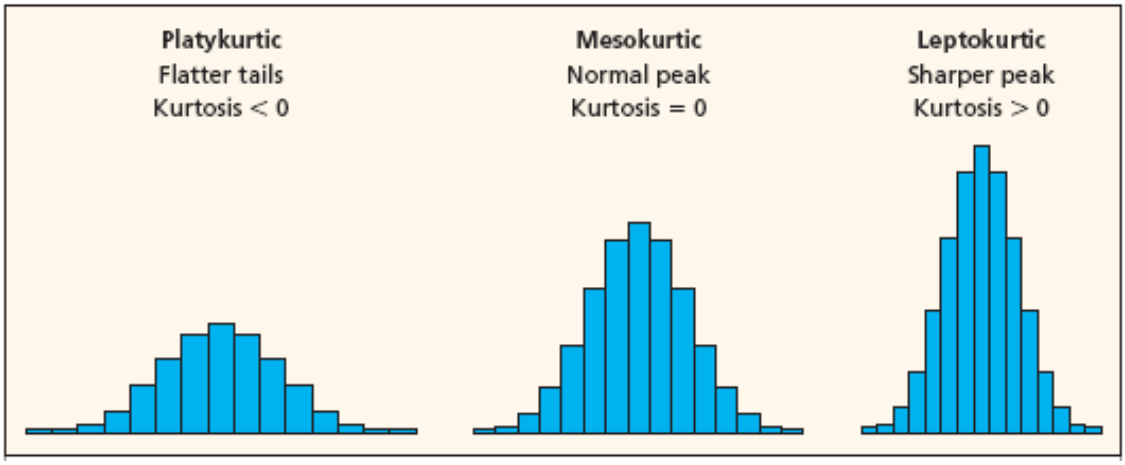


- This visual indicator is imprecise and does not take into consideration sample size n.

Kurtosis

Kurtosis:- It is concerned with the degree of Flatness or Peakedness in a curve.

- Kurtosis is the relative length of the tails and the degree of concentration in the center.
- Consider three kurtosis prototype shapes.



- **kurtosis** (from Greek: κυρτός, kyrtos or kurtos, meaning "curved, arching") is a measure of the "tailedness" of the probability distribution of a real-valued random variable.
- kurtosis is a descriptor of the shape of a probability distribution and, just as for skewness, there are different ways of quantifying it for a theoretical distribution and corresponding ways of estimating it from a sample from a population.
- Depending on the particular measure of kurtosis that is used, there are various interpretations of kurtosis, and of how particular measures should be interpreted.

Types of Kurtosis:-

- Leptokurtic: A curve which is more peaked then the normal.

- Mesokurtic: A normal curve is called mesokurtic curve.
- PlatyKurtic: A flat curve than normal is called platykurtic.

Measures of Kurtosis:-

$\beta_2 = \mu_4/\mu_2^2$

- μ_4 = Forth moment about mean
- μ_2 = Second moment about mean

Interpretation: -

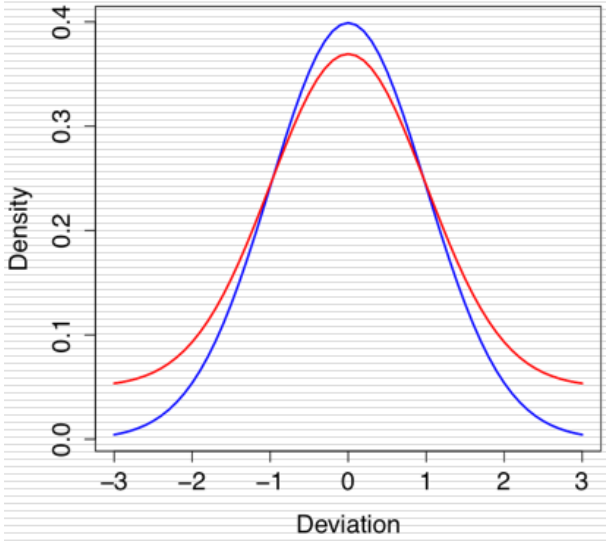
- If $\beta_2 < 3$ Curve is Leptokurtic
- If $\beta_2 = 3$ Curve is Mesokurtic
- If $\beta_2 > 3$ Curve is PlatyKurtic

Kurtosis Formula

Let $x_1, x_2, \ldots x_n$ be n observations. Then,

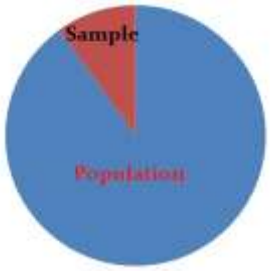
$$\text{Kurtosis} = \frac{n\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^2} - 3$$

Kurtosis



Kurtosis relates to the relative flatness or peakedness of a distribution. A standard normal distribution (blue line: $\mu = 0$; $\sigma = 1$) has kurtosis = 0. A distribution like that illustrated with the red curve has kurtosis > 0 with a lower peak relative to its tails.

Descriptive statistics describes data (for example, a chart or graph) and



inferential statistics allows you to make predictions ("inferences") from that data.

With inferential statistics, you take data from samples and make generalizations about a population.

Inferential statistics use a random sample of data taken from a population to describe and make inferences about the population.

Inferential statistics are valuable when examination of each member of an entire population is not convenient or possible.

For example,

1. you might stand in a mall and ask a sample of 100 people if they like shopping at Sears. You could make a bar chart of yes or no answers (that would be descriptive statistics) or you could use your research (and inferential statistics) to reason that around 75-80% of the population (**all** shoppers in **all malls**) like shopping at Sears.

2. To measure the diameter of each nail that is manufactured in a mill is impractical. You can measure the diameters of a representative random sample of nails. You can use the information from the sample to make generalizations about the diameters of all of the nails.

There are two main areas of inferential statistics:

1. **Estimating parameters:** This means taking a statistic from your sample data (for example the sample mean) and using it to say something about a population parameter (i.e. the population mean).
2. **Hypothesis tests:** This is where you can use sample data to answer research questions. For example, you might be interested in knowing if a new cancer drug is effective. Or if breakfast helps children perform better in schools.

Probability Concepts:

Definition	Example
An **experiment** is a situation involving chance or probability that leads to results called outcomes.	In the problem above, the experiment is spinning the spinner.
An **outcome** is the result of a single trial of an experiment.	The possible outcomes are landing on yellow, blue, green or red.
An **event** is one or more outcomes of an experiment.	One event of this experiment is landing on blue.
Probability is the measure of how likely an event is.	The probability of landing on blue is one fourth.

In order to measure probabilities, mathematicians have devised the following formula for finding the probability of an event.

Probability Of an Event:

$$P(A) = \frac{\text{The Number Of Ways Event A Can Occur}}{\text{The total number Of Possible Outcomes}}$$

The probability of event A is the number of ways event A can occur divided by the total number of possible outcomes. Let's take a look at a slight modification of the problem from the top of the page.

Experiment 1: A spinner has 4 equal sectors colored yellow, blue, green and red. After spinning the spinner, what is the probability of landing on each color?

The possible outcomes of this experiment are yellow, blue, green, and red.

Probabilities:

$$P(\text{yellow}) = \frac{\#\text{ of ways to land on yellow}}{\text{total }\#\text{ of colors}} = \frac{1}{4}$$

$$P(\text{blue}) = \frac{\#\text{ of ways to land on blue}}{\text{total }\#\text{ of colors}} = \frac{1}{4}$$

$$P(\text{green}) = \frac{\#\text{ of ways to land on green}}{\text{total }\#\text{ of colors}} = \frac{1}{4}$$

$$P(\text{red}) = \frac{\#\text{ of ways to land on red}}{\text{total }\#\text{ of colors}} = \frac{1}{4}$$

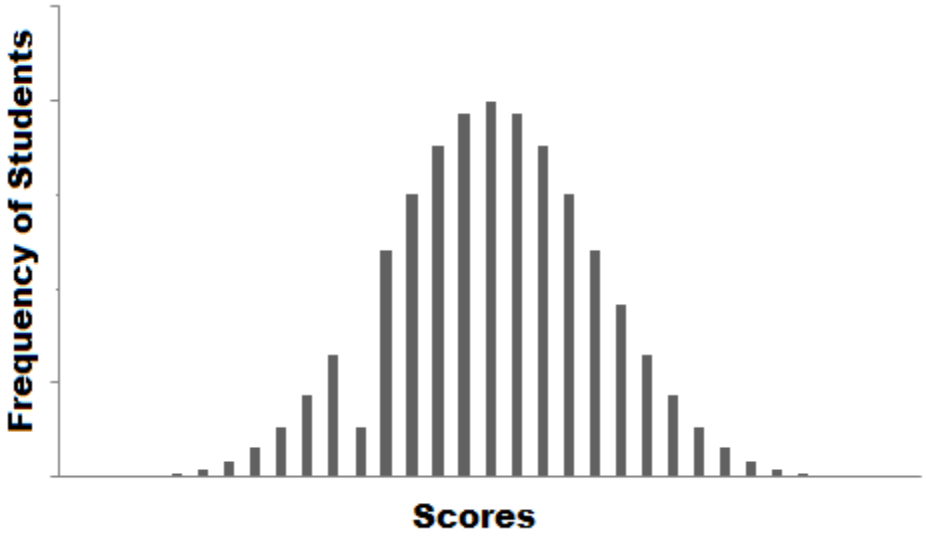
Probability Distributions:

Suppose you are a teacher in a university. After checking assignments for a week, you graded all the students. You gave these graded papers to a data entry guy in the university and tell him to create a spreadsheet containing the grades of all the students. But the guy only stores the grades and not the corresponding students.

S. No.	Scores
1	25
2	27
3	38
4	42
5	
6	16
7	35
8	46
9	48
10	31

He made another blunder, he missed a couple of entries in a hurry and we have no idea whose grades are missing. Let's find a way to solve this.

One way is that you visualize the grades and see if you can find a trend in the data.



The graph that you have plot is called the frequency distribution of the data. You see that there is a smooth curve like structure that defines our data, but do you notice an anomaly? We have an abnormally low frequency at a particular score range. So the best guess would be to have missing values that remove the dent in the distribution.

This is how you would try to solve a real-life problem using data analysis. For any Data Scientist, a student or a practitioner, distribution is a must know concept. It provides the basis for analytics and inferential statistics.

While the concept of probability gives us the mathematical calculations, distributions help us actually visualize what's happening underneath.

In this article, I have covered some important probability distributions which are explained in a lucid as well as comprehensive manner.

Note: This article assumes you have a basic knowledge of probability. If not, you can refer this article on basics of probability

Table of Contents

1. Common Data Types
2. Types of Distributions
    1. Bernoulli Distribution
    2. Uniform Distribution
    3. Binomial Distribution
    4. Normal Distribution
    5. Poisson Distribution
    6. Exponential Distribution

Common Data Types

Before we jump on to the explanation of distributions, let's see what kind of data can we encounter. The data can be discrete or continuous.

Discrete Data, as the name suggests, can take only specified values. For example, when you roll a die, the possible outcomes are 1, 2, 3, 4, 5 or 6 and not 1.5 or 2.45.

Continuous Data can take any value within a given range. The range may be finite or infinite. For example, A girl's weight or height, the length of the road. The weight of a girl can be any value from 54 kgs, or 54.5 kgs, or 54.5436kgs.

Now let us start with the types of distributions.

Types of Distributions

Bernoulli Distribution

Let's start with the easiest distribution that is Bernoulli Distribution. It is actually easier to understand than it sounds!

All you cricket junkies out there! At the beginning of any cricket match, how do you decide who is going to bat or ball? A toss! It all depends on whether you win or lose the toss, right? Let's say if the toss results in a head, you win. Else, you lose. There's no midway.

A **Bernoulli distribution** has only two possible outcomes, namely 1 (success) and 0 (failure), and a single trial. So the random variable X which has a Bernoulli distribution can take value 1 with the probability of success, say p, and the value 0 with the probability of failure, say q or 1-p.

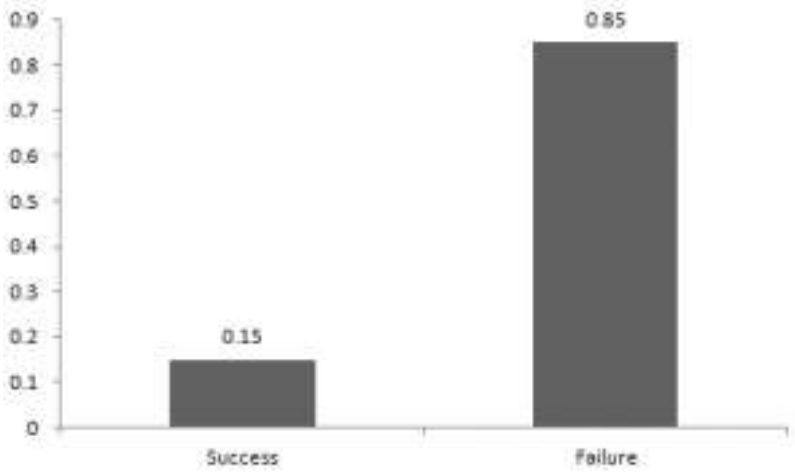
Here, the occurrence of a head denotes success, and the occurrence of a tail denotes failure. Probability of getting a head = 0.5 = Probability of getting a tail since there are only two possible outcomes.

The probability mass function is given by: $p^x(1-p)^{1-x}$ where x € (0, 1). It can also be written as

$$P(x) = \begin{cases} 1-p, & x = 0 \\ p, & x = 1 \end{cases}$$

The probabilities of success and failure need not be equally likely, like the result of a fight between me and Undertaker. He is pretty much certain to win. So in this case probability of my success is 0.15 while my failure is 0.85

Here, the probability of success(p) is not same as the probability of failure. So, the chart below shows the Bernoulli Distribution of our fight.



Here, the probability of success = 0.15 and probability of failure = 0.85. The expected value is exactly what it sounds. If I punch you, I may expect you to punch me back. Basically expected value of any distribution is the mean of the distribution. The expected value of a random variable X from a Bernoulli distribution is found as follows:

$E(X) = 1*p + 0*(1-p) = p$

The variance of a random variable from a bernoulli distribution is:

$V(X) = E(X^2) - [E(X)]^2 = p - p^2 = p(1-p)$

There are many examples of Bernoulli distribution such as whether it's going to rain tomorrow or not where rain denotes success and no rain denotes failure and Winning (success) or losing (failure) the game.

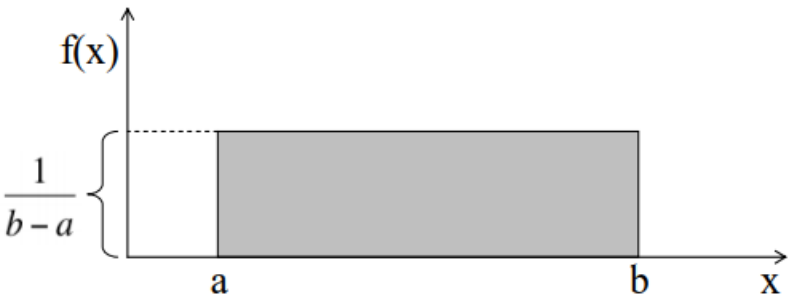
Uniform Distribution

When you roll a fair die, the outcomes are 1 to 6. The probabilities of getting these outcomes are equally likely and that is the basis of a uniform distribution. Unlike Bernoulli Distribution, all the n number of possible outcomes of a uniform distribution are equally likely.

A variable X is said to be uniformly distributed if the density function is:

$$f(x) = \frac{1}{b-a} \qquad for\ -\infty < a \le x \le b < \infty$$

The graph of a uniform distribution curve looks like



You can see that the shape of the Uniform distribution curve is rectangular, the reason why Uniform distribution is called rectangular distribution.

For a Uniform Distribution, a and b are the parameters.

The number of bouquets sold daily at a flower shop is uniformly distributed with a maximum of 40 and a minimum of 10.

Let's try calculating the probability that the daily sales will fall between 15 and 30.

The probability that daily sales will fall between 15 and 30 is (30-15)*(1/(40-10)) = 0.5

Similarly, the probability that daily sales are greater than 20 is = 0.667

The mean and variance of X following a uniform distribution is:

Mean -> E(X) = (a+b)/2

Variance -> V(X) = $(b-a)^2/12$

The standard uniform density has parameters a = 0 and b = 1, so the PDF for standard uniform density is given by:

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & otherwise \end{cases}$$

Binomial Distribution

Let's get back to cricket. Suppose that you won the toss today and this indicates a successful event. You toss again but you lost this time. If you win a toss today, this does not necessitate that you will win the toss tomorrow. Let's assign a random variable, say X, to the number of times you won the toss. What can be the possible value of X? It can be any number depending on the number of times you tossed a coin.

There are only two possible outcomes. Head denoting success and tail denoting failure. Therefore, probability of getting a head = 0.5 and the probability of failure can be easily computed as: q = 1- p = 0.5.

A distribution where only two outcomes are possible, such as success or failure, gain or loss, win or lose and where the probability of success and failure is same for all the trials is called a Binomial Distribution.

The outcomes need not be equally likely. Remember the example of a fight between me and Undertaker? So, if the probability of success in an experiment is 0.2 then the probability of failure can be easily computed as q = 1 – 0.2 = 0.8.

Each trial is independent since the outcome of the previous toss doesn't determine or affect the outcome of the current toss. An experiment with only two possible outcomes repeated n number of times is called binomial. The parameters of a binomial distribution are n and p where n is the total number of trials and p is the probability of success in each trial.

On the basis of the above explanation, the properties of a Binomial Distribution are

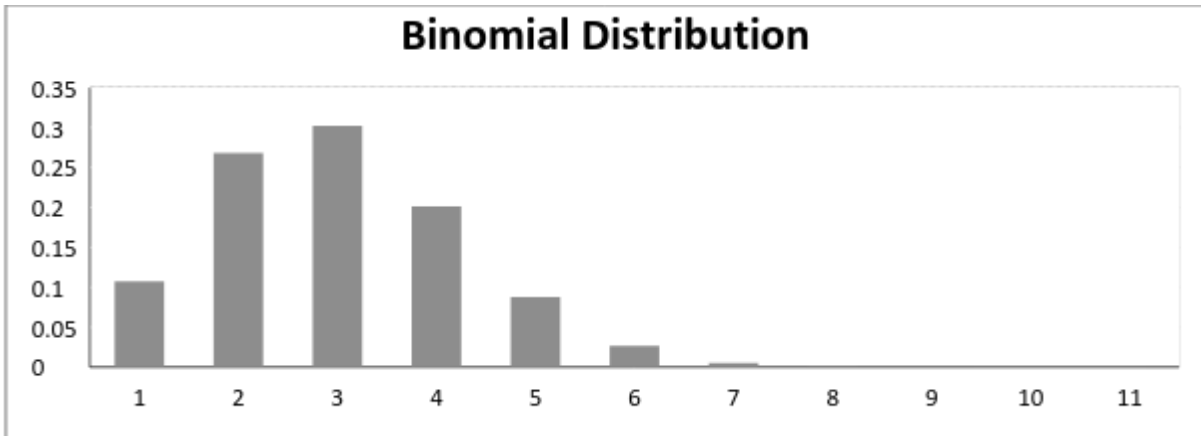
1. Each trial is independent.

2. There are only two possible outcomes in a trial- either a success or a failure.
3. A total number of n identical trials are conducted.
4. The probability of success and failure is same for all trials. (Trials are identical.)

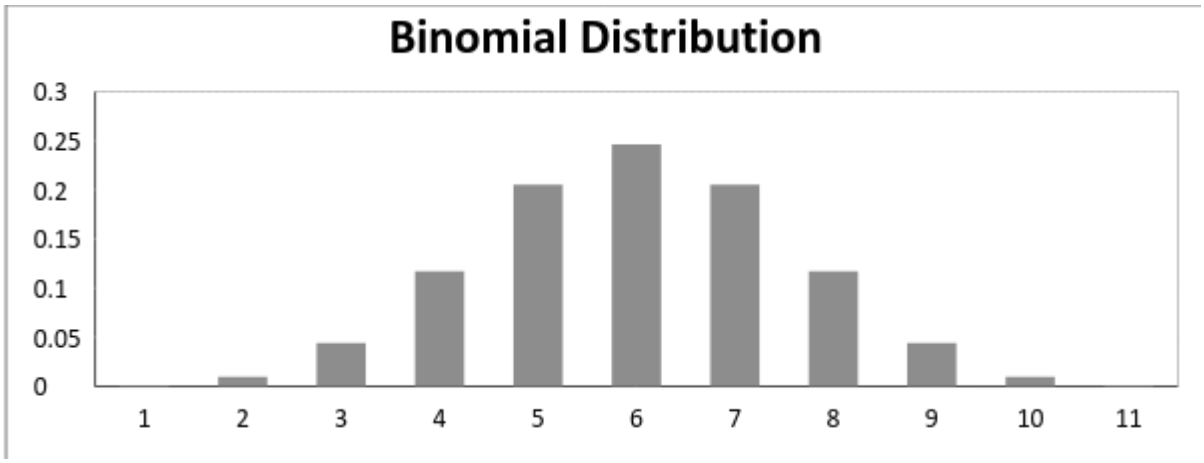
The mathematical representation of binomial distribution is given by:

$$P(x) = \frac{n!}{(n-x)!\,x!} p^x q^{n-x}$$

A binomial distribution graph where the probability of success does not equal the probability of failure looks like



Now, when probability of success = probability of failure, in such a situation the graph of binomial distribution looks like



The mean and variance of a binomial distribution are given by:

Mean -> μ = n*p

Variance -> Var(X) = n*p*q

Normal Distribution

Normal distribution represents the behavior of most of the situations in the universe (That is why it's called a "normal" distribution. I guess!). The large sum of (small) random variables often turns out to be normally distributed, contributing to its widespread application. Any distribution is known as Normal distribution if it has the following characteristics:

1. The mean, median and mode of the distribution coincide.
2. The curve of the distribution is bell-shaped and symmetrical about the line x=μ.
3. The total area under the curve is 1.
4. Exactly half of the values are to the left of the center and the other half to the right.

A normal distribution is highly different from Binomial Distribution. However, if the number of trials approaches infinity then the shapes will be quite similar.

The PDF of a random variable X following a normal distribution is given by:

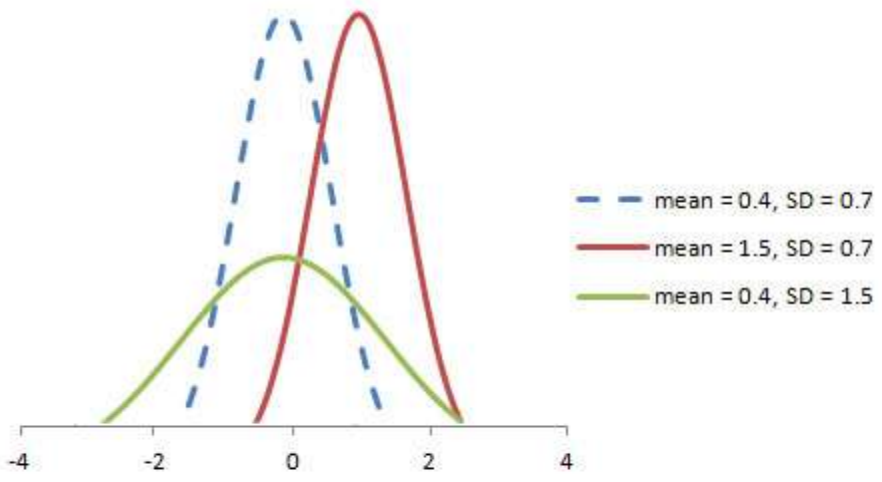
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\{-\frac{1}{2}(\frac{x-u}{\sigma})^2\}} \qquad for -\infty < x < \infty.$$

The mean and variance of a random variable X which is said to be normally distributed is given by:

Mean -> E(X) = μ

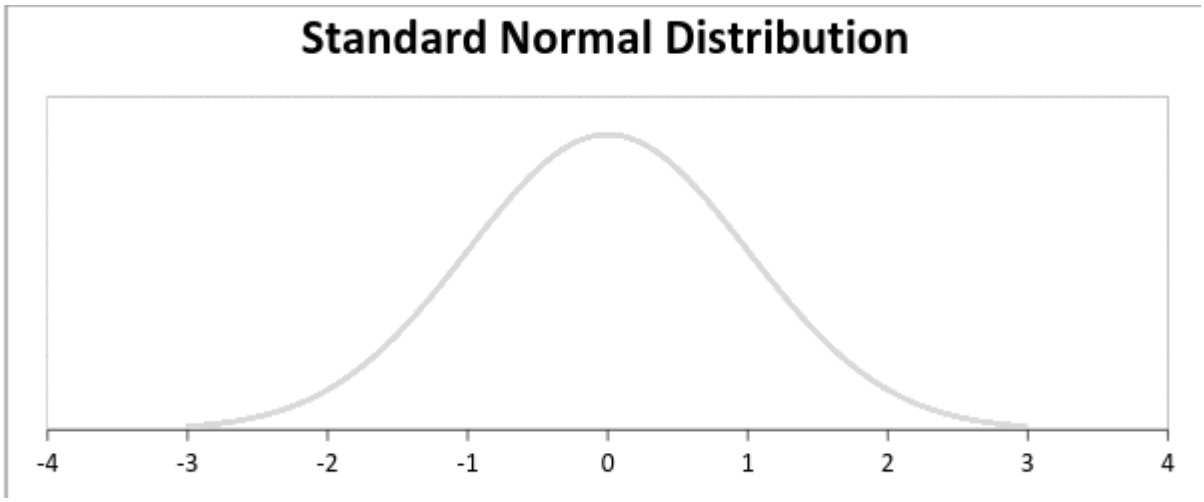
Variance -> Var(X) = σ^2

Here, μ (mean) and σ (standard deviation) are the parameters.
The graph of a random variable X ~ N (μ, σ) is shown below.



A standard normal distribution is defined as the distribution with mean 0 and standard deviation 1. For such a case, the PDF becomes:

$$f(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2/2} \qquad for\ -\infty < x < \infty$$



Poisson Distribution

Suppose you work at a call center, approximately how many calls do you get in a day? It can be any number. Now, the entire number of calls at a call center in a day is modeled by Poisson distribution. Some more examples are

1. The number of emergency calls recorded at a hospital in a day.
2. The number of thefts reported in an area on a day.
3. The number of customers arriving at a salon in an hour.
4. The number of suicides reported in a particular city.
5. The number of printing errors at each page of the book.

You can now think of many examples following the same course. Poisson Distribution is applicable in situations where events occur at random points of time and space wherein our interest lies only in the number of occurrences of the event.

A distribution is called **Poisson distribution** when the following assumptions are valid:

1. Any successful event should not influence the outcome of another successful event.
2. The probability of success over a short interval must equal the probability of success over a longer interval.
3. The probability of success in an interval approaches zero as the interval becomes smaller.

Now, if any distribution validates the above assumptions then it is a Poisson distribution. Some notations used in Poisson distribution are:

- λ is the rate at which an event occurs,
- t is the length of a time interval,
- And X is the number of events in that time interval.

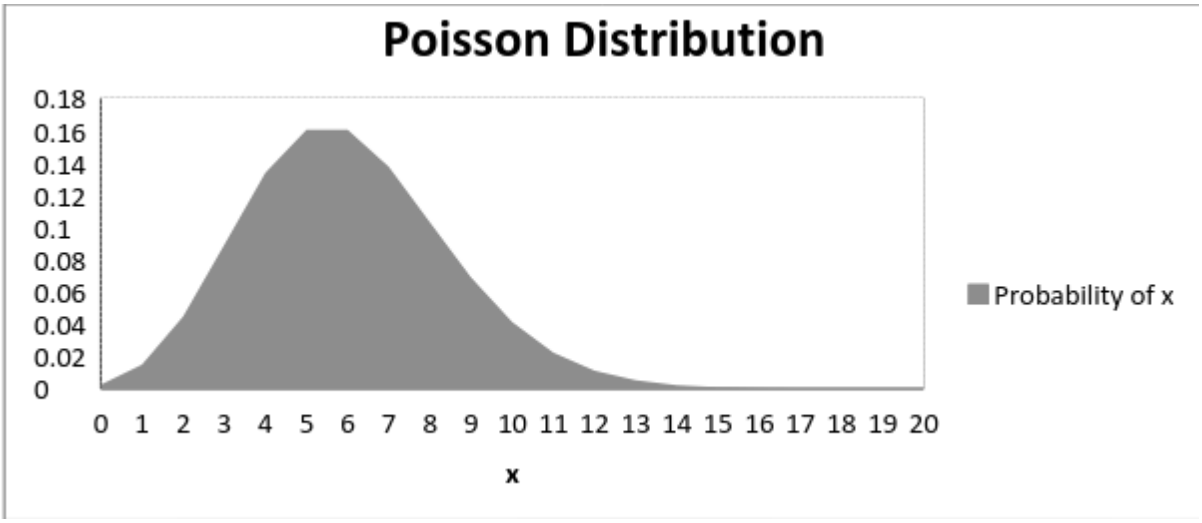
Here, X is called a Poisson Random Variable and the probability distribution of X is called Poisson distribution.

Let μ denote the mean number of events in an interval of length t. Then, μ = λ*t.

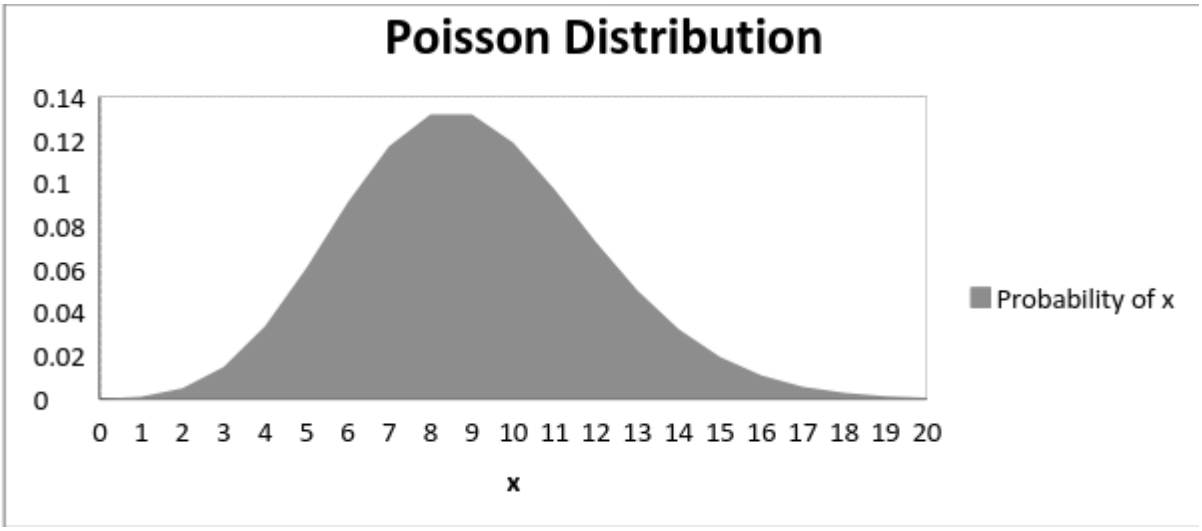
The PMF of X following a Poisson distribution is given by:

$$P(X = x) = e^{-\mu}\frac{\mu^x}{x!} \qquad for\ x = 0, 1, 2, \ldots..$$

The mean μ is the parameter of this distribution. μ is also defined as the λ times length of that interval. The graph of a Poisson distribution is shown below:



The graph shown below illustrates the shift in the curve due to increase in mean.



It is perceptible that as the mean increases, the curve shifts to the right.

The mean and variance of X following a Poisson distribution:

Mean -> E(X) = μ
Variance -> Var(X) = μ

Exponential Distribution

Let's consider the call center example one more time. What about the interval of time between the calls ? Here, exponential distribution comes to our rescue. Exponential distribution models the interval of time between the calls.

Other examples are:

1. Length of time beteeen metro arrivals

2. Length of time between arrivals at a gas station

3. The life of an Air Conditioner

Exponential distribution is widely used for survival analysis. From the expected life of a machine to the expected life of a human, exponential distribution successfully delivers the result.

A random variable X is said to have an **exponential distribution** with PDF:

$f(x) = \{ \lambda e^{-\lambda x}, \ x \geq 0$

and parameter **$\lambda > 0$** which is also called the rate.

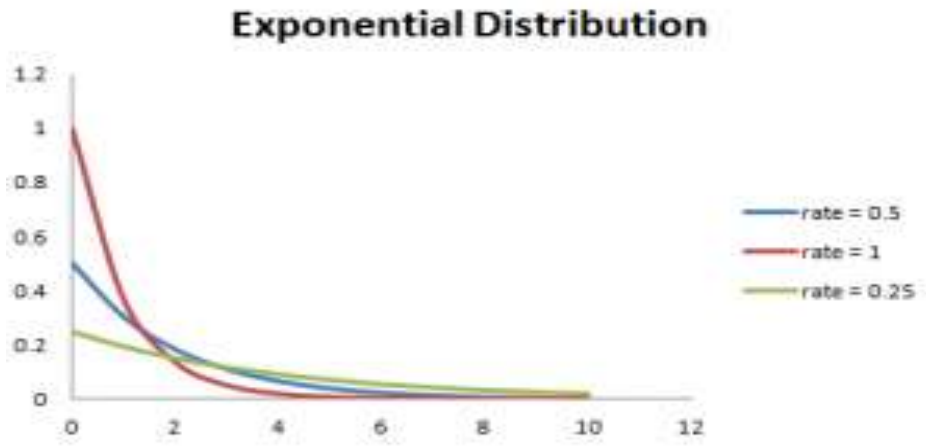
For survival analysis, λ is called the failure rate of a device at any time t, given that it has survived up to t.

Mean and Variance of a random variable X following an exponential distribution:

Mean -> $E(X) = 1/\lambda$

Variance -> $Var(X) = (1/\lambda)^2$

Also, the greater the rate, the faster the curve drops and the lower the rate, flatter the curve. This is explained better with the graph shown below.



To ease the computation, there are some formulas given below. $P\{X \leq x\} = 1 - e^{-\lambda x}$, corresponds to the area under the density curve to the left of x.

$P\{X > x\} = e^{-\lambda x}$, corresponds to the area under the density curve to the right of x.

$P\{x1 < X \leq x2\} = e^{-\lambda x1} - e^{-\lambda x2}$, corresponds to the area under the density curve between x1 and x2.

Hypothesis Testing

$$z = \frac{\hat{p} - p}{\sqrt{p\,q/n}}$$ The main purpose of statistics is to test a hypothesis. For example, you might run an experiment and find that a certain drug is effective at treating headaches. But if you can't repeat that experiment, no one will take your results seriously. A good example of this was the cold fusion discovery, which petered into obscurity because no one was able to duplicate the results.

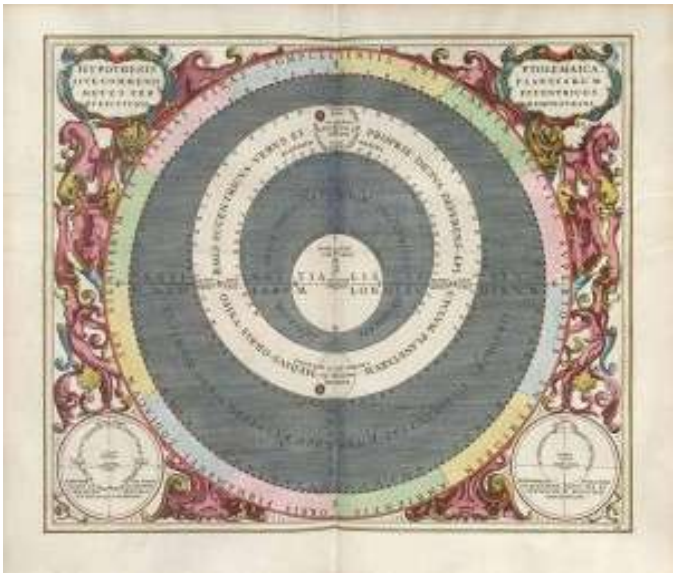
Contents :

1. What is a Hypothesis?
2. What is Hypothesis Testing?
3. Hypothesis Testing Examples (One Sample Z Test).
4. Hypothesis Test on a Mean (TI 83).
5. Bayesian Hypothesis Testing.
6. More Hypothesis Testing Articles

See also:

- Critical Values
- What is the Null Hypothesis?

What is a Hypothesis?



Andreas Cellarius hypothesis, showing the planetary motions.

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.

What is a Hypothesis Statement?

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).
- If I (give exams at noon instead of 7) then (student test scores will improve).
- If I (look in this certain location) then (I am more likely to find new species).

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?

$$z = \frac{\hat{p} - p}{\sqrt{p\,q/n}}$$

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 have to 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:

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

How do I State the Null Hypothesis?

You won't be required to actually perform a real experiment or survey in elementary statistics (or even disprove a fact like "Pluto is a planet"!), so you'll be given word problems from real-life situations. You'll need to figure out what your hypothesis is from the problem. This can be a little trickier than just figuring out what the accepted fact is. With word problems, you are looking to find a fact that is nullifiable (i.e. something you can reject).

Hypothesis Testing Examples #1: Basic Example

A researcher thinks that if knee surgery patients go to physical therapy twice a week (instead of 3 times), their recovery period will be longer. Average recovery times for knee surgery patients is 8.2 weeks.

The hypothesis statement in this question is that the researcher believes the average recovery time is more than 8.2 weeks. It can be written in mathematical terms as: H_1: $\mu > 8.2$

Next, you'll need to **state the null hypothesis** (See: How to state the null hypothesis). That's what will happen if the researcher is wrong. In the above example, if the researcher is wrong then the recovery time is less than or equal to 8.2 weeks. In math, that's: H_0 $\mu \leq 8.2$

Rejecting the null hypothesis

Ten or so years ago, we believed that there were 9 planets in the solar system. Pluto was demoted as a planet in 2006. The null hypothesis of "Pluto is a planet" was replaced by "Pluto is not a planet." Of course, rejecting the null hypothesis isn't always that easy — the hard part is usually figuring out what your null hypothesis is in the first place.

Hypothesis Testing Examples (One Sample Z Test)

The one sample z test isn't used very often (because we rarely know the actual population standard deviation). However, it's a good idea to understand how it works as it's one of the simplest tests you can perform in hypothesis testing. In English class you got to learn the basics (like grammar and spelling) before you could write a story; think of one sample z tests as the foundation for understanding more complex hypothesis testing. This page contains two hypothesis testing examples for one sample z-tests.

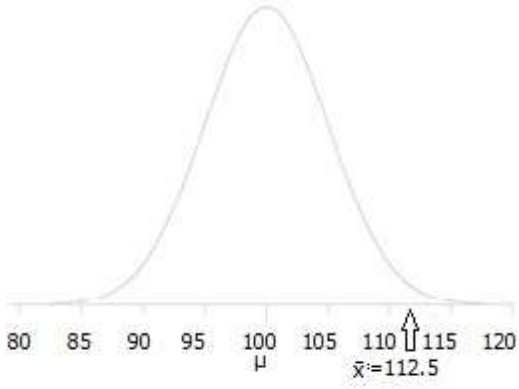
One Sample Hypothesis Testing Examples: #2

A principal at a certain school claims that the students in his school are above average intelligence. A random sample of thirty students IQ scores have a mean score of 112. Is there sufficient evidence to support the principal's claim? The mean population IQ is 100 with a standard deviation of 15.

Step 1: State the Null hypothesis. The accepted fact is that the population mean is 100, so: H_0: $\mu=100$.

Step 2: State the Alternate Hypothesis. The claim is that the students have above average IQ scores, so: H_1: $\mu > 100$.
The fact that we are looking for scores "greater than" a certain point means that this is a one-tailed test.

Step 3: Draw a picture to help you visualize the problem.



Step 4: State the alpha level. If you aren't given an alpha level, use 5% (0.05).

Step 5: Find the rejection region area (given by your alpha level above) from the z-table. An area of .05 is equal to a z-score of 1.645.

Step 6: Find the test statistic using this formula: $Z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}}$
For this set of data: z= (112.5-100) / (15/√30)=4.56.

Step 6: If Step 6 is greater than Step 5, reject the null hypothesis. If it's less than Step 5, you cannot reject the null hypothesis. In this case, it is greater (4.56 > 1.645), so you can reject the null.

One Sample Hypothesis Testing Examples: #3

Blood glucose levels for obese patients have a mean of 100 with a standard deviation of 15. A researcher thinks that a diet high in raw cornstarch will have a positive or negative effect on blood glucose levels. A sample of 30 patients who have tried the raw cornstarch diet have a mean glucose level of 140. Test the hypothesis that the raw cornstarch had an effect.

Step 1: State the null hypothesis: H_0:μ=100

Step 2: State the alternate hypothesis: H_1:≠100

Step 3: State your alpha level. We'll use 0.05 for this example. As this is a two-tailed test, split the alpha into two. 0.05/2=0.025.

Step 4: Find the z-score associated with your alpha level. You're looking for the area in one tail only. A z-score for 0.75(1-0.025=0.975) is 1.96. As this is a two-tailed test, you would also be considering the left tail (z=1.96)

Step 5: Find the test statistic using this formula: $Z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}}$
z=(140-100)/(15/√30)=14.60.
Step 6: If Step 5 is less than -1.96 or greater than 1.96 (Step 3), reject the null hypothesis. In this case, it is greater, so you can reject the null.

*This process is made much easier if you use a TI-83 or Excel to calculate the z-score (the "critical value").

See:

- Critical z value TI 83
- Z Score in Excel

Hypothesis Testing Examples: Mean (Using TI 83)

You can use the **TI 83** calculator for hypothesis testing, but the calculator won't figure out the null and alternate hypotheses; that's up to you to read the question and input it into the calculator.

Sample problem: A sample of 200 people has a mean age of 21 with a population standard deviation (σ) of 5. Test the hypothesis that the population mean is 18.9 at α = 0.05.

Step 1: State the null hypothesis. In this case, the null hypothesis is that the population mean is 18.9, so we write: **H_0: μ = 18.9**

Step 2: State the alternative hypothesis. We want to know if our sample, which has a mean of 21 instead of 18.9, really is different from the population, therefore our alternate hypothesis:
H_1: μ ≠ 18.9

Step 3: Press Stat then press the **right arrow** twice to select TESTS.

Step 4: Press 1 to select **1:Z-Test…**. Press ENTER.

Step 5: Use the **right arrow** to select **Stats**.

Step 6: Enter the data from the problem:
μ_0: 18.9
σ: 5
x: 21
n: 200
μ: $\neq \mu_0$

Step 7: Arrow down to **Calculate** and press ENTER. The calculator shows the p-value: $p = 2.87 \times 10^{-9}$

This is smaller than our alpha value of .05. That means we should **reject the null**

Hypothesis.

Bayesian Hypothesis Testing: What is it?

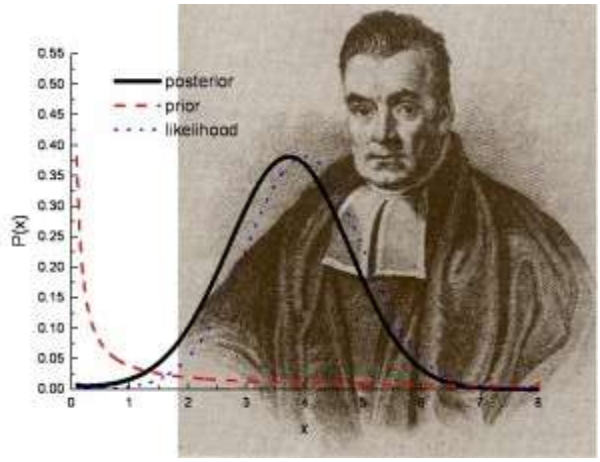


Image: Los Alamos National Lab.

Bayesian hypothesis testing helps to answer the question: Can the results from a test or survey be repeated?

Why do we care if a test can be repeated? Let's say twenty people in the same village came down with leukemia. A group of researchers find that cell-phone towers are to blame. However, a second study found that cell-phone towers had nothing to do with the cancer cluster in the village. In fact, they found that the cancers were completely random. If that sounds impossible, it actually can happen! Clusters of cancer can happen simply by chance. There could be many reasons why the first study was faulty. One of the main reasons could be that they just didn't take into account that sometimes things happen randomly and we just don't know why.

P Values.

It's good science to let people know if your study results are solid, or if they could have happened by chance. The usual way of doing this is to test your results with a p-value. A p value is a number that you get by running a hypothesis test on your data. A P value of 0.05 (5%) or less is usually enough to claim that your results are repeatable. However, there's another way to test the validity of your results: Bayesian Hypothesis testing. This type of testing gives you another way to test the strength of your results.

Bayesian Hypothesis Testing.

Traditional testing (the type you probably came across in elementary stats or AP stats) is called Non-Bayesian. It is how often an outcome happens over repeated runs of the experiment. It's an **objective** view of whether an experiment is repeatable. Bayesian hypothesis testing is a **subjective** view of the same thing. It takes into account

how much faith you have in your results. In other words, would you wager money on the outcome of your experiment?

Differences Between Traditional and Bayesian Hypothesis Testing.

Traditional testing (Non Bayesian) requires you to repeat sampling over and over, while Bayesian testing does not. The main different between the two is in the first step of testing: stating a probability model. In Bayesian testing you add prior knowledge to this step. It also requires use of a posterior probability, which is the conditional probability given to a random event after all the evidence is considered.

Arguments for Bayesian Testing.

Many researchers think that it is a better alternative to traditional testing, because it:

1. Includes prior knowledge about the data.
2. Takes into account personal beliefs about the results.

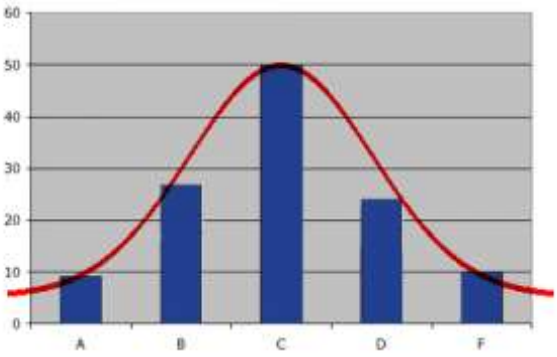
Arguments against.

1. Including prior data or knowledge isn't justifiable.
2. It is difficult to calculate compared to non-Bayesian testing.

Z-TEST:

A Z-test is a type of hypothesis test. Hypothesis testing is just a way for you to figure out if results from a test are valid or repeatable.

- A hypothesis test will tell you if it's probably true, or probably not true. A Z test, is used when your data is approximately.
- A z-test is a statistical test used to determine whether two population means are different when the variances are known and the sample size is large. The test statistic is assumed to have a normal distribution, and nuisance parameters such as standard deviation should be known in order for an accurate z-test to be performed.



- A z-test is a statistical test used to determine whether two population means are different when the variances are known and the sample size is large.

- The test statistic is assumed to have a normal distribution, and nuisance parameters such as standard deviation should be known in order for an accurate z-test to be performed.

BREAKING DOWN 'Z-Test'

- A one-sample location test, two-sample location test, paired difference test and maximum likelihood estimate are examples of tests that can be conducted as z-tests.
- Z-tests are closely related to t-tests, but t-tests are best performed when an experiment has a small sample size.
- Also, t-tests assume the standard deviation is unknown, while z-tests assume it is known. If the standard deviation of the population is unknown, the assumption of the sample variance equaling the population variance is made.

Hypothesis Test

- The z-test is also a hypothesis test in which the z-statistic follows a normal distribution.
- The z-test is best used for greater than 30 samples because, under the central limit theorem, as the number of samples gets larger, the samples are considered to be approximately normally distributed.
- When conducting a z-test, the null and alternative hypotheses, alpha and z-score should be stated. Next, the test statistic should be calculated, and the results and conclusion stated.

One-Sample Z-Test Example

For example, assume an investor wishes to test whether the average daily return of a stock is greater than 1%. A simple random sample of 50 returns is calculated and has an average of 2%.

Assume the standard deviation of the returns is 2.50%. Therefore, the null hypothesis is when the average, or mean, is equal to 3%.

Conversely, the alternative hypothesis is whether the mean return is greater than 3%. Assume an alpha of 0.05% is selected with a two-tailed test.

Consequently, there is 0.025% of the samples in each tail, and the alpha has a critical value of 1.96 or -1.96. If the value of z is greater than 1.96 or less than -1.96, the null hypothesis is rejected.

The value for z is calculated by subtracting the value of the average daily return selected for the test, or 1% in this case, from the observed average of the samples.

Next, divide the resulting value by the standard deviation divided by the square root of the number of observed values.

Therefore, the test statistic is calculated to be 2.83, or (0.02 - 0.01) / (0.025 / (50)^(1/2)). The investor rejects the null hypothesis since z is greater than 1.96, and concludes that the average daily return is greater than 1%.

There are different types of Z-test each for different purpose. Some of the popular types are outlined below:

1.z test for single proportion is used to test a hypothesis on a specific value of the population proportion:

Statistically speaking, we test the null hypothesis H_0: p = p_0 against the alternative hypothesis H_1: p >< p_0 where p is the population proportion and p_0 is a specific value of the population proportion we would like to test for acceptance.

The example on tea drinkers explained above requires this test. In that example, p_0 = 0.5. Notice that in this particular example, proportion refers to the proportion of tea drinkers.

2.z test for difference of proportions is used to test the hypothesis that two populations have the same proportion.

Example;- suppose one is interested to test if there is any significant difference in the habit of tea drinking between male and female citizens of a town. In such a situation, Z-test for difference of proportions can be applied.

One would have to obtain two independent samples from the town- one from males and the other from females and determine the proportion of tea drinkers in each sample in order to perform this test.

3.z -test for single mean is used to test a hypothesis on a specific value of the population mean.

Statistically speaking, we test the null hypothesis H_0: μ = μ_0 against the alternative hypothesis H_1: μ >< μ_0 where μ is the population mean and μ_0 is a specific value of the population that we would like to test for acceptance.

Unlike the t-test for single mean, this test is used if n ≥ 30 and population standard deviation is known.

4.z test for single variance is used to test a hypothesis on a specific value of the population variance.

Statistically speaking, we test the null hypothesis H_0: $\sigma = \sigma_0$ against H_1: $\sigma >< \sigma_0$ where σ is the population mean and σ_0 is a specific value of the population variance that we would like to test for acceptance.

In other words, this test enables us to test if the given sample has been drawn from a population with specific variance σ_0. Unlike the chi square test for single variance, this test is used if $n \geq 30$.

5.Z-test for testing equality of variance is used to test the hypothesis of equality of two population variances when the sample size of each sample is 30 or larger.

T-test

The t-test was described by 1908 by William Sealy Gosset for monitoring the brewing at Guinness in Dublin. Guinness considered the use of statistics a trade secret, so he published his test under the pen-name 'Student' -- hence the test is now often called the 'Student's t-test'.

The t-test is a basic test that is limited to two groups. For multiple groups, you would have to compare each pair of groups, for example with three groups there would be three tests (AB, AC, BC).

The t-test (or student's t-test) gives an indication of the separateness of two sets of measurements, and is thus used to check whether two sets of measures are essentially different (and usually that an experimental effect has been demonstrated). The typical way of doing this is with the null hypothesis that means of the two sets of measures are equal.

The t-test assumes:

- A normal distribution (parametric data)
- Underlying variances are equal (if not, use Welch's test)

It is used when there is random assignment and only two sets of measurement to compare.

There are two main types of t-test:

- Independent-measures t-test: when samples are not matched.
- Matched-pair t-test: When samples appear in pairs (eg. before-and-after).

Independent one-sample t-test

- In testing the null hypothesis that the population mean is equal to a specified value μ0, one uses the statistic

$$t = \frac{\overline{x} - \mu_0}{\frac{s}{\sqrt{n}}},$$

where s is the sample standard deviation of the sample and n is the sample size. The degrees of freedom used in this test is n − 1.

Independent two-sample t-test:

Equal sample sizes, equal variance

This test is only used when both:

- the two sample sizes (that is, the number, n, of participants of each group) are equal;
- it can be assumed that the two distributions have the same variance.

Violations of these assumptions are discussed below.

The t statistic to test whether the means are different can be calculated as follows:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_{X_1X_2} \cdot \sqrt{\frac{2}{n}}} \quad S_{X_1X_2} = \sqrt{\frac{S_{X_1}^2 + S_{X_2}^2}{2}}$$

where Here Sx1x2 is the grand standard deviation (or pooled standard deviation), 1 = group one, 2 = group two. The denominator of t is the standard error of the difference between two means.

For significance testing, the degrees of freedom for this test is 2n − 2 where n is the number of participants in each group.

Unequal sample sizes, equal variance :

This test is used only when it can be assumed that the two distributions have the same variance. (When this assumption is violated, see below.) The t statistic to test whether the means are different can be calculated as follows:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_{X_1X_2} \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \quad \text{where} \quad S_{X_1X_2} = \sqrt{\frac{(n_1-1)S_{X_1}^2 + (n_2-1)S_{X_2}^2}{n_1+n_2-2}}.$$

Note that the formulae above are generalizations for the case where both samples have equal sizes

Sx1x2 is an estimator of the common standard deviation of the two samples: it is defined in this way so that its square is an unbiased estimator of the common variance whether or not the population means are the same. In these formulae, n = number of participants, 1 = group one, 2 = group two. n − 1 is the number of degrees of freedom for either group, and the total sample size minus two (that is, n1 + n2 − 2) is the total number of degrees of freedom, which is used in significance testing.

Unequal sample sizes, unequal variance

This test is used only when the two population variances are assumed to be different (the two sample sizes may or may not be equal) and hence must be estimated separately. The t statistic to test whether the population means are different can be calculated as follows:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{s_{\overline{X}_1 - \overline{X}_2}} \qquad \text{where} \qquad s_{\overline{X}_1 - \overline{X}_2} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}.$$

Where s2 is the unbiased estimator of the variance of the two samples, n = number of participants, 1 = group one, 2 = group two. Note that in this case, $s^2_{\overline{X}_1 - \overline{X}_2}$ is not a pooled variance. For use in significance testing, the distribution of the test statistic is approximated as being an ordinary Student's t distribution with the degrees of freedom calculated using

$$\text{d.f.} = \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{(s_1^2/n_1)^2/(n_1 - 1) + (s_2^2/n_2)^2/(n_2 - 1)}.$$

This is called the Welch–Satterthwaite equation. Note that the true distribution of the test statistic actually depends (slightly) on the two unknown variances.

This test is used when the samples are dependent; that is, when there is only one sample that has been tested twice (repeated measures) or when there are two samples that have been matched or "paired". This is an example of a paired difference test.

$$t = \frac{\overline{X}_D - \mu_0}{s_D / \sqrt{n}}.$$

For this equation, the differences between all pairs must be calculated. The pairs are either one person's pre-test and post-test scores or between pairs of persons matched into meaningful groups (for instance drawn from the same family or age group). The average (XD) and standard deviation (sD) of those differences are used in the equation. The constant μ0 is non-zero if you want to test whether the average of the difference is significantly different from μ0. The degree of freedom used is n − 1.

F – Test :

An F-test (Snedecor and Cochran, 1983) is used to test if the standard deviations of two populations are equal. This test can be a two-tailed test or a one-tailed test. The two-tailed version tests against the alternative that the standard deviations are not equal. The one-

tailed version only tests in one direction, that is the standard deviation from the first population is either greater than or less than (but not both) the second population standard deviation . The choice is determined by the problem. For example, if we are testing a new process, we may only be interested in knowing if the new process is less variable than the old process. The test statistic for F is simply

$$F = \frac{s_1^2}{s_2^2}$$

where s_1^2 and s_2^2 are the sample variances with N1 and N2 number of observations respectively,. The more this ratio deviates from 1, the stronger the evidence for unequal population variances.

The variance are arranged so that F>1. That is; s12>s22.

We use the F-test as the Student's t test, only we are testing for significant differences in the variances.

The hypothesis that the two standard deviations are equal is rejected if

$$F > F_{(\alpha,N1-1,N2-1)} \quad \text{for an upper one-tailed test}$$
$$F < F_{(1-\alpha,N1-1,N2-1)} \quad \text{for a lower one-tailed test}$$
$$F < F_{(1-\alpha/2,N1-1,N2-1)} \quad \text{for a two-tailed test}$$

or

$$F > F_{(\alpha/2,N1-1,N2-1)}$$

Where $F_{(\alpha,k-1,N-k)}$ is the critical value of the F distribution with and degrees of freedom and a significance level of .

ANOVA (ANALYSY OF VARIENCE):

- **Analysis of variance** (**ANOVA**) is a collection of statistical models and their associated estimation procedures (such as the "variation" among and between groups) used to analyze the differences among group means in a sample. ...
- **ANOVA** is useful for comparing (testing) three or more group means for statistical significance.
- Analysis of variance (ANOVA) is an analysis tool used in statistics that splits the aggregate variability found inside a data set into two parts: systematic factors and random factors.
- The systematic factors have a statistical influence on the given data set, but the random factors do not.

- Analysts use the analysis of the variance test to determine the result that independent variables have on the dependent variable amid a regression study.

The ANOVA Test:

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.

Types of ANOVA

- There are two types of analysis of variance: one-way (or unidirectional) and two-way.
- One-way or two-way refers to the number of independent variables in your Analysis of Variance test.
- A one-way ANOVA evaluates the impact of a sole factor on a sole response variable. It determines whether all the samples are the same.
- The one-way ANOVA is used to determine whether there are any statistically significant differences between the means of three or more independent (unrelated) groups.
- 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. For example, a two-way ANOVA allows a company to compare worker productivity based on two independent variables, say salary and skill set. It is utilized to observe the interaction between the two factors.
- It tests the effect of two factors at the same time.

What Does "One-Way" or "Two-Way Mean?

One-way or two-way refers to the number of independent variables (IVs) in your Analysis of Variance test.

One-way has one independent variable (with 2 levels) and two-way has two independent variables (can have multiple levels).

For example, a one-way Analysis of Variance could have one IV (brand of cereal) and a two-way Analysis of Variance has two IVs (brand of cereal, calories).

One way analysis: When we are comparing more than three groups based on one factor variable, then it said to be one way analysis of variance (ANOVA). For example, if we want to compare whether or not the mean output of three workers is the same based on the working hours of the three workers.

Two way analysis: When factor variables are more than two, then it is said to be two way analysis of variance (ANOVA). For example, based on working condition and working hours, we can compare whether or not the mean output of three workers is the same.

K-way analysis: When factor variables are k, then it is said to be the k-way analysis of variance (ANOVA).

What are "Groups" or "Levels"?

Groups or levels are different groups in 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.

The use of this parametric statistical technique involves certain key assumptions, including the following:

1. **Independence of case:** Independence of case assumption means that the case of the dependent variable should be independent or the sample should be selected randomly. There should not be any pattern in the selection of the sample.

2. **Normality:** Distribution of each group should be normal. The Kolmogorov-Smirnov or the Shapiro-Wilk test may be used to confirm normality of the group.

3. **Homogeneity:** Homogeneity means variance between the groups should be the same. Levene's test is used to test the homogeneity between groups.

Key terms and concepts:

Sum of square between groups: For the sum of the square between groups, we calculate the individual means of the group, then we take the deviation from the individual mean for each group. And finally, we will take the sum of all groups after the square of the individual group.
Sum of squares within group: In order to get the sum of squares within a group, we calculate the grand mean for all groups and then take the deviation from the individual group. The sum of all groups will be done after the square of the deviation.

F –ratio: To calculate the F-ratio, the sum of the squares between groups will be divided by the sum of the square within a group.

Degree of freedom: To calculate the degree of freedom between the sums of the squares group, we will subtract one from the number of groups. The sum of the square within the group's degree of freedom will be calculated by subtracting the number of groups from the total observation.

BSS df = (g-1) for BSS is between the sum of squares, where g is the group, and df is the degree of freedom.

WSS df = (N-g) for WSS within the sum of squares, where N is the total sample size.

Significance:

- At a predetermine level of significance (usually at 5%), we will compare and calculate the value with the critical table value.
- Today, however, computers can automatically calculate the probability value for F-ratio.
- If p-value is lesser than the predetermined significance level, then group means will be different. Or, if the p-value is greater than the predetermined significance level, we can say that there is no difference between the groups' mean.

Extension:

- **MANOVA:** Analysis of variance (ANOVA) is performed when we have one dependent metric variable and one nominal independent variable.
- However, when we have more than one dependent variable and one or more independent variable, then we will use multivariate analysis of variance (MANOVA).
- **ANCOVA:** Analysis of covariance (ANCOVA) test is used to know whether or not certain factors have an effect on the outcome variable after removing the variance for quantitative predictors (covariates).

Uses of statistics in Machine Learning:

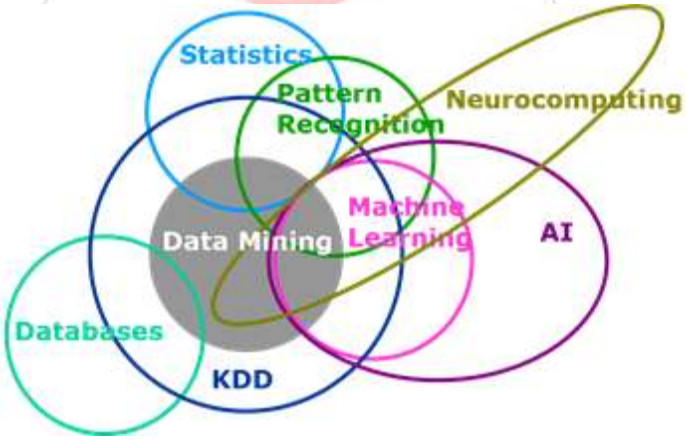
Many people have this doubt, what's the difference between statistics and machine learning? Is there something like machine learning vs. statistics?

From a traditional data analytics standpoint, the answer to the above question is simple.

- Machine Learning is an algorithm that can learn from data without relying on rules-based programming.
- Statistical modeling is a formalization of relationships between variables in the data in the form of mathematical equations.
- Machine learning is all about predictions, supervised learning, unsupervised learning, etc.
- Statistics is about sample, population, hypothesis, etc.

A statistician and machine learning expert at Stanford, calls machine learning "glorified statistics".

- Nowadays, both machine learning and statistics techniques are used in pattern recognition, knowledge discovery and data mining. The two fields are converging more and more even though the below figure may show them as almost exclusive.



machine learning and statistics share the same goal: **Learning from data**. Both these methods focus on drawing knowledge or insights from the data. But, their methods are affected by their inherent cultural differences.

- Machine learning is a subfield of computer science and artificial intelligence. It deals with building systems that can learn from data, instead of explicitly programmed instructions.
- A statistical model, on the other hand, is a subfield of mathematics.
- Machine learning is comparatively a new field.

The five tribes are

- **Symbolists:** The origin of this tribe is in logic and philosophy. This group relies on inverse deduction to solve problems.
- **Connectionists:** The origin of this tribe is in neuroscience. This group relies on backpropagation to solve problems.
- **Evolutionaries:** The origin of this tribe is in evolutionary biology. This group relies on genetic programming to solve problems.
- **Bayesians:** This origin of this tribe is in statistics. This group relies on probabilistic inference to solve problems.
- **Analogizers:** The origin of this tribe is in psychology. This group relies on kernel machines to solve problems.

CALCULAS

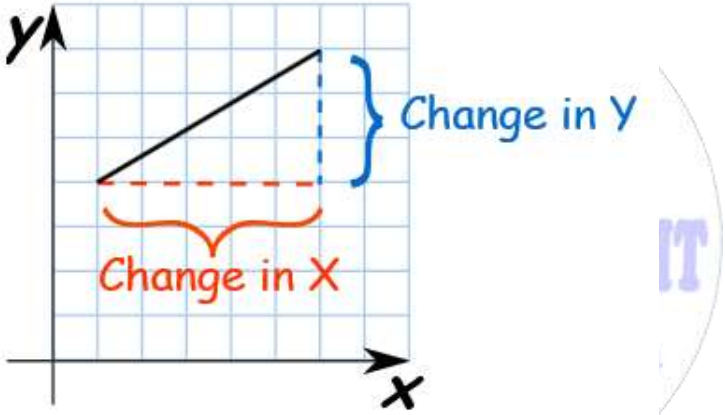
The word Calculus comes from Latin meaning "small stone", Because it is like understanding something by looking at small pieces.

Calculus (from Latin calculus, literally 'small pebble', used for counting and calculations, as on an abacus), is the mathematical study of continuous change, in the same way that geometry is the study of shape and algebra is the study of generalizations of arithmetic operations.

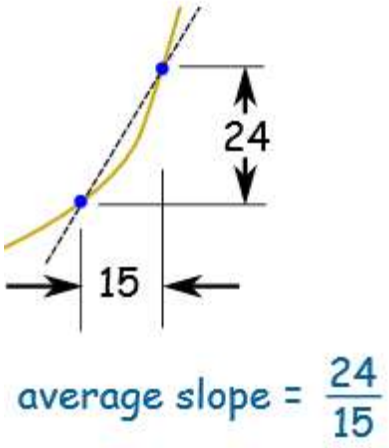
Differential Calculus cuts something into small pieces to find how it changes.

Integral Calculus joins (integrates) the small pieces together to find how much there is.

Slope = Change in Y/**Change in X**

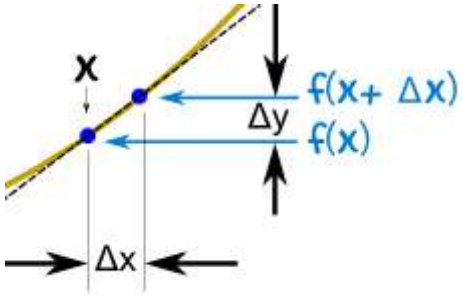


We can find an **average** slope between two points.



To find the derivative of a function y = f(x) we use the slope formula:

Slope = Change in Y /**Change in X** = Δy**Δx**



And (from the diagram) we see that:

x changes from x to x+Δx

y changes from f(x) to f(x+Δx)

Now follow these steps:

- Fill in this slope formula: Δy**Δx** = f(x+Δx) − f(x)**Δx**
- Simplify it as best we can
- Then make **Δx** shrink towards zero.

Notation

"Shrink towards zero" is actually written as a limit like this:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

"The derivative of **f** equals **the limit as Δx goes to zero** of f(x+Δx) - f(x) over Δx"

Or sometimes the derivative is written like this (explained on Derivatives as dy/dx):

$$\frac{dy}{dx} = \frac{f(x+dx) - f(x)}{dx}$$

The process of finding a derivative is called "differentiation".

What is a 'Derivative':

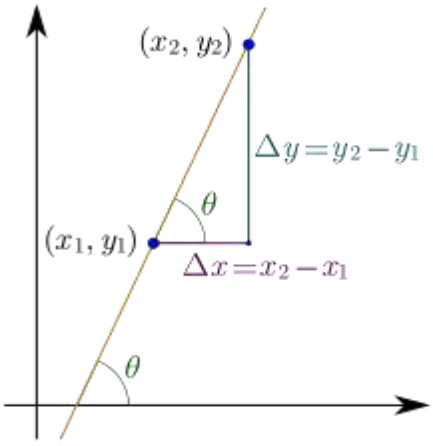
- A derivative is a financial security with a value that is reliant upon or derived from an underlying asset or group of assets.
- The derivative itself is a contract between two or more parties based upon the asset or assets.
- Its price is determined by fluctuations in the underlying asset. The most common underlying assets include stocks, bonds, commodities, currencies, interest rates and market indexes.

Definition: A derivative is a contract between two parties which derives its value/price from an underlying asset. The most common types of derivatives are futures, options, forwards and swaps.

Description: It is a financial instrument which derives its value/price from the underlying assets. Originally, underlying corpus is first created which can consist of one security or a combination of different securities. The value of the underlying asset is bound to change as the value of the underlying assets keep changing continuously.

Gradient or slope:

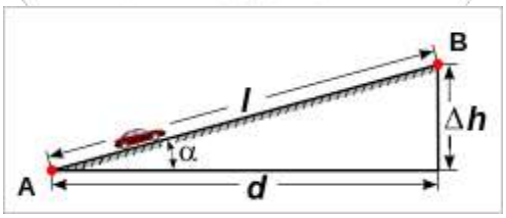
- A gradient is a vector, and slope is a scalar. Gradients really become meaningful in multivarible functions, where the gradient is a vector of partial derivatives. With single variable functions, the gradient is a one dimensional vector with the slope as its single coordinate (so, not very different to the slope at all).
- The **slope** or **gradient** of a line is a number that describes both the direction and the steepness of the line. Slope is often denoted by the letter m; there is no clear answer to the question why the letter m is used for slope, but it might be from the "m for multiple" in the equation of a straight line "y = mx + b" or "y = mx + c".



The slope of a line in the plane containing the x and y axes is generally represented by the letter m, and is defined as the change in the y coordinate divided by the corresponding change in the x coordinate, between two distinct points on the line. This is described by the following equation:

$$m = \frac{\Delta y}{\Delta x} = \frac{\text{vertical change}}{\text{horizontal change}} = \frac{\text{rise}}{\text{run}}.$$

- The **grade** (also called **slope**, **incline**, **gradient**, **mainfall**, **pitch** or **rise**) of a physical feature, landform or constructed line refers to the tangent of the angle of that surface to the horizontal.
- It is a special case of the slope, where zero indicates horizontality. A larger number indicates higher or steeper degree of "tilt".
- Often slope is calculated as a ratio of "rise" to "run", or as a fraction ("rise over run") in which run is the horizontal distance and rise is the vertical distance.
- The grades or slopes of existing physical features such as canyons and hillsides, stream and river banks and beds are often described.
- Grades are typically specified for new linear constructions (such as roads, landscape grading, roof pitches, railroads, aqueducts, and pedestrian or bicycle circulation routes). The grade may refer to the longitudinal slope or the perpendicular cross slope.



d = run
Δh = rise
l = slope length
α = angle of inclination

There are several ways to express slope:

1. as an angle of inclination to the horizontal. (This is the angle α opposite the "rise" side of a triangle with a right angle between vertical rise and horizontal run.)
2. as a percentage, the formula for which is 100(run/rise)which could also be

expressed as the tangent of the angle of inclination times 100. In the U.S., this percentage "grade" is the most commonly used unit for communicating slopes in transportation (streets, roads, highways and rail tracks), surveying, construction, and civil engineering.

3. as a per mille figure, the formula for which is 100(run/rise)which could also be expressed as the tangent of the angle of inclination times 1000. This is commonly used in Europe to denote the incline of a railway.
4. as a ratio of one part rise to so many parts run. For example, a slope that has a rise of 5 feet for every 100 feet of run would have a slope ratio of 1 in 20. (The word "in" is normally used rather than the mathematical ratio notation of "1:20"). This is generally the method used to describe railway grades in Australia and the UK.
    1. as a ratio of many parts run to one part rise, which is the inverse of the previous expression (depending on the country and the industry standards). For example, "slopes are expressed as ratios such as 4:1. This means that for every 4 units (feet or meters) of horizontal distance there is a 1-unit (foot or meter) vertical change either up or down."

Any of these may be used. Grade is usually expressed as a percentage, but this is easily converted to the angle α from horizontal or the other expressions.

- Slope may still be expressed when the horizontal run is not known: the rise can be divided by the hypotenuse (the slope length).

Interesting Functions

Try finding the slope of y = x^2 at:

- x = 1
- x = 2
- x = 3

Try finding the slope of y = ln(x) at:

- x = 1
- x = 1.5
- x = 2

Try finding the slope of y = e^x at:

- **y** = 1 (x=0)

- **y** = 1.2
- **y** = 1.5

Accuracy

The are only a few hundred pixels in either direction, and so the calculations are not totally accurate. But they should give you a good feel for what is going on.

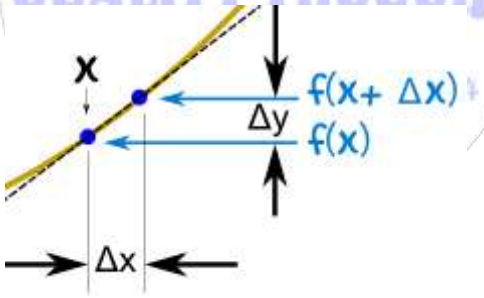
And don't worry, you can usually use **differential calculus** to find an accurate answer.

Derivatives are all about **change** ...

... they show how fast something is changing (called the **rate of change**) at any point.

In Introduction to Derivatives (please read it first!) we looked at how to do a derivative using **differences** and **limits**.

Here we look at doing the same thing but using the "dy/dx" notation (also called Leibniz's notation) instead of limits.



We start by calling the function "y":

y = f(x)

1. Add Δx

When x increases by Δx, then y increases by Δy :

y + Δy = f(x + Δx)

2. Subtract the Two Formulas

From: y + Δy = f(x + Δx)

Subtract: $y = f(x)$

To Get: $y + \Delta y - y = f(x + \Delta x) - f(x)$

Simplify: $\Delta y = f(x + \Delta x) - f(x)$

3. Rate of Change

To work out how fast (called the **rate of change**) we **divide by Δx**:

$\Delta y \mathbf{\Delta x} = f(x + \Delta x) - f(x) \mathbf{\Delta x}$

4. Reduce Δx close to 0

We can't let Δx become 0 (because that would be dividing by 0), but we can make it **head towards zero** and call it "dx":

Δx → dx

You can also think of "dx" as being **infinitesimal**, or infinitely small.

Likewise Δy becomes very small and we call it "dy", to give us:

dy **dx** = f(x + dx) − f(x) **dx**

Try It On A Function

Let's try $f(x) = x^2$

dy **dx**	= f(x + dx) − f(x) **dx**	
	= $(x + dx)^2 - x^2$ **dx**	$f(x) = x^2$
	= $x^2 + 2x(dx) + (dx)^2 - x^2$ **dx**	Expand $(x+dx)^2$
	= $2x(dx) + (dx)^2$ **dx**	$x^2 - x^2 = 0$
	= 2x + dx	Simplify fraction
	= 2x	dx goes towards 0

So the derivative of $\mathbf{x^2}$ is **2x**

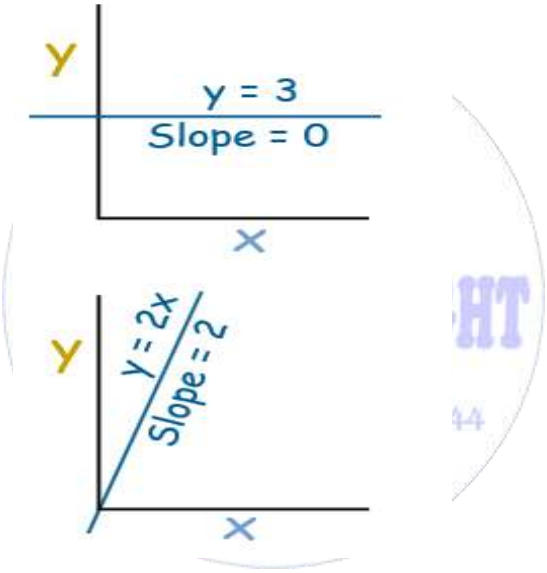
Derivative Rules:

There are **rules** we can follow to find many derivatives.

For example:

- The slope of a **constant** value (like 3) is always 0
- The slope of a **line** like 2x is 2, or 3x is 3 etc
- and so on.

Here are useful rules to help you work out the derivatives of many functions (with examples below). Note: the little mark ’ means "Derivative of".



"The derivative of" is also written d **dx**

So d **dx** sin(x) and sin(x)’ both mean "The derivative of sin(x)"

Power Rule

The derivative of x^n is $nx^{(n-1)}$."The derivative of" can be shown with the little mark ’

So we get this definition:

$f'(x^n) = nx^{(n-1)}$

Example: What is the derivative of x^3 ?

$f'(x^3) = 3x^{3-1} = \mathbf{3x^2}$

Partial derivatives:

a **partial derivative** of a function of several variables is its derivative with respect to one of those variables, with the others held constant (as opposed to the total derivative, in which all variables are allowed to vary). Partial derivatives are used in vector calculus and differential geometry.

The partial derivative of a function f(x,y…..)with respect to the variable Z is variously denoted by

$$f'_x,\ f_x,\ \partial_x f,\ D_x f,\ D_1 f,\ \frac{\partial}{\partial x} f,\ \text{or}\ \frac{\partial f}{\partial x}.$$

Defining the Partial Derivative

When we write u = u(x,y), we are saying that we have a function, u, which depends on two independent variables: x and y. We can consider the change in u with respect to either of these two independent variables by using the partial derivative. The partial derivative of u with respect to x is written as

$$\frac{\partial u}{\partial x}$$

What this means is to take the usual derivative, but only x will be the variable. All other variables will be treated as constants. We can also determine how u changes with y when x is held constant. This is the partial of u with respect to y. It is written as

$$\frac{\partial u}{\partial y}$$

For example, if u = y * x^2, then

$$\frac{\partial u}{\partial x} = 2xy$$

Likewise, we can differentiate with respect to y and treat x as a constant:

$$\frac{\partial u}{\partial y} = x^2$$

The **rule** for partial derivatives is that we differentiate with respect to one variable while keeping all the other variables constant. As another example, find the partial derivatives of u with respect to x and with respect to y for

$$u = e^{-2x} \cos 3y$$

To do this example, we will need the derivative of an exponential,

$$\frac{\mathrm{d}}{\mathrm{d}x} e^{ax} = \left(\frac{\mathrm{d}}{\mathrm{d}x}(ax) \right) e^{ax} = ae^{ax}$$

and the derivative of a cosine,

$$\frac{\mathrm{d}}{\mathrm{d}x} \cos ax = \left(\frac{\mathrm{d}}{\mathrm{d}x}(ax) \right) (-\sin ax) = -a \sin ax$$

Thus,

$$\frac{\partial u}{\partial x} = -2e^{-2x} \cos 3y$$

and

$$\frac{\partial u}{\partial y} = -3e^{-2x} \sin 3y$$

Taking the Partial Derivative of a Partial Derivative

So far we have defined and given examples for first-order partial derivatives. **Second-order partial derivatives** are simply the partial derivative of a first-order partial derivative. We can have four second-order partial derivatives:

$$\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) = \frac{\partial^2 u}{\partial x^2}$$
$$\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) = \frac{\partial^2 u}{\partial y^2}$$
$$\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right) = \frac{\partial^2 u}{\partial y \partial x}$$
$$\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}\right) = \frac{\partial^2 u}{\partial x \partial y}$$

Continuing with our first example of u = y * x^2,

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)$$
$$= \frac{\partial}{\partial x}(2x) = 2$$

and

$$\frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)$$
$$= \frac{\partial}{\partial y}(x^2) = 0$$

Likewise,

$$\frac{\partial^2 u}{\partial y \partial x} = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right)$$
$$= \frac{\partial}{\partial y}(2xy) = 2x$$

and

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}\right)$$
$$= \frac{\partial}{\partial x}(x^2) = 2x$$

And, in our second example,

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)$$

$$= \frac{\partial}{\partial x}\left(-2e^{-2x}\cos 3y\right)$$

$$= 4e^{-2x}\cos 3y$$

and

$$\frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)$$

$$= \frac{\partial}{\partial y}\left(-3e^{-2x}\sin 3y\right)$$

$$= -9e^{-2x}\cos 3y$$

The **mixed partial derivatives** become

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}\right)$$

$$= \frac{\partial}{\partial x}\left(-3e^{-2x}\sin 3y\right)$$

$$= 6e^{-2x}\sin 3y$$

and

$$\frac{\partial^2 u}{\partial y \partial x} = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right)$$

$$= \frac{\partial}{\partial y}\left(-2e^{-2x}\cos 3y\right)$$

$$= 6e^{-2x}\sin 3y$$

Note that we will always get

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial^2 u}{\partial y \partial x}$$

This can be used to check our work.

Extending the Idea of Partial Derivatives

What if the variables x and y also depend on other variables? For example, we could have x = x(s,t) and y = y(s,t).

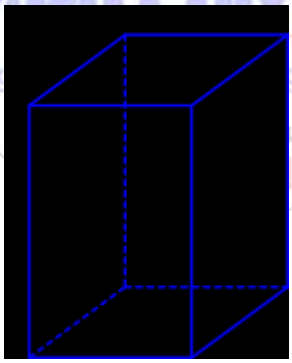
Then,

$$\frac{\partial u}{\partial s} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial s}$$

and

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$$

Example: The surface area of a square prism.



The surface is: the top and bottom with areas of x^2 each, and 4 sides of area xy:

$f(x,y) = 2x^2 + 4xy$

$f'_x = 4x + 4y$

$f'_y = 0 + 4x = 4x$

Chain rule (back propagation algorithm for training neural networks) :

The **chain rule** is used to differentiate composite functions. Specifically, it allows us to use differentiation rules on more complicated functions by differentiating the inner function and outer function separately.

The chain rule states that given a composite function f(g(x)), its derivative is the derivative of the outer function (leaving the inner function unchanged and in place) multiplied by the derivative of the inner function. Concisely,

Chain Rule

$$\frac{d}{dx} f(g(x)) \equiv f'(g(x)) \cdot g'(x)$$

The alternative statement of the chain in terms of function composition is

$$(f \circ g)' = (f' \circ g) \cdot g'.$$

Another way of stating this which makes it intuitive (almost trivial) would be

$$\frac{df \circ g}{dx} = \frac{df \circ g}{dg} \frac{dg}{dx}.$$

The chain rule says:

$$\frac{d}{dx}\left[f\left(g(x)\right)\right] = f'\left(g(x)\right)g'(x)$$

It tells us how to differentiate composite functions.

Chain Rule

The **chain rule** provides us a technique for finding the derivative of composite functions, with the number of functions that make up the composition determining how many differentiation steps are necessary. For example, if a composite function f(x) is defined as

$$f(x) = (g \circ h)(x) = g[h(x)]$$
$$\text{then } f'(x) = g'[h(x)] \cdot h'(x)$$

Note that because two functions, g and h, make up the composite function f, you have to consider the derivatives g′ and h′ in differentiating f(x).

Different forms of chain rule:

Consider the two functions f (x) and g (x). These two functions are differentiable. Then, the chain rule has two different forms as given below:

1. If F(x) = (f o g)(x), then the derivative of the composite function F(x) is,

F ′ (x) = f ′ (g(x)) g ′ (x)

2. If y = f (u) and u = g (x), then the derivative of the function y is,

$$\frac{dy}{dx} = \frac{dy}{du} \times \frac{du}{dx}$$

Uses of Calculus in Machine Learning:

- Derivatives are an important concept for machine learning. In the training process of many machine learning algorithms we use gradient descent (or one of its close relatives).
- Gradient descent uses the derivative of the sum of errors to update the systems parameters a little bit in such a way that the error decreases as much as possible.
- After every update the system learns to predict with a lower error. Let it run many iterations and it will converge at some (local) optimum. The system is now ready to start making predictions.
- The gradient descent algorithm is often used as a vectorised implementation which uses matrix calculations for highly parallel calculation on a GPU.
- Optimization algorithms like gradient descent use derivatives to decide whether to increase or decrease weights in order to maximize or minimize some objective (e.g. a model's accuracy or error functions).
- Derivatives also help us approximate nonlinear functions as linear functions (tangent lines), which have constant slopes. With a constant slope we can decide whether to move up or down the slope (increase or decrease our weights) to get closer to the target value (class label).

PYTHON

What is Python?

1. Python is an easy to learn, powerful programming language. The application development process much faster and easier
2. The programming language Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at Netherlands as a successor to the ABC (programming language)
3. Python First release happened in 1991.
4. Python was named for the BBC TV show Monty Python's Flying Circus.

Why python?

1. Easy to understand
2. Beginners language
3. Portable
4. Less lines of code
5. Simple to implement
6. Huge libraries supports

Features of python:

1. Easy to learn and use.
2. Expressive language
3. Interpreted language.
4. Cross platform language(windows, linux, mac)
5. Free to install and opensource
6. Object oriented language
7. Extensible, Awesome online community
8. Large standard library (numpy, scipy)
9. GUI programming (Tkinter)

Python Implementation alternatives:

1. CPython(stadnard implemenation of python)
2. Jython(Python for java)
3. IronPython(Python for .net)
4. Stackless (Python for concurrency)
5. PyPy (Python for speed)

Python Packages :

1. Web devlopment - Django,Flask frameworks,Pylons,Web2py framworks.
2. Artificiat Intelligence : Scikit-learn,Keras,TensorFlow,OpenCV
3. GUI - TKinter
4. Desktop Applications : Jython, WxPython
5. Game Development : pygame
6. Testing : Spliter Tool,pytest framework
7. Bigdata : Pydoop,DASK,PySpark Libraries

8. DataScience : NumPy, Pandas, matplotlib, seaborn libraries
9. AWS : boto
10. Robotic process : pyro
11. Web Scraping : Beautifulsoup4,urllib2,mechanize
12. Devops & System Admin : Os,Sys,Shutil,Glob,Subprocess,PathLib,fabric
13. Networking : Twisted,socket,client and server

Who uses Python :

- ✓ Data engineers, data scientists, System administrators and developers.
- ✓ Python is not industry specific, but task specific .Great for data processing, business intelligence, and some application development
- ✓ Google, You Tube,
- ✓ Instagram, DropBox,
- ✓ Survey Monkey, Quora,
- ✓ Pinterest, Reddit
- ✓ Yahoo Maps,…..

Python Variable:

variable :

Think of a number. Any number. Now, before you forget that number, let's store it for later.

When you think of a number, you are holding that value in your head. If you want to remember it you will write it down on a piece of paper.

And if it's really important, you will put it in a safe place.

In computer science, that safe place is a variable. They're called variables because, well, they're "capable of being varied or changed"

- ✓ A variable is a memory location where a programmer can store a value. Example : roll_no, amount, name etc.
- ✓ Value is either string, numeric etc. Example : "Sara", 120, 25.36
- ✓ Variables are created when first assigned.
- ✓ The value stored in a variable can be accessed or updated later.
- ✓ No declaration required
- ✓ The type (string, int, float etc.) of the variable is determined by Python
- ✓ The interpreter allocates memory on the basis of the data type of a variable.

Rules for python variables :

- Must begin with a letter (a - z, A - B) or underscore (_)

```
>>> @account_number = 34525
SyntaxError: invalid syntax
>>> _account_number=34525
>>> print(_account_number)
34525
```

--Other characters can be letters, numbers or _

- Must not contain any special characters !, @, #, $, %

```
>>> a@ = 20
SyntaxError: invalid syntax
>>> a$ = 49
SyntaxError: invalid syntax
```

- Case Sensitive

```
>>> product_name = 'TV'
>>> print(product_name)
TV
>>> print(Product_name)
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    print(Product_name)
NameError: name 'Product_name' is not defined
>>> print(PRODUCT_NAME)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print(PRODUCT_NAME)
NameError: name 'PRODUCT_NAME' is not defined
```

- There are some reserved words which you cannot use as a variable name because Python uses them for other things.

```
>>> for = 10
SyntaxError: invalid syntax
```

Good Variable Name :

- Choose meaningful name instead of short name. roll_no is better than rn.
- Maintain the length of a variable name. Roll_no_of_a-student is too long?
- Be consistent; roll_no or RollNo
- Begin a variable name with an underscore(_) character for a special case.

Multi assignment

```
a=10
print(a)
Name='RAVI'
```

Age=21

a = b = c = 1

print(a)

print(b)

print(c)

Swaping variable

Python swap values in a single line and this applies to all objects in python.

Syntax:

var1, var2 = var2, var1

Example:

```
>>> x = 10
>>> y = 20
>>> print(x)
10
>>> print(y)
20
>>> x, y = y, x
>>> print(x)
20
>>> print(y)
10
```


Input() Function :

```
>>> a = input()
12
>>> print(a)
12
>>> age = input("Enter your age \n")
Enter your age
28
>>> print(age)
28
```

Python Datatypes :

A datatype represents the type of data stored into a variable or memory.

Buitin datatypes -- Already available in python

User defined datatypes -- Datatypes created by programmers

1. Built in datatypes :

* None Type
* Numaric Types --> int,float,complex
* Sequences --> str,bytes,bytearray,list,tuple,range
* Sets --> set,frozenset
* Mappings --> dict

None :

'None' datatype represents an object that does not contain any value.

In java - NULL

In Python - None

Numeric data types :

1. int
2. float
3. complex

int :

* Int represents an integer number.
* It is number without decimal part and fraction part
* There is no limit for size of int datatype. It can store very integer number conveniently.

Eg :
```
>>> a = 20
>>> type(a)
<class 'int'>
```

Float :

* Float represents floating number
* A floating number contains decimal part

Eg:
```
>>> a = 223.345
>>> type(a)
<class 'float'>
```

```
>>> a = 22e5
>>> print(a)
2200000.0
```

Complex Data type :

* complex number is number that is written in the form of a+bj or a+bJ

a : real part

b : imaginary part

Eg :
```
>>> a = 1+2j
>>> b = 2+3j
>>> c = a+b
>>> print(c)
(3+5j)
```

bool data type :

The bool datatype in python represents boolean values.

```
>>> a = 20
>>> b = 10
>>> print(a>b)
True
>>> print(a<b)
False
>>> c=print(a>b)
True
>>> print(c)
None
>>> c=a>b
>>> print(c)
True
>>> True + True
2
>>> True - False
1
```

```
>>> file1 = True
>>> file2 = True
>>> file3 = True
>>> print(file1+file2+file3)
3
```

Sequences in Python :

A sequence represents a group of elements or items.

eg : group of integer numbers will form sequence

There are six types of sequences in python :

1. str
2. bytes
3. bytearray
4. list
5. tuple
6. range

str datatype :

- str represents string datatype.
- string represents group of characters
- string enclosed in single quotes or double quotes('',"")
- string can also be represent in """ or '''(if assign to variable then string otherwise it would be comment only)

  eg :

```
>>> print(word)
welcome
>>> word = "welcome"
>>> print(word)
welcome
>>> word = '''welcome'''
>>> print(word)
welcome
```

```
>>> word = """welcome"""
>>> print(word)
welcome
for = 'Quality Thought!'
SyntaxError: invalid syntax (for is key word, variable name should not be key word)
name = 'Quality Thought!'
print(name)        # Prints complete string
print (name[0])      # Prints first character of the string
print (name[2:5])    # Prints characters starting from 3rd to 5th
print (name[2:])     # Prints string starting from 3rd character
print (name * 2)     # Prints string two times
print (name + "INSTITUTE") # Prints concatenated string
Quality Thought!
Q
ali
ality Thought!
Quality Thought!Quality Thought!
Quality Thought!INSTITUTE

Note : explain about file processing , file name contain date
>>> name[0:4]
'Qual'
>>> name[0:6:2]
'Qai'
>>> print(name[0:4])
Qual
>>> print(name[0:6:2])
Qai
There is no char datatype like C in python.
>>> ch = 'A'
>>> type(ch)
<class 'str'>
```

```
>>> ch = 'A'
>>> type(ch)
<class 'str'>
>>> name = "srinivas"
>>> name[0]
's'
```

bytes data type :

The bytes data type represents a group of byte numbers just like an array does.

- should be 0 to 255
- Does not support negative numbers

```
>>> a = [12,23,45]
>>> x = bytes(a)
>>> type(x)
<class 'bytes'>
>>> x[0]=55
Traceback (most recent call last):
File "<pyshell#120>", line 1, in <module>
    x[0]=55
TypeError: 'bytes' object does not support item assignment
>>> a = [12,23,45,345]
>>> x = bytes(a)
Traceback (most recent call last):
File "<pyshell#122>", line 1, in <module>
    x = bytes(a)
ValueError: bytes must be in range(0, 256)
>>> a = [12,23,45,-23]
>>> x = bytes(a)
Traceback (most recent call last):
File "<pyshell#124>", line 1, in <module>
x = bytes(a)
ValueError: bytes must be in range(0, 256)
```

byte array data type :

- The bytearray datatype is similar to bytes data type. The difference is that the bytes type array can not be modified but bytearray can be modified

```
>>> a = [12,23,34,54]
>>> x = bytearray(a)
>>> print(x[0])
12
>>> x[0] = 55
>>> for i in x:
        print(i)
55
23
34
54
```

List data type :

- Array : store only similar datatype elements, fixed size
- List : Store different datatype elements, can grow dyanamically.

* List represent in [] and elements separated by ,

```
>>> a= [101,"srinivas",'NRT']
>>> b =[102,"phani","HYD"]
>>> a[0]
101
>>> a[0:2]
[101, 'srinivas']
>>> b[:3]
[102, 'phani', 'HYD']
>>> print(a)
[101, 'srinivas', 'NRT']
>>> type(a)
<class 'list'>

>>> print(a)
```

```
[101, 'srinivas', 'NRT']
>>> a[0]
101
>>> a[0] = 111
>>> print(a)
[111, 'srinivas', 'NRT']
```

list elements can be modified

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print(list)        # Prints complete list
print(list[0])     # Prints first element of the list
print(list[1:3])   # Prints elements starting from 2nd till 3rd
print(list[2:])    # Prints elements starting from 3rd element
print(tinylist * 2)  # Prints list two times
print(list + tinylist) # Prints concatenated lists
```

This produce the following result –

```
['abcd', 786, 2.23, 'john', 70.200000000000003]
abcd
[786, 2.23]
[2.23, 'john', 70.200000000000003]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john']
```

tuple data type :

- Tuple is similar to list.
- Can contain different datatypes of elements
- Represents ()
- Differene with list and tuple is
- can not modify the tuple so tuple is read only list.

```
>>> account_details = (101,"srinivas","NRT")
>>> type(account_details)
```

```
<class 'tuple'>
>>> account_details[0]
101
>>> account_details[1]
'srinivas'
>>> account_details[1] = "PHANI"
Traceback (most recent call last):
File "<pyshell#171>", line 1, in <module>
        account_details[1] = "PHANI"
TypeError: 'tuple' object does not support item assignment
```

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')
print(tuple)          # Prints complete list
print(tuple[0])       # Prints first element of the list
print(tuple[1:3])     # Prints elements starting from 2nd till 3rd
print(tuple[2:])      # Prints elements starting from 3rd element
print(tinytuple * 2)  # Prints list two times
print(tuple + tinytuple) # Prints concatenated lists
```

range datatype :

- range datatype represents seqence of numbers.
- The numbers in range are not modifiable
- Generally range is used for repeating a for loop for specific number of times.

```
>>> a = range(5)
>>> print(a)
range(0, 5)
>>> type(a)
<class 'range'>
>>> r = range(10,30,3)
>>> for i in r:
        print(i)
```

```
10
13
16
19
22
25
28
>>> r = range(10,30,3)
>>> for i in r:
        print(i)
10
13
16
19
22
25
28
```


Sets :

A set is an unordered collection of elements much like a set in mathematics.

- Order of elements is not maintained. It means elements may not appear in the same order as they entered in to set.
- Set does not accept duplicate elements

Two types of sets

1. set datatype
2. frozenset datatype

set datatype :

- set elements should separated with ,
- set always print only unique elements.

```
>>> s = {10,20,30,40,50}
>>> print(s)
{40, 10, 50, 20, 30}
>>> type(s)
```

```
<class 'set'>
>>> s = {10,10,20,20,30,30}
>>> print(s)
{10, 20, 30}

>>> str1 = set("srinivas")
>>> print(str1)
{'r', 'v', 'n', 'a', 's', 'i'}

>>> str2 = list(str1)
>>> print(str2)
['r', 'v', 'n', 'a', 's', 'i']
>>> print(str2[0])
r
>>> print(str2[1])
v
>>> s = {10,20,30,40}
>>> s.update([50])      // adding element to set
>>> print(s)
{40, 10, 50, 20, 30}
>>>
>>> print(s)
{40, 10, 50, 20, 30}
>>> s.remove(50)        // remove element from set
>>> print(s)
{40, 10, 20, 30}
```


fronzenset data type:

- Create frozenset by passing set data
- Cannot be modified (update and remove methods will not work)

```
>>> s = {50,60,70,80,90}
>>> fs = frozenset(s)
>>> type(fs)
```

```
<class 'frozenset'>
>>> print(fs)
frozenset({80, 50, 70, 90, 60})
>>> s = {50,50,60,60,70}
>>> fs1 = frozenset(s)
>>> type(fs1)
<class 'frozenset'>
>>> print(fs1)
frozenset({50, 60, 70})
```

Mapping Type :

- map represents a group of elements in the form of key values pairs so that when key is given will retrieve a value.
- The 'dict' datatype is an example of map
- dict represents dictionary that contains of pair of elements first one is Key and second one is Value
- key and value separated by ':'

```
>>> d = {101:"Ram",102:"Ravi",103:"Rani"}
>>> print(d)
{101: 'Ram', 102: 'Ravi', 103: 'Rani'}
>>> d[101]
'Ram'
>>> d[102]
'Ravi'
>>> type(d)
<class 'dict'>
```

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
print(dict['one'])      # Prints value for 'one' key
print(dict[2])          # Prints value for 2 key
```

```
print(tinydict)        # Prints complete dictionary
print(tinydict.keys())   # Prints all the keys
print(tinydict.values()) # Prints all the values
```

Command Line Arguments:

While running program arguments which are passing through command line
are called as command line arguments .
Here arguments are separated by space.
>> python add.py 10 20

* These arguments are stored by default in the form of strings in a list with name argv.
this is available in sys module.

argv[0] --> add.py
argv[1] --> 10
argv[2] --> 20

cmnd_line.py

```
===============
import sys
a = int(sys.argv[1])
b = int(sys.argv[2])
print(type(a))
print(type(b))
add = a + b
print(add)
```

NOTE : default it will take string so we have convert to integer
execution :
C:\Users\welcome\Desktop>python cmnd_line.py 10 20

Control statements(if..elif..else)

```
============================
```

If any statement expecting its sub statements then it should be ended with :

- conditional execution :

```
if x > 0 :
   print('x is positive')
```

- Alternative execution :

```
if x%2 == 0 :
   print('x is even')
else :
   print('x is odd')
```

- Chained conditionals :

```
if x < y:
   print 'x is less than y'
elif x > y:
   print 'x is greater than y'
else:
   print 'x and y are equal'
```


What is loop?

A loop is a sequence of instructions that is continually repeated until a certain condition reached.

Types of loops in Python :

1. for loop
2. while loop
3. Nested loop

for loop :

=========

for(i=0;i<n;i++) ===> (which is not implemented in python)

```
>>> for i in range(5):
        print(i)
0
1
2
```

```
3
4
>>> for i in range(1,5):
        print(i)
1
2
3
4
```

for_example.py

---

1. To do operation on each and every element of list.

```
a = [1,2,3,4]
s = 0
for i in a:
        s = s+i
print(s)
```


2. return to list

```
a = [1,2,3,4]

for i in a:
        print(i ** 2)
b = [  (i**2) for i in a]
print(b)
```

for with if :

```
student_marks = [12,3,42,34,4,2,2]
for data in student_marks:
        if( data % 2 == 0):
                print(data," is even number ")
        else:
                print(data," is odd number")
```

output:

12 is even number

3 is odd number

42 is even number

34 is even number

4 is even number

2 is even number

2 is even number

usage of end in print statement :

```
student_marks = [12,3,42,34,4,2,2]
for data in student_marks:
    if( data % 2 == 0):
        print(data," is even number",end = ":\n")
    else:
        print(data," is odd number",end = ":\n")
```

12 is even number:

3 is odd number:

42 is even number:

34 is even number:

4 is even number:

2 is even number:

2 is even number:

```
student_marks = [12,3,42,34,4,2,2]
for data in student_marks:
    if( data % 2 == 0):
        print(" %d is even number " %data)
    else:
        print(" %d is odd number " %data)
```

%d,%i --> integer

%s --> string

%f ---> float

```
>>> name = "ravi"
>>> age = 24

>>> print(" stuedent name %s and age is %d" %(name,age))
 stuedent name ravi and age is 24

dataset =  ['python','java','perl']

for i in dataset:
        print(i.upper())
PYTHON
JAVA
PERL

for i in dataset:
        print(i[0].upper()+i[1:])
Python
Java
Perl
```


for loop with else clause :

```
numbers = [10,20,30,40,50]
for i in numbers:
        print(i)
else:
        print("Loop completed successfuly")
```

Looping control statement:

A statement that alters the execution of loop from its designated sequence is called loop control statement

1. Break:

To break out the loop we can use break function.

syntax :

```
for varaiable_name in sequence :
        statement1
        statement2
        if(condition):
                break
```

```
>>> lst = [10,20,30,40]
>>> for i in lst:
                if(i == 30):
                        break
                print(i)
10
20
```

Note: if we stop loop using break statement then else part will not execute.

```
lst = [10,20,30,40]
for i in lst:
                if(i == 30):
                        break
                        print(i)
else :
        print("loop completed")
10
20
```

Continue statement:

Continue statement is used to tell python to jump to the next iteration of loop.

```
lst = [10,20,30,40]
for i in lst:
                if(i == 30):
```

```
                    continue
            print(i)
else :
    print("loop completed")
```

10

20

40

loop completed

while loop:

while loop is used to execute no.of statements till the condition passed in while loop.

once condition is false, the control will come out the loop.

syntax :

```
while<expression>:
    statement1
    statement2
>>> while(i < n):
    print(i)
    i += 1
>>> while(i < n):
    print(i)
```

infinite loop

while else loop:

```
a = int(input("Enter integer less 10\n"))
print(a)
```

Pythone Nested Loops:

Nested loop --> loop inside another loop

mulitiplication table :

```
for i in range(1,5):
    for j in range(1,10):
        print('%d * %d = %d' %(i,j,i*j))
        print("\n")
```

Introduction to Numpy:

Numpy is module that contains several classes,functions or variables. etc. to deal with scientific calculations in python. Numpy is useful to create and also process single and multidimensional arrays.

An array is an object that store a group of elements of same data type.

It means in case of numbes we can store only integer or only float but not one integer and one is float.

To work with numpy we should import numpy module.

```
import numpy
arr = numpy.array([10,20,30])
print(arr)
```

output:

```
array([10, 20, 30])
```



```
import numpy as np
arr = np.array([10,20,30])
print(arr)
```

output:

```
array([10, 20, 30])
```

```
from numpy import *
arr = array([10,20,30])
print(arr)
```

output:

```
[10, 20, 30]
```

Creating array can be done in several ways.

1. Using array() function
2. Using arange() function
3. Using zeros() and onces() function

1. Using array() function

```
>>> arr = array([10,20,30],int)
>>> type(arr)
<class 'numpy.ndarray'>
>>> arr
array([10, 20, 30])
>>> arr = array([10.4,20.3,30.5],int)
>>> arr
array([10, 20, 30])
>>> arr = array([10.4,20.3,30.5],float)
>>> arr
array([10.4, 20.3, 30.5])
>>> arr = array([10.4,20.3,30.5])
>>> arr
array([10.4, 20.3, 30.5])

>>> arr = array([10,20.3,30])
>>> arr
array([10. , 20.3, 30. ])
```


For string no need to specify data type

```
>>> arr = array(['a','b','c'])
>>> print(arr)
['a' 'b' 'c']
>>> arr[0]
'a'
```

Also we can create any array from other array.

```
>>> a = array([1,2,3,4])
>>> a
array([1, 2, 3, 4])
>>> b = array(a)
```

```
>>> b
array([1, 2, 3, 4])
>>> c = b
>>> c
array([1, 2, 3, 4])
>>>
```

2. creating array using arange() :

arange(start,stop,steppoint)

```
>>> from numpy import *
>>> a = arange(2,11,2)
>>> a
array([ 2,  4,  6,  8, 10])
```

3. creating array using zeros() and ones() function.

zeros(n,datatype) --> create array with all zeros

ones(n,datatype) --> creat array with all 1's.

```
>>> from numpy import *
>>> a = zeros(5,int)
>>> a
array([0, 0, 0, 0, 0])
>>> b = ones(5,int)
>>> b
array([1, 1, 1, 1, 1])
>>>
```

Mathematical operations on arrays:

```
>>> arr = array([1,2,3,4,5])
>>> arr1 = arr + 5
>>> arr1
array([ 6,  7,  8,  9, 10])
```

```
>>> arr2 = arr1 + 5
>>> arr2
array([11, 12, 13, 14, 15])
>>> arr3 = arr2 - arr1
>>> arr3
array([5, 5, 5, 5, 5])
```

sin(arr)
cos(arr)
tan(arr)
arcsin(arr)- sin inverse
arcos(arr) - cos inverse
arctan(arr)
log(arr) - logerthemic value
sum(arr)
prod(arr)
min(arr)
max(arr)
mean(arr)
median(arr)
var(arr)
std(arr)
unique(arr)
sort(arr)


```
>>>
>>> arr1
array([ 6,  7,  8,  9, 10])
>>> min(arr1)
6
>>> max(arr1)
10
```

```
>>> sum(arr1)
40
```

comparision arrays:

```
>>> from numpy import *
>>> a = array([1,2,3,4])
>>> b = array([0,2,2,4])
>>> c = a == b
>>> c
array([False,  True, False,  True])
>>>
>>> d = a>b
>>> d
array([ True, False,  True, False])

>>> print("check if any one element is true",any(d))
check if any one element is true True
>>> print("check if any one element is true",all(d))
check if any one element is true False
```

Slicing and indexing in Numpy arrays:

```
from numpy import *
a = arange(10,16)
print(a)
b = a[1:6:2]
print(b)
```

output:

```
[10 11 12 13 14 15]
[11 13 15]
```

Dimensions of array:

1 Dimension : When array contains only one row or one column.

```
>>> a = array([1,2,3,4])
>>> a
array([1, 2, 3, 4])

>>> b = array([10,
...            20,
...            30,
...            40
...            ])
>>> b
array([10, 20, 30, 40])
>>>
```

2 Dimension : Array contains more than one row and one column.
combination of several 1D arrays.

```
>>> arr2 = array([[10,20,30],
...               [20,30,40]])
>>> arr2
array([[10, 20, 30],
       [20, 30, 40]])
```

It is 2D array with 2 rows and 3 columns.

3 Dimension : combination of 2D arrays.

```
>>> arr4 = array([[[10,20],[30,40]], [[40,50],[50,60]] ])
>>> print(arr4)
[[[10 20]
  [30 40]]
```

```
 [[40 50]
  [50 60]]]
```

Attributes of array:

---

1. ndim attribute
2. shape attribute
3. size attribute

ndim attribute : represents no.of dimensions or axes of the array.

```
>>> arr4.ndim
3
```

shape attribute : Share attribute gives the shape of an array. The shape is a tuple listing number of elements along each dimension.

1D--> shape represents no.of elements.

2D --> no.of rows or columns of each

```
>>> arr1= array([1,2,3,4,5])
>>> print(arr1.shape)
(5,)

>>> arr2= array([[1,2,3],[4,5,6]])
>>> arr2.shape
(2, 3)
```

size attribute : Size attribute gives total number of elements in array.

```
>>> arr1 = array([1,2,3,4])
>>> arr1.size
4
>>> arr2 = array ([[1,2,3],[4,5,6]])
>>> arr2.size
6
```

The reshape method():

The reshape() is used to change the shape of array. The new array should have same elements of original array.

```
arr1 = arange(10)
print(arr1)

>>> arr1 = arange(10)
>>> arr1
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> arr2 = arr1.reshape(2,5)
>>> arr2
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
>>> arr3 = arr1.reshape(5,2)
>>> arr3
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> arr4 = arr1.reshape(4,3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: cannot reshape array of size 10 into shape (4,3)
```


The flatten() method :

The flatten() is used to return a copy of array collapsed into one dimension.

```
>>> arr1 = array([[1,2,3,4],[4,5,6,7]])
>>> print(arr1.flatten())
[1 2 3 4 4 5 6 7]
```

Working with multi dimensional arrays :

2D arrays,3D arrays etc. are called multi dimensional arrays.

we can create multi dimensional arrays in the following ways.

1. using array() function
2. Using ones() and zeros() function
3. Using eye() function
4. Using reshape() function

The array() function:

```
>>> arr_1D = array([1,2,3,4])
>>> arr_1D.ndim
1
>>> arr_2D = array([[1,2,3,4],[3,4,5,6]])
>>> arr_2D.ndim
2
>>> arr_3D = array([[[1,2],[3,4]],[[3,4],[5,6]]])
>>> arr_3D.ndim
3
```


ones , zeros functions :

---

ones((r,c),dtype)

```
>>> arr_1 = ones((4,3),int)
>>> arr_1
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]])
```

```
>>> arr_1 = ones((4,3),float)
>>> arr_1
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

zeros((r,c),dtype)

```
>>> arr_0 = zeros((4,3),int)
>>> arr_0
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
>>> arr_0 = zeros((4,3),float)
>>> arr_0
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

>>>eye() function : eye() creates a 2D array and fills the elements in

---

the diagonal with 1's. The general format of using that function is

eye(n,dtype=datatype)

the default datatype is false

```
>>> arr_e = eye(3,dtype=int)
>>> arr_e = eye(3)
>>> arr_e
[[1 0 0]
```

```
 [0 1 0]
 [0 0 1]]
>>> arr_e = eye(3,dtype=int)
>>> arr_e
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])
```

reshape() Function :

As we discussed it will convert 1D to multi dimensional arrays. The syntax is as below.

reshape(arryname,(n,r,c))


```
>>> a= array([1,2,3,4,5,6,7,8])
>>> b = reshape(a,(2,4))
>>> b
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> b = reshape(a,(4,2))
>>> b
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])

>>> a= array([1,2,3,4,5,6,7,8,9,10,11,12])
>>> b=a.reshape(3,4)
>>> b
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
>>> b = a.reshape(4,3)
>>>
>>> b
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

Matrices in numpy :

In mathematics, matrices are represents a rectangular array of elements arranged in rows and columns.

If matrix has only one row it is called row matrix. if it has only 1 column it is called as column matrix. Row and column matrix are nothing but 1D array.

When a matrix has more than 1 row and 1 column then it is called m x n matrix. where m is no.of rows and n is no.of columns.
using 'matrix' keyword directly we can convert array to matrix.


```
a = array([[1,2,3],[4,5,6]])
print(a)
print(type(a))
b = matrix(a)
print(type(b))
```

Using 'matrix' directly we can create matrix.

```
a = matrix([[1,2,3],[4,5,6]])
type(a)
```

```
str = '1 2;3 4;5 6'
b = matrix(str)
print(b)
```

output:

[[1 2]
[3 4]
[5 6]]

c = matrix("1 3;3 5;6 7")
print(c)

output:

[[1 3]
[3 5]
[6 7]]

To get diagonal elements of matrix :

---

c = matrix("1 3 6;3 5 7;6 7 8")
print(c)

b = diagonal(c)
print(b)

output :

[[1 3 6]
[3 5 7]
[6 7 8]]

[1 5 8]

max(),min(),sum(),mean():

---

a = matrix([[1,2,3],[4,5,6],[7,8,9]])
print(a)

```
print("max element in a is",a.max())
print("min element in a is",a.min())
print("sum of elements in a is",a.sum())
print("mean of a is",a.mean())
```

output :

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
max element in a is 9
min element in a is 1
sum of elements in a is 45
mean of a is 5.0
```

product of elments

---

prod(0): product of elements in columns level

prod(1): product of elements in row level.

```
>>> a = matrix(arange(12).reshape(3,4))
>>> a
matrix([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])
>>> a.prod(0)
matrix([[  0,  45, 120, 231]])
>>> a.prod(1)
matrix([[   0],
        [ 840],
        [7920]])
```

Transpose of matrix :

Rewriting matrix rows into columns and vice versa is called 'transpose'.

we can use transpose() or getT()

```
>>> a = matrix("1 2 3;4 5 6 ; 7 8 9")
>>> a
matrix([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
>>> b = a.transpose()
>>> b
matrix([[1, 4, 7],
        [2, 5, 8],
        [3, 6, 9]])
>>> c = a.getT()
>>> c
matrix([[1, 4, 7],
        [2, 5, 8],
        [3, 6, 9]])
```

Matrix add,sub,div:

```
>>> a = matrix(" 1 2; 2 3")
>>> a
matrix([[1, 2],
        [2, 3]])
>>> b = matrix("2 4 ; 4 5")
>>> b
matrix([[2, 4],
        [4, 5]])
```

```
>>> a
matrix([[1, 2],
        [2, 3]])
>>> b
matrix([[2, 4],
        [4, 5]])
>>> add = a+b
>>> add
matrix([[3, 6],
        [6, 8]])
>>> sub = b-a
>>> sub
matrix([[1, 2],
        [2, 2]])
>>> div = b/a
>>> div
matrix([[2.        , 2.        ],
        [2.        , 1.66666667]])
```

Matrix multiplication

---

a = m x n matrix

b = n x p matrix

no.of columns of a should be equal to no.of rows of b

result --> m x p matrix

```
>>> a = matrix(" 1 2; 2 3")
>>> a
matrix([[1, 2],
        [2, 3]])
>>> b = matrix("2 4 ; 4 5")
>>> b
matrix([[2, 4],
```

```
      [4, 5]])

>>> c = a * b
>>> c
matrix([[10, 14],
      [16, 23]])
```

Machine Learning

Machine Learning Introduction: -

1) Machine learning is a field of data science which prevents you from explicitly coding rules on a system but making the system learn by itself using some algorithms.

2) Machine Learning refers to the techniques involved in dealing with vast data in the most intelligent fashion (by developing algorithms) to derive actionable insights.

Machine learning - is the science of creating algorithms and program which learn on their own. Once designed, they do not need a human to become better. Some of the common applications of machine learning include following: Web Search, spam filters, recommender system, ad placement, credit scoring, fraud detection, stock trading, computer vision and drug design. An easy way to understand is this - it is humanly impossible to create models for every possible search or spam, so you make the machine intelligent enough to learn by itself. **When you automate the later part of data mining - it is known as machine learning.**

Evolution of machine learning

Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results. It's a science that's not new – but one that has gained fresh momentum.

While many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data – over and over, faster and faster – is a recent development. Here are a few widely publicized examples of machine learning applications you may be familiar with:

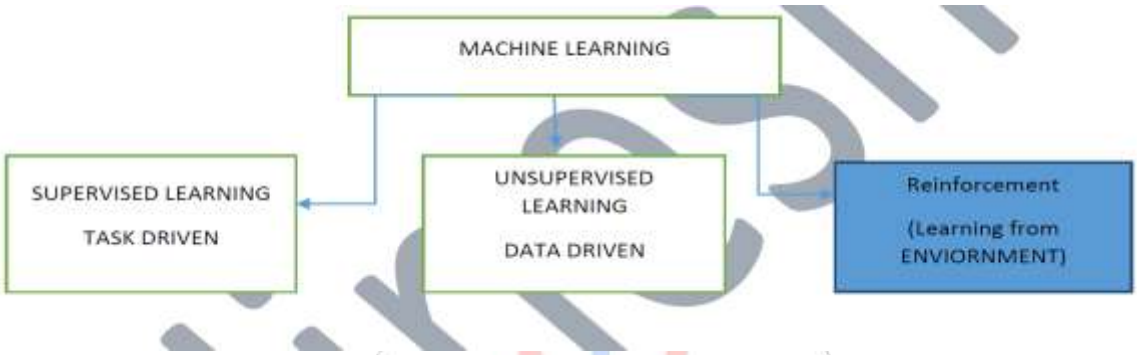
- The heavily hyped, self-driving Google car? The essence of machine learning.
- Online recommendation offers such as those from Amazon and Netflix? Machine learning applications for everyday life.
- Knowing what customers are saying about you on Twitter? Machine learning combined with linguistic rule creation.
- Fraud detection? One of the more obvious, important uses in our world today.
- ML focus on Development of computer Program that can change when Exposed to new data

• Process of ML is similar to that of data mining both the system search through data to look for patterns . However instead of extracting data for human compression

• ML uses data to detect patterns in data and adjust program accordingly

• Example: Facebook News Feed –Stastical and Predictive analytics

Types of Machine Learning: -



Supervised Learning / Predictive models:

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

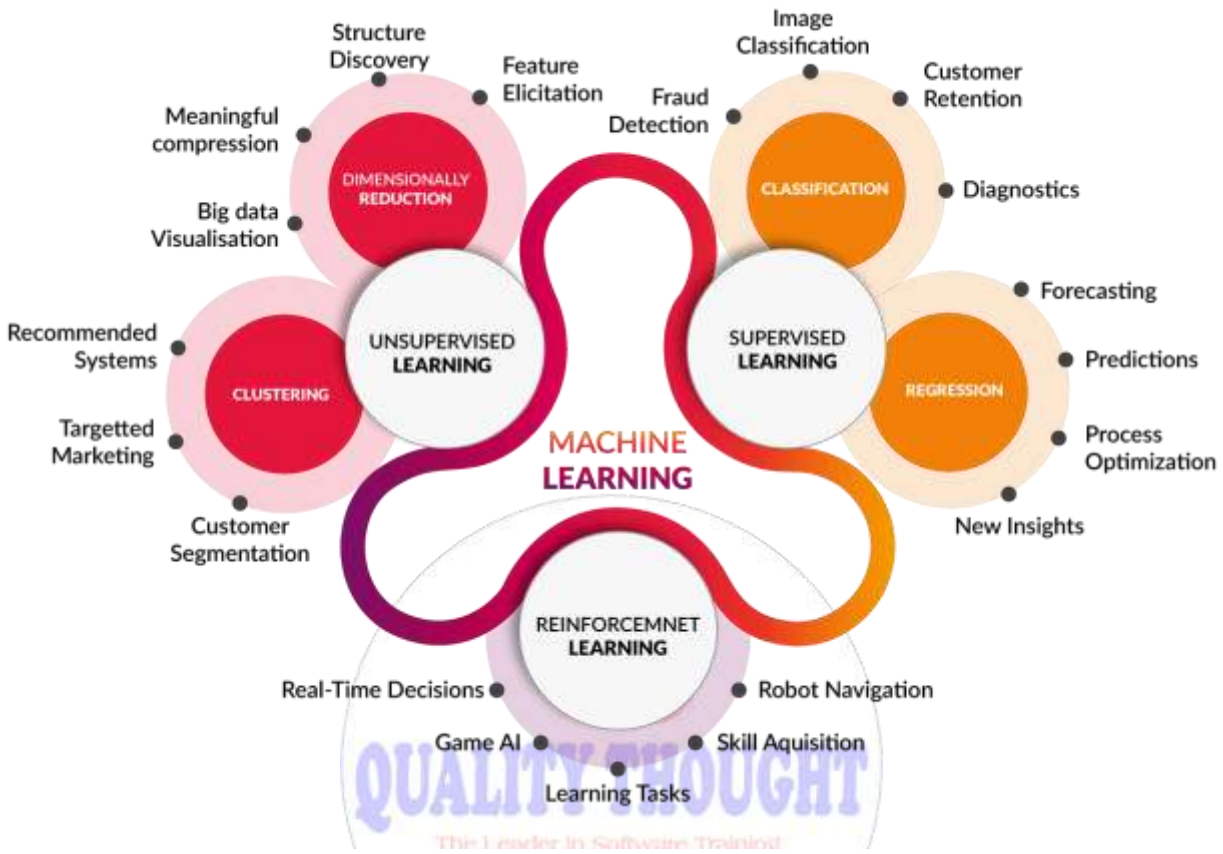
The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

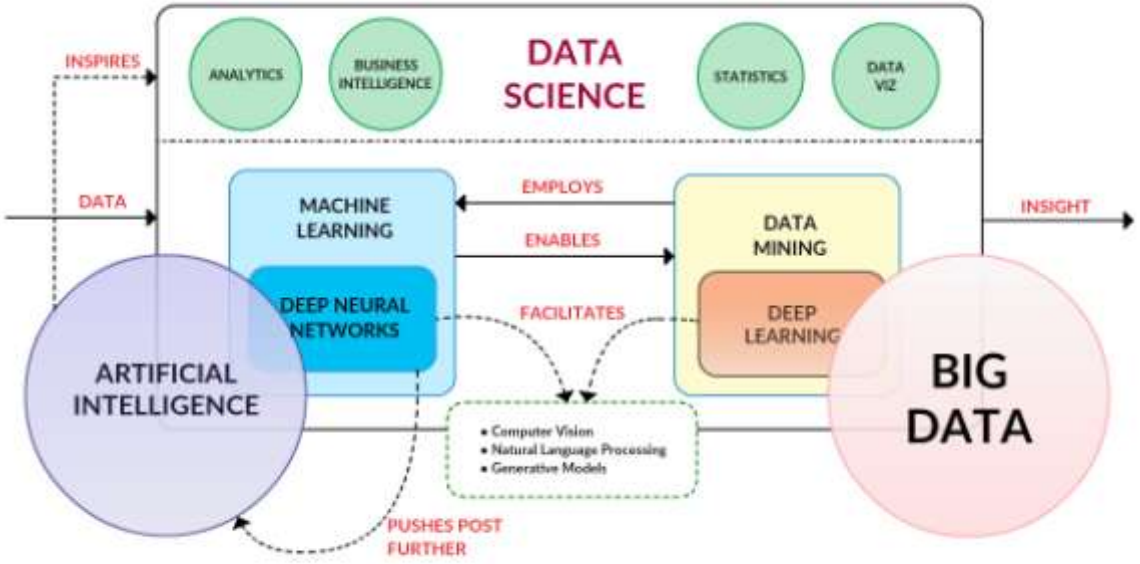
- It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process.
- We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.
- Supervised learning problems can be further grouped into regression and classification problems.
- The set of data (training data) consists of a set of input data and correct responses corresponding to every piece of data.
    - Based on this training data, the algorithm has to generalize such

  that it is able to correctly (or with a low margin of error) respond to all possible inputs.
    - In essence: The algorithm should produce sensible outputs for inputs that weren't encountered during training.
    - Also called learning from exemplars

- Classification: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

- **Regression:** A regression problem is when the output variable is a real value, such as “dollars” or “weight”.





Supervised Learning: Classification Problems

- "Consists of taking input vectors and deciding which of the N classes they belong to, based on training from exemplars of each class. "
- Is discrete (most of the time). i.e. an example belongs to precisely one class, and the set of classes covers the whole possible output space.
- How it's done: Find 'decision boundaries' that can be used to separate out the different classes.
- Given the features that are used as inputs to the classifier, we need to identify some values of those features that will enable us to decide which class the current input belongs to

Supervised Learning: Regression Problems

- Given some data, you assume that those values come from some sort of function and try to find out what the function is.
- In essence: You try to fit a mathematical function that describes a curve, such that the curve passes as close as possible to all the data points.
- So, regression is essentially a problem of function approximation or interpolation

Unsupervised Learning /Descriptive models: Unsupervised learning is where you only have input data (X) and no corresponding output variables.

- The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.
- These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher.
- Algorithms are left to their own devises to discover and present the interesting structure in the data.
- Unsupervised learning problems can be further grouped into clustering and association problems.

• **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

• **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y

Unsupervised Learning:

- Conceptually Different Problem.

- No information about correct outputs are available.

- No Regression No guesses about the function can be made

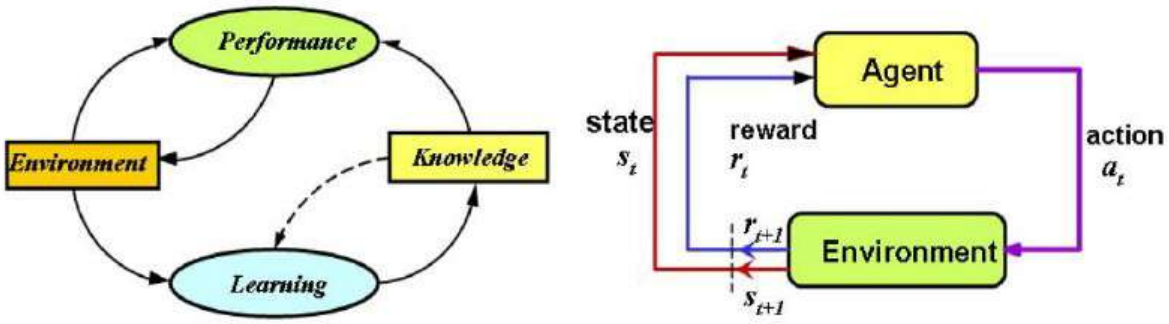
- Classification: - No information about the correct classes.

  But if we design our algorithm so that it exploits similarities between inputs so as to cluster inputs that are similar together, this might perform classification automatically

- In essence: The aim of unsupervised learning is to find clusters of similar inputs in the data without being explicitly told that some data points belong to one class and the other in other classes. The algorithm has to discover this similarity by itself

Reinforcement Learning:

- Stands in the middle ground between supervised and unsupervised learning.
- The algorithm is provided information about whether or not the answer is correct but not how to improve it
- The reinforcement learner has to try out different strategies and see which works best
- In essence: The algorithm searches over the state space of possible inputs and outputs in order to maximize a reward



Neural networks overview :-

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

"Connectionist" systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

Ex: in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge about cats, e.g., that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

Why use neural networks?

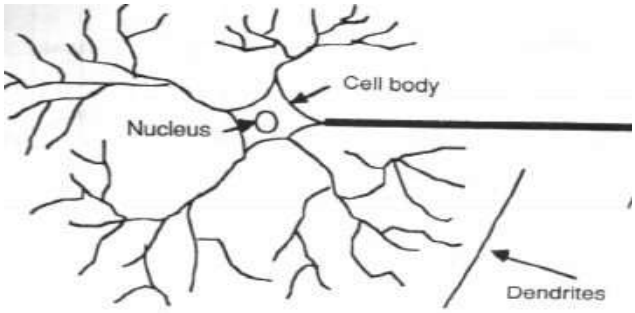
Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages include:

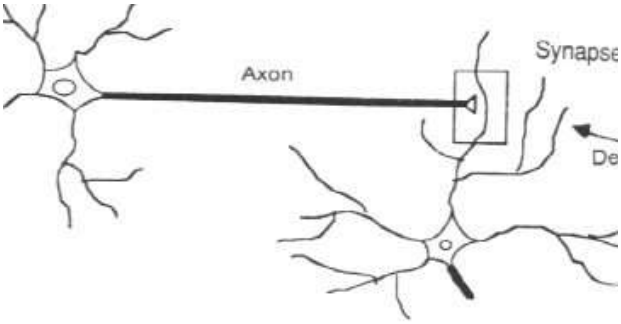
1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

How the Human Brain Learns?

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurones. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.



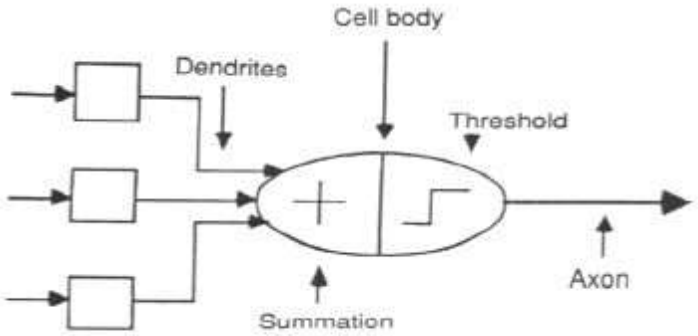
Components of a neuron



The synapse

2.2 From Human Neurones to Artificial Neurones

We conduct these neural networks by first trying to deduce the essential features of neurones and their interconnections. We then typically program a computer to simulate these features. However because our knowledge of neurones is incomplete and our computing power is limited, our models are necessarily gross idealisations of real networks of neurones.

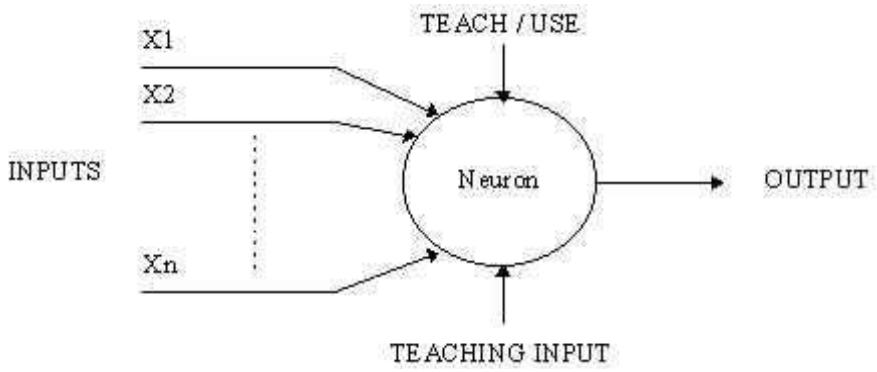


The neuron model

An engineering approach

1. A simple neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.



A simple neuron

2 Firing rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained.

A simple firing rule can be implemented by using Hamming distance technique. The rule goes as follows:

Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison , they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern in the 0-taught set. If there is a tie, then the pattern remains in the undefined state.

For example, a 3-input neuron is taught to output 1 when the input (X1,X2 and X3) is 111 or 101 and to output 0 when the input is 000 or 001. Then, before applying the firing rule, the truth table is;

X1:		0	0	0	0	1	1	1	1
X2:		0	0	1	1	0	0	1	1
X3:		0	1	0	1	0	1	0	1
OUT:		0	0	0/1	0/1	0/1	1	0/1	1

As an example of the way the firing rule is applied, take the pattern 010. It differs from 000 in 1 element, from 001 in 2 elements, from 101 in 3 elements and from 111 in 2 elements. Therefore, the 'nearest' pattern is 000 which belongs in the 0-taught set. Thus the firing rule requires that the neuron should not fire when the input is 001. On the other hand, 011

is equally distant from two taught patterns that have different outputs and thus the output stays undefined (0/1).

By applying the firing in every column the following truth table is obtained;

X1:		0	0	0	0	1	1	1	1
X2:		0	0	1	1	0	0	1	1
X3:		0	1	0	1	0	1	0	1
OUT:		0	0	0	0/1	0/1	1	1	1

The difference between the two truth tables is called the generalisation of the neuron. Therefore the firing rule gives the neuron a sense of similarity and enables it to respond 'sensibly' to patterns not seen during training.

3 Pattern Recognition - an example

An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward (figure 1) neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

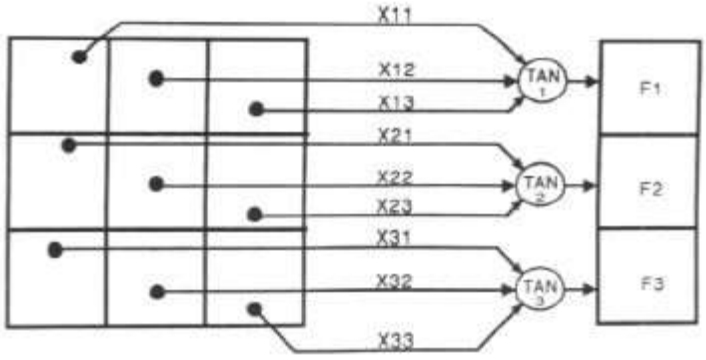


Figure 1.

For example: The network of figure 1 is trained to recognise the patterns T and H. The associated patterns are all black and all white respectively as shown below.

If we represent black squares with 0 and white squares with 1 then the truth tables for the 3 neurones after generalisation are;

X11:		0	0	0	0	1	1	1	1
X12:		0	0	1	1	0	0	1	1
X13:		0	1	0	1	0	1	0	1
OUT:		0	0	1	1	0	0	1	1

Top neuron

X21:		0	0	0	0	1	1	1	1
X22:		0	0	1	1	0	0	1	1
X23:		0	1	0	1	0	1	0	1
OUT:		1	0/1	1	0/1	0/1	0	0/1	0

Middle neuron

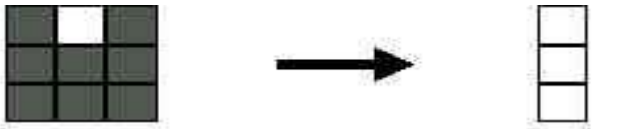
X21:		0	0	0	0	1	1	1	1
X22:		0	0	1	1	0	0	1	1
X23:		0	1	0	1	0	1	0	1
OUT:		1	0	1	1	0	0	1	0

Bottom neuron

From the tables it can be seen the following associasions can be extracted:

INPUT OUTPUT

In this case, it is obvious that the output should be all blacks since the input pattern is almost the same as the 'T' pattern.



INPUT OUTPUT

Here also, it is obvious that the output should be all whites since the input pattern is almost the same as the 'H' pattern.

INPUT OUTPUT

Here, the top row is 2 errors away from the a T and 3 from an H. So the top output is black. The middle row is 1 error away from both T and H so the output is random. The bottom row is 1 error away from T and 2 away from H. Therefore the output is black. The total output of the network is still in favour of the T shape.

4 A more complicated neuron

The previous neuron doesn't do anything that conventional conventional computers don't do already. A more sophisticated neuron (figure 2) is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are 'weighted', the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

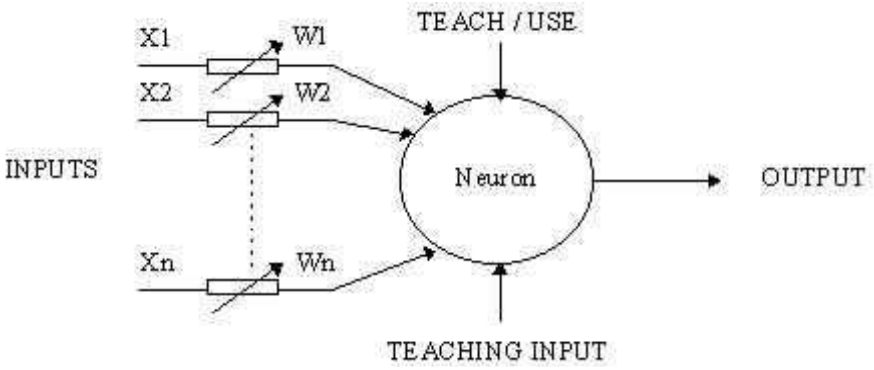


Figure 2. An MCP neuron

In mathematical terms, the neuron fires if and only if;

X1W1 + X2W2 + X3W3 + ... > T

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold. Various algorithms exist that cause the neuron to 'adapt'; the most used ones are the Delta rule and the back error propagation. The former is used in feed-forward networks and the latter in feedback networks.

Architecture of neural networks

1. Feed-forward networks

Feed-forward ANNs (figure 1) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organisation is also referred to as bottom-up or top-down.

2 Feedback networks

Feedback networks (figure 1) can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

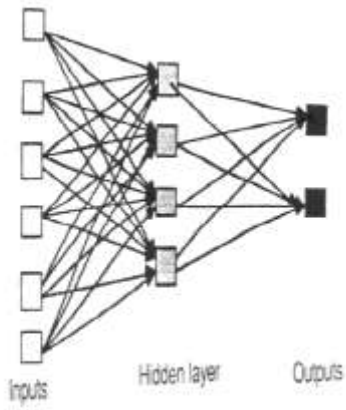


Figure 4.1 An example of a simple feedforward network

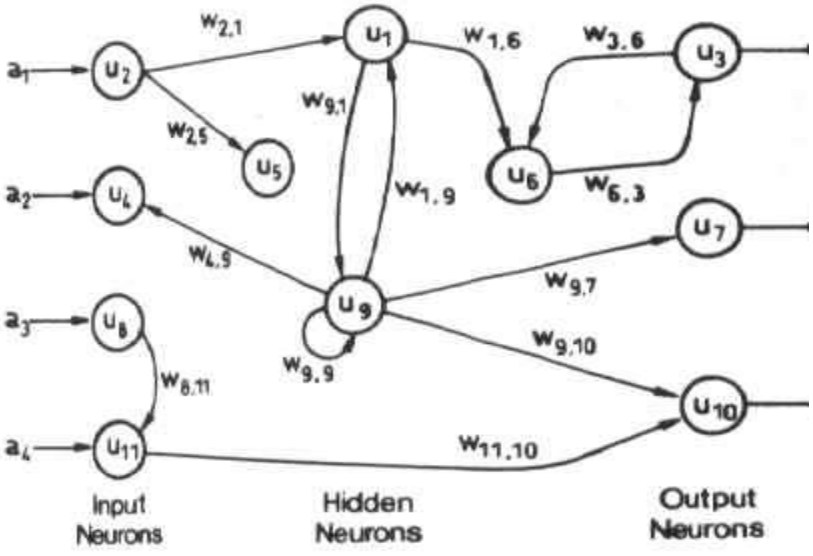


Figure 4.2 An example of a complicated network

3 Network layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "**input**" units is connected to a layer of "**hidden**" units, which is connected to a layer of **"output**" units. (see Figure 4.1)

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organisation, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organisations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

4 Perceptrons

The most influential work on neural nets in the 60's went under the heading of 'perceptrons' a term coined by Frank Rosenblatt. The perceptron (figure 4.4) turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, pre--processing. Units labelled A1, A2, Aj , Ap are called association units and their task is to extract specific, localised featured from the input images. Perceptrons mimic the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.

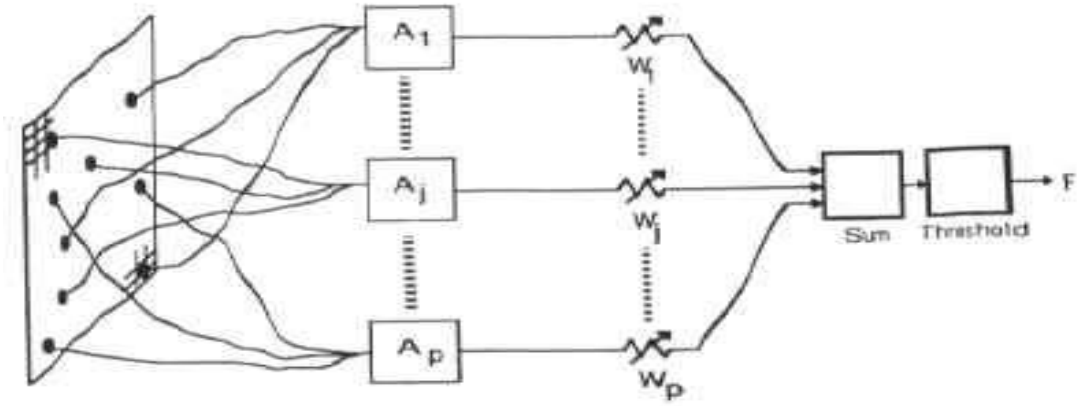


Figure 4.4

In 1969 Minsky and Papert wrote a book in which they described the limitations of single layer Perceptrons. The impact that the book had was tremendous and caused a lot of neural network researchers to loose their interest. The book was very well written and showed mathematically that single layer perceptrons could not do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected or not. What they did not realised, until the 80's, is that given the appropriate training, multilevel perceptrons can do these operations.

The main steps for building a Neural Network are:

- Define the model structure (such as number of input features and outputs)
- Initialize the model's parameters.
- Loop.
    - Calculate current loss (forward propagation)
    - Calculate current gradient (backward propagation)
    - Update parameters (gradient descent)
- Preprocessing the dataset is important.

- Tuning the learning rate (which is an example of a "hyperparameter") can make a big difference to the algorithm.

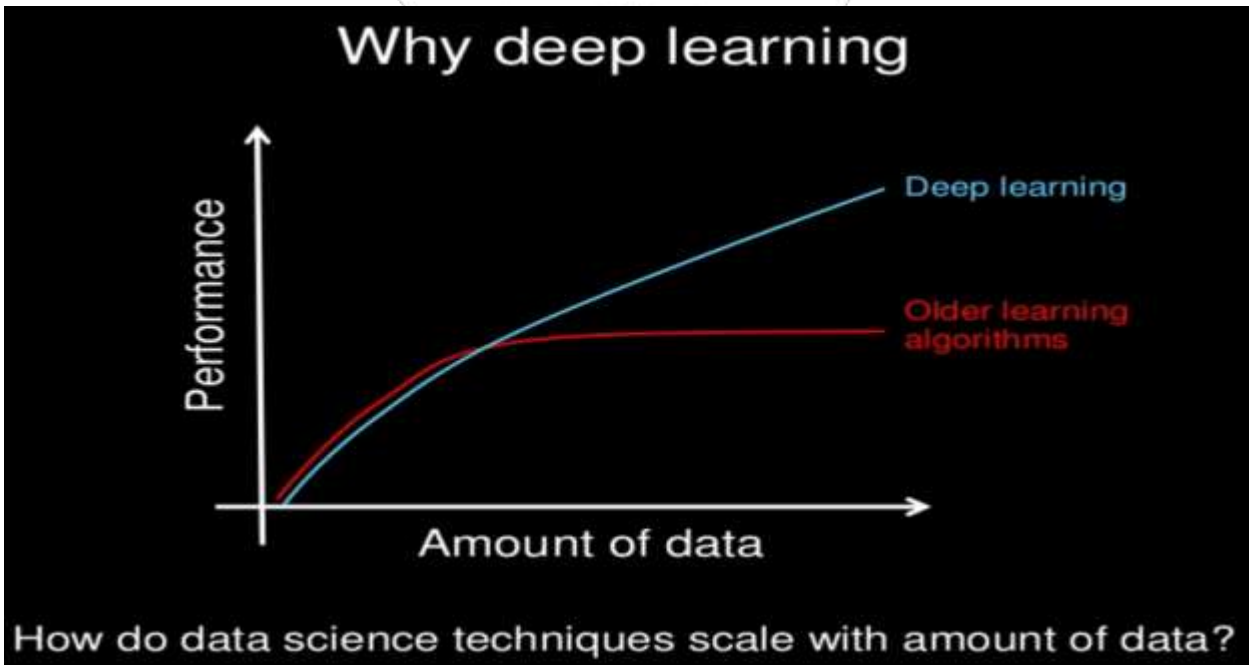
Deep Learning overview:-

Deep learning (also known as **deep structured learning** or **hierarchical learning**) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.[1][2][3]

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts.[4][5][6]

Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains, which make them incompatible with neuroscience evidences.

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.



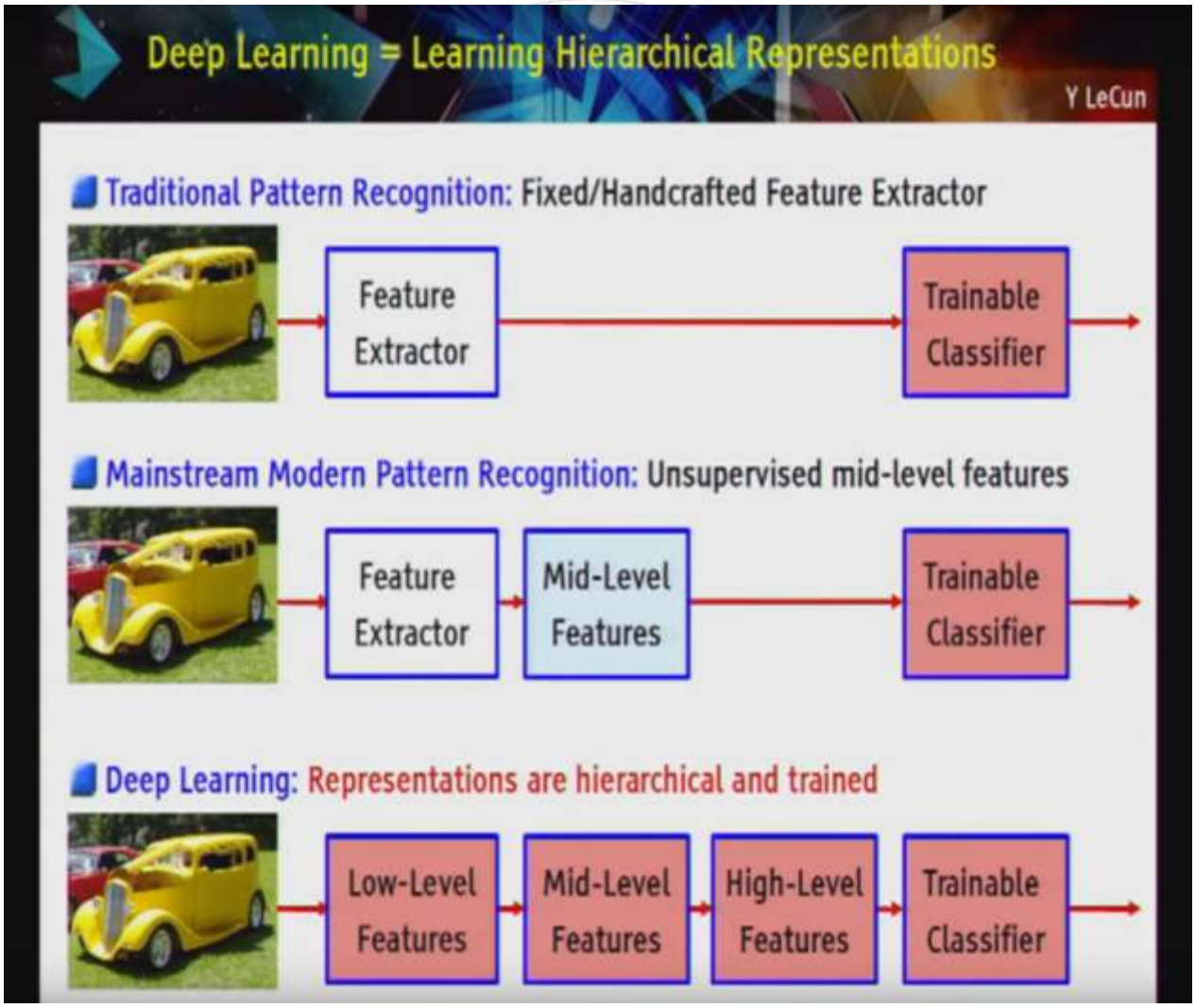
Why Call it "Deep Learning"?

Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory.

We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

It has been obvious since the 1980s that backpropagation through deep autoencoders would be very effective for nonlinear dimensionality reduction, provided that computers were fast enough, data sets were big enough, and the initial weights were close enough to a good solution. All three conditions are now satisfied.

Deep learning [is] … a pipeline of modules all of which are trainable. … deep because [has] multiple stages in the process of recognizing an object and all of those stages are part of the training"



Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. [...] The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

Deep learning is focused on training deep (many layered) neural network models using the backpropagation algorithm. The most popular techniques are:

- Multilayer Perceptron Networks.
- Convolutional Neural Networks.
- Long Short-Term Memory Recurrent Neural Networks.

Life Cycle of Machine Learning Model development:

Data Preparation

Stages of data preparation

Data preparation usually involves iterative steps. In this documentation the steps for getting data ready for supervised machine learning are identified as:

1. Gather data.
2. Clean the data.
3. Split the data.
4. Engineer features.
5. Preprocess the features.

Gather data

Finding data, especially data with the labels you need, can be a challenge. Your sources might vary significantly from one machine learning project to the next. If you find that you are merging data from different sources, or getting data entry from multiple places, you'll need to be extra careful in the next step.

Clean the data

Cleaning data is the process of checking for integrity and consistency. At this stage you shouldn't be looking at the data overall for patterns. Instead, you clean data by column (attribute), looking for such anomalies as:

- Instances with missing features.

- Multiple methods of representing a feature. For example, some instances might list a length measurement in inches, and others might list it in centimeters. It is crucial that all instances of a given feature use the same scale and follow the same format.
- Features with values far out of the typical range (outliers), which may be data-entry anomalies or other invalid data.
- Significant changes in the data over distances in time, geographic location, or other recognizable characteristics.
- Incorrect labels or poorly defined labeling criteria.

Split the data

You need at least three subsets of data in a supervised learning scenario: training data, evaluation data, and test data.

Training data is the data that you'll get acquainted with. You use it to train your model, yes, but you also analyze it as you develop the model in the first place.

Evaluation data is what you use to check your model's performance during the regular training cycle. It is your primary tool in ensuring that your model is generalizable to data beyond the training set.

Test data is used to test a model that's close to completion, usually after multiple training iterations. You should never analyze or scrutinize your test data, instead keeping it fresh until needed to test your model. That way you can be assured that you aren't making assumptions based on familiarity with the data that can pollute your training results.

Here are some important things to remember when you split your data:

- It is better to randomly sample the subsets from one big dataset than to use some pre-divided data, such as instances from two distinct date ranges or data-collection systems. The latter approach has an increased risk of non-uniformity that can lead to overfitting.
- Ideally you should assign instances to a dataset and keep those associations throughout the process.
- Experts disagree about the right proportions for the different datasets. However, regardless of the specific ratios you should have more training data than evaluation data, and more evaluation data than test data.

Engineer features

Before you develop your model, you should get acquainted with your training data. Look for patterns in your data, and think about what values could influence your target attribute.

This process of deciding which data is important for your model is called feature engineering.

Feature engineering is not just about deciding which attributes you have in your raw data that you want in your model. The harder and often more important work is extracting generalizable, indicative features from specific data. That means combining the data you have with your knowledge about the problem space to get the data you really need. It can be a complex process, and doing it right depends on understanding the subject matter and the goals of your problem. Here are a couple of examples:

Example data: residential address

Data about people often includes a residential address, which is a complex string, often hard to make consistent, and not particularly useful for many applications on its own. You should usually extract a more meaningful feature from it. Here are some examples of things you could extract from an address:

- Longitude and latitude
- Neighborhood
- Closest elementary school
- Legislative district
- Relative position to a landmark

Example data: timestamp

Another common item of data is a timestamp, which is usually a large numerical value indicating the amount of time elapsed since a common reference point. Here are some examples of things you might extract from a precise timestamp:

- Hour of the day
- Elapsed time since another event
- Time of day (morning, afternoon, evening, night)
- Whether some facility was open or closed at that time
- Frequency of an event (in combination with other instances)
- Position of the sun (in combination with latitude and longitude)

Here are some important things to note about the examples above:

- You can combine multiple attributes to make one generalizable feature. For example, address and timestamp can get you the position of the sun.

- You can use feature engineering to simplify data. For example, timestamp to time of day takes an attribute with seemingly countless values and reduces it to four categories.
- You can get useful features, and reduce the number of instances in your dataset, by engineering across instances. For example, use multiple instances to calculate the frequency of something.

When you're done you'll have a list of features to include when training your model.

One of the most difficult parts of the process is deciding when you have the right set of features. It's sometimes difficult to know which features are likely to affect your prediction accuracy. Machine learning experts often stress that it's a field that requires flexibility and experimentation. You'll never get it perfect the first try, so make your best guess and use the results to inform your next iteration.

Preprocessing data

So far this page has described generally applicable steps to take when getting your data ready to train your model. It hasn't mattered up to this point how your data is represented and formatted. Preprocessing is the next step: getting your prepared data into a format that works with the tools and techniques you use to train a model.

Data formats and Cloud ML Engine

Cloud ML Engine doesn't get involved in your data format; you can use whatever input format is convenient for your training application. That said, you'll need to have your input data in a format that TensorFlow can read. You also need to have your data in a location that your Cloud ML Engine project can access. The simplest solution is often to use a CSV file in a Google Cloud Storage bucket that your Google Cloud Platform project has access to. Some types of data, such as sparse vectors and binary data, can be better represented using TensorFlow'stf.train.Example format serialized in a TFRecords file.

Transforming data

There are many transformations that might be useful to perform on your raw feature data. Some of the more common ones are:

- Normalizing numerical values to be represented at a consistent scale (commonly a range between -1 and 1 or between 0 and 1).
- Representing non-numeric data numerically, such as changing categorical features to index values or one-hot vectors.
- Changing raw text strings to a more compact representation, like a bag of words.

Summary of data preparation and preprocessing

Cloud ML Engine doesn't impose specific requirements on your input data, leaving you to use whatever format works for your training application. Follow TensorFlow's data reading procedures.

- Use the raw data you have to get the data you need.
- Split your dataset into training, validation, and test subsets.
- Store your data in a location that your Cloud ML Engine project can access—a Cloud Storage bucket is often the easiest approach.
- Transform features to suit the operations you perform on them.

How Do You Start Machine Learning in Python?

The best way to learn machine learning is by designing and completing small projects.

Python Can Be Intimidating When Getting Started

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems.

There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will given you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

Beginners Need A Small End-to-End Project

Books and courses are frustrating. They give you lots of recipes and snippets, but you never get to see how they all fit together.

When you are applying machine learning to your own datasets, you are working on a project.

A machine learning project may not be linear, but it has a number of well known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

If you can do that, you have a template that you can use on dataset after dataset. You can fill in the gaps such as further data preparation and improving result tasks later, once you have more confidence.

Hello World of Machine Learning

The best small project to start with on a new tool is the classification of iris flowers (e.g. the iris dataset).
This is a good project because it is so well understood.

- Attributes are numeric so you have to figure out how to load and handle data.
- It is a classification problem, allowing you to practice with perhaps an easier type of supervised learning algorithm.
- It is a multi-class classification problem (multi-nominal) that may require some specialized handling.
- It only has 4 attributes and 150 rows, meaning it is small and easily fits into memory (and a screen or A4 page).
- All of the numeric attributes are in the same units and the same scale, not requiring any special scaling or transforms to get started.

Let's get started with your hello world machine learning project in Python.

Machine Learning in Python: Step-By-Step

In this section, we are going to work through a small machine learning project end-to-end.

Here is an overview of what we are going to cover:

- Installing the Python and SciPy platform.
- Loading the dataset.
- Summarizing the dataset.
- Visualizing the dataset.
- Evaluating some algorithms.
- Making some predictions.

Take your time. Work through each step.

Try to type in the commands yourself or copy-and-paste the commands to speed things up.

If you have any questions at all, please leave a comment at the bottom of the post.

1. Downloading, Installing and Starting Python SciPy

Get the Python and SciPy platform installed on your system if it is not already.

I do not want to cover this in great detail, because others already have. This is already pretty straightforward, especially if you are a developer. If you do need help, ask a question in the comments.

1.1 Install SciPy Libraries

This tutorial assumes Python version 2.7 or 3.5.

There are 5 key libraries that you will need to install. Below is a list of the Python SciPy libraries required for this tutorial:

- scipy
- numpy
- matplotlib
- pandas
- sklearn

There are many ways to install these libraries. My best advice is to pick one method then be consistent in installing each library.

The scipy installation page provides excellent instructions for installing the above libraries on multiple different platforms, such as Linux, mac OS X and Windows. If you have any doubts or questions, refer to this guide, it has been followed by thousands of people.

- On Mac OS X, you can use macports to install Python 2.7 and these libraries.
- On Linux you can use your package manager, such as yum on Fedora to install RPMs.
- If you are on Windows or you are not confident, I would recommend installing the free version of Anaconda that includes everything you need.

1.2 Start Python and Check Versions

It is a good idea to make sure your Python environment was installed successfully and is working as expected.

The script below will help you test out your environment. It imports each library required in this tutorial and prints the version.

1 Python

I recommend working directly in the interpreter or writing your scripts and running them on the command line rather than big editors and IDEs. Keep things simple and focus on the machine learning not the toolchain.

Type or copy and paste the following script:

```
# Check the versions of libraries
# Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```



```
Python: 2.7.11 (default, Mar  1 2016, 18:40:10)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)]
scipy: 0.17.0
numpy: 1.10.4
matplotlib: 1.5.1
pandas: 0.17.1
sklearn: 0.18.1
```

Compare the above output to your versions.

Ideally, your versions should match or be more recent. The APIs do not change quickly, so do not be too concerned if you are a few versions behind, Everything in this tutorial will very likely still work for you.

If you get an error, stop. Now is the time to fix it.

If you cannot run the above script cleanly you will not be able to complete this tutorial.

My best advice is to Google search for your error message or post a question on Stack Exchange.

2. Load The Data

We are going to use the iris flowers dataset. This dataset is famous because it is used as the "hello world" dataset in machine learning and statistics by pretty much everyone.

The dataset contains 150 observations of iris flowers. There are four columns of measurements of the flowers in centimeters. The fifth column is the species of the flower observed. All observed flowers belong to one of three species.

You can learn more about this dataset on Wikipedia.
In this step we are going to load the iris data from CSV file URL.

2.1 Import libraries

First, let's import all of the modules, functions and objects we are going to use in this tutorial.

```
# Load libraries
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

Everything should load without error. If you have an error, stop. You need a working SciPy environment before continuing. See the advice above about setting up your environment.

2.2 Load Dataset

We can load the data directly from the UCI Machine Learning repository.

We are using pandas to load the data. We will also use pandas next to explore the data both with descriptive statistics and data visualization.

Note that we are specifying the names of each column when loading the data. This will help later when we explore the data.

```
1 # Load dataset
2 url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
3 names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
4 dataset = pandas.read_csv(url, names=names)
```

The dataset should load without incident.

If you do have network problems, you can download the iris.data file into your working directory and load it using the same method, changing URL to the local file name.

3. Summarize the Dataset

Now it is time to take a look at the data. In this step we are going to take a look at the data a few different ways:

- Dimensions of the dataset.
- Peek at the data itself.
- Statistical summary of all attributes.
- Breakdown of the data by the class variable.

Don't worry, each look at the data is one command. These are useful commands that you can use again and again on future projects.

3.1 Dimensions of Dataset

We can get a quick idea of how many instances (rows) and how many attributes (columns) the data contains with the shape property.

```
1 # shape
2 print(dataset.shape)
```

You should see 150 instances and 5 attributes:

1 (150, 5)

3.2 Peek at the Data

It is also always a good idea to actually eyeball your data.

```
1 # head
2 print(dataset.head(20))
```

You should see the first 20 rows of the data:

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa

3.3 Statistical Summary

Now we can take a look at a summary of each attribute.

This includes the count, mean, the min and max values as well as some percentiles.

```
1 # descriptions
2 print(dataset.describe())
```

We can see that all of the numerical values have the same scale (centimeters) and similar ranges between 0 and 8 centimeters.

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667

4 std	0.828066	0.433594	1.764420	0.763161
5 min	4.300000	2.000000	1.000000	0.100000
6 25%	5.100000	2.800000	1.600000	0.300000
7 50%	5.800000	3.000000	4.350000	1.300000
8 75%	6.400000	3.300000	5.100000	1.800000
9 max	7.900000	4.400000	6.900000	2.500000

3.4 Class Distribution

Let's now take a look at the number of instances (rows) that belong to each class. We can view this as an absolute count.

```
1 # class distribution
2 print(dataset.groupby('class').size())
```

We can see that each class has the same number of instances (50 or 33% of the dataset).

```
1 Class
2 Iris-setosa       50
3 Iris-versicolor   50
4 Iris-virginica    50
```

4. Data Visualization

We now have a basic idea about the data. We need to extend that with some visualizations.

We are going to look at two types of plots:

1. Univariate plots to better understand each attribute.
2. Multivariate plots to better understand the relationships between attributes.

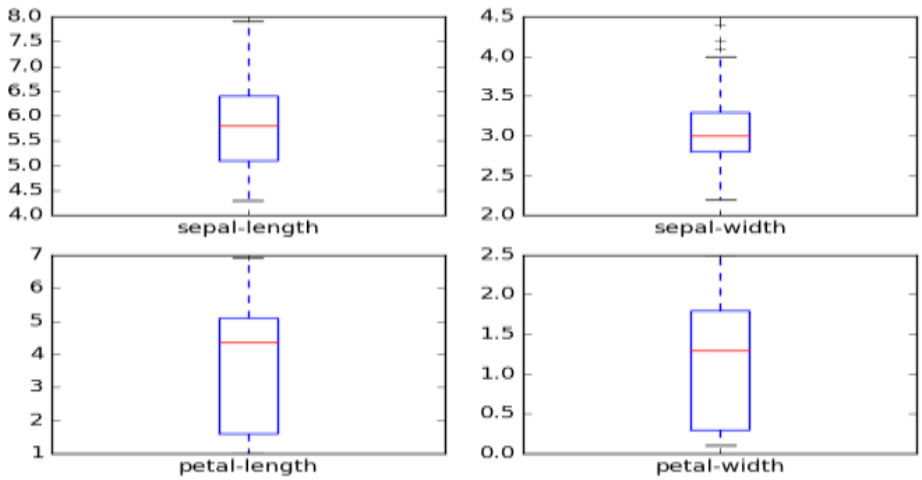
4.1 Univariate Plots

We start with some univariate plots, that is, plots of each individual variable.

Given that the input variables are numeric, we can create box and whisker plots of each.

```
1 # box and whisker plots
2 dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
3 plt.show()
```

This gives us a much clearer idea of the distribution of the input attributes:

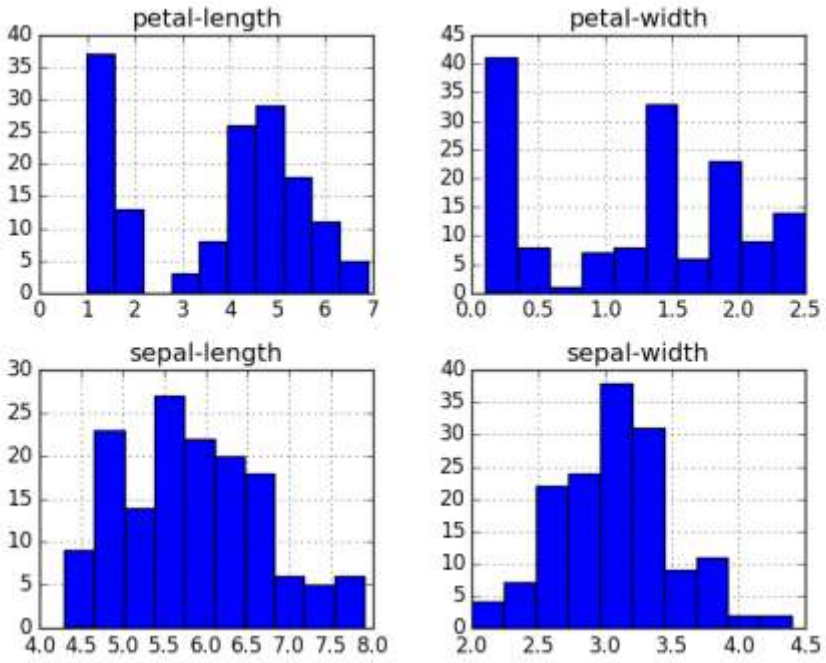


Box and Whisker Plots

We can also create a histogram of each input variable to get an idea of the distribution.

```
1 # histograms
2 dataset.hist()
3 plt.show()
```

It looks like perhaps two of the input variables have a Gaussian distribution. This is useful to note as we can use algorithms that can exploit this assumption.



Histogram Plots

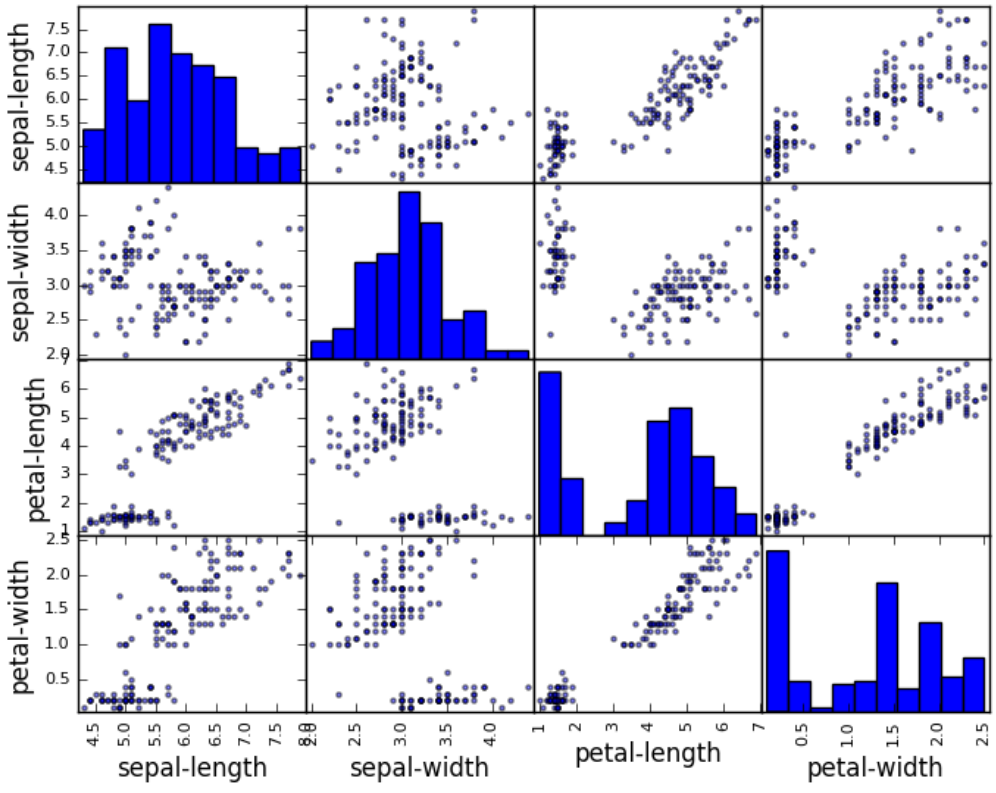
4.2 Multivariate Plots

Now we can look at the interactions between the variables.

First, let's look at scatterplots of all pairs of attributes. This can be helpful to spot structured relationships between input variables.

```
1 # scatter plot matrix
2 scatter_matrix(dataset)
3 plt.show()
```

Note the diagonal grouping of some pairs of attributes. This suggests a high correlation and a predictable relationship.



Scatterplot Matrix

5. Evaluate Some Algorithms

Now it is time to create some models of the data and estimate their accuracy on unseen data. Here is what we are going to cover in this step:

Separate out a validation dataset.

- Set-up the test harness to use 10-fold cross validation.
- Build 5 different models to predict species from flower measurements
- Select the best model.

5.1 Create a Validation Dataset

We need to know that the model we created is any good.

Later, we will use statistical methods to estimate the accuracy of the models that we create on unseen data. We also want a more concrete estimate of the accuracy of the best model on unseen data by evaluating it on actual unseen data.

That is, we are going to hold back some data that the algorithms will not get to see and we will use this data to get a second and independent idea of how accurate the best model might actually be.

We will split the loaded dataset into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset.

```
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
test_size=validation_size, random_state=seed)
```


You now have training data in the X_train and Y_train for preparing models and a X_validation and Y_validation sets that we can use later.

5.2 Test Harness

We will use 10-fold cross validation to estimate accuracy.

This will split our dataset into 10 parts, train on 9 and test on 1 and repeat for all combinations of train-test splits.

```
# Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

The specific random seed does not matter, learn more about pseudorandom number generators here:

- Introduction to Random Number Generators for Machine Learning in Python
We are using the metric of 'accuracy' to evaluate models. This is a ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate). We will be using the scoring variable when we run build and evaluate each model next.

5.3 Build Models

We don't know which algorithms would be good on this problem or what configurations to use. We get an idea from the plots that some of the classes are partially linearly separable in some dimensions, so we are expecting generally good results.

Let's evaluate 6 different algorithms:

- Logistic Regression (LR)
- Linear Discriminant Analysis (LDA)
- K-Nearest Neighbors (KNN).
- Classification and Regression Trees (CART).
- Gaussian Naive Bayes (NB).
- Support Vector Machines (SVM).

This is a good mixture of simple linear (LR and LDA), nonlinear (KNN, CART, NB and SVM) algorithms. We reset the random number seed before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.

Let's build and evaluate our five models:

```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=seed)
        cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold,
scoring=scoring)
```

```
16          results.append(cv_results)
17          names.append(name)
18          msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
            print(msg)
```

5.4 Select Best Model

We now have 6 models and accuracy estimations for each. We need to compare the models to each other and select the most accurate.

Running the example above, we get the following raw results:

```
1 LR: 0.966667 (0.040825)
2 LDA: 0.975000 (0.038188)
3 KNN: 0.983333 (0.033333)
4 CART: 0.975000 (0.038188)
5 NB: 0.975000 (0.053359)
6 SVM: 0.981667 (0.025000)
```

Note, you're results may differ. For more on this see the post:

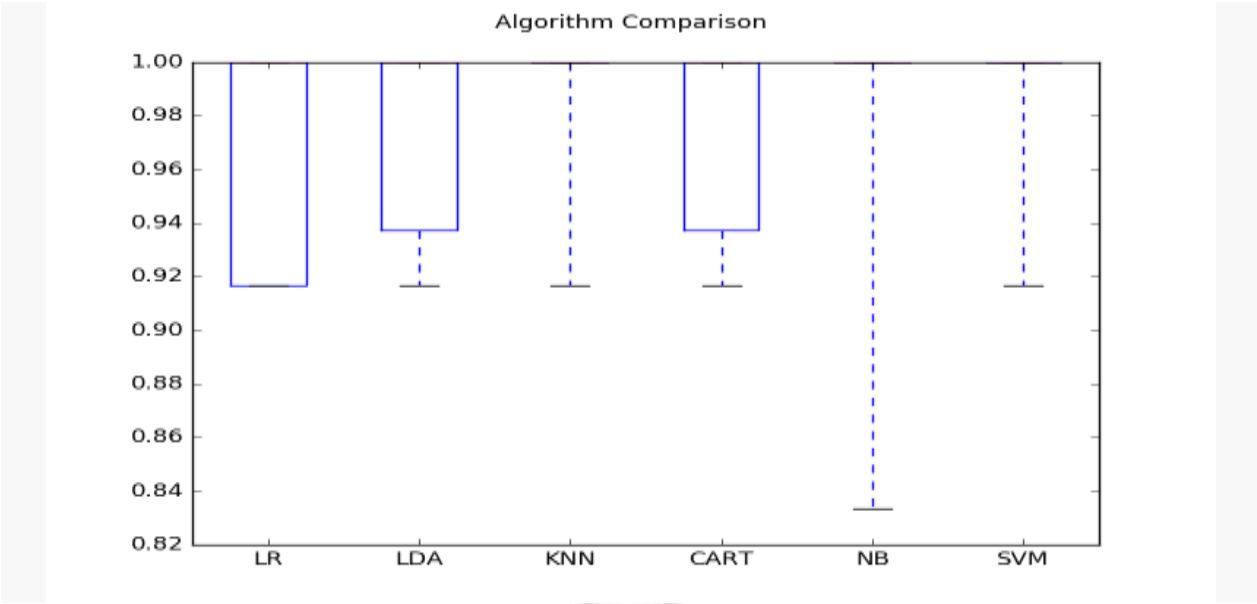
- Embrace Randomness in Machine Learning

We can see that it looks like KNN has the largest estimated accuracy score.

We can also create a plot of the model evaluation results and compare the spread and the mean accuracy of each model. There is a population of accuracy measures for each algorithm because each algorithm was evaluated 10 times (10 fold cross validation).

```
1 # Compare Algorithms
2 fig = plt.figure()
3 fig.suptitle('Algorithm Comparison')
4 ax = fig.add_subplot(111)
5 plt.boxplot(results)
6 ax.set_xticklabels(names)
7 plt.show()
```

You can see that the box and whisker plots are squashed at the top of the range, with many samples achieving 100% accuracy.



Compare Algorithm Accuracy

6. Make Predictions

The KNN algorithm was the most accurate model that we tested. Now we want to get an idea of the accuracy of the model on our validation set.

This will give us an independent final check on the accuracy of the best model. It is valuable to keep a validation set just in case you made a slip during training, such as overfitting to the training set or a data leak. Both will result in an overly optimistic result.

We can run the KNN model directly on the validation set and summarize the results as a final accuracy score, a confusion matrix and a classification report.

```
1 # Make predictions on validation dataset
2 knn = KNeighborsClassifier()
3 knn.fit(X_train, Y_train)
4 predictions = knn.predict(X_validation)
5 print(accuracy_score(Y_validation, predictions))
6 print(confusion_matrix(Y_validation, predictions))
7 print(classification_report(Y_validation, predictions))
```

We can see that the accuracy is 0.9 or 90%. The confusion matrix provides an indication of the three errors made. Finally, the classification report provides a breakdown of each class by precision, recall, f1-score and support showing excellent results (granted the validation dataset was small).

```
1  0.9
2
```

```
 3  [[ 7  0  0]
 4   [ 0 11  1]
 5   [ 0  2  9]]
 6
 7              precision    recall  f1-score   support
 8
 9  Iris-setosa       1.00      1.00      1.00         7
10 Iris-versicolor   0.85      0.92      0.88        12
11 Iris-virginica    0.90      0.82      0.86        11
12
13 avg / total       0.90      0.90      0.90        30
```

1. Business Understanding

Goal

The goals of this stage are:

- Clearly and explicitly specify the model target(s) as 'sharp' question(s) which is used to drive the customer engagement.
- Clearly specifying where to find the data sources of interest. Define the predictive model target in this step and determine if we need to bring in ancillary data from other sources.

How to do it

In this stage, you work with your customer and stakeholder to understand the business problems that can be greatly enhanced with predictive analytics. A central objective of this step is to identify the key business variables (sales forecast or the probability of an order being fraudulent, for example) that the analysis needs to predict (also known as model targets) to satisfy these requirements. In this stage you also to develop an understanding of the data sources needed to address the objectives of the project from an analytical perspective. There are two main aspects of this stage - Define Objectives and Identify data sources.

1.1 Define Objectives

1. Understand the customer business domain, key variables by which success is defined in that space. Then understand what business problems are we trying to address using data science to affect those key metrics?
2. Define the project goals with 'sharp' question(s). A fine description of what a sharp question is, and how you can ask it, can be found in this article. As per the article, here is a very useful tip to ask a sharp question - "When choosing your question,

imagine that you are approaching an oracle that can tell you anything in the universe, as long as the answer is a number or a name". Data science / machine learning is typically used to answer these five types of questions:

- How much or how many? (regression)
- Which category? (classification)
- Which group? (clustering)
- Is this weird? (anomaly detection)
- Which option should be taken? (recommendation)

3. Define the project team, the role and responsibilities. Develop a high level milestone plan that you iterate upon as more information is discovered.
4. Define success metrics. The metrics must be SMART (Specific, Measurable, Achievable, Relevant and Time-bound). For example: Achieve customer churn prediction accuracy of X% by the end of this 3-month project so that we can offer promotions to reduce churn.

1.2 Identify Data Sources

Identify data sources that contain known examples of answers to the sharp questions. Look for the following:

- Data that is **Relevant** to the question. Do we have measures of the target and features that are related to the target?
- Data that is an **Accurate** measure of our model target and the features of interest.

It is not uncommon, for example, to find that existing systems need to collect and log additional kinds of data to address the problem and achieve the project goals. In this case, you may want to look for external data sources or update your systems to collect newer data.

Artifacts

The following are the deliverables in this stage.

- **Charter Document** : A standard template is provided in the TDSP project structure definition. This is a living document that is updated throughout the project as new discoveries are made and as business requirements change. The key is to iterate upon this document with finer details as you progress through the discovery process. Be sure to keep the customer and stakeholders involved in the changes and clearly communicate the reasons for the change.

- **Data Sources**: This is part of the Data Report that is found in the TDSP project structure. It describes the sources for the raw data. In later stages you will fill in additional details like scripts to move the data to your analytic ernvironment.
- **Data Dictionaries** : This document provides the descriptions and the schema (data types, information on any validation rules) of the data which will be used to answer the question. If available, the entity-relation diagrams or descriptions are included too.

2. Data Acquisition and Understanding

Goal

The goals for this stage are:

- Ingest the data into the target analytic environment
- To determine if the data we have can be used to answer the question.

How to do it

In this stage, you will start developing the process to move the data from the source location to the target locations where the analytics operations like training and predictions (also known as scoring) will be run. For technical details and options on how to do this on various Azure data services, see Load data into storage environments for analytics.

Before you train your models, you need to develop a deep understanding about the data. Real world data is often messy with incomplete or incorrect data. By data summarization and visualization of the data, you can quickly identify the quality of your data and inform how to deal with the data quality. For guidance on cleaning the data, see this article.

Data visualization can be particularly useful to answer questions like - Have we measured the features consistently enough for them to be useful or are there a lot of missing values in the data? Has the data been consistently collected over the time period of interest or are there blocks of missing observations? If the data does not pass this quality check, we may need to go back to the previous step to correct or get more data.

Otherwise, you can start to better understand the inherent patterns in the data that will help you develop a sound predictive model for your target. Specifically you look for evidence for how well connected is the data to the target and whether the data is large enough to move forward with next steps. As we determine if the data is connected or if we have enough data, we may need to find new data sources with more accurate or more relevant data to complete the data set initially identified in the previous stage. TDSP also provides automated utility called IDEAR to help visualize the data and prepare data summary reports. We recommend starting with IDEAR first to explore the data to help develop initial data understanding interactively with no coding and then write custom code for data exploration and visualization.

In addition to the initial ingestion of data, you will typically need to setup a process to score new data or refresh the data regularly as part of an ongoing learning process. This can be done by setting up a data pipeline or workflow. Here is an example of how to setup a pipeline with Azure Data Factory. A solution architecture of the data pipeline is developed in this stage. The pipeline is developed in parallel in the following stages of the data science project. The pipeline may be batch based or a streaming/real-time or a hybrid depending on your business need and the constraints of your existing systems into which this solution is being integrated.

Artifacts

The following are the deliverables in this stage.

- **Data Quality Report** : This report contains data summaries, relationships between each attribute and target, variable ranking etc. The IDEAR tool provided as part of TDSP can help with the quickly generating this report on any tabular dataset like a CSV or relational table.
- Solution Architecture: This can be a diagram and/or description of your data pipeline used to run scoring or predictions on new data once you have built a model. It will also contain the pipeline to retrain your model based on new data. The document is stored in this directory when using the TDSP directory structure template.

Checkpoint Decision: Before we begin to do the full feature engineering and model building process, we can reevaluate the project to determine value in continuing this effort. We may be ready to proceed, need to collect more data, or it's possible the data does not exist to answer the question.

3. Modeling

Goal

The goals for this stage are:

- Develop new attributes or data features (also known as feature engineering), for building the machine learning model.
- Construct and evaluate an informative model to predict the target.
- Determine if we have a model that is suitable for production use

How to do it

There are two main aspects in this stage - Feature Engineering and Model training. They are described in following sub-sections.

3.1 Feature Engineering

Feature engineering involves inclusion, aggregation and transformation of raw variables to create the features used in the analysis. If we want insight into what is driving the model, then we need to understand how features are related to each other, and how the machine learning method will be using those features. This step requires a creative combination of domain expertise and the insights obtained from the data exploration step. This is a balancing act of including informative variables without including too many unrelated variables. Informative variables will improve our result; unrelated variables will introduce unnecessary noise into the model. You will also need to be able to generate these features for new data during scoring. So there should not be any dependency on generating these features on any piece of data that is unavailable at the time of scoring. For technical guidance on feature engineering when using various Azure data technologies, see this article.

3.2 Model Training

Depending on type of question you are trying answer, there are multiple modeling algorithms options available. For guidance on choosing the algorithms, see this article. NOTE: Though this article is written for Azure Machine Learning, it should be generally useful even when using other frameworks.

The process for model training is:

- The input data for modeling is usually split randomly into a training data set and a test data set.
- The models are built using the training data set.
- Evaluate (training and test dataset) a series of competing machine learning algorithms along with the various associated tuning parameters (also known as parameter sweep) that are geared toward answering the question of interest with the data we currently have at hand.
- Determine the "best" solution to answer the question by comparing the success metric between alternative methods.

[NOTE] Avoid leakage: Leakage is caused by including variables that can perfectly predict the target. These are usually variables that may have been used to detect the target initially. As the target is redefined, these dependencies can be hidden from the original definition. To avoid this often requires iterating between building an analysis data set, and creating a model and evaluating the accuracy. Leakage is a major reason data scientists get nervous when they get really good predictive results.

We provide a Automated Modeling and Reporting tool with TDSP that is able to run through multiple algorithms and parameter sweeps to produce a baseline model. It will also produce a baseline modeling report summarizing performance of each model and

parameter combination including variable importance. This can further drive further feature engineering.

Artifacts

The artifacts produced in this stage includes:

- **Feature Sets**: The features developed for the modeling are described in the in the Feature Set section of the Data Definition report. It contains pointers to the code to generate the features and description on how the feature was generated.
- **Modeling Report**: For each models that are tried, a standard report following a specified TDSP template is produced. The

Checkpoint Decision: We evaluate whether the model performing is acceptable enough to deploy it to a production system here. Some of the questions to ask include:

- Does the model answer the question sufficiently given the test data?
- Should we go back and collect more data or do more feature engineering or try other algorithms?

4. Deployment

Goal

Deploy models and pipeline to a production or production-like environment for final user acceptance.

How to do it

Once we have a set of models that perform well, they can be operationalized for other applications to consume. Depending on the business requirements, predictions are made either in real time or on a batch basis. To be operationalized, the models have to be exposed with an open API interface that is easily consumed from various applications such online website, spreadsheets, dashboards, or line of business and backend applications. See example of model operationalization with Azure Machine Learning web service in this article. It is also a good idea to build in telemetry and monitoring of the production model deployment and the data pipeline to help with system status reporting and troubleshooting.

Artifacts

- Status dashboard of system health and key metrics
- Final modeling report with deployment details
- Final solution architecture document

5. Customer Acceptance

Goal

To finalize the project deliverables by confirming the pipeline, the model, and their deployment in a production environment.

How to do it

The customer would validate that the system meet their business need and the answers the questions with acceptable accuracy to deploy the system to production for use by their client application. All the documentation are finalized and reviewed. A handoff of the project to the entity responsible for operations is done. Thbis could be an IT or data science team at the customer or an agent of the customer that will be responsible for running the system in production.

Artifacts

The main artifact produced in this final stage is the **Project Final Report**. This is the project technical report containing all details of the project that useful to learn and operate the system. A template is provided by TDSP that can be used as is or customized for specific client needs.

Summary

We have seen the Team Data Science Process lifecycle which is modeled as a sequence of iterated steps that provide guidance on the tasks needed to use predictive models that can be deployed in a production environment to be leveraged to build intelligent applications. The goal of this process lifecycle is to continue to move a data science project forward towards a clear engagement end point. While it is true that data science is an exercise in research and discovery, being able to clearly communicate this to customers using a well defined set of artifacts in a standardized template can help avoid misunderstanding and increase the odds of success.

Gathering Data

Once we have our equipment and booze, it's time for our first real step of machine learning: **gathering data**. This step is very important because the quality and quantity of data that you gather will directly determine how good your predictive model can be. In this case, the data we collect will be the color and the alcohol content of each drink.

Color (nm)	Alcohol %	Beer or Wine?
610	5	Beer
599	13	Wine
693	14	Wine

This will yield a table of color, alcohol%, and whether it's beer or wine. This will be our **training data.**

Data preparation

A few hours of measurements later, we have gathered our training data. Now it's time for the next step of machine learning: **Data preparation**, where we load our data into a suitable place and prepare it for use in our machine learning training.

We'll first put all our data together, and then randomize the ordering. We don't want the order of our data to affect what we learn, since that's not part of determining whether a drink is beer or wine. In other words, we make a determination of what a drink is, independent of what drink came before or after it.

We'll also need to split the data in two parts. The first part, used in training our model, will be the majority of the dataset. The second part will be used for evaluating our trained model's performance. We don't want to use the same data that the model was trained on for evaluation, since it could then just memorize the "questions", just as you wouldn't use the same questions from your math homework on the exam.

Sometimes the data we collect needs other forms of adjusting and manipulation. Things like de-duping, normalization, error correction, and more. These would all happen at the data preparation step. In our case, we don't have any further data preparation needs, so let's move forward.

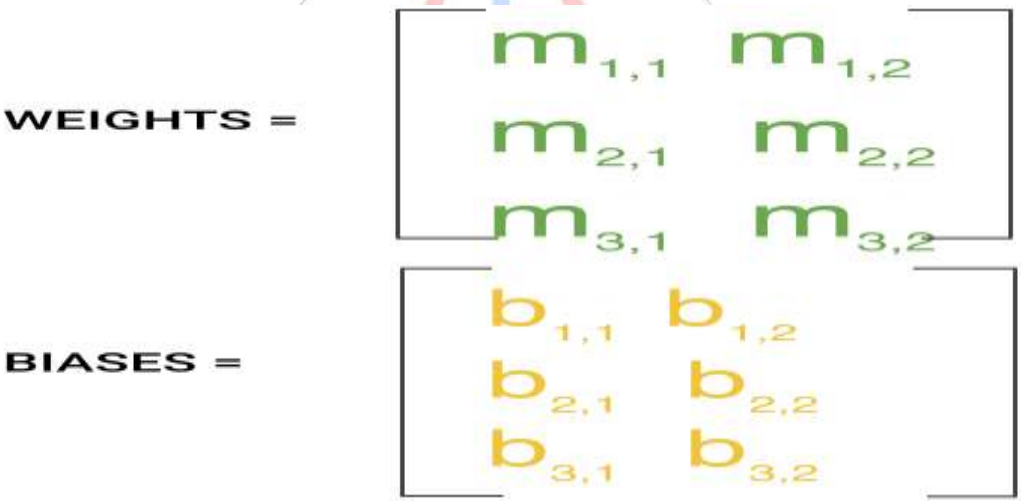
Choosing a model

The next step in our workflow is choosing a model. There are many models that researchers and data scientists have created over the years. Some are very well suited for image data, others for sequences (like text, or music), some for numerical data, others for text-based data. In our case, since we only have 2 features, color and alcohol%, we can use a small linear model, which is a fairly simple one that should get the job done.

Training

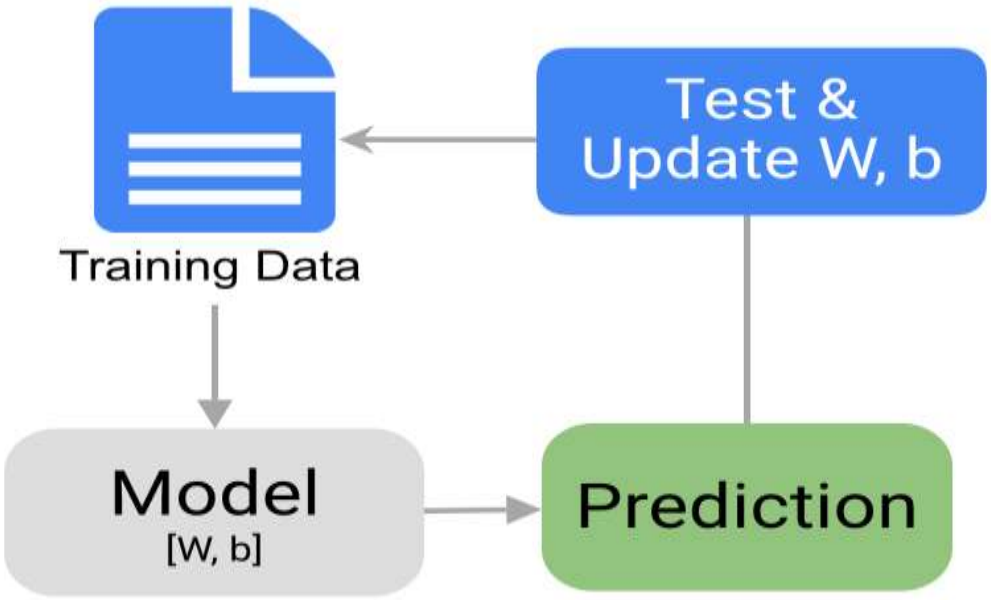
Now we move onto what is often considered the bulk of machine learning—the **training**. In this step, we will use our data to incrementally improve our model's ability to predict whether a given drink is wine or beer.

We will do this on a much smaller scale with our drinks. In particular, the formula for a straight line is y=m*x+b, where x is the input, m is the slope of that line, b is the y-intercept, and y is the value of the line at the position x. The values we have available to us for adjusting, or "training", are m and b. There is no other way to affect the position of the line, since the only other variables are x, our input, and y, our output.



In machine learning, there are many m's since there may be many features. The collection of these m values is usually formed into a matrix, that we will denote W, for the "weights" matrix. Similarly for b, we arrange them together and call that the biases.

The training process involves initializing some random values for W and b and attempting to predict the output with those values. As you might imagine, it does pretty poorly. But we can compare our model's predictions with the output that it should produced, and adjust the values in W and b such that we will have more correct predictions.



This process then repeats. Each iteration or cycle of updating the weights and biases is called one training "step".

Let's look at what that means in this case, more concretely, for our dataset. When we first start the training, it's like we drew a random line through the data. Then as each step of the training progresses, the line moves, step by step, closer to an ideal separation of the wine and beer.

Evaluation

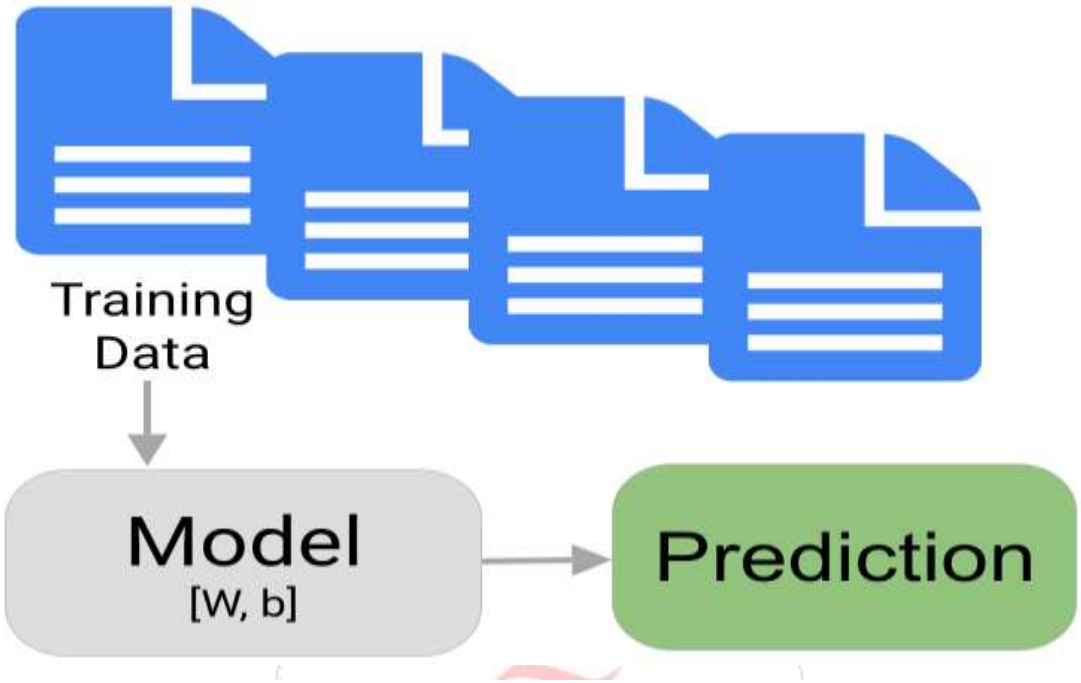
Once training is complete, it's time to see if the model is any good, using **Evaluation**. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

A good rule of thumb I use for a training-evaluation split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don't need as big of a fraction for the evaluation dataset.

Parameter Tuning

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by **tuning our parameters**. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

One example is how many times we run through the training dataset during training. What I mean by that is we can "show" the model our full dataset multiple times, rather than just once. This can sometimes lead to higher accuracies.



Supervised Machine Learning Algorithms

Predictive models:

Regression models:

Linear Regression Machine Learning Algorithm

Linear Regression algorithm shows the relationship between 2 variables and how the change in one variable impacts the other. The algorithm shows the impact on the dependent variable on changing the independent variable. The independent variables are referred as explanatory variables, as they explain the factors the impact the dependent variable. Dependent variable is often referred to as the factor of interest or predictor.

Advantages of Linear Regression Machine Learning Algorithm

- It is one of the most interpretable machine learning algorithms, making it easy to explain to others.
- It is easy of use as it requires minimal tuning.
- It is the mostly widely used machine learning technique that runs fast.

Applications of Linear Regression

- **Estimating Sales**

  Linear Regression finds great use in business, for sales forecasting based on the trends. If a company observes steady increase in sales every month - a linear regression analysis of the monthly sales data helps the company forecast sales in upcoming months.

  Access the Solution to Kaggle Data Science Challenge -Walmart Store Sales Forecasting

- **Risk Assessment**

  Linear Regression helps assess risk involved in insurance or financial domain. A health insurance company can do a linear regression analysis on the number of claims per customer against age. This analysis helps insurance companies find, that older customers tend to make more insurance claims. Such analysis results play a vital role in important business decisions and are made to account for risk.

  Data Science Libraries in Python to implement Linear Regression – statsmodel and SciKit

  Data Science Libraries in R to implement Linear Regression – stats

  Explanations about the top machine learning algorithms will continue, as it is a work in progress. Stay tuned to our blog to learn more about the popular machine learning algorithms and their applications!!!

  Linear regression requires a linear model. No surprise, right? But what does that really mean?

  A model is linear when each term is either a constant or the product of a parameter and a predictor variable. A linear equation is constructed by adding the results for each term. This constrains the equation to just one basic form:

Response = constant + parameter * predictor + ... + parameter * predictor

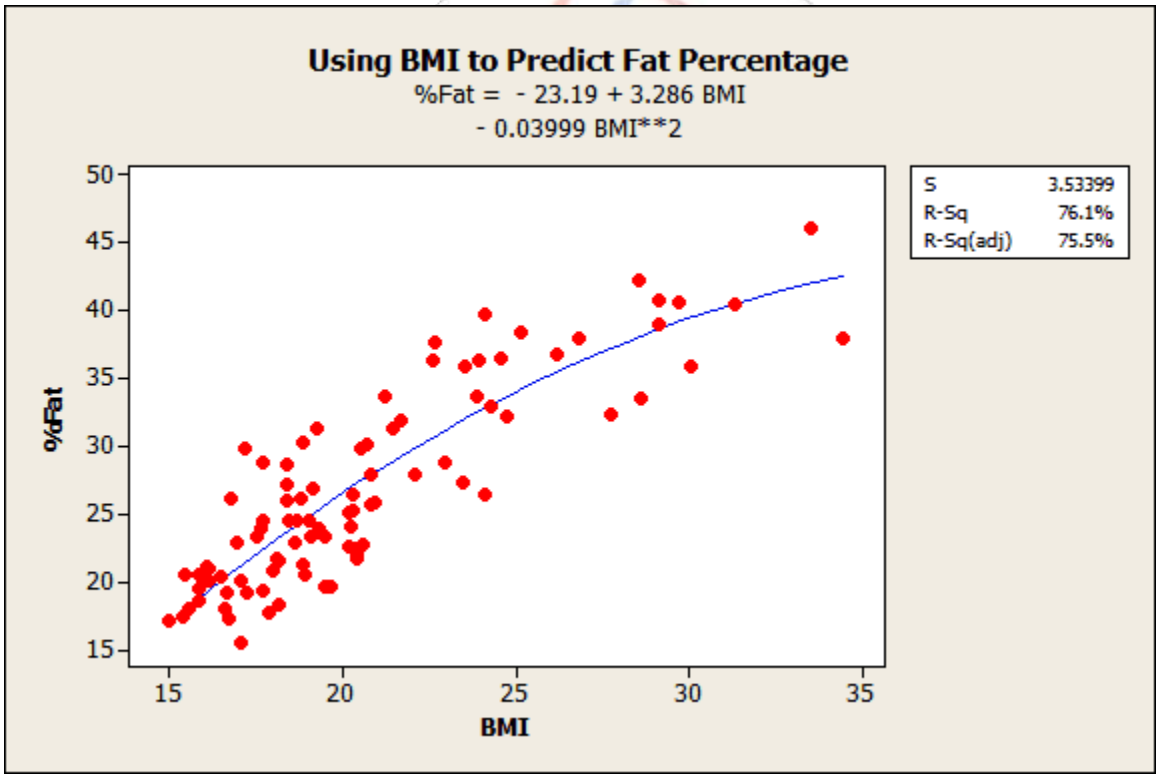
$Y = b_o + b_1X_1 + b_2X_2 + ... + b_kX_k$

In statistics, a regression equation (or function) is linear when it is linear in the parameters. While the equation must be linear in the parameters, you can transform the predictor variables in ways that produce curvature. For instance, you can include a squared variable to produce a U-shaped curve.

$Y = b_o + b_1X_1 + b_2X_1^2$

This model is still linear in the parameters even though the predictor variable is squared. You can also use log and inverse functional forms that are linear in the parameters to produce different types of curves.

Example :- of a linear regression model that uses a squared term to fit the curved relationship between BMI and body fat percentage.

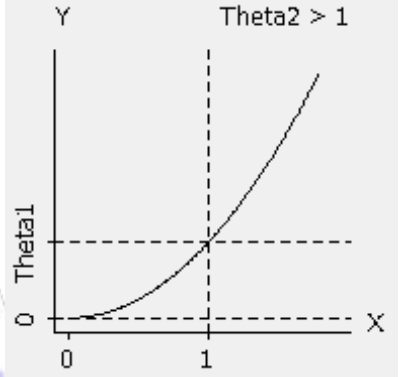
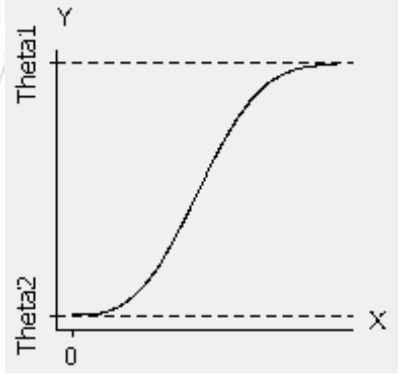
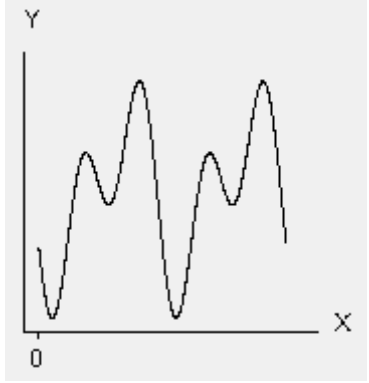


Non linear regression:

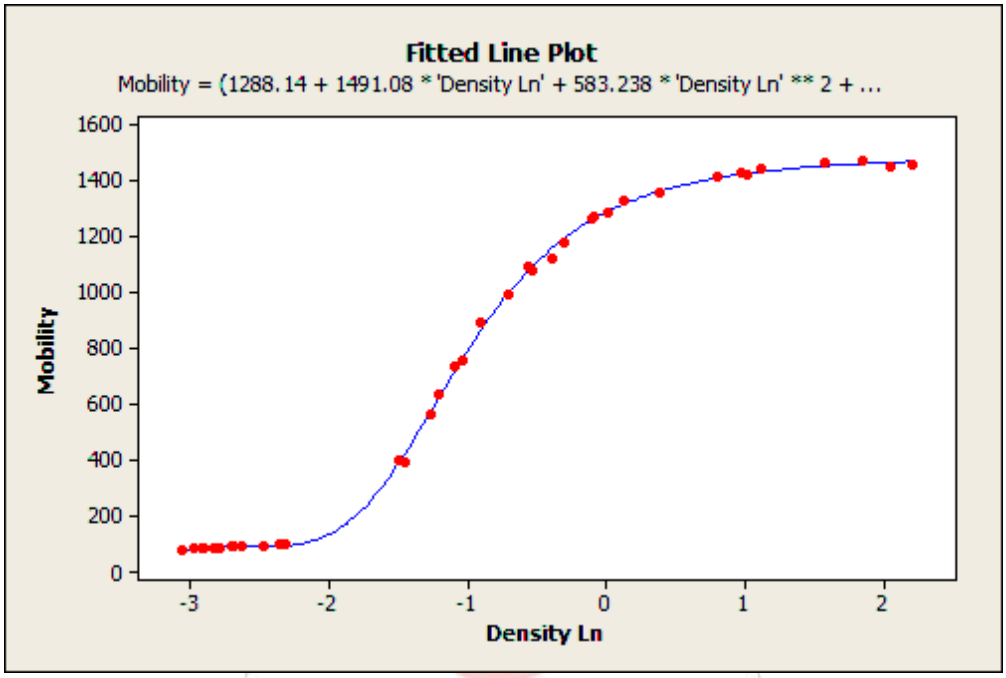
nonlinear regression is a form of regression analysis in which observational data are modeled by a function which is a nonlinear combination of the model parameters and depends on one or more independent variables. The data are fitted by a method of successive approximations.

While a linear equation has one basic form, nonlinear equations can take many different forms. The easiest way to determine whether an equation is nonlinear is to focus on the term "nonlinear" itself. Literally, it's not linear. If the equation doesn't meet the criteria above for a linear equation, it's nonlinear.

That covers many different forms, which is why nonlinear regression provides the most flexible curve-fitting functionality. Here are several examples from Minitab's nonlinear function catalog. Thetas represent the parameters and X represents the predictor in the nonlinear functions. Unlike linear regression, these functions can have more than one parameter per predictor variable.

Nonlinear function	**One possible shape**
Power (convex): Theta1 * X^Theta2	
Weibull growth: Theta1 + (Theta2 - Theta1) * exp(-Theta3 * X^Theta4)	
Fourier: Theta1 * cos(X + Theta4) + (Theta2 * cos(2*X + Theta4) + Theta3	

Here is an example of a nonlinear regression model of the relationship between density and electron mobility.



The nonlinear equation is so long it that it doesn't fit on the graph:

Mobility = (1288.14 + 1491.08 * Density Ln + 583.238 * Density Ln^2 + 75.4167 * Density Ln^3) / (1 + 0.966295 * Density Ln + 0.397973 * Density Ln^2 + 0.0497273 * Density Ln^3)

Linear and nonlinear regression are actually named after the functional form of the models that each analysis accepts. I hope the distinction between linear and nonlinear equations is clearer and that you understand how it's possible for linear regression to model curves! It also explains why you'll see R-squared displayed for some curvilinear models even though it's impossible to calculate R-squared for nonlinear regression.

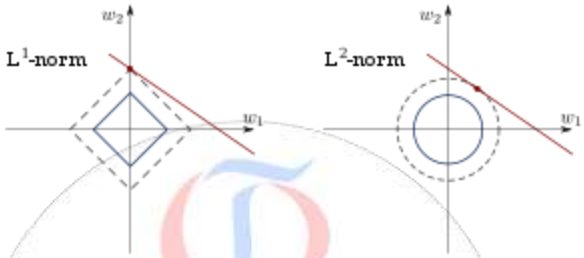
Lasso Regression:-

What is Lasso Regression?

Lasso regression is a type of **linear regression** that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of muticollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

The acronym "LASSO" stands for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator.

- method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.
- Lasso was originally formulated for least squares models and this simple case reveals a substantial amount about the behavior of the estimator, including its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates need not be unique if covariates are collinear.
- Lasso's ability to perform subset selection relies on the form of the constraint and has a variety of interpretations including in terms of geometry, Bayesian statistics, and convex analysis.



L1 Regularization

Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This type of regularization can result in sparse models with few coefficients; Some coefficients can become zero and eliminated from the model. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models. On the other hand, L2 regularization (e.g. Ridge regression) doesn't result in elimination of coefficients or sparse models. This makes the Lasso far easier to interpret than the Ridge.

Performing the Regression

Lasso solutions are quadratic programming problems, which are best solved with software (like Matlab). The goal of the algorithm is to minimize:

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Which is the same as minimizing the sum of squares with constraint $\Sigma |B_j \leq s$. Some of the βs are shrunk to exactly zero, resulting in a regression model that's easier to interpret.

A **tuning parameter**, λ controls the strength of the L1 penalty. λ is basically the amount of shrinkage:

- When λ = 0, no parameters are eliminated. The estimate is equal to the one found with linear regression.
- As λ increases, more and more coefficients are set to zero and eliminated (theoretically, when λ = ∞, all coefficients are eliminated).
- As λ increases, bias increases.
- As λ decreases, variance increases.

If an intercept is included in the model, it is usually left unchanged.

Ridge Regression:-

Ridge regression is a way to create a parsimonious model when the number of predictor variables in a set exceeds the number of observations, or when a data set has multicollinearity (correlations between predictor variables).

Tikhivov's method is basically the same as ridge regression, except that Tikhonov's has a larger set. It can produce solutions even when your data set contains a lot of statistical noise (unexplained variation in a sample).

Ridge regression avoids all of these problems. It works in part because it doesn't require unbiased estimators;

Ridge regression adds just enough bias to make the estimates reasonably reliable approximations to true population values.

Shrinkage

Ridge regression uses a type of shrinkage estimator called a ridge estimator. Shrinkage estimators theoretically produce new estimators that are shrunk closer to the "true" population parameters. The ridge estimator is especially good at improving the least-squares estimate when multicollinearity is present.

Regularization

Ridge regression belongs a class of regression tools that use L2 regularization. The other type of regularization, **L1 regularization**, limits the size of the coefficients by adding an L1 penalty equal to the absolute value of the magnitude of coefficients. This sometimes results in the elimination of some coefficients altogether, which can yield sparse models. **L2 regularization** adds an L2 penalty, which equals the square of the magnitude of coefficients. All coefficients are shrunk by the same factor (so none are eliminated). Unlike L1 regularization, L2 will not result in sparse models.

A tuning parameter (λ) controls the strength of the penalty term. When λ = 0, ridge regression equals least squares regression. If λ = ∞, all coefficients are shrunk to zero. The ideal penalty is therefore somewhere in between 0 and ∞.

On Mathematics

OLS regression uses the following formula to estimate coefficients:

$$\hat{\underline{B}} = (\underline{X}'\underline{X})^{-1}\underline{X}'\underline{Y}$$

If X is a centered and scaled matrix, the crossproduct matrix (X`X) is nearly singular when the X-columns are highly correlated. Ridge regression adds a ridge parameter (k), of the identity matrix to the cross product matrix, forming a new matrix (X`X + kI). It's called ridge regression because the diagonal of ones in the correlation matrix can be described as a ridge. The new formula is used to find the coefficients:

$$\tilde{\underline{B}} = (\underline{X}'\underline{X} + k\underline{I})^{-1}\underline{X}'\underline{Y}$$

Choosing a value for k is not a simple task, which is perhaps one major reason why ridge regression isn't used as much as least squares or logistic regression. You can read one way to find k in Dorugade and D. N. Kashid's paper Alternative Method for Choosing Ridge Parameter for Regression..

Note:-

- It shrinks the parameters, therefore it is mostly used to prevent multicollinearity.
- It reduces the model complexity by coefficient shrinkage.
- It uses L2 regularization technique.

The result is the ridge regression estimator

$\hat{}\beta ridge = (X'X + \lambda Ip) - 1X'Y$

Ridge regression places a particular form of constraint on the parameters (β

's): $\hat{}\beta ridge$

is chosen to minimize the penalized sum of squares:

$n\sum i=1(yi - p\sum j=1xij\beta j)2 + \lambda p\sum j=1\beta 2j$

which is equivalent to minimization of $\sum ni=1(yi - \sum pj=1xij\beta j)2$

subject to, for some $c>0$, $\sum pj=1\beta 2j<c$

, i.e. constraining the sum of the squared coefficients.

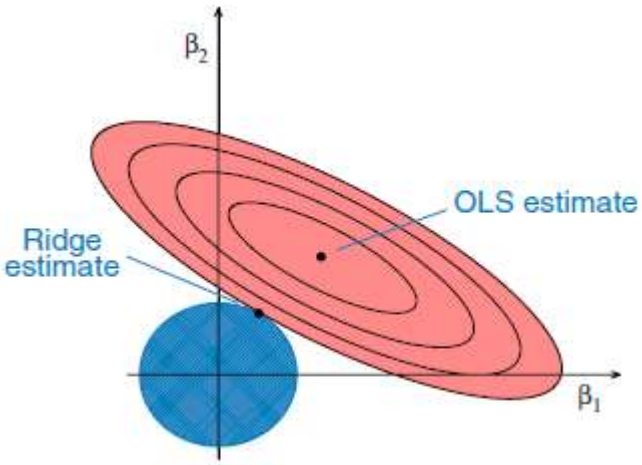
Therefore, ridge regression puts further constraints on the parameters, βj

's, in the linear model. In this case, what we are doing is that instead of just minimizing the residual sum of squares we also have a penalty term on the β's. This penalty term is λ (a pre-chosen constant) times the squared norm of the β vector. This means that if the βj's

take on large values, the optimization function is penalized. We would prefer to take smaller βj's, or βj

's that are close to zero to drive the penalty term small.

Geometric Interpretation of Ridge Regression:



The ellipses correspond to the contours of residual sum of squares (RSS): the inner ellipse has smaller RSS, and RSS is minimized at ordinal least square (OLS) estimates.

For p=2

, the constraint in ridge regression corresponds to a circle, ∑pj=1β2j<c

.

We are trying to minimize the ellipse size and circle simultanously in the ridge regression. The ridge estimate is given by the point at which the ellipse and the circle touch.

Gradient descent algorithm:-

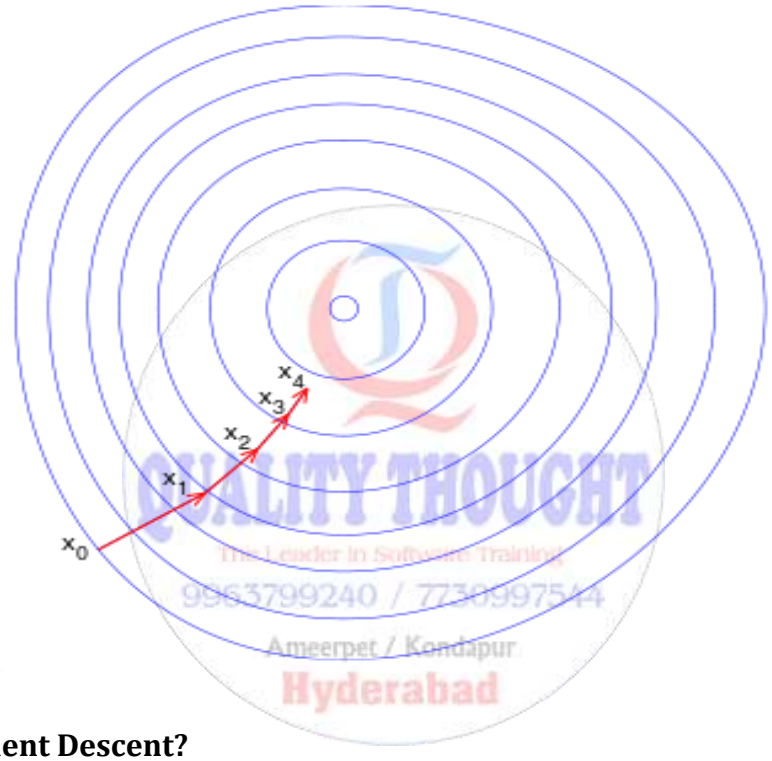
Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as **gradient ascent**.

Gradient descent is also known as **steepest descent**. However, gradient descent should not be confused with the method of steepest descent for approximating integrals.

Gradient Descent

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

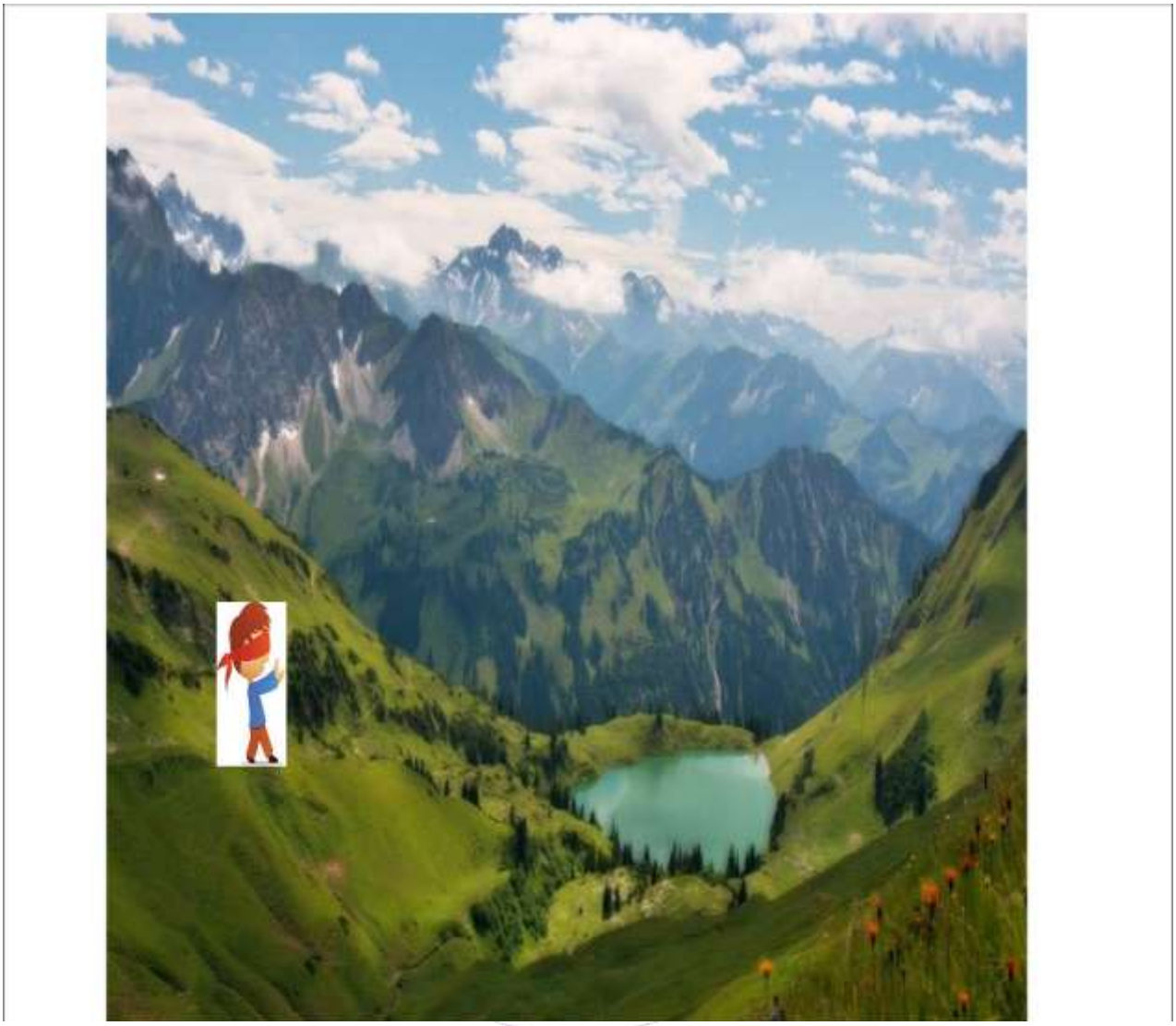
Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.



1. What is Gradient Descent?

To explain Gradient Descent I'll use the classic mountaineering example.

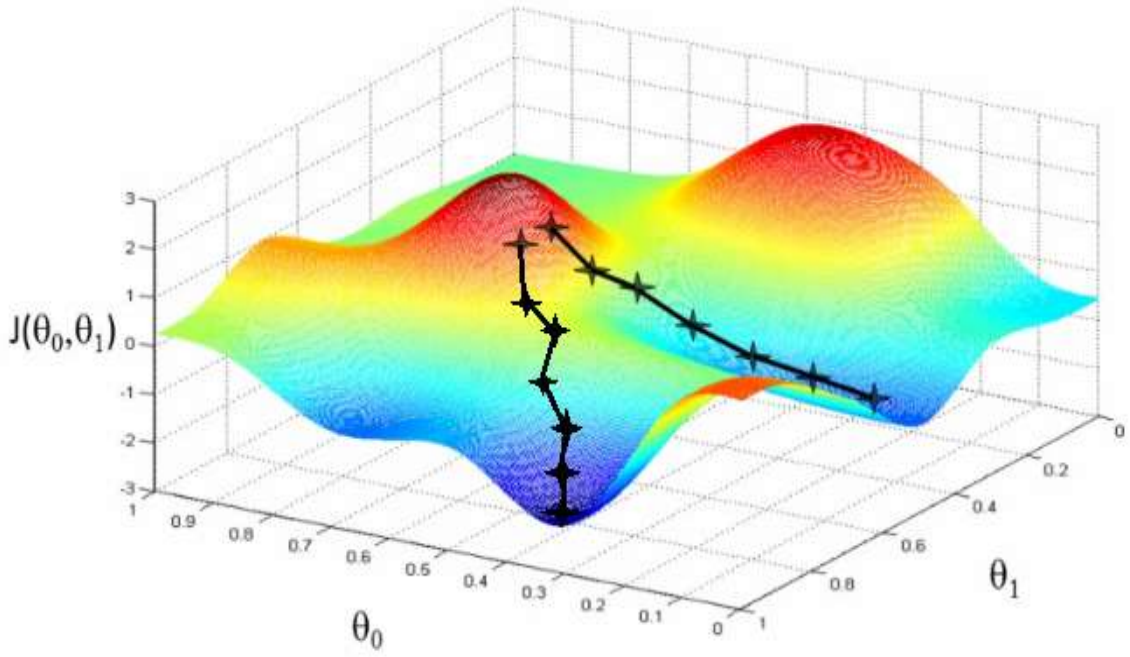
Suppose you are at the top of a mountain, and you have to reach a lake which is at the lowest point of the mountain (a.k.a valley). A twist is that you are blindfolded and you have zero visibility to see where you are headed. So, what approach will you take to reach the lake?



Source

The best way is to check the ground near you and observe where the land tends to descend. This will give an idea in what direction you should take your first step. If you follow the descending path, it is very likely you would reach the lake.

To represent this graphically, notice the below graph.



Source

Let us now map this scenario in mathematical terms.

Suppose we want to find out the best parameters (θ1) and (θ2) for our learning algorithm. Similar to the analogy above, we see we find similar mountains and valleys when we plot our "cost space". Cost space is nothing but how our algorithm would perform when we choose a particular value for a parameter.

So on the y-axis, we have the cost J(θ) against our parameters θ1 and θ2 on x-axis and z-axis respectively. Here, hills are represented by red region, which have high cost, and valleys are represented by blue region, which have low cost.

Now there are many types of gradient descent algorithms. They can be classified by two methods mainly:

- **On the basis of data ingestion**
    1. Full Batch Gradient Descent Algorithm
    2. Stochastic Gradient Descent Algorithm

In full batch gradient descent algorithms, you use whole data at once to compute the gradient, whereas in stochastic you take a sample while computing the gradient.

- **On the basis of differentiation techniques**
    1. First order Differentiation
    2. Second order Differentiation

Gradient descent requires calculation of gradient by differentiation of cost function. We can either use first order differentiation or second order differentiation.

When applying gradient descent, you can look at these points which might be helpful in circumventing the problem:

- **Error rates** – You should check the training and testing error after specific iterations and make sure both of them decreases. If that is not the case, there might be a problem!
- **Gradient flow in hidden layers –** Check if the network doesn't show a vanishing gradient problem or exploding gradient problem.
- **Learning rate** – which you should check when using adaptive techniques.

gradient descent algorithm and its variants: Batch Gradient Descent, Mini-batch Gradient Descent, and Stochastic Gradient Descent.

Batch Gradient Descent

Batch Gradient Descent, also called vanilla gradient descent, calculates the error for each example within the training dataset, but only after all training examples have been evaluated, the model gets updated. This whole process is like a cycle and called a training epoch.

Advantages of it are that it's computational efficient, it produces a stable error gradient and a stable convergence. Batch Gradient Descent has the disadvantage that the stable error gradient can sometimes result in a state of convergence that isn't the best the model can achieve. It also requires that the entire training dataset is in memory and available to the algorithm.

Stochastic Gradient Descent

Stochastic gradient descent (SGD) in contrary, does this for each training example within the dataset. This means that it updates the parameters for each training example, one by one. This can make SGD faster than Batch Gradient Descent, depending on the problem. One advantage is that the frequent updates allow us to have a pretty detailed rate of improvement.

The thing is that the frequent updates are more computationally expensive as the approach of Batch Gradient Descent. The frequency of those updates can also result in noisy gradients, which may cause the error rate to jump around, instead of slowly decreasing.

Mini Batch Gradient Descent

Mini-batch Gradient Descent is the go-to method since it's a combination of the concepts of SGD and Batch Gradient Descent. It simply splits the training dataset into small batches and performs an update for each of these batches. Therefore it creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent.

Common mini-batch sizes range between 50 and 256, but like for any other machine learning techniques, there is no clear rule, because they can vary for different applications. Note that it is the go-to algorithm when you are training a neural network and it is the most common type of gradient descent within deep learning.

Classification models

- Logistic Regression:-

Logistic Regression

The name of this algorithm could be a little confusing in the sense that Logistic Regression machine learning algorithm is for classification tasks and not regression problems. The name 'Regression' here implies that a linear model is fit into the feature space. This algorithm applies a logistic function to a linear combination of features to predict the outcome of a categorical dependent variable based on predictor variables.

The odds or probabilities that describe the outcome of a single trial are modelled as a function of explanatory variables. Logistic regression algorithms helps estimate the probability of falling into a specific level of the categorical dependent variable based on the given predictor variables.

Just suppose that you want to predict if there will be a snowfall tomorrow in New York. Here the outcome of the prediction is not a continuous number because there will either be snowfall or no snowfall and hence linear regression cannot be applied. Here the outcome variable is one of the several categories and using logistic regression helps.

Based on the nature of categorical response, logistic regression is classified into 3 types –

· **Binary Logistic Regression** – The most commonly used logistic regression when the categorical response has 2 possible outcomes i.e. either yes or not. Example –Predicting whether a student will pass or fail an exam, predicting whether a student will have low or high blood pressure, predicting whether a tumour is cancerous or not.

· **Multi-nominal Logistic Regression** - Categorical response has 3 or more possible outcomes with no ordering. Example- Predicting what kind of search engine (Yahoo, Bing, Google, and MSN) is used by majority of US citizens.

· **Ordinal Logistic Regression** - Categorical response has 3 or more possible outcomes with natural ordering. Example- How a customer rates the service and quality of food at a restaurant based on a scale of 1 to 10.

Let us consider a simple example where a cake manufacturer wants to find out if baking a cake at 160°C, 180°C and 200°C will produce a 'hard' or 'soft' variety of cake (assuming the fact that the bakery sells both the varieties of cake with different names and prices).

Logistic regression is a perfect fit in this scenario instead of other statistical techniques. For example, if the manufactures produces 2 cake batches wherein the first batch contains 20 cakes (of which 7 were hard and 13 were soft) and the second batch of cake produced consisted of 80 cakes (of which 41 were hard and 39 were soft cakes). Here in this case if linear regression algorithm is used it will give equal importance both the batches of cakes regardless of the number of cakes in each batch. Applying a logistic regression algorithm will consider this factor and give the second batch of cakes more weightage than the first batch.

When to Use Logistic Regression Machine Learning Algorithm

- Use logistic regression algorithms when there is a requirement to model the probabilities of the response variable as a function of some other explanatory variable. For example, probability of buying a product X as a function of gender
- Use logistic regression algorithms when there is a need to predict probabilities that categorical dependent variable will fall into two categories of the binary response as a function of some explanatory variables. For example, what is the probability that a customer will buy a perfume given that the customer is a female?
- Logistic regression algorithms is also best suited when the need is to classify elements two categories based on the explanatory variable. For example-classify females into 'young' or 'old' group based on their age.

Advantages of Using Logistic Regression

- Easier to inspect and less complex.
- Robust algorithm as the independent variables need not have equal variance or normal distribution.
- These algorithms do not assume a linear relationship between the dependent and independent variables and hence can also handle non-linear effects.
- Controls confounding and tests interaction.

Drawbacks of Using Logistic Regression

- When the training data is sparse and high dimensional, in such situations a logistic model may over fit the training data.
- Logistic regression algorithms cannot predict continuous outcomes. For instance, logistic regression cannot be applied when the goal is to determine how heavily it will rain because the scale of measuring rainfall is continuous. Data scientists can predict heavy or low rainfall but this would make some compromises with the precision of the dataset.
- Logistic regression algorithms require more data to achieve stability and meaningful results. These algorithms require minimum of 50 data points per predictor to achieve stable outcomes.
- It predicts outcomes depending on a group of independent variables and if a data scientist or a machine learning expert goes wrong in identifying the independent variables then the developed model will have minimal or no predictive value.
- It is not robust to outliers and missing values.

Applications of Logistic Regression

- Logistic regression algorithm is applied in the field of epidemiology to identify risk factors for diseases and plan accordingly for preventive measures.
- Used to predict whether a candidate will win or lose a political election or to predict whether a voter will vote for a particular candidate.
- Used to classify a set of words as nouns, pronouns, verbs, adjectives.
- Used in weather forecasting to predict the probability of rain.
- Used in credit scoring systems for risk management to predict the defaulting of an account.

The Data Science libraries in Python language to implement Logistic Regression Machine Learning Algorithm is Sci-Kit Learn.

The Data Science libraries in R language to implement Logistic Regression Machine Learning Algorithm is stats package (glm () function)

Naïve Bayes Classifier Algorithm

It would be difficult and practically impossible to classify a web page, a document, an email or any other lengthy text notes manually. This is where Naïve Bayes Classifier machine learning algorithm comes to the rescue. A classifier is a function that allocates a population's element value from one of the available categories. For instance, Spam Filtering is a popular application of Naïve Bayes algorithm. Spam filter here, is a classifier that assigns a label "Spam" or "Not Spam" to all the emails.

Naïve Bayes Classifier is amongst the most popular learning method grouped by similarities, that works on the popular Bayes Theorem of Probability- to build machine learning models particularly for disease prediction and document classification. It is a simple classification of words based on Bayes Probability Theorem for subjective analysis of content.

When to use the Machine Learning algorithm - Naïve Bayes Classifier?

- If you have a moderate or large training data set.
- If the instances have several attributes.
- Given the classification parameter, attributes which describe the instances should be conditionally independent.

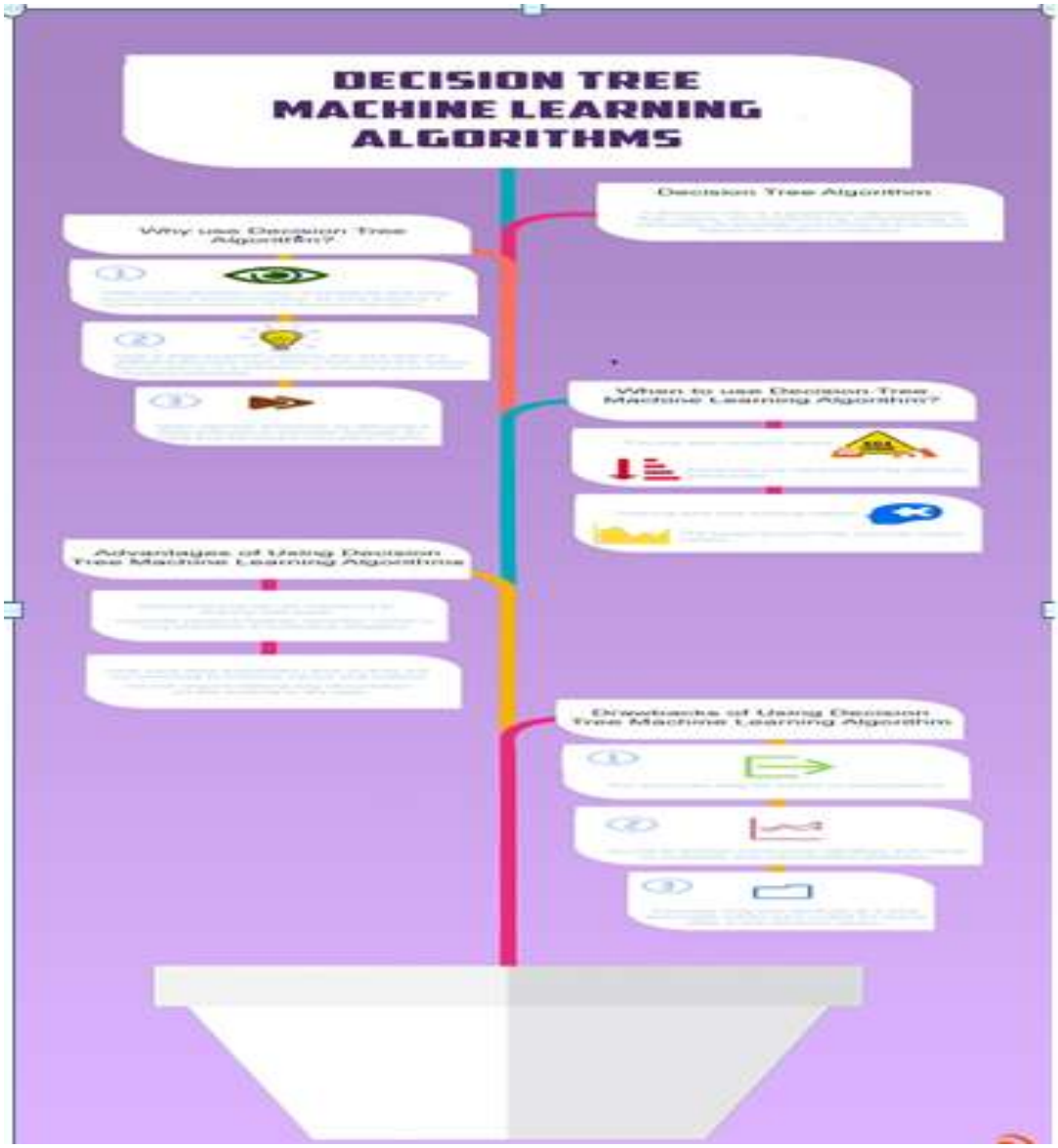
Applications of Naïve Bayes Classifier

1. **Sentiment Analysis**- It is used at Facebook to analyse status updates expressing positive or negative emotions.
2. **Document Categorization**- Google uses document classification to index documents and find relevancy scores i.e. the PageRank. PageRank mechanism considers the pages marked as important in the databases that were parsed and classified using a document classification technique.
3. Naïve Bayes Algorithm is also used for classifying news articles about Technology, Entertainment, Sports, Politics, etc.
4. **Email Spam Filtering**-Google Mail uses Naïve Bayes algorithm to classify your emails as Spam or Not Spam

Advantages of the Naïve Bayes Classifier Machine Learning Algorithm

1. Naïve Bayes Classifier algorithm performs well when the input variables are categorical.
2. A Naïve Bayes classifier converges faster, requiring relatively little training data than other discriminative models like logistic regression, when the Naïve Bayes conditional independence assumption holds.
3. With Naïve Bayes Classifier algorithm, it is easier to predict class of the test data set. A good bet for multi class predictions as well.

Decision Tree Machine Learning Algorithm



You are making a weekend plan to visit the best restaurant in town as your parents are visiting but you are hesitant in making a decision on which restaurant to choose. Whenever you want to visit a restaurant you ask your friend Tyrion if he thinks you will like a particular place. To answer your question, Tyrion first has to find out, the kind of restaurants you like. You give him a list of restaurants that you have visited and tell him whether you liked each restaurant or not (giving a labelled training dataset). When you ask Tyrion that whether you will like a particular restaurant R or not, he asks you various questions like “Is “R” a roof top restaurant?” , “Does restaurant “R” serve Italian cuisine?”, “Does R have live music?”, “Is restaurant R open till midnight?” and so on. Tyrion asks you several informative questions to maximize the information gain and gives you YES or NO

answer based on your answers to the questionnaire. Here Tyrion is a decision tree for your favourite restaurant preferences.

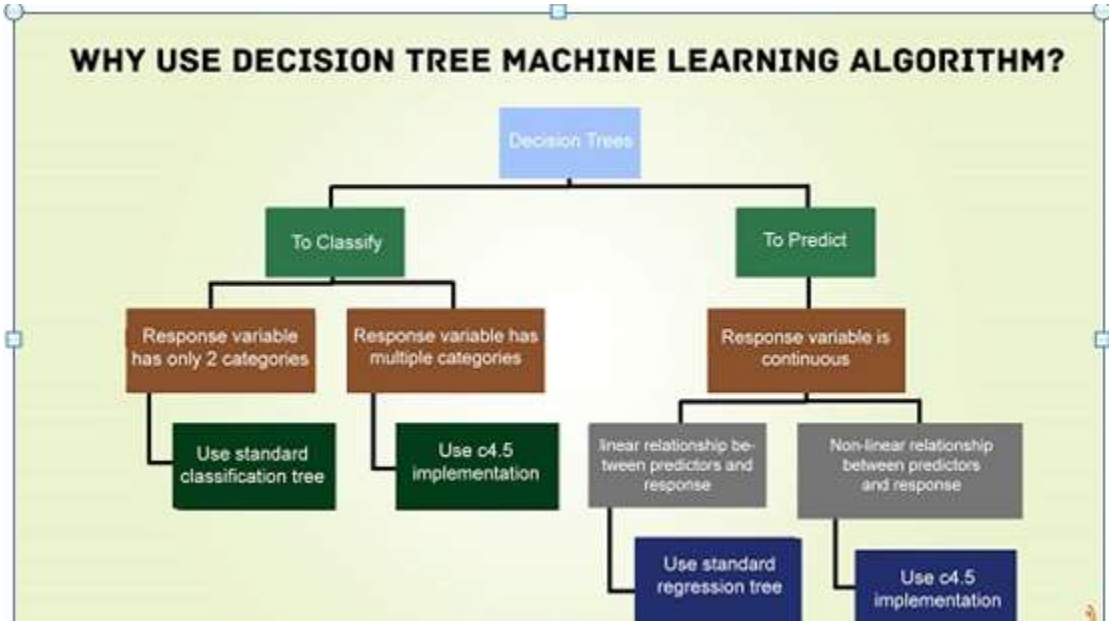
A decision tree is a graphical representation that makes use of branching methodology to exemplify all possible outcomes of a decision, based on certain conditions. In a decision tree, the internal node represents a test on the attribute, each branch of the tree represents the outcome of the test and the leaf node represents a particular class label i.e. the decision made after computing all of the attributes. The classification rules are represented through the path from root to the leaf node.

Types of Decision Trees

Classification Trees- These are considered as the default kind of decision trees used to separate a dataset into different classes, based on the response variable. These are generally used when the response variable is categorical in nature.

Regression Trees-When the response or target variable is continuous or numerical, regression trees are used. These are generally used in predictive type of problems when compared to classification.

Decision trees can also be classified into two types, based on the type of target variable- Continuous Variable Decision Trees and Binary Variable Decision Trees. It is the target variable that helps decide what kind of decision tree would be required for a particular problem.



Why should you use Decision Tree Machine Learning algorithm?

- These machine learning algorithms help make decisions under uncertainty and help you improve communication, as they present a visual representation of a decision situation.
- Decision tree machine learning algorithms help a data scientist capture the idea that if a different decision was taken, then how the operational nature of a situation or model would have changed intensely.

- Decision tree algorithms help make optimal decisions by allowing a data scientist to traverse through forward and backward calculation paths.

When to use Decision Tree Machine Learning Algorithm

- Decision trees are robust to errors and if the training data contains errors- decision tree algorithms will be best suited to address such problems.
- Decision trees are best suited for problems where instances are represented by attribute value pairs.
- If the training data has missing value then decision trees can be used, as they can handle missing values nicely by looking at the data in other columns.
- Decision trees are best suited when the target function has discrete output values.

Advantages of Using Decision Tree Machine Learning Algorithms

- Decision trees are very instinctual and can be explained to anyone with ease. People from a non-technical background, can also decipher the hypothesis drawn from a decision tree, as they are self-explanatory.
- When using decision tree machine learning algorithms, data type is not a constraint as they can handle both categorical and numerical variables.
- Decision tree machine learning algorithms do not require making any assumption on the linearity in the data and hence can be used in circumstances where the parameters are non-linearly related. These machine learning algorithms do not make any assumptions on the classifier structure and space distribution.
- These algorithms are useful in data exploration. Decision trees implicitly perform feature selection which is very important in predictive analytics. When a decision tree is fit to a training dataset, the nodes at the top on which the decision tree is split, are considered as important variables within a given dataset and feature selection is completed by default.
- Decision trees help save data preparation time, as they are not sensitive to missing values and outliers. Missing values will not stop you from splitting the data for building a decision tree. Outliers will also not affect the decision trees as data splitting happens based on some samples within the split range and not on exact absolute values.

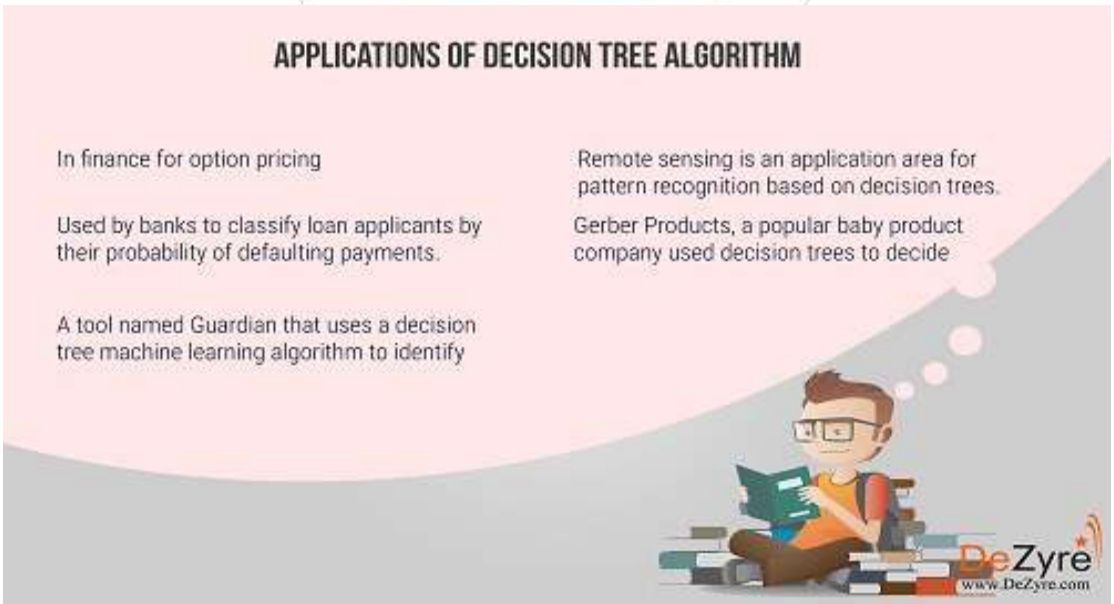
Drawbacks of Using Decision Tree Machine Learning Algorithms

- The more the number of decisions in a tree, less is the accuracy of any expected outcome.
- A major drawback of decision tree machine learning algorithms, is that the outcomes may be based on expectations. When decisions are made in real-time, the payoffs and resulting outcomes might not be the same as expected or planned. There are chances that this could lead to unrealistic decision trees leading to bad decision making. Any irrational expectations could lead to major errors and flaws in decision tree analysis, as it is not always possible to plan for all eventualities that can arise from a decision.
- Decision Trees do not fit well for continuous variables and result in instability and classification plateaus.

- Decision trees are easy to use when compared to other decision making models but creating large decision trees that contain several branches is a complex and time consuming task.
- Decision tree machine learning algorithms consider only one attribute at a time and might not be best suited for actual data in the decision space.
- Large sized decision trees with multiple branches are not comprehensible and pose several presentation difficulties.

Applications of Decision Tree Machine Learning Algorithm

- Decision trees are among the popular machine learning algorithms that find great use in finance for option pricing.
- Remote sensing is an application area for pattern recognition based on decision trees.
- Decision tree algorithms are used by banks to classify loan applicants by their probability of defaulting payments.
- Gerber Products, a popular baby product company, used decision tree machine learning algorithm to decide whether they should continue using the plastic PVC (Poly Vinyl Chloride) in their products.
- Rush University Medical Centre has developed a tool named Guardian that uses a decision tree machine learning algorithm to identify at-risk patients and disease trends.
- The Data Science libraries in Python language to implement Decision Tree Machine Learning Algorithm are – SciPy and Sci-Kit Learn.
- The Data Science libraries in R language to implement Decision Tree Machine Learning Algorithm is caret.



Random Forest Machine Learning Algorithm

Let's continue with the same example that we used in decision trees, to explain how Random Forest Machine Learning Algorithm works. Tyrion is a decision tree for your

restaurant preferences. However, Tyrion being a human being does not always generalize your restaurant preferences with accuracy. To get more accurate restaurant recommendation, you ask a couple of your friends and decide to visit the restaurant R, if most of them say that you will like it. Instead of just asking Tyrion, you would like to ask Jon Snow, Sandor, Bronn and Bran who vote on whether you will like the restaurant R or not. This implies that you have built an ensemble classifier of decision trees - also known as a forest.

You don't want all your friends to give you the same answer - so you provide each of your friends with slightly varying data. You are also not sure of your restaurant preferences and are in a dilemma.You told Tyrion that you like Open Roof Top restaurants but maybe, just because it was summer when you visited the restaurant you could have liked it then. You may not be a fan of the restaurant during the chilly winters. Thus, all your friends should not make use of the data point that you like open roof top restaurants, to make their recommendations for your restaurant preferences.

By providing your friends with slightly different data on your restaurant preferences, you make your friends ask you different questions at different times. In this case just by slightly altering your restaurant preferences, you are injecting randomness at model level (unlike randomness at data level in case of decision trees). Your group of friends now form a random forest of your restaurant preferences.

Random Forest is the go to machine learning algorithm that uses a bagging approach to create a bunch of decision trees with random subset of the data. A model is trained several times on random sample of the dataset to achieve good prediction performance from the random forest algorithm.In this ensemble learning method, the output of all the decision trees in the random forest, is combined to make the final prediction. The final prediction of the random forest algorithm is derived by polling the results of each decision tree or just by going with a prediction that appears the most times in the decision trees.

For instance, in the above example - if 5 friends decide that you will like restaurant R but only 2 friends decide that you will not like the restaurant then the final prediction is that, you will like restaurant R as majority always wins.

Access the Solution to Kaggle Data Science Challenge - Expedia Hotel Recommendations

Why use Random Forest Machine Learning Algorithm?

- There are many good open source, free implementations of the algorithm available in Python and R.
- It maintains accuracy when there is missing data and is also resistant to outliers.
- Simple to use as the basic random forest algorithm can be implemented with just a few lines of code.
- Random Forest machine learning algorithms help data scientists save data preparation time, as they do not require any input preparation and are capable of handling numerical, binary and categorical features, without scaling, transformation or modification.

- Implicit feature selection as it gives estimates on what variables are important in the classification.

Advantages of Using Random Forest Machine Learning Algorithms

- Overfitting is less of an issue with Random Forests, unlike decision tree machine learning algorithms. There is no need of pruning the random forest.
- These algorithms are fast but not in all cases. A random forest algorithm, when run on an 800 MHz machine with a dataset of 100 variables and 50,000 cases produced 100 decision trees in 11 minutes.
- Random Forest is one of the most effective and versatile machine learning algorithm for wide variety of classification and regression tasks, as they are more robust to noise.
- It is difficult to build a bad random forest. In the implementation of Random Forest Machine Learning algorithms, it is easy to determine which parameters to use because they are not sensitive to the parameters that are used to run the algorithm. One can easily build a decent model without much tuning.
- Random Forest machine learning algorithms can be grown in parallel.
- This algorithm runs efficiently on large databases.
- Has higher classification accuracy.

Drawbacks of Using Random Forest Machine Learning Algorithms

- They might be easy to use but analysing them theoretically, is difficult.
- Large number of decision trees in the random forest can slow down the algorithm in making real-time predictions.
- If the data consists of categorical variables with different number of levels, then the algorithm gets biased in favour of those attributes that have more levels. In such situations, variable importance scores do not seem to be reliable.
- When using RandomForest algorithm for regression tasks, it does not predict beyond the range of the response values in the training data.

Applications of Random Forest Machine Learning Algorithms

- Random Forest algorithms are used by banks to predict if a loan applicant is a likely high risk.
- They are used in the automobile industry to predict the failure or breakdown of a mechanical part.
- These algorithms are used in the healthcare industry to predict if a patient is likely to develop a chronic disease or not.
- They can also be used for regression tasks like predicting the average number of social media shares and performance scores.
- Recently, the algorithm has also made way into predicting patterns in speech recognition software and classifying images and texts.
- Data Science libraries in Python language to implement Random Forest Machine Learning Algorithm is Sci-Kit Learn.

SVM Classifier (support Vector machines):- Support Vector Machine Learning Algorithm

Support Vector Machine is a supervised machine learning algorithm for classification or regression problems where the dataset teaches SVM about the classes so that SVM can classify any new data. It works by classifying the data into different classes by finding a line (hyperplane) which separates the training data set into classes. As there are many such linear hyperplanes, SVM algorithm tries to maximize the distance between the various classes that are involved and this is referred as margin maximization. If the line that maximizes the distance between the classes is identified, the probability to generalize well to unseen data is increased.

SVM's are classified into two categories:

Linear SVM's – In linear SVM's the training data i.e. classifiers are separated by a hyperplane.

Non-Linear SVM's- In non-linear SVM's it is not possible to separate the training data using a hyperplane. For example, the training data for Face detection consists of group of images that are faces and another group of images that are not faces (in other words all other images in the world except faces). Under such conditions, the training data is too complex that it is impossible to find a representation for every feature vector. Separating the set of faces linearly from the set of non-face is a complex task.

Advantages of Using SVM

- SVM offers best classification performance (accuracy) on the training data.SVM renders more efficiency for correct classification of the future data.The best thing about SVM is that it does not make any strong assumptions on data.It does not over-fit the data.

Applications of Support Vector Machine

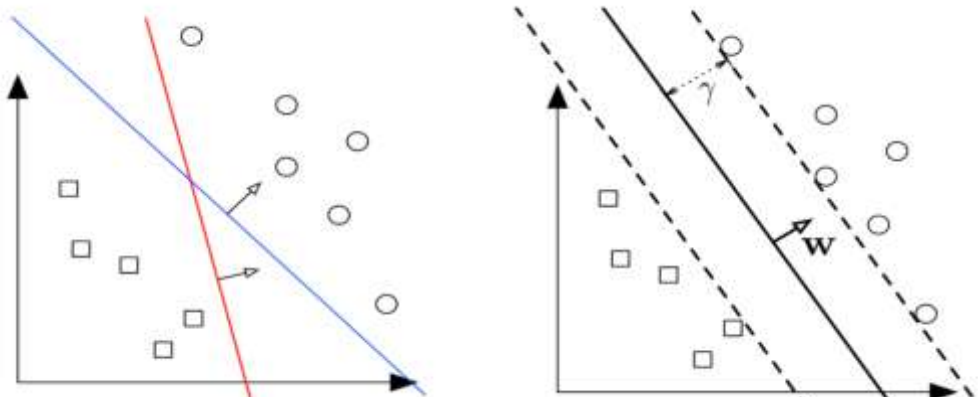
SVM is commonly used for stock market forecasting by various financial institutions. For instance, it can be used to compare the relative performance of the stocks when compared to performance of other stocks in the same sector. The relative comparison of stocks helps manage investment making decisions based on the classifications made by the SVM learning algorithm.

Data Science Libraries in Python to implement Support Vector Machine –SciKit Learn, PyML , SVM^{Struct} Python , LIBSVM

Data Science Libraries in R to implement Support Vector Machine – klar, e1071

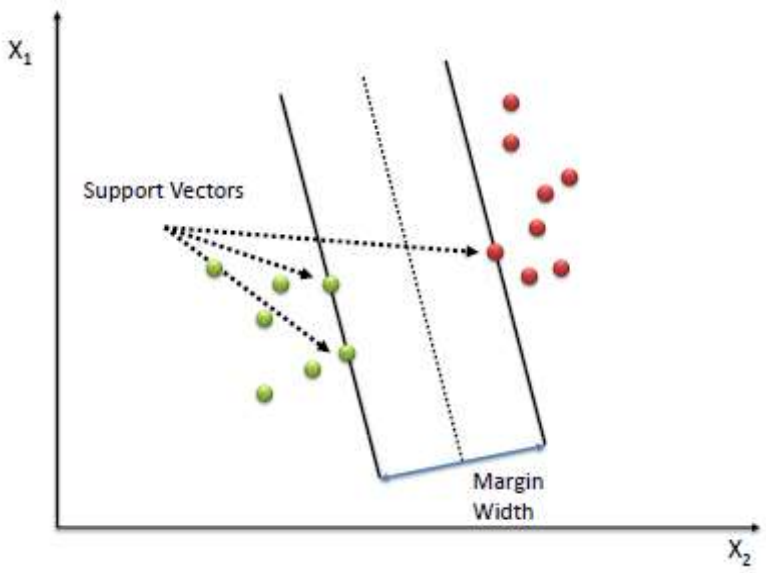
A **Support Vector Machine (SVM)** is a very powerful and flexible Machine Learning Model, capable of performing linear or nonlinear classification, regression, and even outlier detection. It is one of the most popular models in Machine Learning, and anyone interested in ML should have it in their toolbox. SVMs are particularly well suited for classification of complex but small or medium sized datasets.

Linear SVM Let's say we have 2 classes of data which we want to classify using SVM as shown in the figure.



The 2 classes can clearly be separated easily with a straight line (linearly separable). The left plot shows the decision boundaries of 2 possible linear classifiers. An SVM model is all about generating the right line (called **Hyperplane** in higher dimension) that classifies the data very well. In the left plot, even though red line classifies the data, it might not perform very well on new instances of data. We can draw many lines that classifies this data, but among all these lines blue line separates the data most. The same blue line is shown on the right plot. This line (hyperplane) not only separates the two classes but also stays as far away from the closest training instances possible. You can think of an SVM classifier as fitting the widest possible street (represented by parallel dashed lines on the right plot) between the classes. This is called Large Margin Classification.

This best possible decision boundary is determined (or "supported") by the instances located on the edge of the street. These instances are called the **support vectors**. The distance between the edges of "the street" is called **margin**.

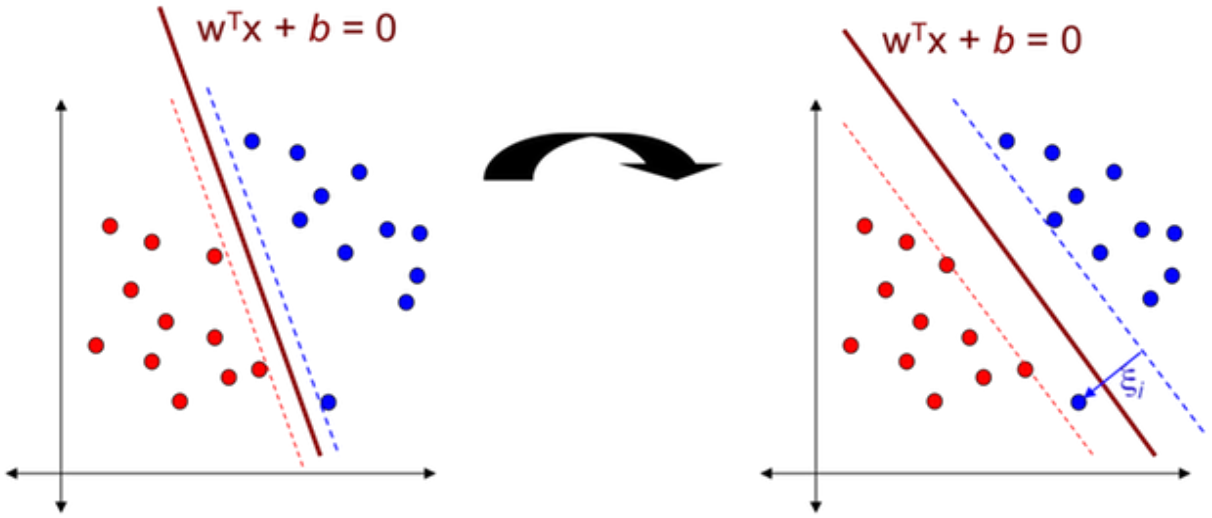


Soft Margin Classification

If we strict our instances be off the "street" and on the correct side of the line, this is called Hard margin classification. There are 2 problems with hard margin classification.

1) It only works if the data is linearly seperable.

2) It is quite sensitive to outliers.

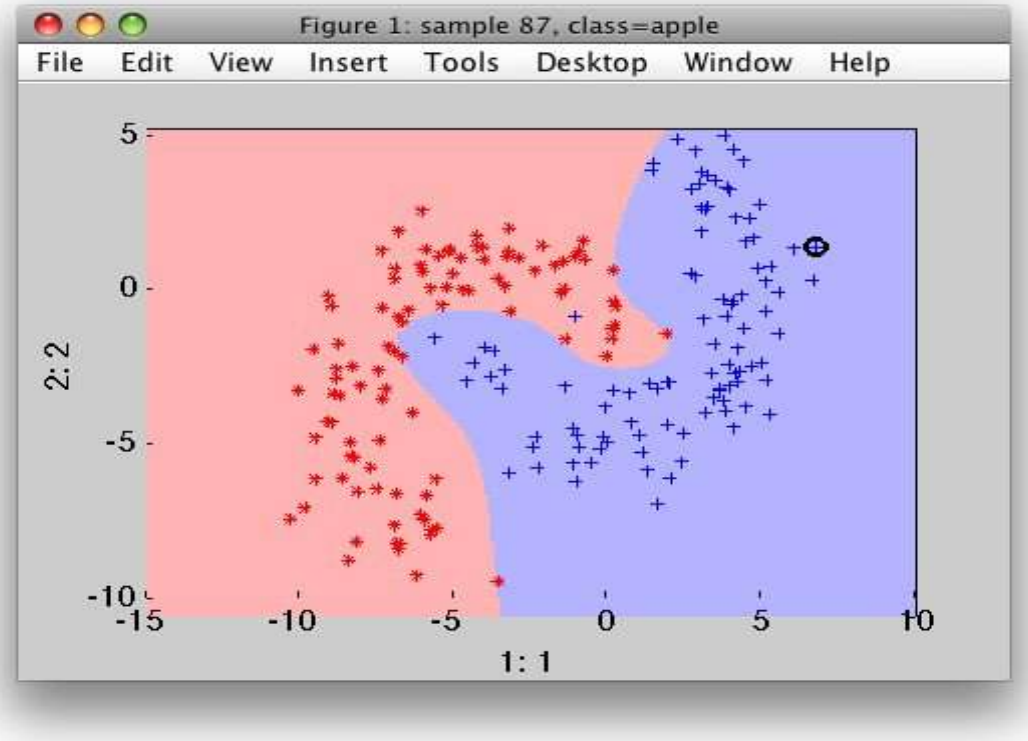


In the above data classes, there is a blue outlier. And if we apply Hard margin classification on this dataset, we will get decision boundary with small margin shown in the left diagram. To avoid these issues it is preferable to to use more flexible model. The objective is to find a good balance between keeping the street as large as possible and limiting the margin violation (i.e., instances that end up in the middle of the street or even on the wrong side). This is called Soft margin classification. If we apply Soft margin classification on this dataset, we will get decision boundary with larger margin than Hard margin classification. This is shown in the right diagram.

Nonlinear SVM

Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly seperable. One simple method to handle nonlinear datasets is to add more features, such as polynomial features and sometimes this can result in a linearly seperable dataset. By generating polynomial features, we will have a new feature matrix consisting of all polynomial combinations of the features with degree

less than or equal to the specified degree. Following image is an example of using Polynomial Features for SVM.



Kernel Trick

Kernel is a way of computing the dot product of two vectors **x** and **y** in some (possibly very high dimensional) feature space, which is why kernel functions are sometimes called "generalized dot product".

Suppose we have a mapping φ:ℝn→ℝm that brings our vectors in ℝn to some feature space ℝm. Then the dot product of x and y in this space is φ(x)Tφ(y). A kernel is a function k that corresponds to this dot product, i.e. k(x,y)=φ(x)Tφ(y). Kernels give a way to compute dot products in some feature space without even knowing what this space is and what is φ.

Polynomial Kernel

Adding polynomial features is very simple to implement. But a low polynomial degree cannot deal with complex datasets, and with high polynomial degree it will create huge number of features, making the model too slow. In these situations we can use a polynomial kernel to avoid this problem. Polynomial kernal is of the following format;

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

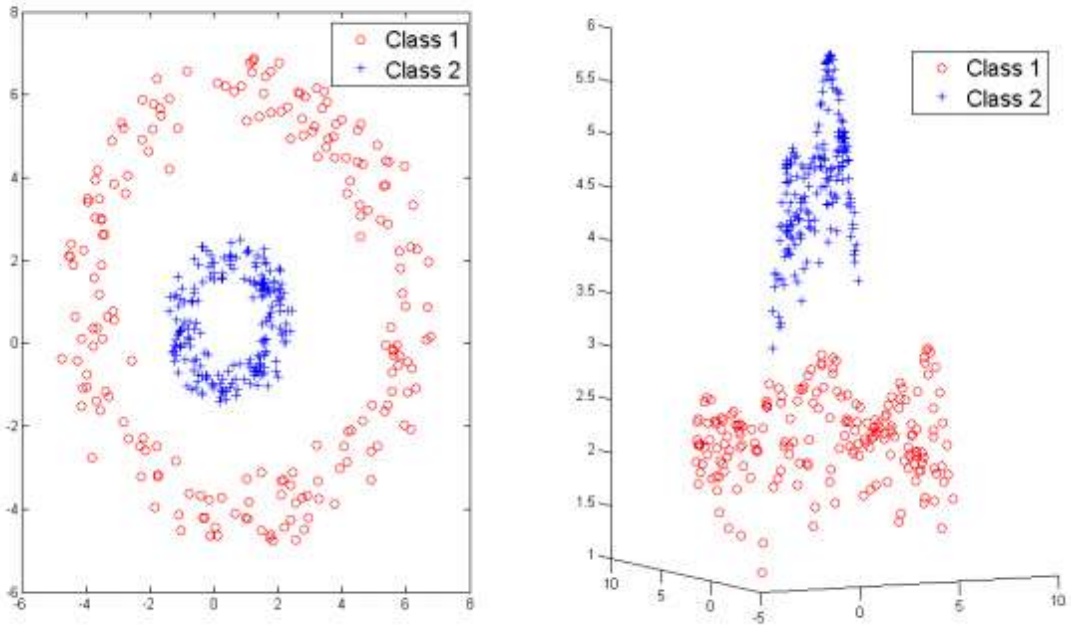
Where d is the degree of the polynomial.

Gaussian RBF Kernel

Gaussian RBF(Radial Basis Function) is another popular Kernel method used in SVM models. Gaussian Kernel is of the following format;

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma|\mathbf{x}-\mathbf{y}|^2}, \gamma > 0$$

RBF Kernels are very useful if we have datasets like the following one;



Hyperparameters

There are 2 important hyperparameters in an SVM model.

C Parameter

The C parameter decides the margin width of the SVM classifier. Large value of C makes the classifier strict and thus small margin width. For large values of C, the model will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the model to look

for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

γ Parameter

The γ parameter defines the influence of each training example reaches. γ parameter is invalid for a linear kernel in scikit-learn.

Implementation using scikit-learn

In this part we will implement SVM using scikit-learn. We will be using artificial datasets.

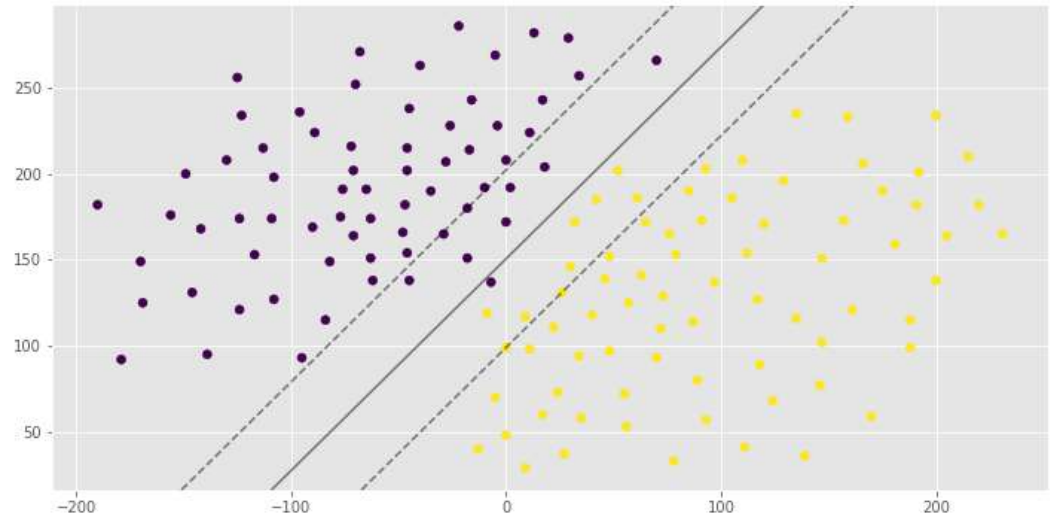
Linear Kernel

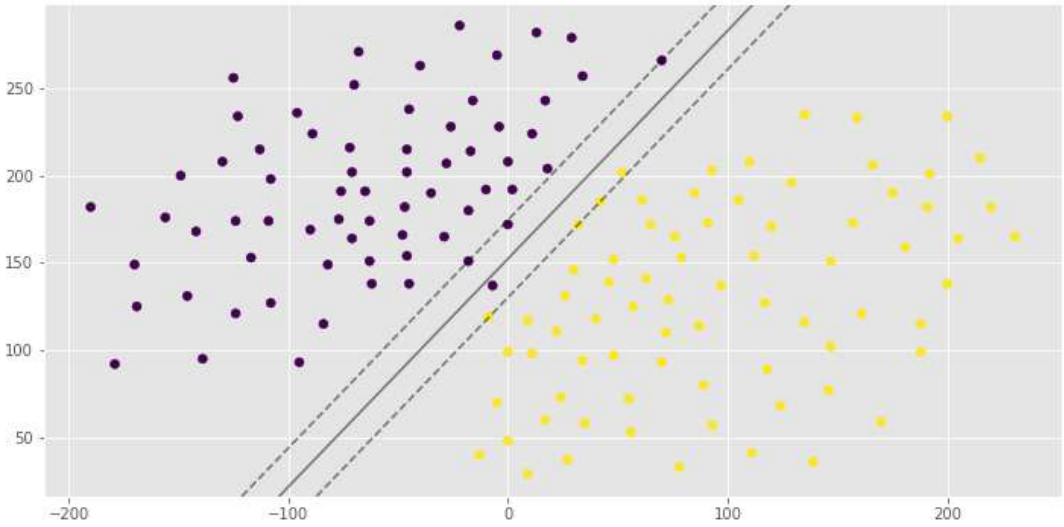
Python Code:

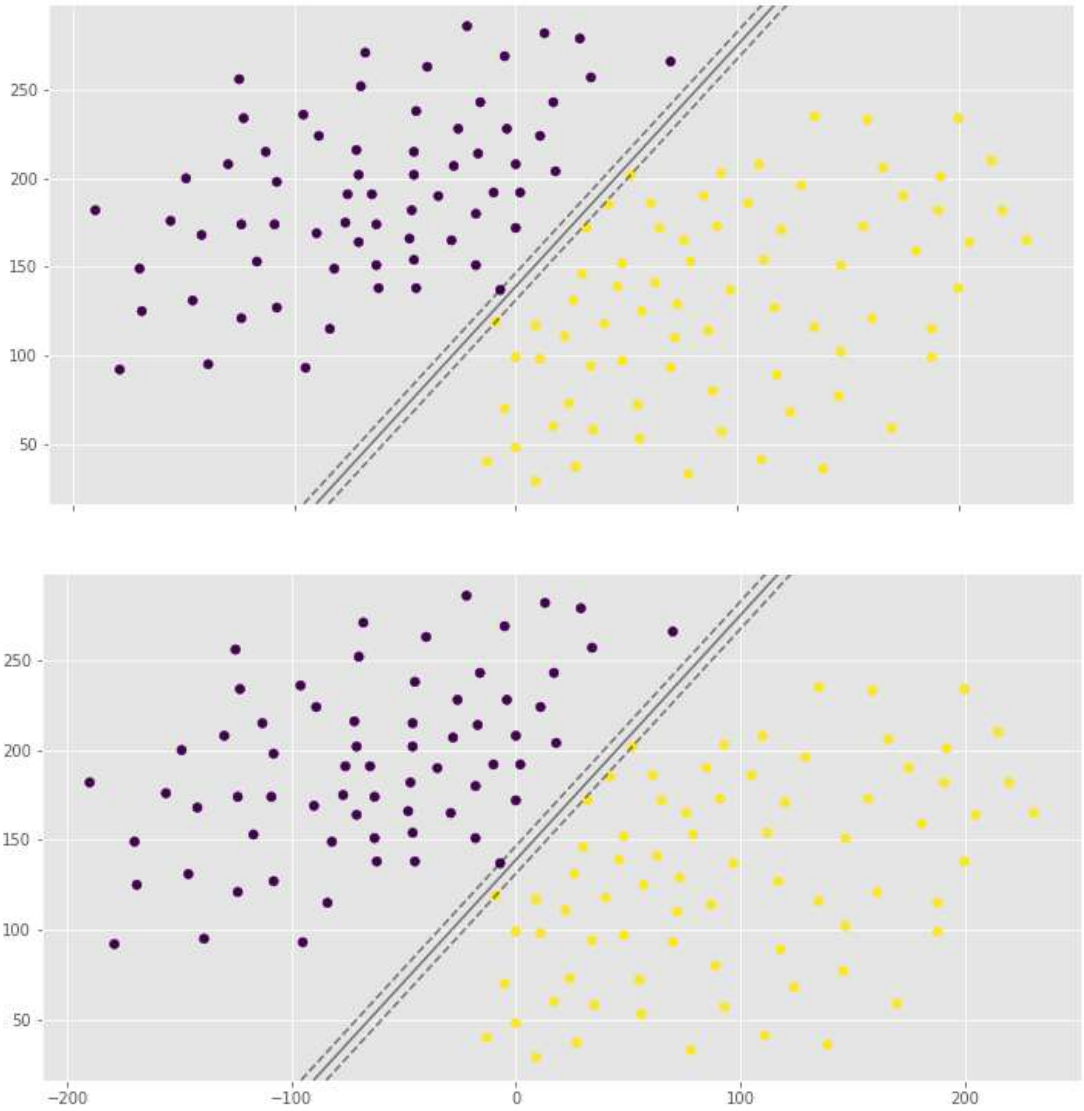
```
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

# Import Dataset
data = pd.read_csv('data.csv', header=None)
X = data.values[:, :2]
y = data.values[:, 2]

# A function to draw hyperplane and the margin of SVM classifier
def draw_svm(X, y, C=1.0):
    # Plotting the Points
    plt.scatter(X[:,0], X[:,1], c=y)

    # The SVM Model with given C parameter
    clf = SVC(kernel='linear', C=C)
```

```
    clf_fit = clf.fit(X, y)

    # Limit of the axes
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Creating the meshgrid
    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    # Plotting the boundary
    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                   alpha=0.5, linestyles=['--', '-', '--'])
    ax.scatter(clf.support_vectors_[:, 0],
               clf.support_vectors_[:, 1],
               s=100, linewidth=1, facecolors='none')
    plt.show()
    # Returns the classifier
    return clf_fit

clf_arr = []
clf_arr.append(draw_svm(X, y, 0.0001))
clf_arr.append(draw_svm(X, y, 0.001))
clf_arr.append(draw_svm(X, y, 1))
clf_arr.append(draw_svm(X, y, 10))

for i, clf in enumerate(clf_arr):
    # Accuracy Score
```

```
print(clf.score(X, y))
pred = clf.predict([(12, 32), (-250, 32), (120, 43)])
print(pred)
```







Output:

0.992907801418

[1 0 1]

0.992907801418

[1 0 1]

1.0

[1 0 1]

1.0

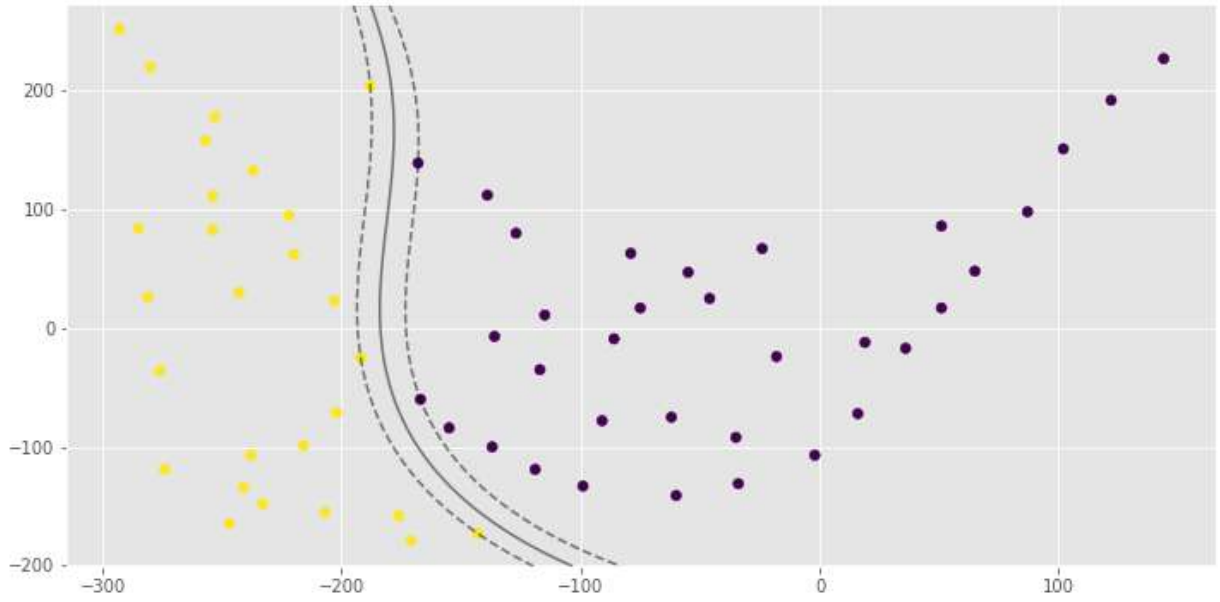
[1 0 1]

You can see the same hyperplane with different margin width. It is depends on the C hyperparameter.

Polynomial Kernel

```
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

data = pd.read_csv('polydata2.csv', header=None)
X = data.values[:, :2]
y = data.values[:, 2]

def draw_svm(X, y, C=1.0):
    plt.scatter(X[:,0], X[:,1], c=y)
    clf = SVC(kernel='poly', C=C)
    clf_fit = clf.fit(X, y)

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                alpha=0.5, linestyles=['--', '-', '--'])
```

```
    ax.scatter(clf.support_vectors_[:, 0],
               clf.support_vectors_[:, 1],
               s=100, linewidth=1, facecolors='none')
    plt.show()
    return clf_fit

clf = draw_svm(X, y)
score = clf.score(X, y)
pred = clf.predict([(-130, 110), (-170, -160), (80, 90), (-280, 20)])
print(score)
print(pred)
```



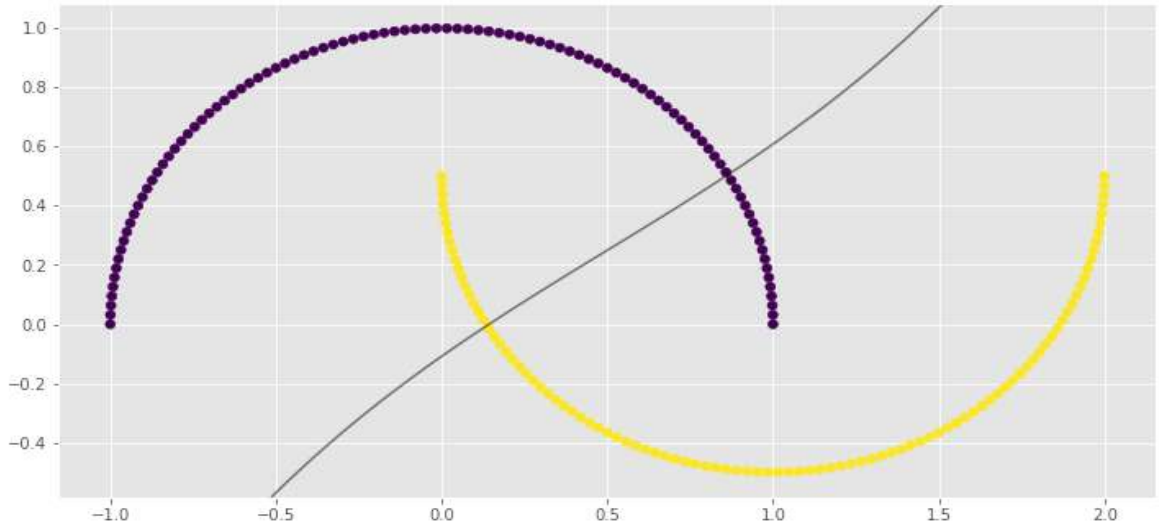

```
1.0
[0 1 0 1]
```

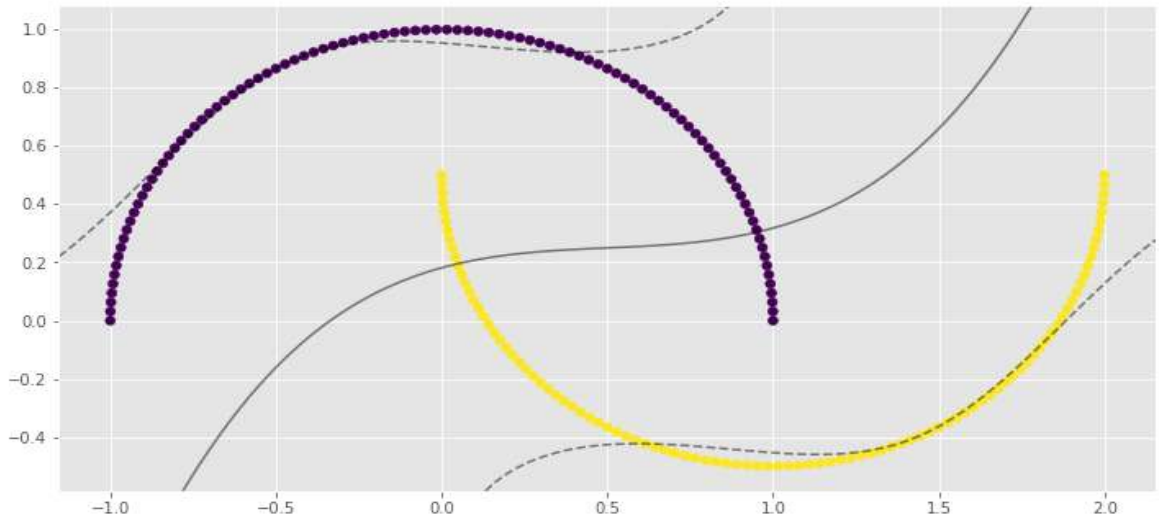
Gaussian Kernel

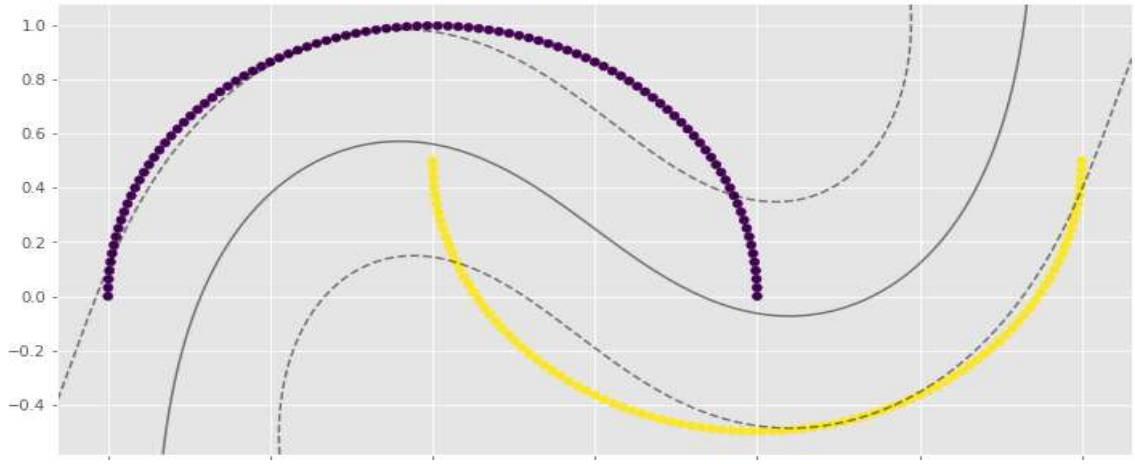
```
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
```

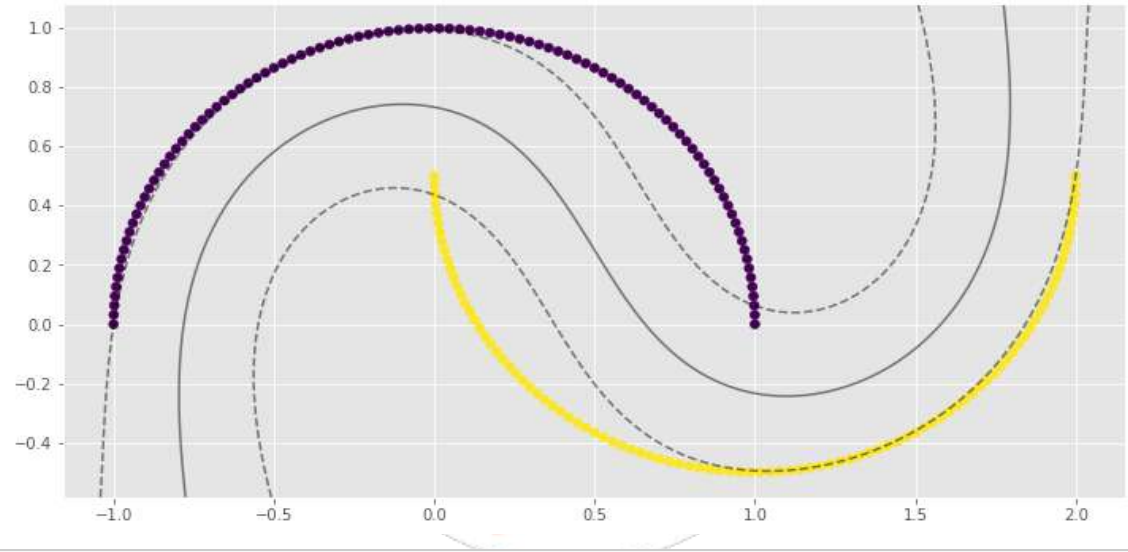
```
from sklearn.datasets import make_classification, make_blobs, make_moons
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

X, y = make_moons(n_samples=200)

# Auto gamma equals 1/n_features
def draw_svm(X, y, C=1.0, gamma='auto'):
    plt.scatter(X[:,0], X[:,1], c=y)
    clf = SVC(kernel='rbf', C=C, gamma=gamma)
    clf_fit = clf.fit(X, y)

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                    alpha=0.5, linestyles=['--', '-', '--'])
    ax.scatter(clf.support_vectors_[:, 0],
               clf.support_vectors_[:, 1],
               s=100, linewidth=1, facecolors='none')
    plt.show()
    return clf_fit

clf_arr = []
```

```
clf_arr.append(draw_svm(X, y, 0.01))
clf_arr.append(draw_svm(X, y, 0.1))
clf_arr.append(draw_svm(X, y, 1))
clf_arr.append(draw_svm(X, y, 10))

for i, clf in enumerate(clf_arr):
   print(clf.score(X, y))
```









0.83

0.9

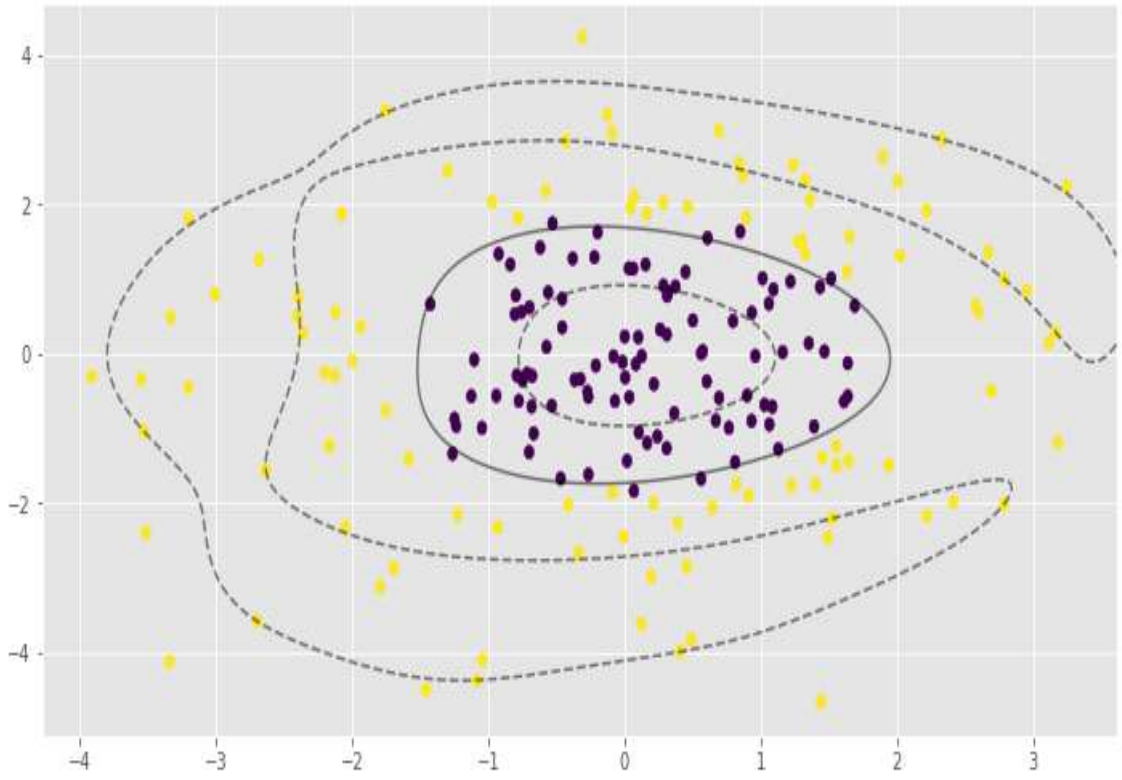
1.0

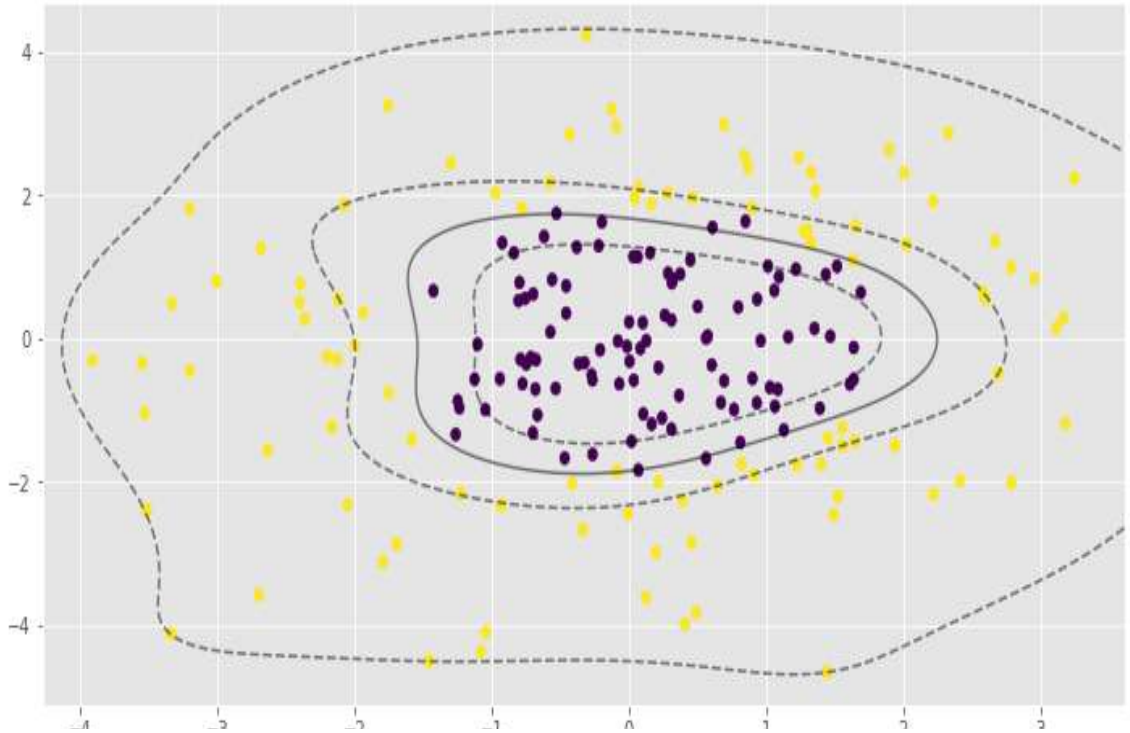
1.0

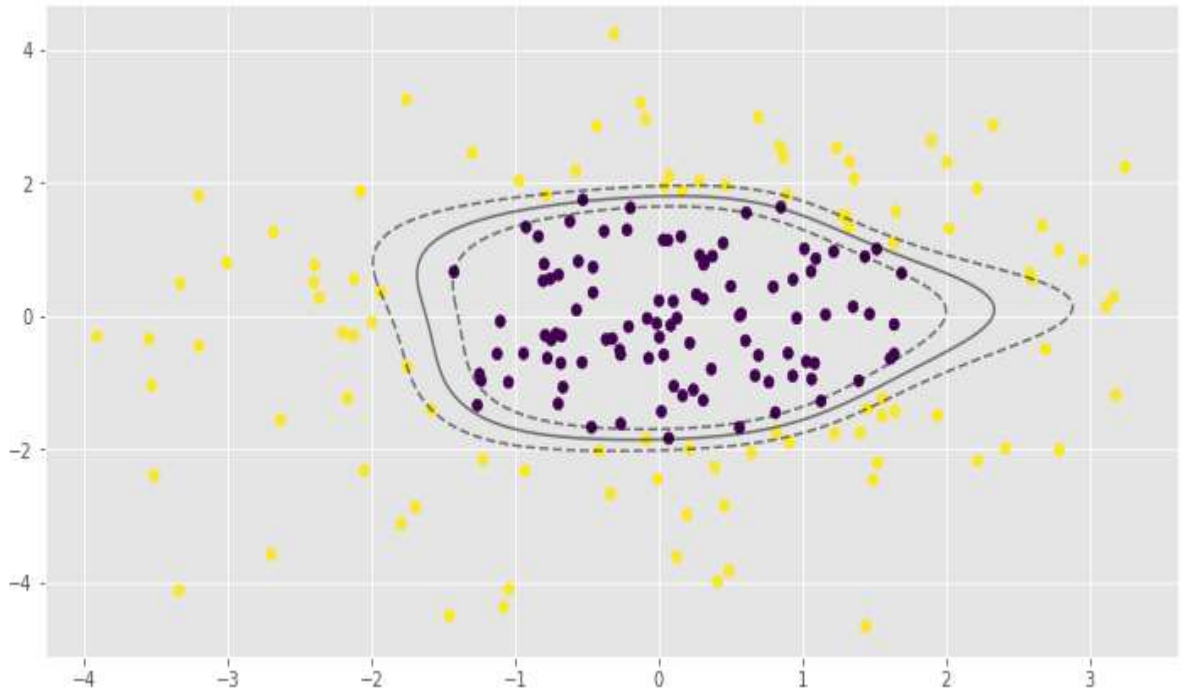
```
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
from sklearn.datasets import make_gaussian_quantiles
import matplotlib.pyplot as plt
```

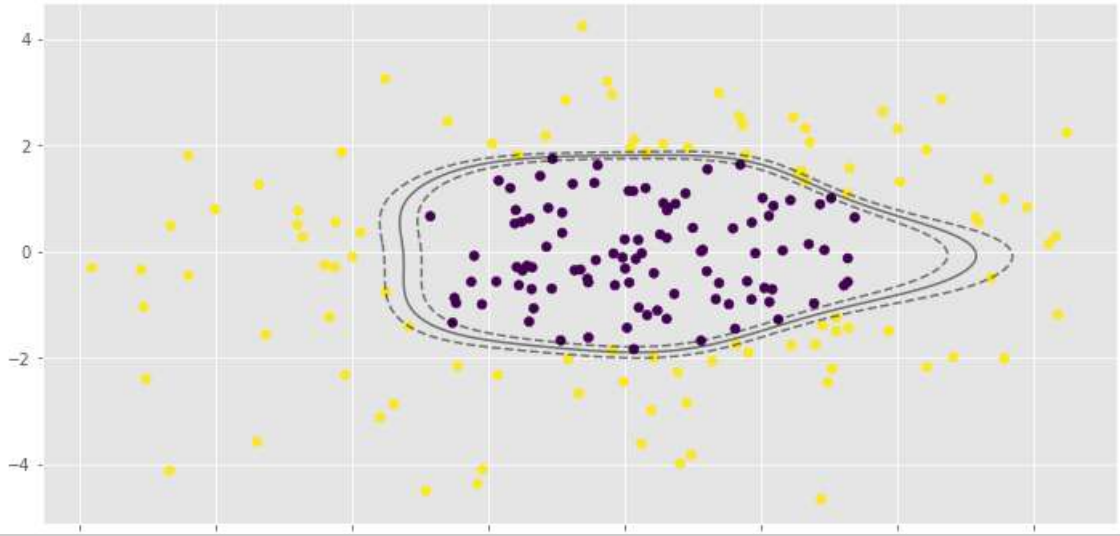
```
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

X, y = make_gaussian_quantiles(n_samples=200, n_features=2, n_classes=2, cov=3)

# Auto gamma equals 1/n_features
def draw_svm(X, y, C=1.0, gamma='auto'):
    plt.scatter(X[:,0], X[:,1], c=y)
    clf = SVC(kernel='rbf', C=C, gamma=gamma)
    clf_fit = clf.fit(X, y)

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                    alpha=0.5, linestyles=['--', '-', '--'])
    ax.scatter(clf.support_vectors_[:, 0],
            clf.support_vectors_[:, 1],
            s=100, linewidth=1, facecolors='none')
    plt.show()
    return clf_fit

clf_arr = []
clf_arr.append(draw_svm(X, y, 0.1))
clf_arr.append(draw_svm(X, y, 1))
```

```
clf_arr.append(draw_svm(X, y, 10))
clf_arr.append(draw_svm(X, y, 100))

for i, clf in enumerate(clf_arr):
    print(clf.score(X, y))
```




4
2
0
−2
−4


4
2
0
−2
−4
−4
−3
−2
−1
0
1
2
3



0.965

0.97

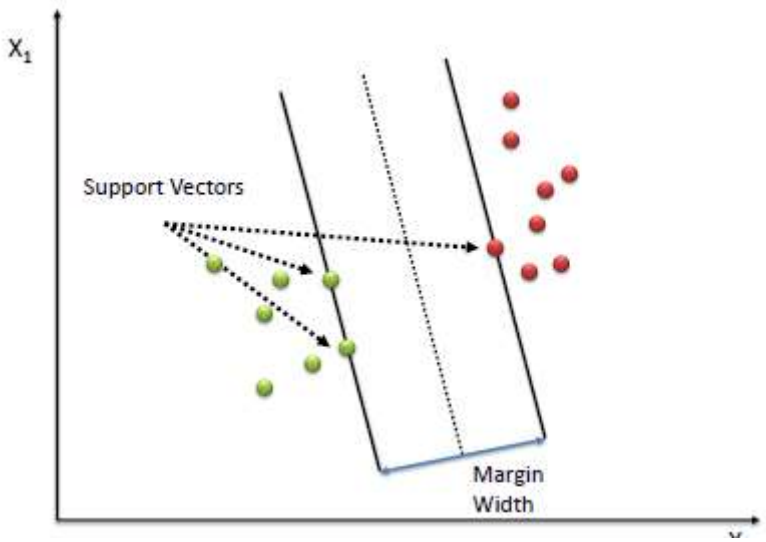
0.985

0.995

γγ parameter is very important to the RBF SVM model. In the first example low value of γγ leads to almost linear classification.

The 2 classes can clearly be seperated easily with a straight line (linearly seperable). The left plot shows the decision boundaries of 2 possible linear classifiers. An SVM model is all about generating the right line (called **Hyperplane** in higher dimension) that classifies the data very well. In the left plot, even though red line classifies the data, it might not perform very well on new instances of data. We can draw many lines that classifies this data, but among all these lines blue line seperates the data most. The same blue line is shown on the right plot. This line (hyperplane) not only seperates the two classes but also stays as far away from the closest training instances possible. You can think of an SVM classifier as fitting the widest possible street (represented by parallel dashed lines on the right plot) between the classes. This is called Large Margin Classification.

This best possible decision boundary is determined (or "supported") by the instances located on the edge of the street. These instances are called the **support vectors**. The distance between the edges of "the street" is called **margin**.

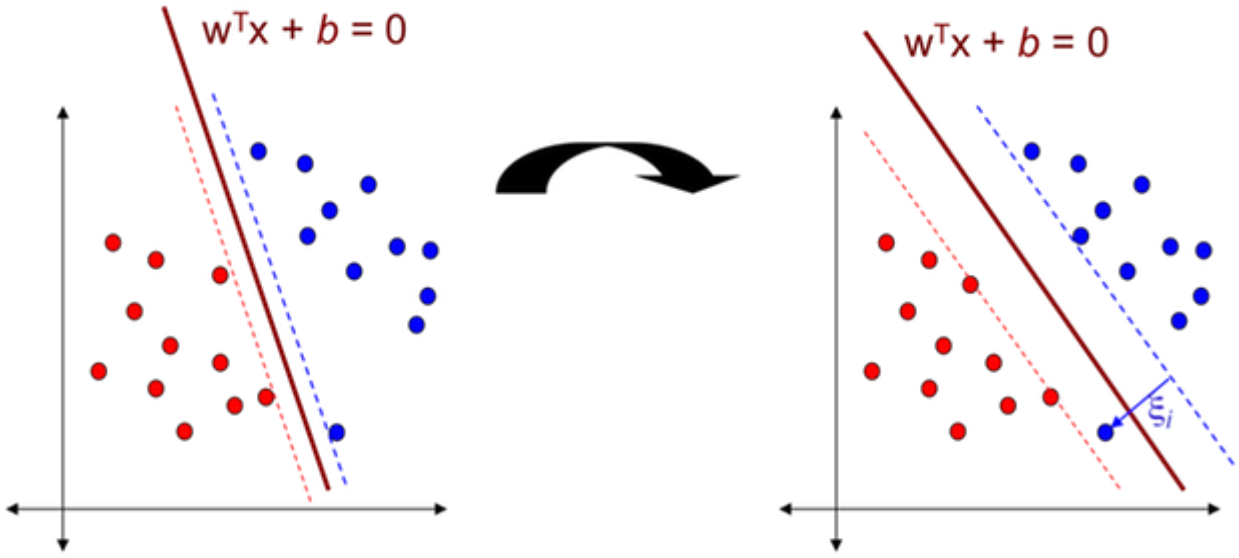


Soft Margin Classification

If we strict our instances be off the "street" and on the correct side of the line, this is called Hard margin classification. There are 2 problems with hard margin classification.

1) It only works if the data is linearly seperable.

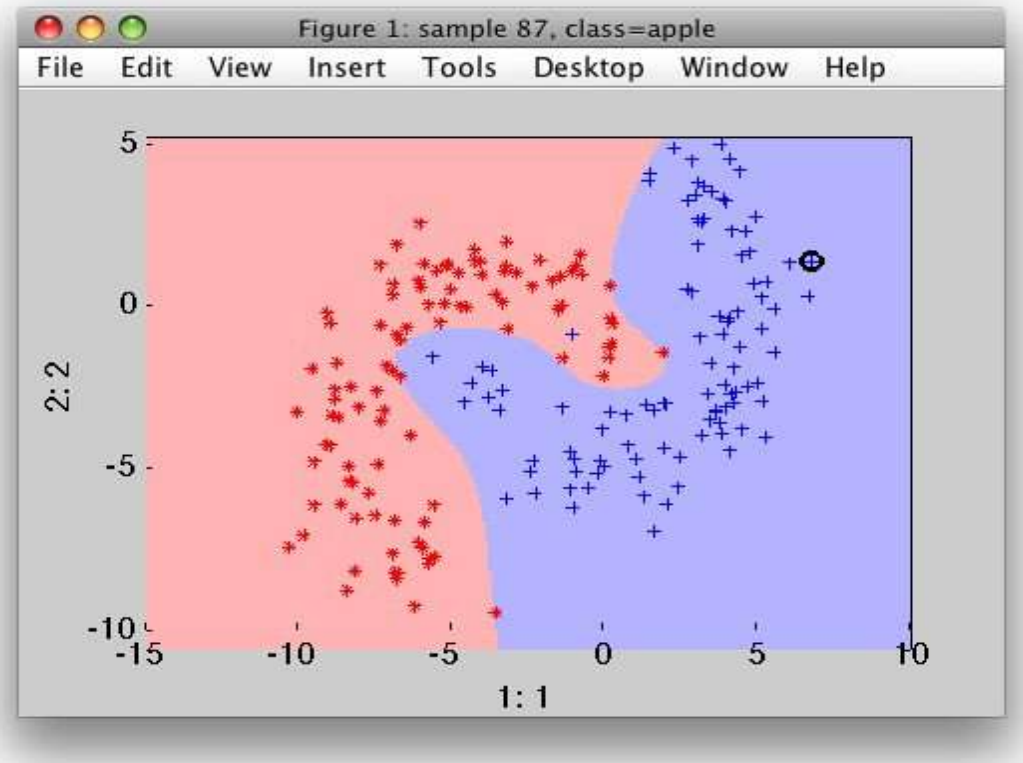
2) It is quite sensitive to outliers.



In the above data classes, there is a blue outlier. And if we apply Hard margin classification on this dataset, we will get decision boundary with small margin shown in the left diagram. To avoid these issues it is preferable to to use more flexible model. The objective is to find a good balance between keeping the street as large as possible and limiting the margin violation (i.e., instances that end up in the middle of the street or even on the wrong side). This is called Soft margin classification. If we apply Soft margin classification on this dataset, we will get decision boundary with larger margin than Hard margin classification. This is shown in the right diagram.

Nonlinear SVM

Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly seperable. One simple method to handle nonlinear datasets is to add more features, such as polynomial features and sometimes this can result in a linearly seperable dataset. By generating polynomial features, we will have a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. Following image is an example of using Polynomial Features for SVM.



Kernel Trick

Kernel is a way of computing the dot product of two vectors **x** and **y** in some (possibly very high dimensional) feature space, which is why kernel functions are sometimes called "generalized dot product". uppose we have a mappingthat brings our vectors in to some feature space ℝmRm. Then the dot product of xx and yy in this space is φ(x)Tφ(y)φ(x)Tφ(y). A kernel is a function kkthat corresponds to this dot product, .e. k(x,y)=φ(x)Tφ(y)k(x,y)=φ(x)Tφ(yKernels give a way to compute dot products in some feature space without even knowing what this space is and what is φφ.

Polynomial Kernel

Adding polynomial features is very simple to implement. But a low polynomial degree cannot deal with complex datasets, and with high polynomial degree it will create huge number of features, making the model too slow. In these situations we can use a polynomial kernel to avoid this problem. Polynomial kernal is of the following format;

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

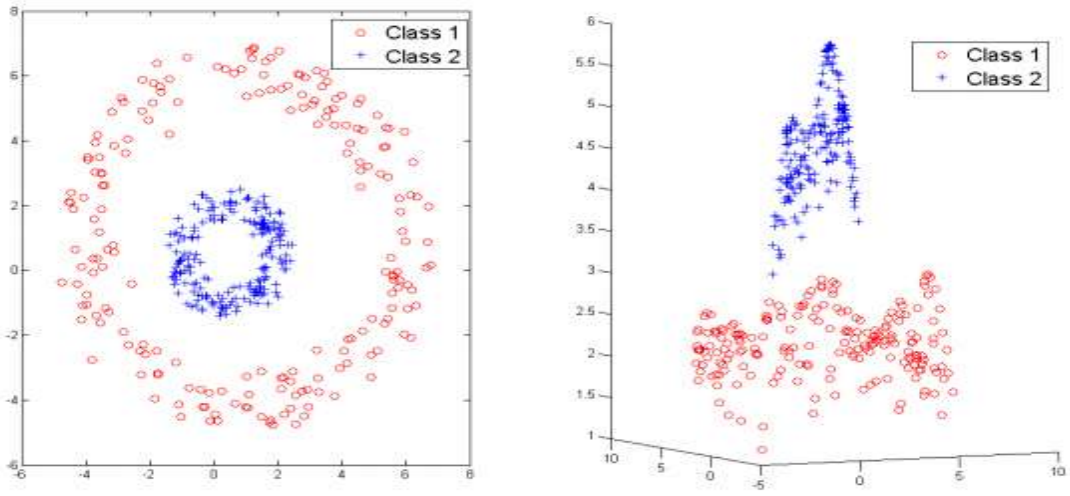
Where d is the degree of the polynomial.

Gaussian RBF Kernel

Gaussian RBF(Radial Basis Function) is another popular Kernel method used in SVM models. Gaussian Kernel is of the following format;

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma|\mathbf{x}-\mathbf{y}|^2}, \gamma > 0$$

RBF Kernels are very useful if we have datasets like the following one;



Hyperparameters

There are 2 important hyperparameters in an SVM model.

C Parameter

The C parameter decides the margin width of the SVM classifier. Large value of C makes the classifier strict and thus small margin width. For large values of C, the model will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the model to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

γ Parameter

The γ parameter defines the influence of each training example reaches. γ parameter is invalid for a linear kernel in scikit-learn.

Implementation using scikit-learn

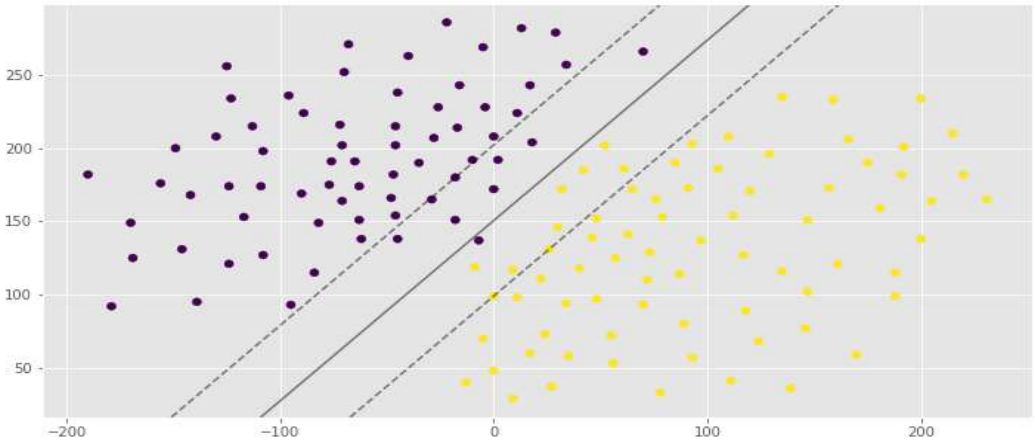
In this part we will implement SVM using scikit-learn. We will be using artificial datasets.

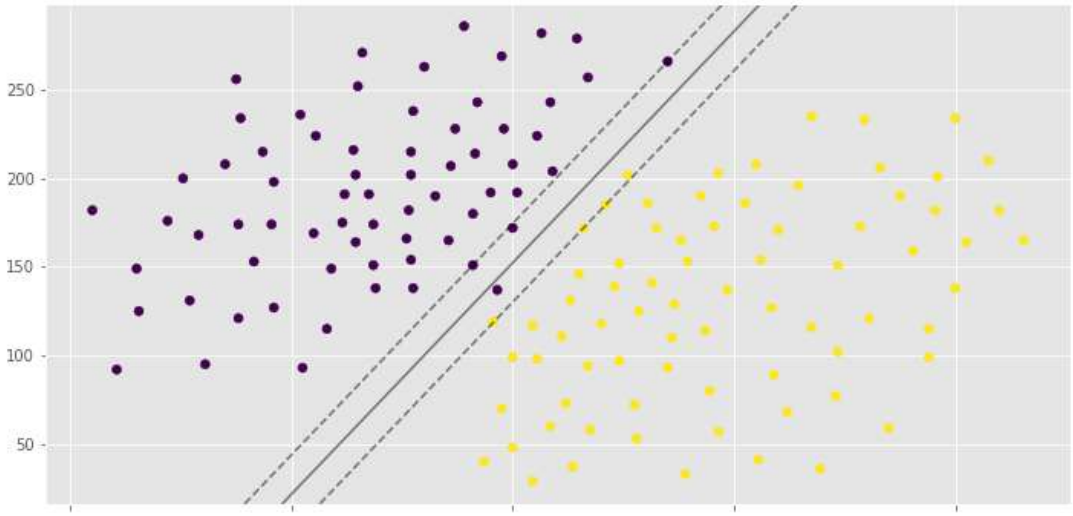
Linear Kernel

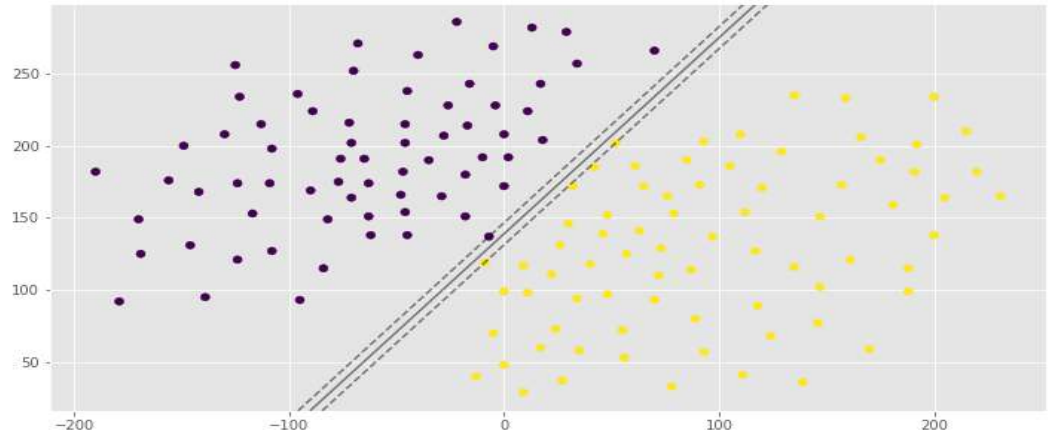
```
import numpy as np
import pandas as pd
```

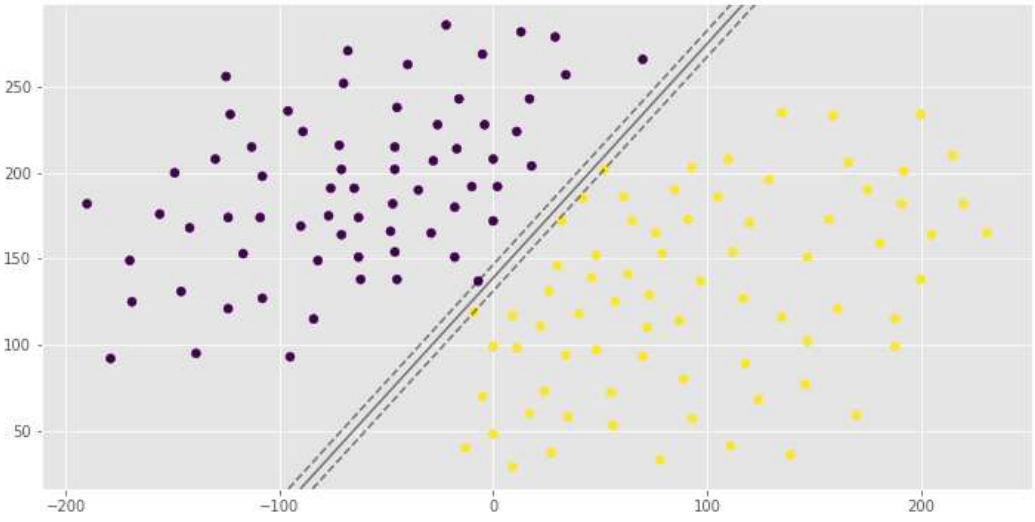
```
from matplotlib import style
from sklearn.svm import SVC
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

# Import Dataset
data = pd.read_csv('data.csv', header=None)
X = data.values[:, :2]
y = data.values[:, 2]

# A function to draw hyperplane and the margin of SVM classifier
def draw_svm(X, y, C=1.0):
    # Plotting the Points
    plt.scatter(X[:,0], X[:,1], c=y)

    # The SVM Model with given C parameter
    clf = SVC(kernel='linear', C=C)
    clf_fit = clf.fit(X, y)

    # Limit of the axes
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Creating the meshgrid
    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)
     # Plotting the boundary
```

```
    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                    alpha=0.5, linestyles=['--', '-', '--'])
    ax.scatter(clf.support_vectors_[:, 0],
               clf.support_vectors_[:, 1],
               s=100, linewidth=1, facecolors='none')
    plt.show()
    # Returns the classifier
    return clf_fit

clf_arr = []
clf_arr.append(draw_svm(X, y, 0.0001))
clf_arr.append(draw_svm(X, y, 0.001))
clf_arr.append(draw_svm(X, y, 1))
clf_arr.append(draw_svm(X, y, 10))

for i, clf in enumerate(clf_arr):
    # Accuracy Score
    print(clf.score(X, y))
    pred = clf.predict([(12, 32), (-250, 32), (120, 43)])
    print(pred)
```




250
200
150
100
50


250
200
150
100
50
−200
−100
0
100
200


250
200
150
100
50
−200
−100
0
100
200

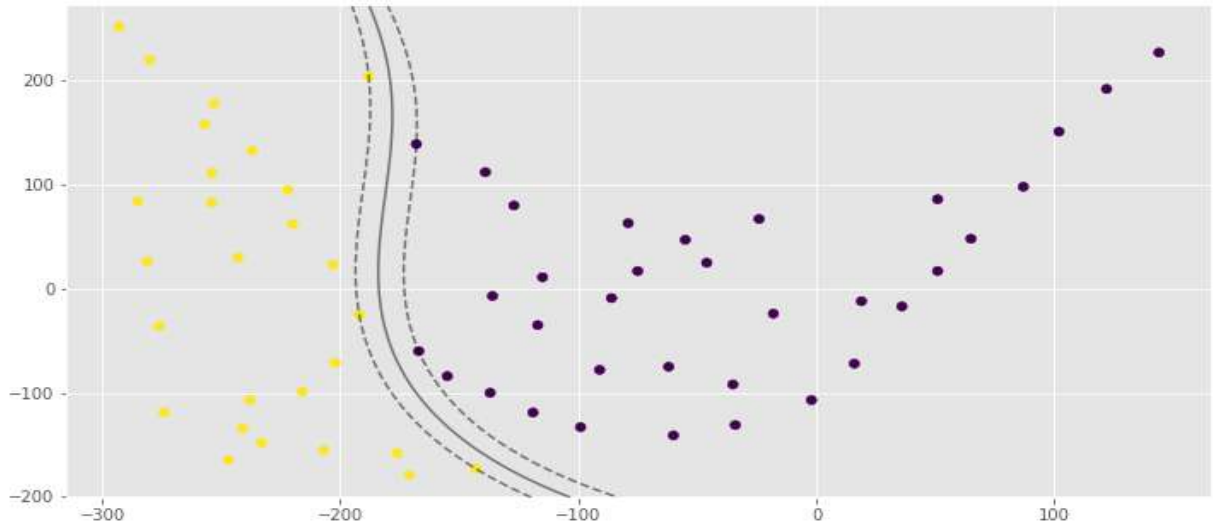
```
0.992907801418
[1 0 1]
0.992907801418
[1 0 1]
1.0
[1 0 1]
1.0
[1 0 1]
```

You can see the same hyperplane with different margin width. It is depends on the C hyperparameter.

Polynomial Kernel

```
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

data = pd.read_csv('polydata2.csv', header=None)
X = data.values[:, :2]
y = data.values[:, 2]

def draw_svm(X, y, C=1.0):
    plt.scatter(X[:,0], X[:,1], c=y)
    clf = SVC(kernel='poly', C=C)
    clf_fit = clf.fit(X, y)

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
```

```
    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                    alpha=0.5, linestyles=['--', '-', '--'])
    ax.scatter(clf.support_vectors_[:, 0],
               clf.support_vectors_[:, 1],
               s=100, linewidth=1, facecolors='none')
    plt.show()
    return clf_fit

clf = draw_svm(X, y)
score = clf.score(X, y)
pred = clf.predict([(-130, 110), (-170, -160), (80, 90), (-280, 20)])
print(score)
print(pred)
```

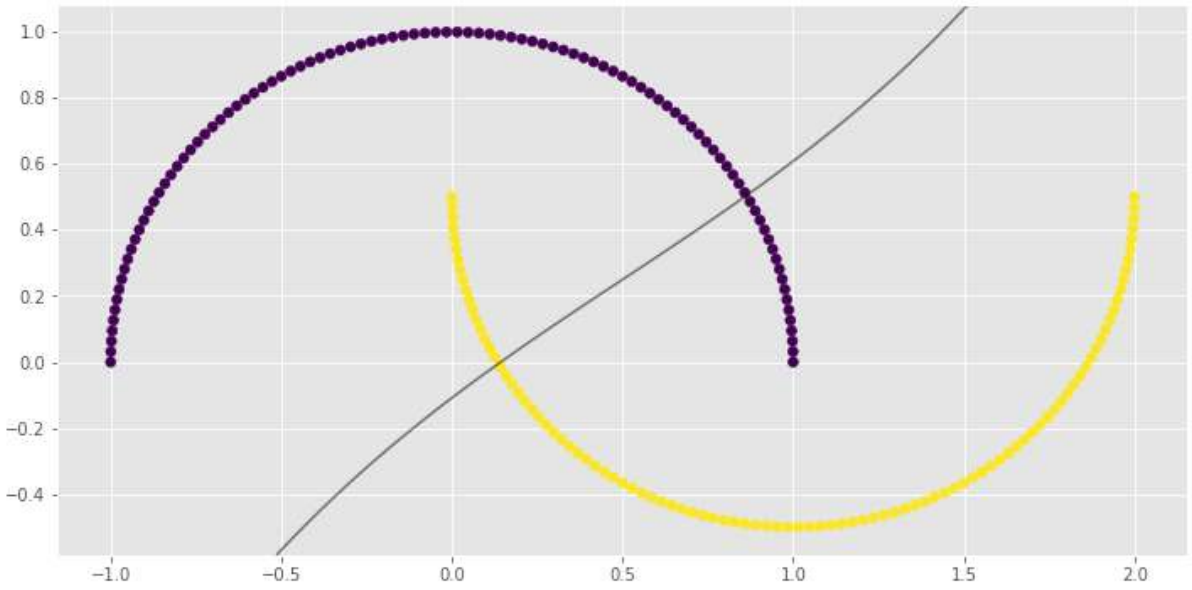


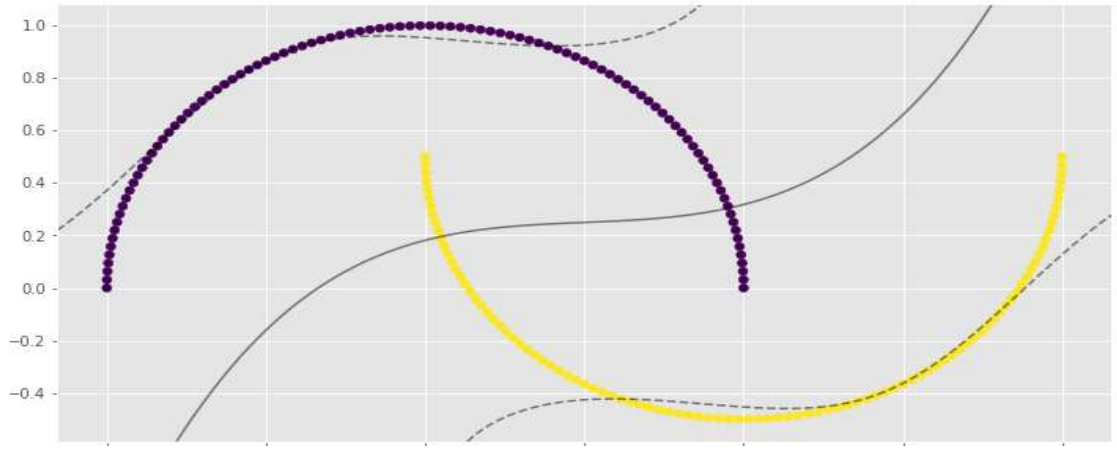
```
1.0
[0 1 0 1]
```

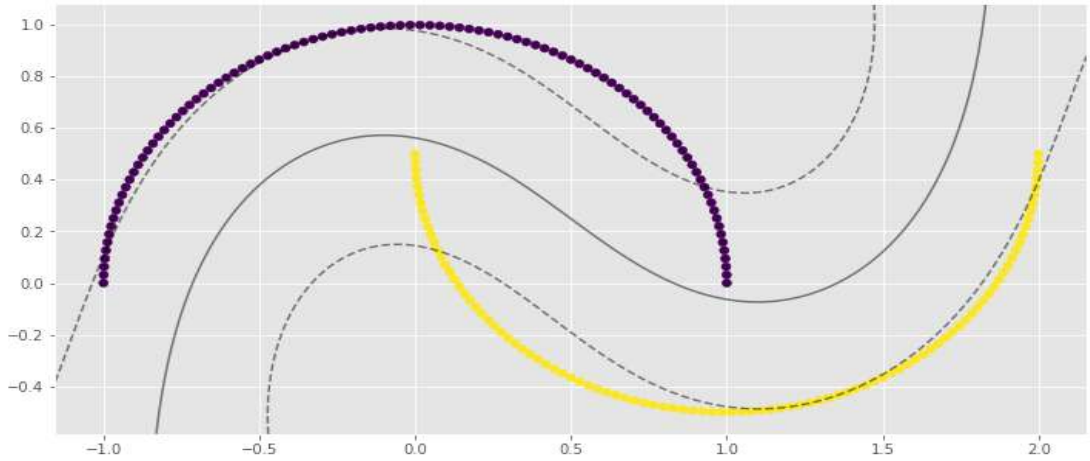
Gaussian Kernel

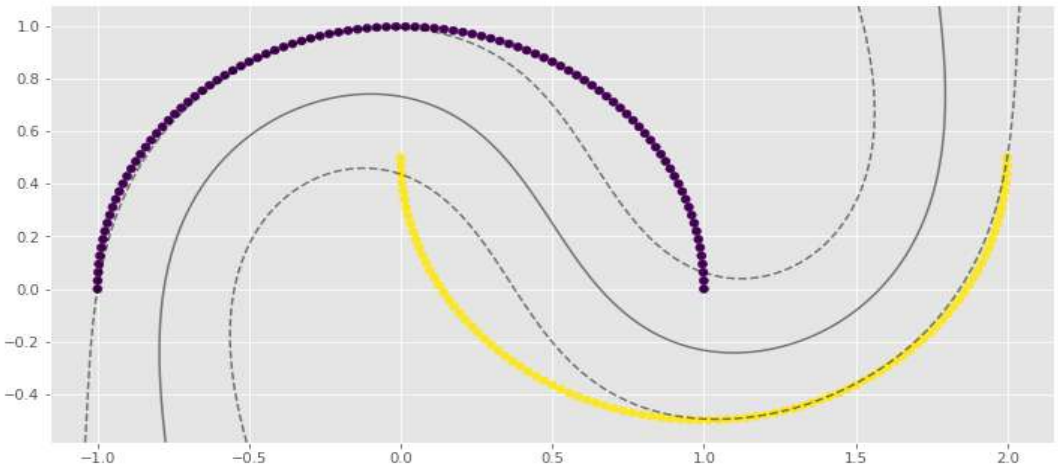
```
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
from sklearn.datasets import make_classification, make_blobs, make_moons
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

X, y = make_moons(n_samples=200)
# Auto gamma equals 1/n_features
def draw_svm(X, y, C=1.0, gamma='auto'):
    plt.scatter(X[:,0], X[:,1], c=y)
    clf = SVC(kernel='rbf', C=C, gamma=gamma)
    clf_fit = clf.fit(X, y)

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                alpha=0.5, linestyles=['--', '-', '--'])
```

```
    ax.scatter(clf.support_vectors_[:, 0],
               clf.support_vectors_[:, 1],
               s=100, linewidth=1, facecolors='none')
    plt.show()
    return clf_fit

clf_arr = []
clf_arr.append(draw_svm(X, y, 0.01))
clf_arr.append(draw_svm(X, y, 0.1))
clf_arr.append(draw_svm(X, y, 1))
clf_arr.append(draw_svm(X, y, 10))

for i, clf in enumerate(clf_arr):
    print(clf.score(X, y))
```




1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

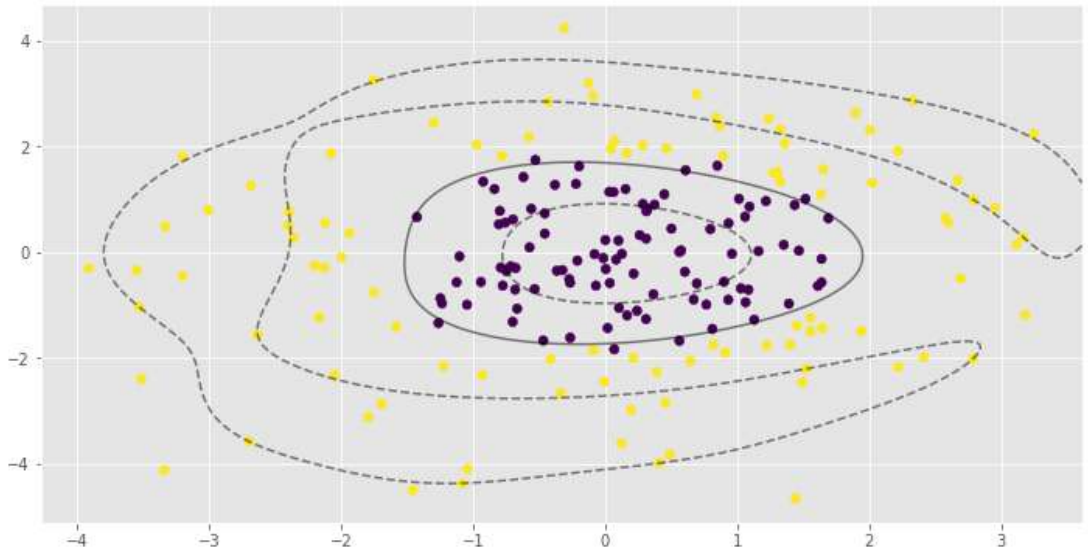

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

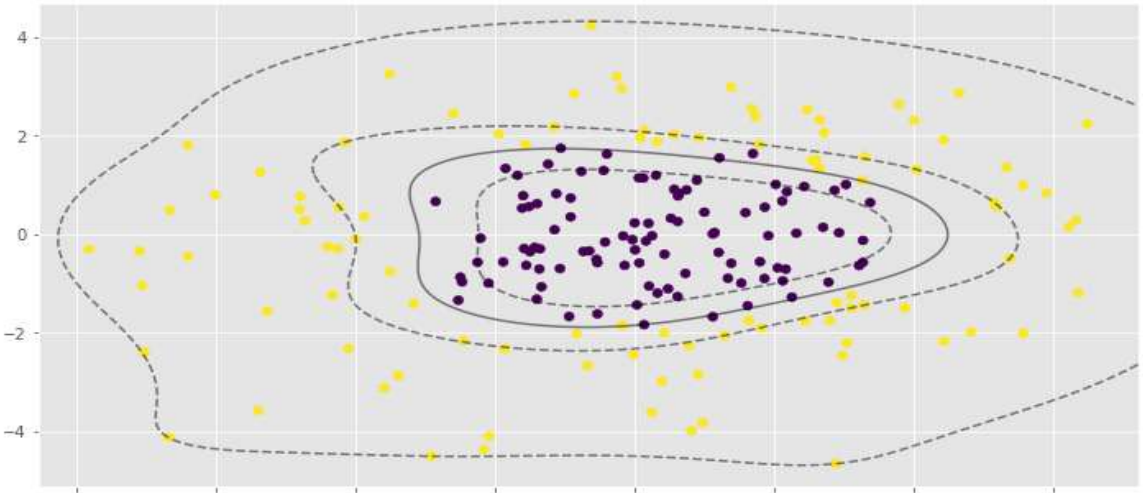
0.83

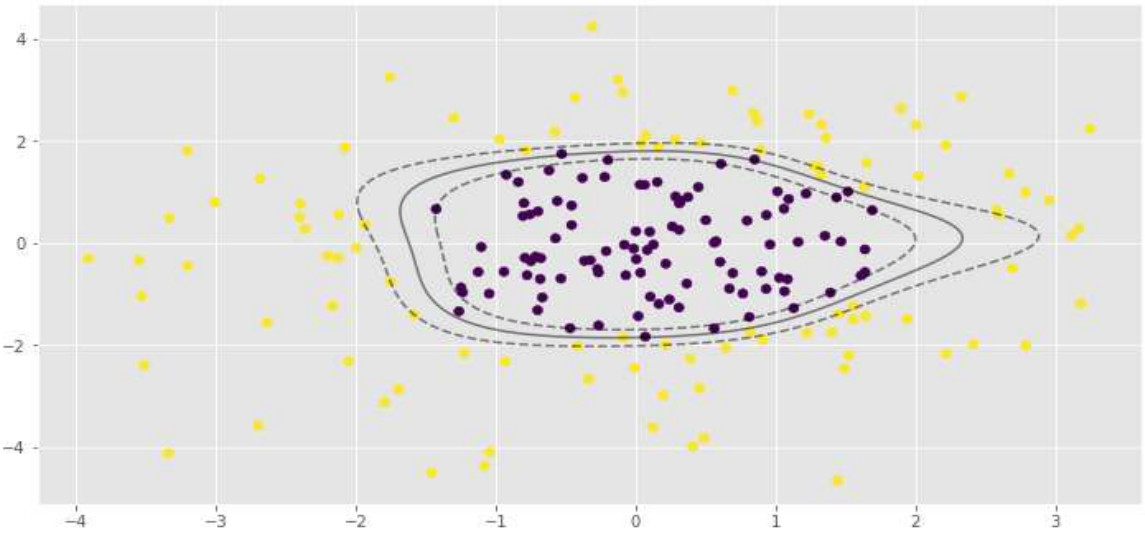
0.9

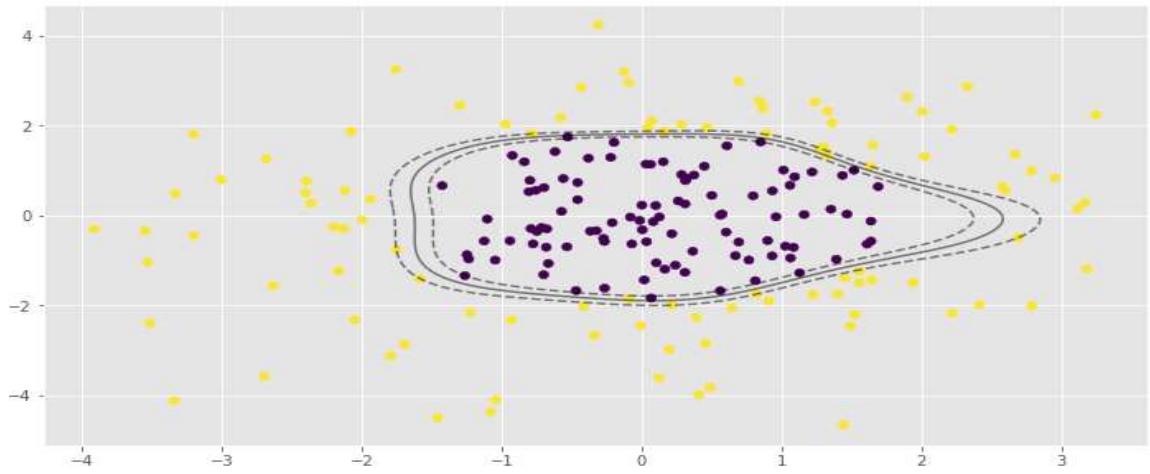
```
1.0
1.0
import numpy as np
import pandas as pd
from matplotlib import style
from sklearn.svm import SVC
from sklearn.datasets import make_gaussian_quantiles
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
style.use('ggplot')

X, y = make_gaussian_quantiles(n_samples=200, n_features=2, n_classes=2, cov=3)

# Auto gamma equals 1/n_features
def draw_svm(X, y, C=1.0, gamma='auto'):
    plt.scatter(X[:,0], X[:,1], c=y)
    clf = SVC(kernel='rbf', C=C, gamma=gamma)
    clf_fit = clf.fit(X, y)

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    xx = np.linspace(xlim[0], xlim[1], 200)
    yy = np.linspace(ylim[0], ylim[1], 200)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1],
                    alpha=0.5, linestyles=['--', '-', '--'])
    ax.scatter(clf.support_vectors_[:, 0],
```

```
            clf.support_vectors_[:, 1],
            s=100, linewidth=1, facecolors='none')
    plt.show()
    return clf_fit

clf_arr = []
clf_arr.append(draw_svm(X, y, 0.1))
clf_arr.append(draw_svm(X, y, 1))
clf_arr.append(draw_svm(X, y, 10))
clf_arr.append(draw_svm(X, y, 100))

for i, clf in enumerate(clf_arr):
    print(clf.score(X, y))
```




4
2
0
−2
−4


4
2
0
−2
−4
−4
−3
−2
−1
0
1
2
3


4
2
0
−2
−4
−4
−3
−2
−1
0
1
2
3

0.965

0.97

0.985

0.995

γ parameter is very important to the RBF SVM model. In the first example low value of γ leads to almost linear classification.

GBM (gradient boosting):-

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. (Wikipedia definition)

The accuracy of a predictive model can be boosted in two ways: Either by embracing feature engineering or by applying boosting algorithms straight away. Having participated in lots of data science competition, I've noticed that people prefer to work with boosting algorithms as it takes less time and produces similar results.

There are multiple boosting algorithms like Gradient Boosting, XGBoost, AdaBoost, Gentle Boost etc. Every algorithm has its own underlying mathematics and a slight variation is observed while applying them. If you are new to this, Great! You shall be learning all these concepts in a week's time from now.

In this article, I've explained the underlying concepts and complexities of Gradient Boosting Algorithm. In addition, I've also shared an example to learn its implementation in R.

While working with boosting algorithms, you'll soon come across two frequently occurring buzzwords: Bagging and Boosting. So, how are they different? Here's a one line explanation:

Bagging: It is an approach where you take random samples of data, build learning algorithms and take simple means to find bagging probabilities.

Boosting: Boosting is similar, however the selection of sample is made more intelligently. We subsequently give more and more weight to hard to classify observations.

Okay! I understand you've questions sprouting up like 'what do you mean by hard? How do I know how much additional weight am I supposed to give to a mis-classified observation.' I shall answer all your questions in subsequent sections. Keep Calm and proceed.

Let's begin with an easy example

Assume, you are given a previous model M to improve on. Currently you observe that the model has an accuracy of 80% (any metric). How do you go further about it?

One simple way is to build an entirely different model using new set of input variables and trying better ensemble learners. On the contrary, I have a much simpler way to suggest. It goes like this:

Y = M(x) + error

What if I am able to see that error is not a white noise but have same correlation with outcome(Y) value. What if we can develop a model on this error term? Like,

error = G(x) + error2

Probably, you'll see error rate will improve to a higher number, say 84%. Let's take another step and regress against error2.

error2 = H(x) + error3

Now we combine all these together :

Y = M(x) + G(x) + H(x) + error3

This probably will have a accuracy of even more than 84%. What if I can find an optimal weights for each of the three learners,

Y = alpha * M(x) + beta * G(x) + gamma * H(x) + error4

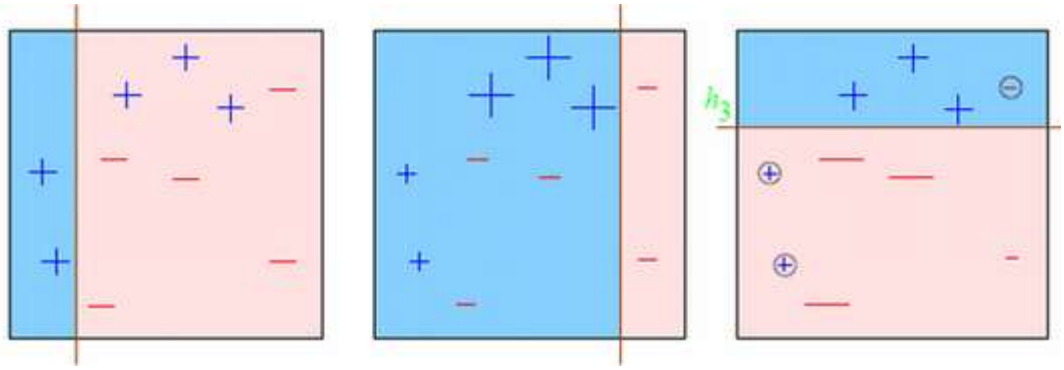
If we found good weights, we probably have made even a better model. This is the underlying principle of a boosting learner. When I read the theory for the first time, I had two quick questions:

1. Do we really see non white noise error in regression/classification equations? If not, how can we even use this algorithm?
2. Wow, if this is possible, why not get near 100% accuracy?

I'll answer these questions in this article, however, in a crisp manner. Boosting is generally done on weak learners, which do not have a capacity to leave behind white noise. Secondly, boosting can lead to overfitting, so we need to stop at the right point.

Let's try to visualize one Classification Problem

Look at the below diagram :



We start with the first box. We see one vertical line which becomes our first week learner. Now in total we have 3/10 mis-classified observations. We now start giving higher weights to 3 plus mis-classified observations. Now, it becomes very important to classify them right. Hence, the vertical line towards right edge. We repeat this process and then combine each of the learner in appropriate weights.

Explaining underlying mathematics

How do we assign weight to observations?

We always start with a uniform distribution assumption. Lets call it as D1 which is 1/n for all n observations.

Step 1 . We assume an alpha(t)

Step 2: Get a weak classifier h(t)

Step 3: Update the population distribution for the next step

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where,

$$Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

Step 4 : Use the new population distribution to again find the next learner

Scared of Step 3 mathematics? Let me break it down for you. Simply look at the argument in exponent. Alpha is kind of learning rate, y is the actual response (+ 1 or -1) and h(x) will be the class predicted by learner. Essentially, if learner is going wrong, the exponent becomes 1*alpha and else -1*alpha. Essentially, the weight will probably increase if the prediction went wrong the last time. So, what's next?

Step 5 : Iterate step 1 – step 4 until no hypothesis is found which can improve further.

Step 6 : Take a weighted average of the frontier using all the learners used till now. But what are the weights? Weights are simply the alpha values. Alpha is calculated as follows:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

How Gradient Boosting Works

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function

The loss function used depends on the type of problem being solved.

It must be differentiable, but many standard loss functions are supported and you can define your own.

For example, regression may use a squared error and classification may use logarithmic loss.

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

2. Weak Learner

Decision trees are used as the weak learner in gradient boosting.

Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions.

Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.

Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.

It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes.

This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

3. Additive Model

Trees are added one at a time, and existing trees in the model are not changed.

A gradient descent procedure is used to minimize the loss when adding trees.

Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.

Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss.

Generally this approach is called functional gradient descent or gradient descent with functions.

One way to produce a weighted combination of classifiers which optimizes [the cost] is by gradient descent in function space

— Boosting Algorithms as Gradient Descent in Function Space [PDF], 1999

The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.

A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

Improvements to Basic Gradient Boosting

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly.

It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.

In this this section we will look at 4 enhancements to basic gradient boosting:

1. Tree Constraints
2. Shrinkage
3. Random sampling
4. Penalized Learning

1. Tree Constraints

It is important that the weak learners have skill but remain weak.

There are a number of ways that the trees can be constrained.

A good general heuristic is that the more constrained tree creation is, the more trees you will need in the model, and the reverse, where less constrained individual trees, the fewer trees that will be required.

Below are some constraints that can be imposed on the construction of decision trees:

- **Number of trees**, generally adding more trees to the model can be very slow to overfit. The advice is to keep adding trees until no further improvement is observed.
- **Tree depth**, deeper trees are more complex trees and shorter trees are preferred. Generally, better results are seen with 4-8 levels.
- **Number of nodes or number of leaves**, like depth, this can constrain the size of the tree, but is not constrained to a symmetrical structure if other constraints are used.
- **Number of observations per split** imposes a minimum constraint on the amount of training data at a training node before a split can be considered
- **Minimim improvement to loss** is a constraint on the improvement of any split added to a tree.

2. Weighted Updates

The predictions of each tree are added together sequentially.

The contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. This weighting is called a shrinkage or a learning rate.

Each update is simply scaled by the value of the "learning rate parameter v"

— Greedy Function Approximation: A Gradient Boosting Machine [PDF], 1999

The effect is that learning is slowed down, in turn require more trees to be added to the model, in turn taking longer to train, providing a configuration trade-off between the number of trees and learning rate.

Decreasing the value of v [the learning rate] increases the best value for M [the number of trees].

— Greedy Function Approximation: A Gradient Boosting Machine [PDF], 1999

It is common to have small values in the range of 0.1 to 0.3, as well as values less than 0.1.

Similar to a learning rate in stochastic optimization, shrinkage reduces the influence of each individual tree and leaves space for future trees to improve the model.

— Stochastic Gradient Boosting [PDF], 1999

3. Stochastic Gradient Boosting

A big insight into bagging ensembles and random forest was allowing trees to be greedily created from subsamples of the training dataset.

This same benefit can be used to reduce the correlation between the trees in the sequence in gradient boosting models.

This variation of boosting is called stochastic gradient boosting.

at each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner.

— Stochastic Gradient Boosting [PDF], 1999

A few variants of stochastic boosting that can be used:

- Subsample rows before creating each tree.
- Subsample columns before creating each tree
- Subsample columns before considering each split.

Generally, aggressive sub-sampling such as selecting only 50% of the data has shown to be beneficial.

According to user feedback, using column sub-sampling prevents over-fitting even more so than the traditional row sub-sampling

— XGBoost: A Scalable Tree Boosting System, 2016

4. Penalized Gradient Boosting

Additional constraints can be imposed on the parameterized trees in addition to their structure.

Classical decision trees like CART are not used as weak learners, instead a modified form called a regression tree is used that has numeric values in the leaf nodes (also called terminal nodes). The values in the leaves of the trees can be called weights in some literature.

As such, the leaf weight values of the trees can be regularized using popular regularization functions, such as:

- L1 regularization of weights.
- L2 regularization of weights.

The additional regularization term helps to smooth the final learnt weights to avoid over-fitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

Un supervised Machine Learning Algorithms

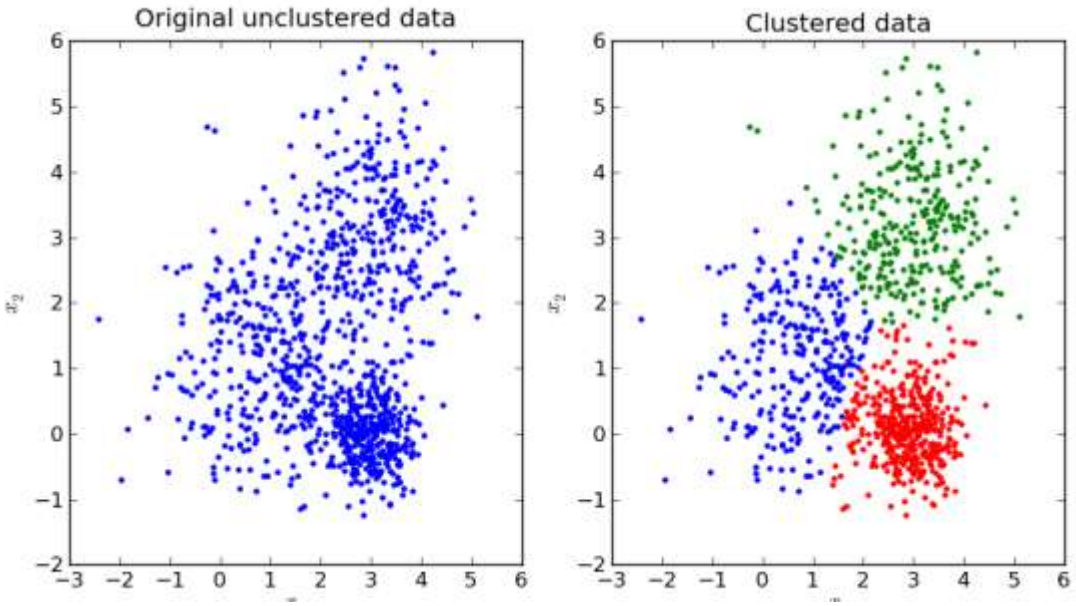
Clustering Models:

- K-Means :-

Clustering is a type of **Unsupervised learning**. This is very often used when you don't have labeled data. **K-Means Clustering** is one of the popular clustering algorithm. The goal of this algorithm is to find groups(clusters) in the given data.We will implement K-Means algorithm using Python from scratch.

K-Means Clustering

K-Means is a very simple algorithm which clusters the data into K number of clusters. The following image from PyPR is an example of K-Means Clustering.



Use Cases

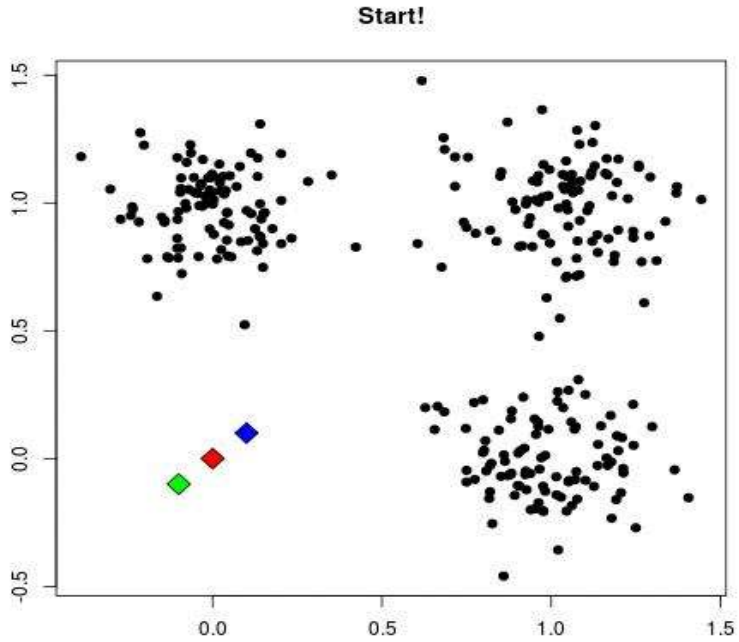
K-Means is widely used for many applications.

- Image Segmentation
- Clustering Gene Segementation Data
- News Article Clustering
- Clustering Languages
- Species Clustering
- Anomaly Detection

Algorithm

Our algorithm works as follows, assuming we have inputs x1,x2,x3,…,xn and value of **K**

- **Step 1** - Pick K random points as cluster centers called centroids.
- **Step 2** - Assign each x_i to nearest cluster by calculating its distance to each centroid.
- **Step 3** - Find new cluster center by taking the average of the assigned points.
- **Step 4** - Repeat Step 2 and 3 until none of the cluster assignments change.



The above animation is an example of running K-Means Clustering on a two dimensional data.

Step 1

We randomly pick **K** cluster centers(centroids). Let's assume these are $c_1, c_2, \dots c_k$ and we can say that;

$$C = c_1, c_2, \dots c_k$$

C is the set of all centroids.

Step 2

In this step we assign each input value to closest center. This is done by calculating Euclidean(L2) distance between the point and the each centroid.

$$\arg\min_{c_i \in C} dist(c_i, x)^2$$

Where is dist(.) the Euclidean distance.

Step 3

In this step, we find the new centroid by taking the average of all the points assigned to that cluster.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

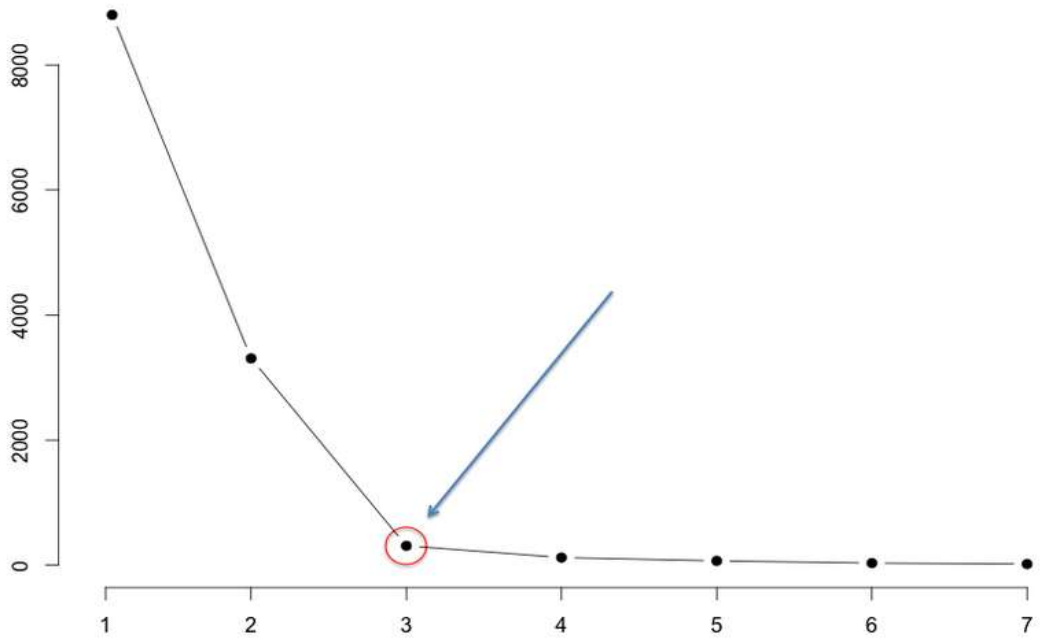
S_i is the set of all points assigned to the i^{th} cluster.

Step 4

In this step, we repeat step 2 and 3 until none of the cluster assignments change. That means until our clusters remain stable, we repeat the algorithm.

Choosing the Value of K

We often know the value of K. In that case we use the value of K. Else we use the **Elbow Method**.



We run the algorithm for different values of **K**(say K = 10 to 1) and plot the K values against SSE(Sum of Squared Errors). And select the value of K for the elbow point as shown in the figure.

Implementation using Python

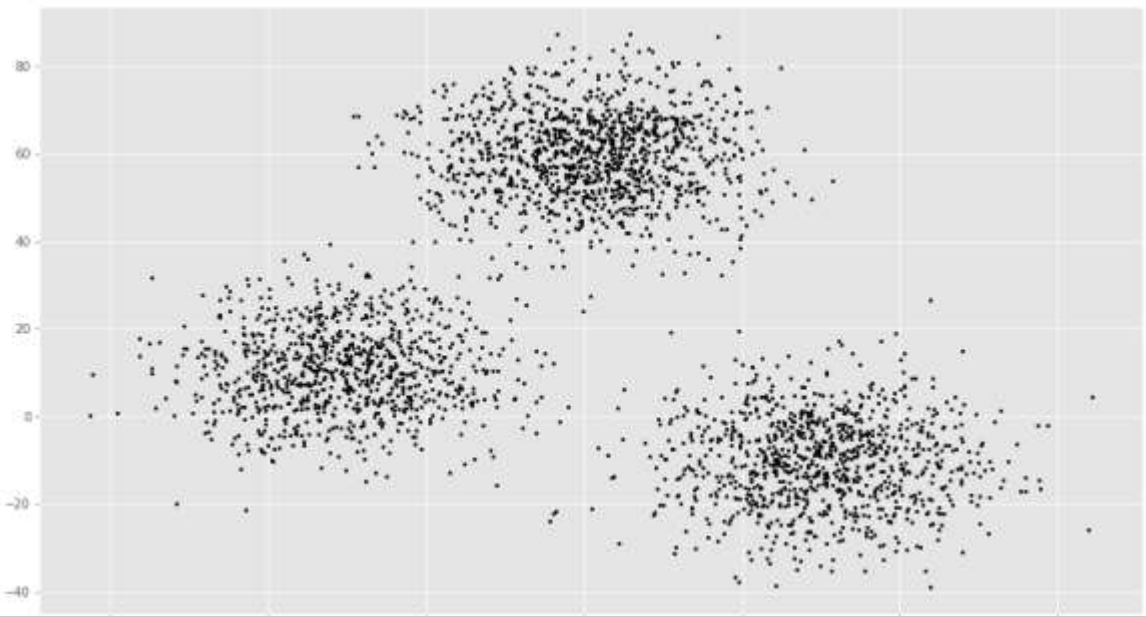
The dataset we are gonna use has 3000 entries with 3 clusters. So we already know the value of K.

We will start by importing the dataset.

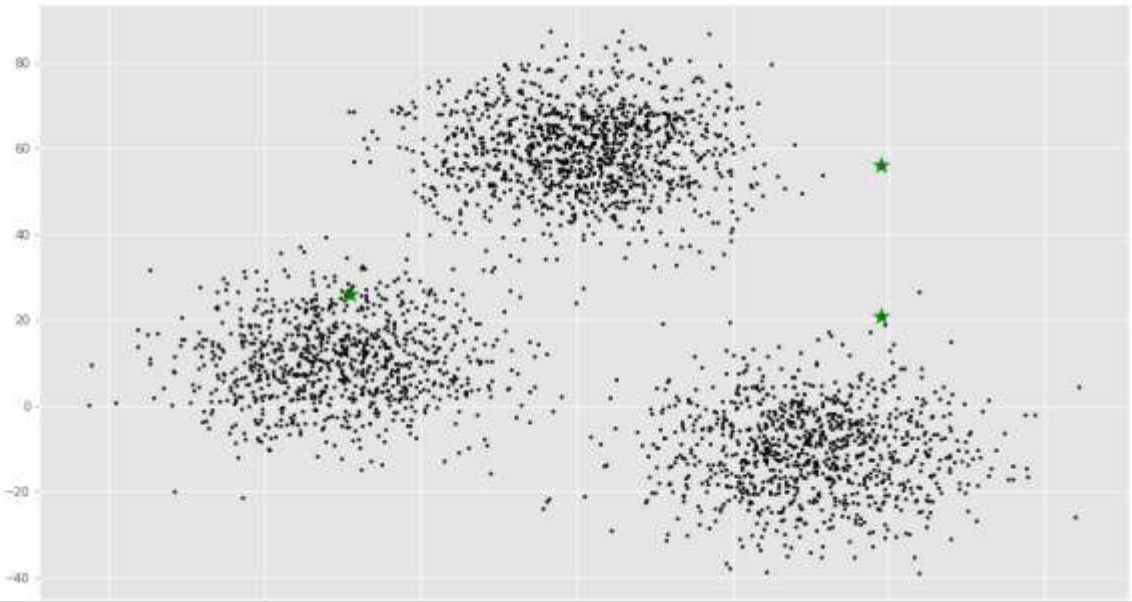
```
%matplotlib inline
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
# Importing the dataset
data = pd.read_csv('xclara.csv')
print(data.shape)
data.head()
(3000, 2)
```

	V1	V2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729


```
# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)
```




```
# Euclidean Distance Caculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)
# Number of clusters
k = 3
# X coordinates of random centroids
C_x = np.random.randint(0, np.max(X)-20, size=k)
# Y coordinates of random centroids
C_y = np.random.randint(0, np.max(X)-20, size=k)
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print(C)
[[ 11.  26.]
 [ 79.  56.]
 [ 79.  21.]]
# Plotting along with the Centroids
plt.scatter(f1, f2, c='#050505', s=7)
plt.scatter(C_x, C_y, marker='*', s=200, c='g')
```

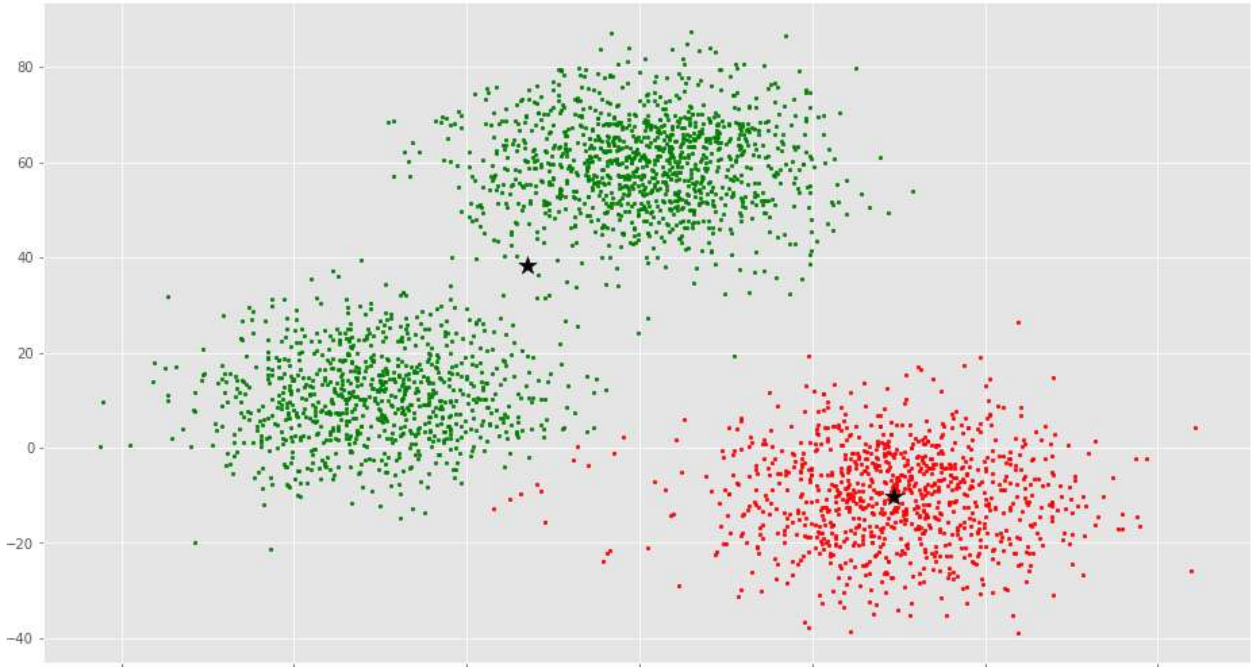


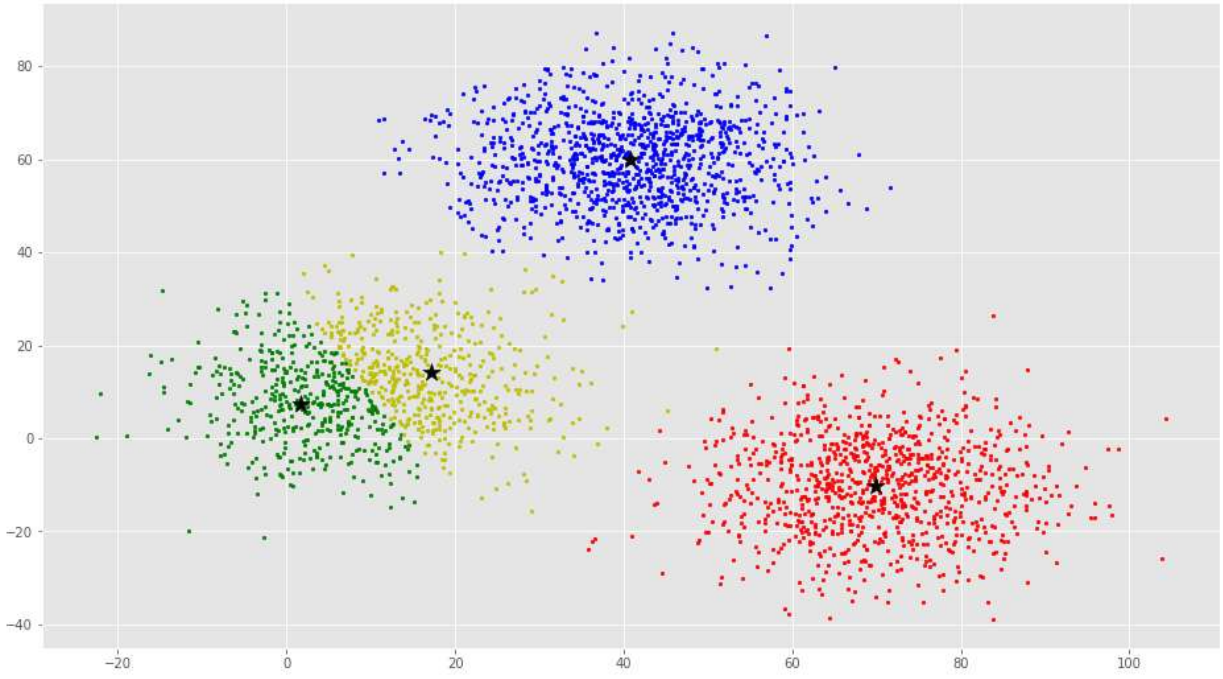

```
# To store the value of centroids when it updates
C_old = np.zeros(C.shape)
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))
# Error func. - Distance between new centroids and old centroids
error = dist(C, C_old, None)
# Loop will run till the error becomes zero
while error != 0:
    # Assigning each value to its closest cluster
    for i in range(len(X)):
        distances = dist(X[i], C)
        cluster = np.argmin(distances)
        clusters[i] = cluster
    # Storing the old centroid values
    C_old = deepcopy(C)
    # Finding the new centroids by taking the average value
    for i in range(k):
        points = [X[j] for j in range(len(X)) if clusters[j] == i]
        C[i] = np.mean(points, axis=0)
    error = dist(C, C_old, None)
```

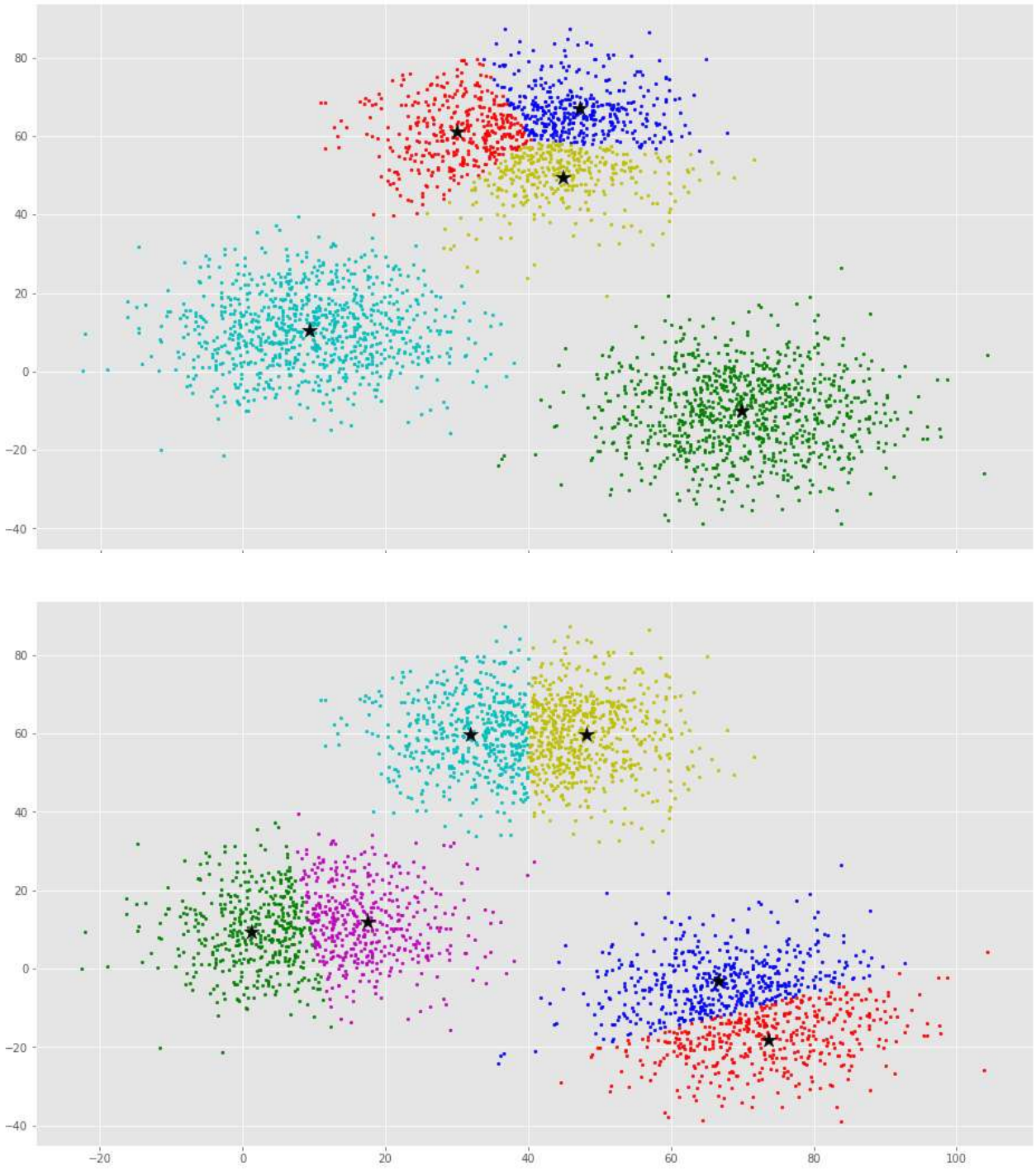
```
colors = ['r', 'g', 'b', 'y', 'c', 'm']
fig, ax = plt.subplots()
for i in range(k):
    points = np.array([X[j] for j in range(len(X)) if clusters[j] == i])
    ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='#050505')
```

From this visualization it is clear that there are 3 clusters with black stars as their centroid.

If you run K-Means with wrong values of K, you will get completely misleading clusters. For example, if you run K-Means on this with values 2, 4, 5 and 6, you will get the following clusters.


80
60
40
20
0
−20
−40


80
60
40
20
0
−20
−40
−20
0
20
40
60
80
100



Now we will see how to implement K-Means Clustering using **scikit-learn**

The scikit-learn approach

Example 1

We will use the same dataset in this example.

```
from sklearn.cluster import KMeans


# Number of clusters
kmeans = KMeans(n_clusters=3)
# Fitting the input data
kmeans = kmeans.fit(X)
# Getting the cluster labels
labels = kmeans.predict(X)
# Centroid values
centroids = kmeans.cluster_centers_
# Comparing with scikit-learn centroids
print(C) # From Scratch
print(centroids) # From sci-kit learn
[[  9.47804546  10.68605232]
 [ 40.68362808  59.71589279]
 [ 69.92418671 -10.1196413 ]]
[[  9.4780459   10.686052  ]
 [ 69.92418447 -10.11964119]
 [ 40.68362784  59.71589274]]
```

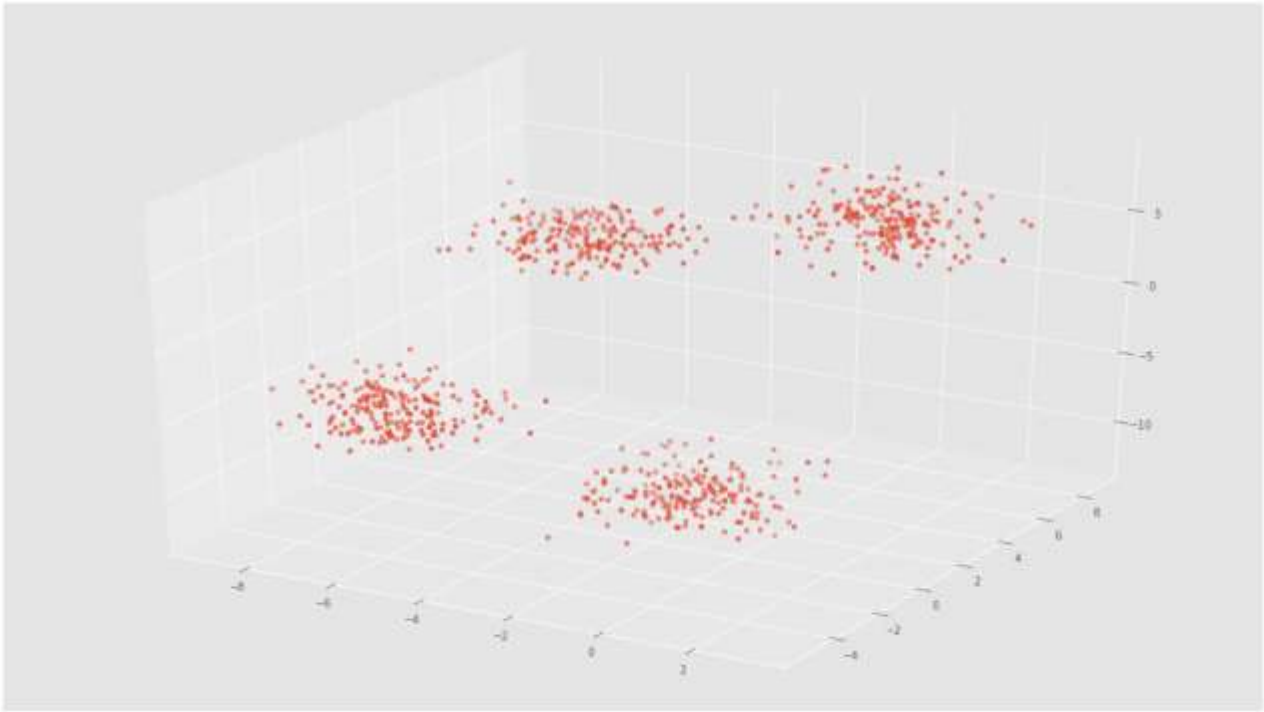
You can see that the centroid values are equal, but in different order.

Example 2

We will generate a new dataset using make_blobs function.

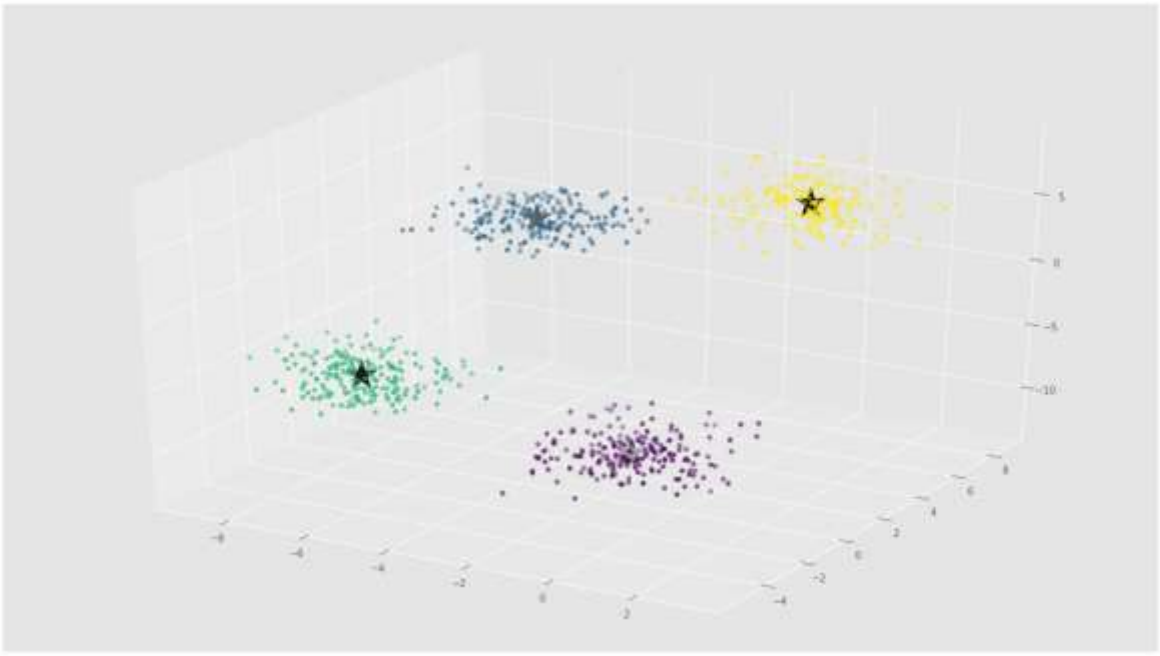
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
```

```
from sklearn.datasets import make_blobs


plt.rcParams['figure.figsize'] = (16, 9)


# Creating a sample dataset with 4 clusters
X, y = make_blobs(n_samples=800, n_features=3, centers=4)
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2])
```



```
# Initializing KMeans
kmeans = KMeans(n_clusters=4)
# Fitting with inputs
kmeans = kmeans.fit(X)
# Predicting the clusters
labels = kmeans.predict(X)
# Getting the cluster centers
```

```
C = kmeans.cluster_centers_
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y)
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='*', c='#050505', s=1000)
```



In the above image, you can see 4 clusters and their centroids as stars. scikit-learn approach is very simple and concise.

K Means Clustering Algorithm

K-means is a popularly used unsupervised machine learning algorithm for cluster analysis. K-Means is a non-deterministic and iterative method. The algorithm operates on a given data set through pre-defined number of clusters, k. The output of K Means algorithm is k clusters with input data partitioned among the clusters.

For instance, let's consider K-Means Clustering for Wikipedia Search results. The search term "Jaguar" on Wikipedia will return all pages containing the word Jaguar which can refer to Jaguar as a Car, Jaguar as Mac OS version and Jaguar as an Animal. K Means clustering algorithm can be applied to group the webpages that talk about similar concepts. So, the algorithm will group all web pages that talk about Jaguar as an Animal into one cluster, Jaguar as a Car into another cluster and so on.

Advantages of using K-Means Clustering Machine Learning Algorithm

- In case of globular clusters, K-Means produces tighter clusters than hierarchical clustering.

- Given a smaller value of K, K-Means clustering computes faster than hierarchical clustering for large number of variables.

CLICK HERE to get the 2017 data scientist salary report delivered to your inbox!

Applications of K-Means Clustering

K Means Clustering algorithm is used by most of the search engines like Yahoo, Google to cluster web pages by similarity and identify the 'relevance rate' of search results. This helps search engines reduce the computational time for the users.

Data Science Libraries in Python to implement K-Means Clustering – SciPy, Sci-Kit Learn, Python Wrapper

Hierarchical Clustering:-

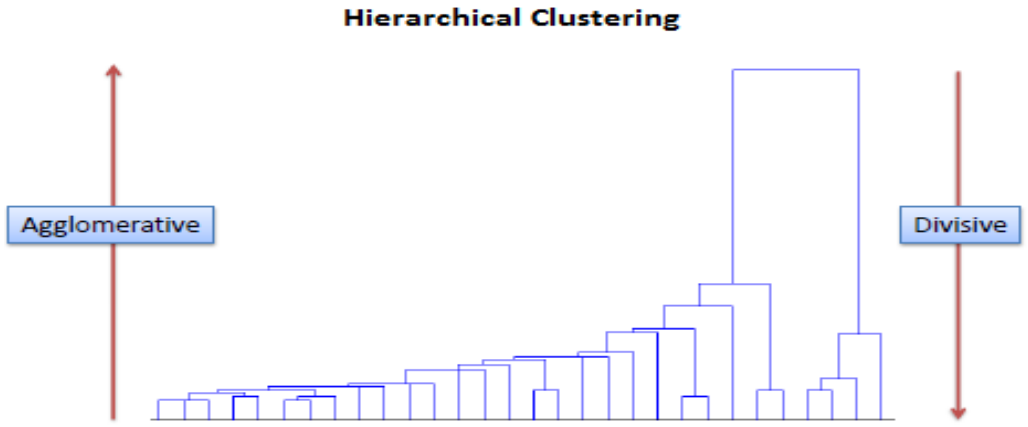
In data mining and statistics, **hierarchical clustering** (also called **hierarchical cluster analysis** or **HCA**) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In **hierarchical clustering**, the two most similar **clusters** are combined and continue to combine until all objects are in the same **cluster**. **Hierarchical clustering** produces a tree (called a dendogram) that shows the **hierarchy** of the **clusters**. ... **Hierarchical clustering** is considered an **unsupervised clustering** method.

Hierarchical clustering (or hierarchic clustering) outputs a hierarchy, a structure that is more informative than the unstructured set of clusters returned by flat clustering. Hierarchical clustering does not require us to prespecify the number of clusters and most hierarchical algorithms that have been used in IR are deterministic. These advantages of hierarchical clustering come at the cost of lower efficiency. The most common hierarchical clustering algorithms have a complexity that is at least quadratic in the number of documents compared to the linear complexity of K-means and EM

Hierarchical clustering involves creating clusters that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a hierarchy. There are two types of hierarchical clustering, Divisive and Agglomerative.



Divisive method

In divisive or top-down clustering method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation. There is evidence that divisive algorithms produce more accurate hierarchies than agglomerative algorithms in some circumstances but is conceptually more complex.

Agglomerative method

In agglomerative or bottom-up clustering method we assign each observation to its own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters. Finally, repeat steps 2 and 3 until there is only a single cluster left. The related algorithm is shown below.

Given:
A set X of objects $\{x_1,\ldots,x_n\}$
A distance function $dist(c_1,c_2)$
for i = 1 to n
  $c_i = \{x_i\}$
end for
$C = \{c_1,\ldots,c_n\}$
$l = n+1$
while C.size > 1 **do**
- (c_{min1},c_{min2}) = minimum $dist(c_i,c_j)$ for all c_i,c_j in C
- remove c_{min1} and c_{min2} from C
- add $\{c_{min1},c_{min2}\}$ to C
- $l = l + 1$

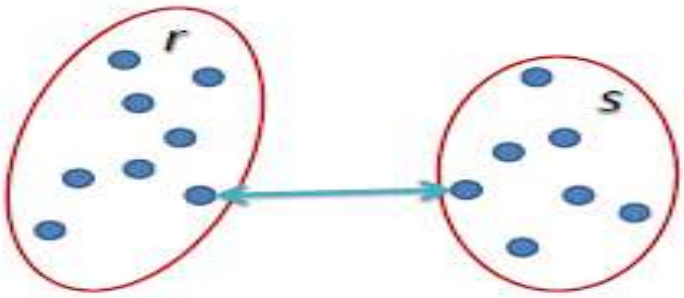
end while

Before any clustering is performed, it is required to determine the proximity matrix containing the distance between each point using a distance function. Then, the matrix is

updated to display the distance between each cluster. The following three methods differ in how the distance between each cluster is measured.

Single Linkage

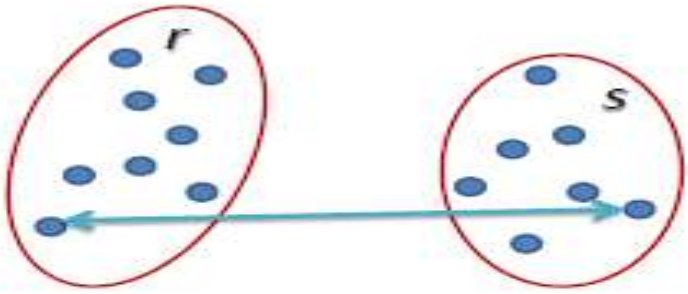
In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster. For example, the distance between clusters "r" and "s" to the left is equal to the length of the arrow between their two closest points.



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete Linkage

In complete linkage hierarchical clustering, the distance between two clusters is defined as the longest distance between two points in each cluster. For example, the distance between clusters "r" and "s" to the left is equal to the length of the arrow between their two furthest points.



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

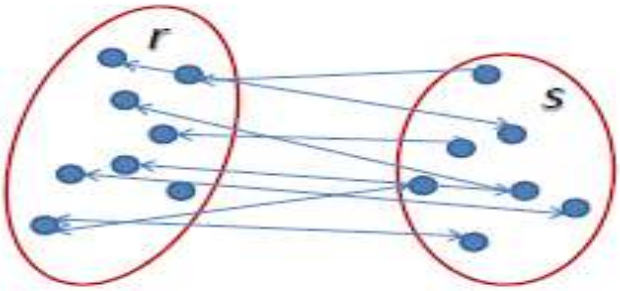
Average Linkage

In average linkage hierarchical clustering, the distance between two clusters is defined

as the average distance between each point in one cluster to every point in the other cluster.

For example, the distance between clusters “r” and “s” to the left is equal to the average length

each arrow between connecting the points of one cluster to the other.



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

TensorFlow

TensorFlow™ is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

This describes how to use machine learning to categorize Iris flowers by species. It uses TensorFlow's eager execution to (1) build a model, (2) train the model on example data, and (3) use the model to make predictions on unknown data. Machine learning experience isn't required to follow this guide, but you'll need to read some Python code.

TensorFlow programming

There many TensorFlow APIs available, but we recommend starting with these high-level TensorFlow concepts:

- Enable an eager execution development environment,
- Import data with the Datasets API,
- Build models and layers with TensorFlow's Keras API.

This shows these APIs and is structured like many other TensorFlow programs:

1. Import and parse the data sets.
2. Select the type of model.
3. Train the model.
4. Evaluate the model's effectiveness.
5. Use the trained model to make predictions.

The TensorFlow programming stack

As the following illustration shows, TensorFlow provides a programming stack consisting of multiple API layers:

Run the notebook

This is available as an interactive Colab notebook for you to run and change the Python code directly in the browser. The notebook handles setup and dependencies while you "play" cells to execute the code blocks. This is a fun way to explore the program and test ideas. If you are unfamiliar with Python notebook environments, there are a couple of things to keep in mind:

1. Executing code requires connecting to a runtime environment. In the Colab notebook menu, select Runtime > Connect to runtime...
2. Notebook cells are arranged sequentially to gradually build the program. Typically, later code cells depend on prior code cells, though you can always rerun a code block. To execute the entire notebook in order, select Runtime > Run all. To rerun a code cell, select the cell and click the play icon on the left.

Setup program

Install the latest version of TensorFlow

This uses eager execution, which is available in TensorFlow 1.8. (You may need to restart the runtime after upgrading.)

```
!pip install -q --upgrade tensorflow
```

Configure imports and eager execution

Import the required Python modules, including TensorFlow, and enable eager execution for this program. Eager execution makes TensorFlow evaluate operations immediately, returning concrete values instead of creating a computational graph that is executed later. If you are used to a REPL or the `python` interactive console, you'll feel at home.

Once eager execution is enabled, it cannot be disabled within the same program. See the eager execution guidefor more details.

```
from __future__ import absolute_import, division, print_function

import os
import matplotlib.pyplot as plt

import tensorflow as tf
import tensorflow.contrib.eager as tfe

tf.enable_eager_execution()

print("TensorFlow version: {}".format(tf.VERSION))
print("Eager execution: {}".format(tf.executing_eagerly()))
```

```
/usr/local/lib/python3.5/dist-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.

 from ._conv import register_converters as _register_converters

TensorFlow version: 1.8.0

Eager execution: True
```

The Iris classification problem

Imagine you are a botanist seeking an automated way to categorize each Iris flower you find. Machine learning provides many algorithms to statistically classify flowers. For instance, a sophisticated machine learning program could classify flowers based on photographs. Our ambitions are more modest—we're going to classify Iris flowers based on the length and width measurements of their sepals and petals.

The Iris genus entails about 300 species, but our program will classify only the following three:

- Iris setosa

- Iris virginica
- Iris versicolor

Fortunately, someone has already created a data set of 120 Iris flowers with the sepal and petal measurements. This is a classic dataset that is popular for beginner machine learning classification problems.

Figure 1. Iris setosa (by Radomil, CC BY-SA 3.0), Iris versicolor, (by Dlanglois, CC BY-SA 3.0), and Iris virginica (by Frank Mayfield, CC BY-SA 2.0).

Import and parse the training dataset

We need to download the dataset file and convert it to a structure that can be used by this Python program.

Download the dataset

Download the training dataset file using the tf.keras.utils.get_file function. This returns the file path of the downloaded file.

```
train_dataset_url = "http://download.tensorflow.org/data/iris_training.csv"

train_dataset_fp = tf.keras.utils.get_file(fname=os.path.basename(train_dataset_url),
                                           origin=train_dataset_url)

print("Local copy of the dataset file: {}".format(train_dataset_fp))
```

```
Downloading data from http://download.tensorflow.org/data/iris_training.csv

8192/2194 [==============================================] - 0s 0us/step

Local copy of the dataset file: /root/.keras/datasets/iris_training.csv
```

Inspect the data

This dataset, iris_training.csv, is a plain text file that stores tabular data formatted as comma-separated values (CSV). Use the head -n5 command to take a peak at the first five entries:

```
!head -n5 {train_dataset_fp}
120,4,setosa,versicolor,virginica

6.4,2.8,5.6,2.2,2

5.0,2.3,3.3,1.0,1

4.9,2.5,4.5,1.7,2

4.9,3.1,1.5,0.1,0
```

From this view of the dataset, we see the following:

1. The first line is a header containing information about the dataset:

- There are 120 total examples. Each example has four features and one of three possible label names.

2. Subsequent rows are data records, one example per line, where:

- The first four fields are features: these are characteristics of an example. Here, the fields hold float numbers representing flower measurements.
- The last column is the label: this is the value we want to predict. For this dataset, it's an integer value of 0, 1, or 2 that corresponds to a flower name.

Each label is associated with string name (for example, "setosa"), but machine learning typically relies on numeric values. The label numbers are mapped to a named representation, such as:

- 0: Iris setosa
- 1: Iris versicolor
- 2: Iris virginica

For more information about features and labels, see the ML Terminology section of the Machine Learning Crash Course.

Parse the dataset

Since our dataset is a CSV-formatted text file, we'll parse the feature and label values into a format our Python model can use. Each line—or row—in the file is passed to the parse_csv function which grabs the first four feature fields and combines them into a single tensor. Then, the last field is parsed as the label. The function returns both the features and label tensors:

```
def parse_csv(line):
  example_defaults = [[0.], [0.], [0.], [0.], [0]]  # sets field types
  parsed_line = tf.decode_csv(line, example_defaults)
  # First 4 fields are features, combine into single tensor
  features = tf.reshape(parsed_line[:-1], shape=(4,))
  # Last field is the label
  label = tf.reshape(parsed_line[-1], shape=())
  return features, label
```

Create the training tf.data.Dataset

TensorFlow's Dataset API handles many common cases for feeding data into a model. This is a high-level API for reading data and transforming it into a form used for training. See the Datasets Quick Start guide for more information.

This program uses tf.data.TextLineDataset to load a CSV-formatted text file and is parsed with our parse_csvfunction. A tf.data.Dataset represents an input pipeline as a collection of elements and a series of transformations that act on those elements. Transformation methods are chained together or called sequentially—just make sure to keep a reference to the returned Dataset object.

Training works best if the examples are in random order. Use tf.data.Dataset.shuffle to randomize entries, setting buffer_size to a value larger than the number of examples (120 in this case). To train the model faster, the dataset's batch size is set to 32 examples to train at once.

```
train_dataset = tf.data.TextLineDataset(train_dataset_fp)
train_dataset = train_dataset.skip(1)             # skip the first header row
train_dataset = train_dataset.map(parse_csv)      # parse each row
train_dataset = train_dataset.shuffle(buffer_size=1000)  # randomize
train_dataset = train_dataset.batch(32)

# View a single example entry from a batch
features, label = iter(train_dataset).next()
print("example features:", features[0])
print("example label:", label[0])
```

example features: tf.Tensor([6. 2.7 5.1 1.6], shape=(4,), dtype=float32)

example label: tf.Tensor(1, shape=(), dtype=int32)

Select the type of model

Why model?

A model is the relationship between features and the label. For the Iris classification problem, the model defines the relationship between the sepal and petal measurements and the predicted Iris species. Some simple models can be described with a few lines of algebra, but complex machine learning models have a large number of parameters that are difficult to summarize.

Could you determine the relationship between the four features and the Iris species without using machine learning? That is, could you use traditional programming techniques (for example, a lot of conditional statements) to create a model? Perhaps—if you analyzed the dataset long enough to determine the relationships between petal and sepal measurements to a particular species. And this becomes difficult—maybe impossible—on more complicated datasets. A good machine learning approach determines the model for you. If you feed enough representative examples into the right machine learning model type, the program will figure out the relationships for you.

Select the model

We need to select the kind of model to train. There are many types of models and picking a good one takes experience. This document uses a neural network to solve the Iris classification problem. Neural networks can find complex relationships between features and the label. It is a highly-structured graph, organized into one or more hidden layers. Each hidden layer consists of one or more neurons. There are several categories of neural networks and this program uses a dense, or fully-connected neural network: the neurons in one layer receive input connections from every neuron in the previous layer. For example, Figure 2 illustrates a dense neural network consisting of an input layer, two hidden layers, and an output layer:

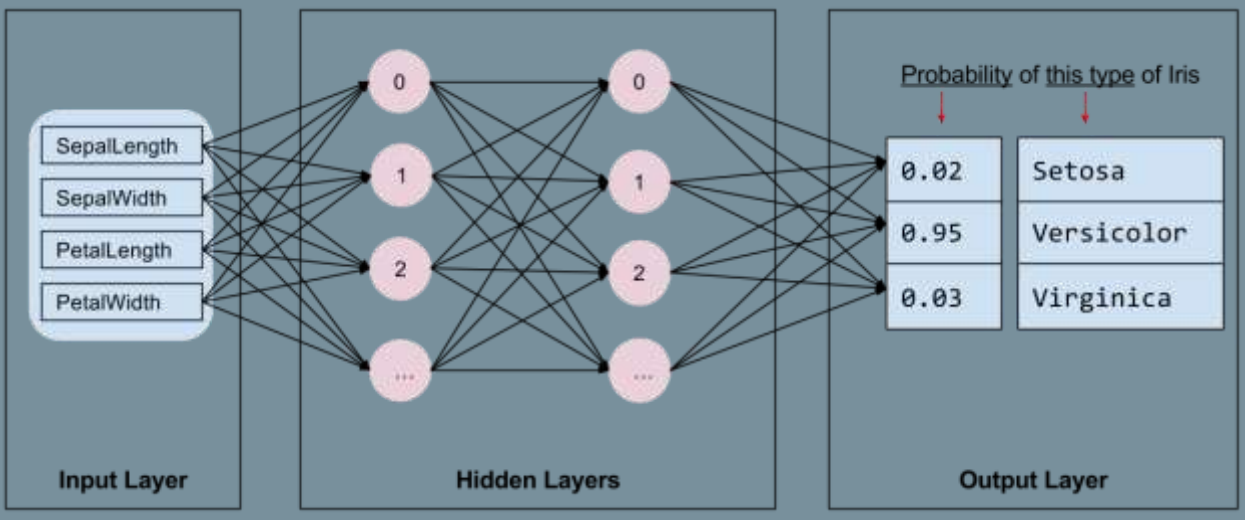


Figure 2. A neural network with features, hidden layers, and predictions.

When the model from Figure 2 is trained and fed an unlabeled example, it yields three predictions: the likelihood that this flower is the given Iris species. This prediction is called inference. For this example, the sum of the output predictions are 1.0. In Figure 2, this prediction breaks down as: 0.03 for Iris setosa, 0.95 for Iris versicolor, and 0.02 for Iris virginica. This means that the model predicts—with 95% probability—that an unlabeled example flower is an Iris versicolor.

Create a model using Keras

The TensorFlow tf.keras API is the preferred way to create models and layers. This makes it easy to build models and experiment while Keras handles the complexity of connecting everything together. See the Keras documentation for details.

The tf.keras.Sequential model is a linear stack of layers. Its constructor takes a list of layer instances, in this case, two Dense layers with 10 nodes each, and an output layer with 3 nodes representing our label predictions. The first layer's input_shape parameter corresponds to the amount of features from the dataset, and is required.

```
model = tf.keras.Sequential([
  tf.keras.layers.Dense(10, activation="relu", input_shape=(4,)),  # input shape required
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(3)
```

The activation function determines the output of a single neuron to the next layer. This is loosely based on how brain neurons are connected. There are many available activations, but ReLU is common for hidden layers.

The ideal number of hidden layers and neurons depends on the problem and the dataset. Like many aspects of machine learning, picking the best shape of the neural network requires a mixture of knowledge and experimentation. As a rule of thumb, increasing the number of hidden layers and neurons typically creates a more powerful model, which requires more data to train effectively.

Train the model

Training is the stage of machine learning when the model is gradually optimized, or the model learns the dataset. The goal is to learn enough about the structure of the training dataset to make predictions about unseen data. If you learn too much about the training dataset, then the predictions only work for the data it has seen and will not be generalizable. This problem is called overfitting—it's like memorizing the answers instead of understanding how to solve a problem.

The Iris classification problem is an example of supervised machine learning: the model is trained from examples that contain labels. In unsupervised machine learning, the examples don't contain labels. Instead, the model typically finds patterns among the features.

Define the loss and gradient function

Both training and evaluation stages need to calculate the model's loss. This measures how off a model's predictions are from the desired label, in other words, how bad the model is performing. We want to minimize, or optimize, this value.

Our model will calculate its loss using the tf.losses.sparse_softmax_cross_entropy function which takes the model's prediction and the desired label. The returned loss value is progressively larger as the prediction gets worse.

```
def loss(model, x, y):
  y_ = model(x)
  return tf.losses.sparse_softmax_cross_entropy(labels=y, logits=y_)


def grad(model, inputs, targets):
  with tf.GradientTape() as tape:
    loss_value = loss(model, inputs, targets)
  return tape.gradient(loss_value, model.variables)
```

The grad function uses the loss function and the tf.GradientTape to record operations that compute the gradients used to optimize our model. For more examples of this, see the eager execution guide.

Create an optimizer

An optimizer applies the computed gradients to the model's variables to minimize the loss function. You can think of a curved surface (see Figure 3) and we want to find its lowest point by walking around. The gradients point in the direction of steepest ascent—so we'll travel the opposite way and move down the hill. By iteratively calculating the loss and gradient for each batch, we'll adjust the model during training. Gradually, the model will find the best combination of weights and bias to minimize loss. And the lower the loss, the better the model's predictions.

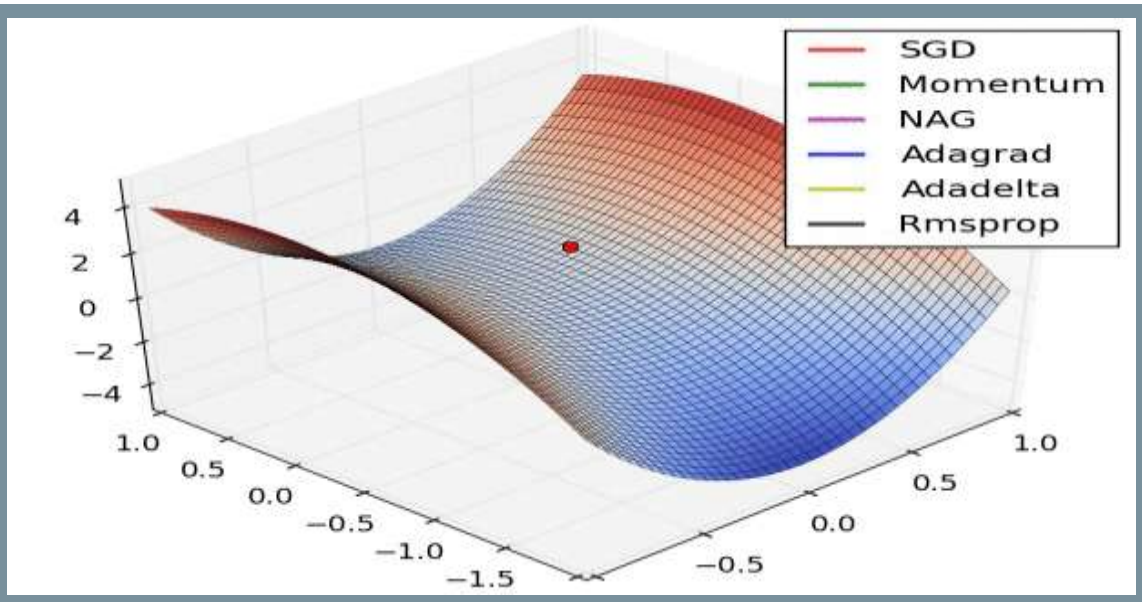


Figure 3. Optimization algorthims visualized over time in 3D space. (Source: Stanford class CS231n, MIT License)

TensorFlow has many optimization algorithms available for training. This model uses the tf.train.GradientDescentOptimizer that implements the stochastic gradient descent (SGD) algorithm. The learning_rate sets the step size to take for each iteration down the hill. This is a hyperparameter that you'll commonly adjust to achieve better results.

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
```

Training loop

With all the pieces in place, the model is ready for training! A training loop feeds the dataset examples into the model to help it make better predictions. The following code block sets up these training steps:

1. Iterate each epoch. An epoch is one pass through the dataset.
2. Within an epoch, iterate over each example in the training Dataset grabbing its features (x) and label (y).
3. Using the example's features, make a prediction and compare it with the label. Measure the inaccuracy of the prediction and use that to calculate the model's loss and gradients.
4. Use an optimizer to update the model's variables.
5. Keep track of some stats for visualization.
6. Repeat for each epoch.

The num_epochs variable is the amount of times to loop over the dataset collection. Counter-intuitively, training a model longer does not guarantee a better model. num_epochs is a hyperparameter that you can tune. Choosing the right number usually requires both experience and experimentation.

```
## Note: Rerunning this cell uses the same model variables

# keep results for plotting
train_loss_results = []
train_accuracy_results = []

num_epochs = 201

for epoch in range(num_epochs):
  epoch_loss_avg = tfe.metrics.Mean()
  epoch_accuracy = tfe.metrics.Accuracy()

  # Training loop - using batches of 32
  for x, y in train_dataset:
    # Optimize the model
    grads = grad(model, x, y)
    optimizer.apply_gradients(zip(grads, model.variables),
                              global_step=tf.train.get_or_create_global_step())

    # Track progress
    epoch_loss_avg(loss(model, x, y))  # add current batch loss
    # compare predicted label to actual label
    epoch_accuracy(tf.argmax(model(x), axis=1, output_type=tf.int32), y)
```

```
# end epoch
train_loss_results.append(epoch_loss_avg.result())
train_accuracy_results.append(epoch_accuracy.result())

if epoch % 50 == 0:
  print("Epoch {:03d}: Loss: {:.3f}, Accuracy: {:.3%}".format(epoch,
                                        epoch_loss_avg.result(),
                                        epoch_accuracy.result()))
```

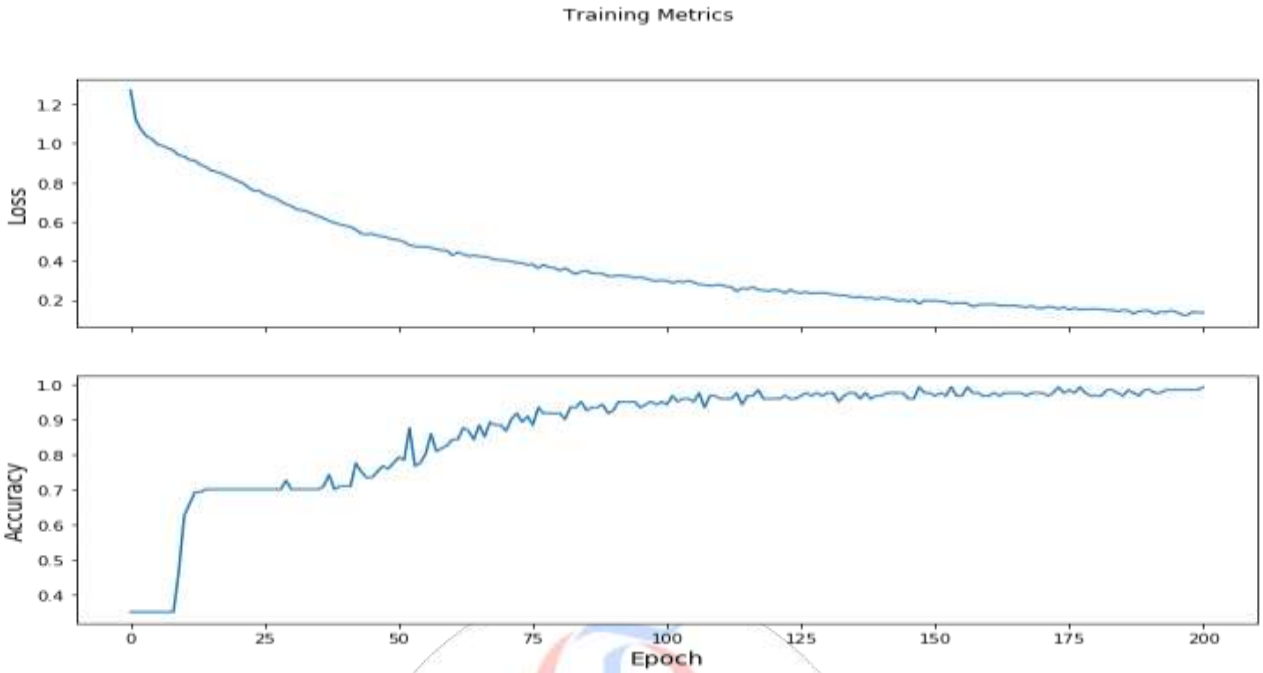
```
Epoch 000: Loss: 1.271, Accuracy: 35.000%

Epoch 050: Loss: 0.507, Accuracy: 79.167%

Epoch 100: Loss: 0.297, Accuracy: 94.167%

Epoch 150: Loss: 0.197, Accuracy: 96.667%

Epoch 200: Loss: 0.136, Accuracy: 99.167%
```

Visualize the loss function over time

While it's helpful to print out the model's training progress, it's often more helpful to see this progress. TensorBoard is a nice visualization tool that is packaged with TensorFlow, but we can create basic charts using the mathplotlib module.

Interpreting these charts takes some experience, but you really want to see the loss go down and the accuracy go up.

```
fig, axes = plt.subplots(2, sharex=True, figsize=(12, 8))
fig.suptitle('Training Metrics')

axes[0].set_ylabel("Loss", fontsize=14)
axes[0].plot(train_loss_results)

axes[1].set_ylabel("Accuracy", fontsize=14)
axes[1].set_xlabel("Epoch", fontsize=14)
axes[1].plot(train_accuracy_results)

plt.show()
```



Evaluate the model's effectiveness

Now that the model is trained, we can get some statistics on its performance.

Example features				Label	Model prediction
5.9	3.0	4.3	1.5	1	1
6.9	3.1	5.4	2.1	2	2
5.1	3.3	1.7	0.5	0	0
6.0	3.4	4.5	1.6	1	2
5.5	2.5	4.0	1.3	1	1

Figure 4. An Iris classifier that is 80% accurate.

Evaluating means determining how effectively the model makes predictions. To determine the model's effectiveness at Iris classification, pass some sepal and petal measurements to the model and ask the model to predict what Iris species they represent. Then compare the

model's prediction against the actual label. For example, a model that picked the correct species on half the input examples has an accuracy of 0.5. Figure 4 shows a slightly more effective model, getting 4 out of 5 predictions correct at 80% accuracy:

Setup the test dataset

Evaluating the model is similar to training the model. The biggest difference is the examples come from a separate test set rather than the training set. To fairly assess a model's effectiveness, the examples used to evaluate a model must be different from the examples used to train the model.

The setup for the test Dataset is similar to the setup for training Dataset. Download the CSV text file and parse that values, then give it a little shuffle:

```
test_url = "http://download.tensorflow.org/data/iris_test.csv"

test_fp = tf.keras.utils.get_file(fname=os.path.basename(test_url),
                                  origin=test_url)

test_dataset = tf.data.TextLineDataset(test_fp)
test_dataset = test_dataset.skip(1)             # skip header row
test_dataset = test_dataset.map(parse_csv)      # parse each row with the funcition created earlier
test_dataset = test_dataset.shuffle(1000)       # randomize
test_dataset = test_dataset.batch(32)           # use the same batch size as the training set


Downloading data from http://download.tensorflow.org/data/iris_test.csv

8192/573 [==============================================] - 0s 0us/step
```

Evaluate the model on the test dataset

Unlike the training stage, the model only evaluates a single epoch of the test data. In the following code cell, we iterate over each example in the test set and compare the model's prediction against the actual label. This is used to measure the model's accuracy across the entire test set.

```
test_accuracy = tfe.metrics.Accuracy()

for (x, y) in test_dataset:
  prediction = tf.argmax(model(x), axis=1, output_type=tf.int32)
  test_accuracy(prediction, y)

print("Test set accuracy: {:.3%}".format(test_accuracy.result()))
```

```
Test set accuracy: 100.000%
```

Use the trained model to make predictions

We've trained a model and "proven" that it's good—but not perfect—at classifying Iris species. Now let's use the trained model to make some predictions on unlabeled examples; that is, on examples that contain features but not a label.

In real-life, the unlabeled examples could come from lots of different sources including apps, CSV files, and data feeds. For now, we're going to manually provide three unlabeled examples to predict their labels. Recall, the label numbers are mapped to a named representation as:

- 0: Iris setosa
- 1: Iris versicolor
- 2: Iris virginica

```
class_ids = ["Iris setosa", "Iris versicolor", "Iris virginica"]

predict_dataset = tf.convert_to_tensor([
    [5.1, 3.3, 1.7, 0.5,],
    [5.9, 3.0, 4.2, 1.5,],
    [6.9, 3.1, 5.4, 2.1]
])

predictions = model(predict_dataset)

for i, logits in enumerate(predictions):
  class_idx = tf.argmax(logits).numpy()
  name = class_ids[class_idx]
  print("Example {} prediction: {}".format(i, name))


Example 0 prediction: Iris setosa

Example 1 prediction: Iris versicolor

Example 2 prediction: Iris virginica
```

These predictions look good!

To dig deeper into machine learning models, take a look at the TensorFlow Programmer's Guide and check out the google ML community.**TensorFlow Linear Model**

In this document, we will use the tf.estimator API in TensorFlow to solve a binary classification problem: Given census data about a person such as age, education, marital

status, and occupation (the features), we will try to predict whether or not the person earns more than 50,000 dollars a year (the target label). We will train a **logistic regression** model, and given an individual's information our model will output a number between 0 and 1, which can be interpreted as the probability that the individual has an annual income of over 50,000 dollars.

Setup

1. Install TensorFlow if you haven't already
2. Download the code. : https://github.com/tensorflow/models/tree/master/official/wide_deep/
3. Execute the data download script we provide to you:

```
$ python data_download.py
```

4. Execute the code with the following command to train the linear model described in this document:

```
$ python wide_deep.py --model_type=wide
```

Read on to find out how this code builds its linear model.

Reading The Census Data

The dataset we'll be using is the Census Income Dataset. We have provided data_download.py which downloads the code and performs some additional cleanup.

Since the task is a binary classification problem, we'll construct a label column named "label" whose value is 1 if the income is over 50K, and 0 otherwise. For reference, see `input_fn` in wide_deep.py.

Next, let's take a look at the dataframe and see which columns we can use to predict the target label. The columns can be grouped into two types—categorical and continuous columns:

- A column is called **categorical** if its value can only be one of the categories in a finite set. For example, the relationship status of a person (wife, husband, unmarried, etc.) or the education level (high school, college, etc.) are categorical columns.
- A column is called **continuous** if its value can be any numerical value in a continuous range. For example, the capital gain of a person (e.g. $14,084) is a continuous column.

Column Name	Type	Description
Age	Continuous	The age of the individual
Workclass	Categorical	The type of employer the individual has (government, military, private, etc.).
Fnlwgt	Continuous	The number of people the census takers believe that observation represents (sample weight). Final weight will not be used.
Education	Categorical	The highest level of education achieved for that individual.
education_num	Continuous	The highest level of education in numerical form.
marital_status	Categorical	Marital status of the individual.
Occupation	Categorical	The occupation of the individual.
Relationship	Categorical	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
Race	Categorical	Amer-Indian-Eskimo, Asian-Pac- Islander, Black, White, Other.
Gender	Categorical	Female, Male.
capital_gain	Continuous	Capital gains recorded.
capital_loss	Continuous	Capital Losses recorded.
hours_per_week	Continuous	Hours worked per week.
native_country	Categorical	Country of origin of the individual.
income_bracket	Categorical	">50K" or "<=50K", meaning whether the person makes more than $50,000 annually.

Here's a list of columns available in the Census Income dataset:

Converting Data into Tensors

When building a tf.estimator model, the input data is specified by means of an Input Builder function. This builder function will not be called until it is later passed to tf.estimator.Estimator methods such as `train` and `evaluate`. The purpose of this function is to construct the input data, which is represented in the form of `tf.Tensor`s or `tf.SparseTensor`s. In more detail, the input builder function returns the following as a pair:

1. `features`: A dict from feature column names to `Tensors` or `SparseTensors`.
2. `labels`: A `Tensor` containing the label column.

The keys of the `features` will be used to construct columns in the next section. Because we want to call the `train` and `evaluate` methods with different data, we define a method that

returns an input function based on the given data. Note that the returned input function will be called while constructing the TensorFlow graph, not while running the graph. What it is returning is a representation of the input data as the fundamental unit of TensorFlow computations, a `Tensor` (or `SparseTensor`).

Each continuous column in the train or test data will be converted into a `Tensor`, which in general is a good format to represent dense data. For categorical data, we must represent the data as a `SparseTensor`. This data format is good for representing sparse data. Our `input_fn` uses the `tf.data` API, which makes it easy to apply transformations to our dataset:

```
def input_fn(data_file, num_epochs, shuffle, batch_size):
 """Generate an input function for the Estimator."""
 assert tf.gfile.Exists(data_file), (
    '%s not found. Please make sure you have either run data_download.py or '
    'set both arguments --train_data and --test_data.' % data_file)

 def parse_csv(value):
  print('Parsing', data_file)
  columns = tf.decode_csv(value, record_defaults=_CSV_COLUMN_DEFAULTS)
  features = dict(zip(_CSV_COLUMNS, columns))
  labels = features.pop('income_bracket')
  return features, tf.equal(labels, '>50K')

 # Extract lines from input files using the Dataset API.
 dataset = tf.data.TextLineDataset(data_file)

 if shuffle:
  dataset = dataset.shuffle(buffer_size=_SHUFFLE_BUFFER)

 dataset = dataset.map(parse_csv, num_parallel_calls=5)

 # We call repeat after shuffling, rather than before, to prevent separate
 # epochs from blending together.
 dataset = dataset.repeat(num_epochs)
 dataset = dataset.batch(batch_size)

 iterator = dataset.make_one_shot_iterator()
 features, labels = iterator.get_next()
 return features, labels
```

Selecting and Engineering Features for the Model

Selecting and crafting the right set of feature columns is key to learning an effective model. A **feature column** can be either one of the raw columns in the original dataframe (let's call them **base feature columns**), or any new columns created based on some transformations

defined over one or multiple base columns (let's call them **derived feature columns**). Basically, "feature column" is an abstract concept of any raw or derived variable that can be used to predict the target label.

Base Categorical Feature Columns

To define a feature column for a categorical feature, we can create a CategoricalColumn using the tf.feature_column API. If you know the set of all possible feature values of a column and there are only a few of them, you can use categorical_column_with_vocabulary_list. Each key in the list will get assigned an auto-incremental ID starting from 0. For example, for the relationship column we can assign the feature string "Husband" to an integer ID of 0 and "Not-in-family" to 1, etc., by doing:

```
relationship = tf.feature_column.categorical_column_with_vocabulary_list(
  'relationship', [
    'Husband', 'Not-in-family', 'Wife', 'Own-child', 'Unmarried',
    'Other-relative'])
```

What if we don't know the set of possible values in advance? Not a problem. We can use categorical_column_with_hash_bucket instead:

```
occupation = tf.feature_column.categorical_column_with_hash_bucket(
  'occupation', hash_bucket_size=1000)
```

What will happen is that each possible value in the feature column occupation will be hashed to an integer ID as we encounter them in training. See an example illustration below:

ID	Feature
9	"Machine-op-inspct"
103	"Farming-fishing"
375	"Protective-serv"

No matter which way we choose to define a SparseColumn, each feature string will be mapped into an integer ID by looking up a fixed mapping or by hashing. Note that hashing collisions are possible, but may not significantly impact the model quality. Under the hood, the LinearModel class is responsible for managing the mapping and creating tf.Variable to store the model parameters (also known as model weights) for each feature ID. The model parameters will be learned through the model training process we'll go through later.

We'll do the similar trick to define the other categorical features:

```
education = tf.feature_column.categorical_column_with_vocabulary_list(
  'education', [
    'Bachelors', 'HS-grad', '11th', 'Masters', '9th', 'Some-college',
    'Assoc-acdm', 'Assoc-voc', '7th-8th', 'Doctorate', 'Prof-school',
    '5th-6th', '10th', '1st-4th', 'Preschool', '12th'])
```

```
marital_status = tf.feature_column.categorical_column_with_vocabulary_list(
   'marital_status', [
      'Married-civ-spouse', 'Divorced', 'Married-spouse-absent',
      'Never-married', 'Separated', 'Married-AF-spouse', 'Widowed'])

relationship = tf.feature_column.categorical_column_with_vocabulary_list(
   'relationship', [
      'Husband', 'Not-in-family', 'Wife', 'Own-child', 'Unmarried',
      'Other-relative'])

workclass = tf.feature_column.categorical_column_with_vocabulary_list(
   'workclass', [
      'Self-emp-not-inc', 'Private', 'State-gov', 'Federal-gov',
      'Local-gov', '?', 'Self-emp-inc', 'Without-pay', 'Never-worked'])

# To show an example of hashing:
occupation = tf.feature_column.categorical_column_with_hash_bucket(
   'occupation', hash_bucket_size=1000)
```

Base Continuous Feature Columns

Similarly, we can define a NumericColumn for each continuous feature column that we want to use in the model:

```
age = tf.feature_column.numeric_column('age')
education_num = tf.feature_column.numeric_column('education_num')
capital_gain = tf.feature_column.numeric_column('capital_gain')
capital_loss = tf.feature_column.numeric_column('capital_loss')
hours_per_week = tf.feature_column.numeric_column('hours_per_week')
```

Making Continuous Features Categorical through Bucketization

Sometimes the relationship between a continuous feature and the label is not linear. As a hypothetical example, a person's income may grow with age in the early stage of one's career, then the growth may slow at some point, and finally the income decreases after retirement. In this scenario, using the raw age as a real-valued feature column might not be a good choice because the model can only learn one of the three cases:

1. Income always increases at some rate as age grows (positive correlation),
2. Income always decreases at some rate as age grows (negative correlation), or
3. Income stays the same no matter at what age (no correlation)

If we want to learn the fine-grained correlation between income and each age group separately, we can leverage **bucketization**. Bucketization is a process of dividing the entire range of a continuous feature into a set of consecutive bins/buckets, and then converting the original numerical feature into a bucket ID (as a categorical feature) depending on which bucket that value falls into. So, we can define a `bucketized_column` over `age` as:

```
age_buckets = tf.feature_column.bucketized_column(
  age, boundaries=[18, 25, 30, 35, 40, 45, 50, 55, 60, 65])
```

where the `boundaries` is a list of bucket boundaries. In this case, there are 10 boundaries, resulting in 11 age group buckets (from age 17 and below, 18-24, 25-29, ..., to 65 and over).

Intersecting Multiple Columns with CrossedColumn

Using each base feature column separately may not be enough to explain the data. For example, the correlation between education and the label (earning > 50,000 dollars) may be different for different occupations. Therefore, if we only learn a single model weight for `education="Bachelors"` and `education="Masters"`, we won't be able to capture every single education-occupation combination (e.g. distinguishing between `education="Bachelors" AND occupation="Exec-managerial"` and `education="Bachelors" AND occupation="Craft-repair"`). To learn the differences between different feature combinations, we can add **crossed feature columns** to the model.

```
education_x_occupation = tf.feature_column.crossed_column(
  ['education', 'occupation'], hash_bucket_size=1000)
```

We can also create a `CrossedColumn` over more than two columns. Each constituent column can be either a base feature column that is categorical (`SparseColumn`), a bucketized real-valued feature column (`BucketizedColumn`), or even another `CrossColumn`. Here's an example:

```
age_buckets_x_education_x_occupation = tf.feature_column.crossed_column(
  [age_buckets, 'education', 'occupation'], hash_bucket_size=1000)
```

Defining The Logistic Regression Model

After processing the input data and defining all the feature columns, we're now ready to put them all together and build a Logistic Regression model. In the previous section we've seen several types of base and derived feature columns, including:

- `CategoricalColumn`
- `NumericColumn`
- `BucketizedColumn`
- `CrossedColumn`

All of these are subclasses of the abstract FeatureColumn class, and can be added to the feature_columnsfield of a model:

```
base_columns = [
  education, marital_status, relationship, workclass, occupation,
  age_buckets,
]
crossed_columns = [
  tf.feature_column.crossed_column(
    ['education', 'occupation'], hash_bucket_size=1000),
  tf.feature_column.crossed_column(
    [age_buckets, 'education', 'occupation'], hash_bucket_size=1000),
]

model_dir = tempfile.mkdtemp()
model = tf.estimator.LinearClassifier(
  model_dir=model_dir, feature_columns=base_columns + crossed_columns)
```

The model also automatically learns a bias term, which controls the prediction one would make without observing any features (see the section "How Logistic Regression Works" for more explanations). The learned model files will be stored in model_dir.

Training and Evaluating Our Model

After adding all the features to the model, now let's look at how to actually train the model. Training a model is just a single command using the tf.estimator API:

```
model.train(input_fn=lambda: input_fn(train_data, num_epochs, True, batch_size))
```

After the model is trained, we can evaluate how good our model is at predicting the labels of the holdout data:

```
results = model.evaluate(input_fn=lambda: input_fn(
  test_data, 1, False, batch_size))
for key in sorted(results):
 print('%s: %s' % (key, results[key]))
```

The first line of the final output should be something like accuracy: 0.83557522, which means the accuracy is 83.6%. Feel free to try more features and transformations and see if you can do even better!

After the model is evaluated, we can use the model to predict whether an individual has an annual income of over 50,000 dollars given an individual's information input.

```
pred_iter = model.predict(input_fn=lambda: input_fn(FLAGS.test_data, 1, False, 1))
for pred in pred_iter:
  print(pred['classes'])
```

The model prediction output would be like `[b'1']` or `[b'0']` which means whether corresponding individual has an annual income of over 50,000 dollars or not.

If you'd like to see a working end-to-end example, you can download our example code and set the `model_type`flag to `wide`.

Adding Regularization to Prevent Overfitting

Regularization is a technique used to avoid **overfitting**. Overfitting happens when your model does well on the data it is trained on, but worse on test data that the model has not seen before, such as live traffic. Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observed training data. Regularization allows for you to control your model's complexity and makes the model more generalizable to unseen data.

In the Linear Model library, you can add L1 and L2 regularizations to the model as:

```
model = tf.estimator.LinearClassifier(
   model_dir=model_dir, feature_columns=base_columns + crossed_columns,
   optimizer=tf.train.FtrlOptimizer(
      learning_rate=0.1,
      l1_regularization_strength=1.0,
      l2_regularization_strength=1.0))
```

One important difference between L1 and L2 regularization is that L1 regularization tends to make model weights stay at zero, creating sparser models, whereas L2 regularization also tries to make the model weights closer to zero but not necessarily zero. Therefore, if you increase the strength of L1 regularization, you will have a smaller model size because many of the model weights will be zero. This is often desirable when the feature space is very large but sparse, and when there are resource constraints that prevent you from serving a model that is too large.

In practice, you should try various combinations of L1, L2 regularization strengths and find the best parameters that best control overfitting and give you a desirable model size.

How Logistic Regression Works

Finally, let's take a minute to talk about what the Logistic Regression model actually looks like in case you're not already familiar with it. We'll denote the label as Y, and the set of observed features as a feature vector x=[x1,x2,...,xd]. We define Y=1 if an individual earned > 50,000 dollars and Y=0 otherwise. In Logistic Regression, the probability of the label being positive (Y=1) given the features x is given as:

$P(Y=1|x)=11+exp(-(wTx+b))$

where w=[w1,w2,...,wd] are the model weights for the features x=[x1,x2,...,xd]. b is a constant that is often called the **bias** of the model. The equation consists of two parts—A linear model and a logistic function:

- **Linear Model**: First, we can see that wTx+b=b+w1x1+...+wdxd is a linear model where the output is a linear function of the input features x. The bias b is the prediction one would make without observing any features. The model weight wi reflects how the feature xi is correlated with the positive label. If xi is positively correlated with the positive label, the weight wi increases, and the probability P(Y=1|x) will be closer to 1. On the other hand, if xi is negatively correlated with the positive label, then the weight wi decreases and the probability P(Y=1|x) will be closer to 0.
- **Logistic Function**: Second, we can see that there's a logistic function (also known as the sigmoid function) S(t)=1/(1+exp(−t)) being applied to the linear model. The logistic function is used to convert the output of the linear model wTx+b from any real number into the range of [0,1], which can be interpreted as a probability.

Model training is an optimization problem: The goal is to find a set of model weights (i.e. model parameters) to minimize a **loss function** defined over the training data, such as logistic loss for Logistic Regression models. The loss function measures the discrepancy between the ground-truth label and the model's prediction. If the prediction is very close to the ground-truth label, the loss value will be low; if the prediction is very far from the label, then the loss value would be high.

-------o------

Note: Book will be updated based on the Technology changes, updates will be available in the next version.