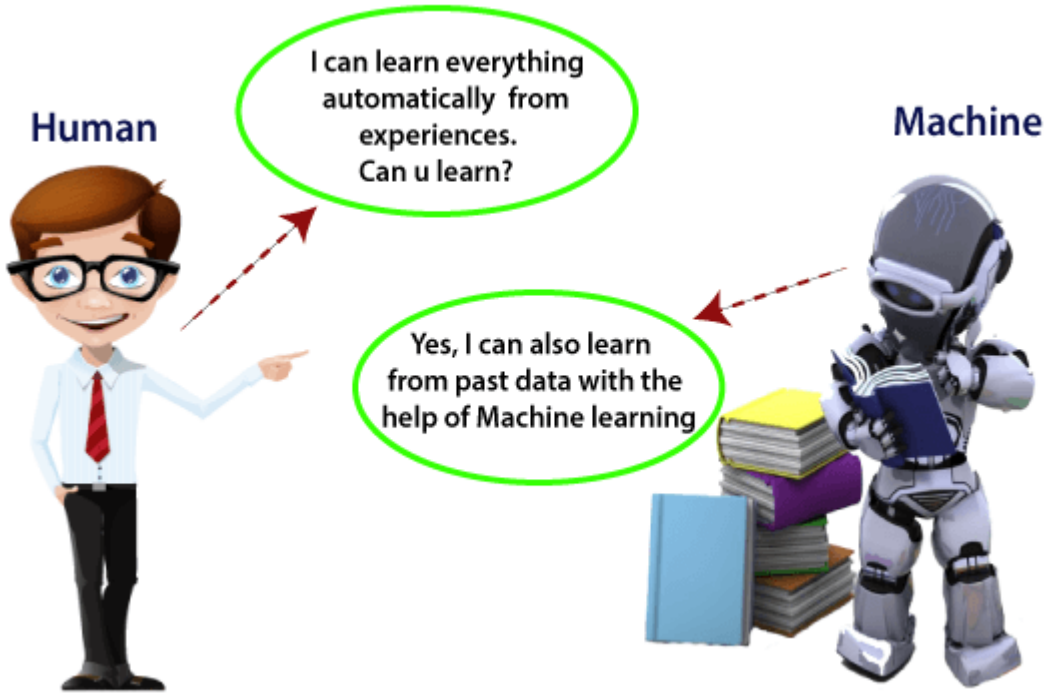
Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for **building mathematical models and making predictions using historical data or information**. Currently, it is being used for various tasks such as **image recognition**, **speech recognition**, **email filtering**, **Facebook auto-tagging**, **recommender system**, and many more.

This machine learning material gives you an introduction to machine learning along with the wide range of machine learning techniques such as **Supervised**, **Unsupervised**, and **Reinforcement** learning. You will learn about regression and classification models, clustering methods, hidden Markov models, and various sequential models.

What is Machine Learning

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning**.



Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**. We can define it in a summarized way as:

> Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine

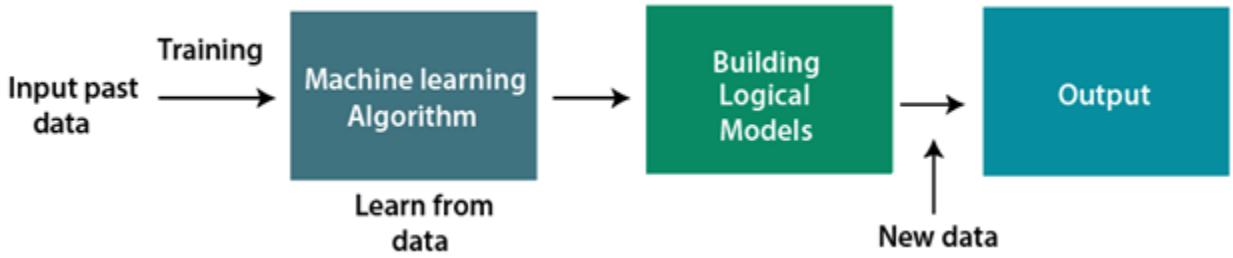
learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

A machine has the ability to learn if it can improve its performance by gaining more data.

How does Machine Learning work

A Machine Learning system **learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it**. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Need for Machine Learning

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.

We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases, currently, machine learning is used in **self-driving cars**, **cyber fraud detection**, **face recognition**, and **friend suggestion by Facebook**, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyze the user interest and recommend product accordingly.

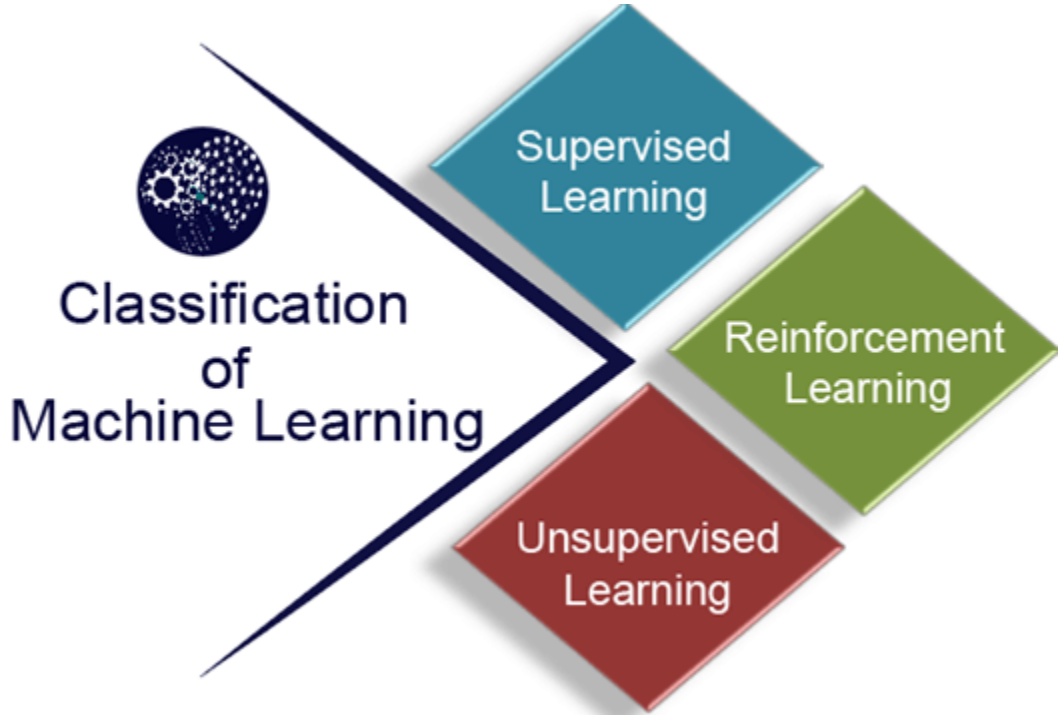
Following are some key points which show the importance of Machine Learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**



1) Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is **spam filtering**.

Supervised learning can be grouped further in two categories of algorithms:

- **Classification**
- **Regression**

2) Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we do not have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:

- **Clustering**
- **Association**

3) Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

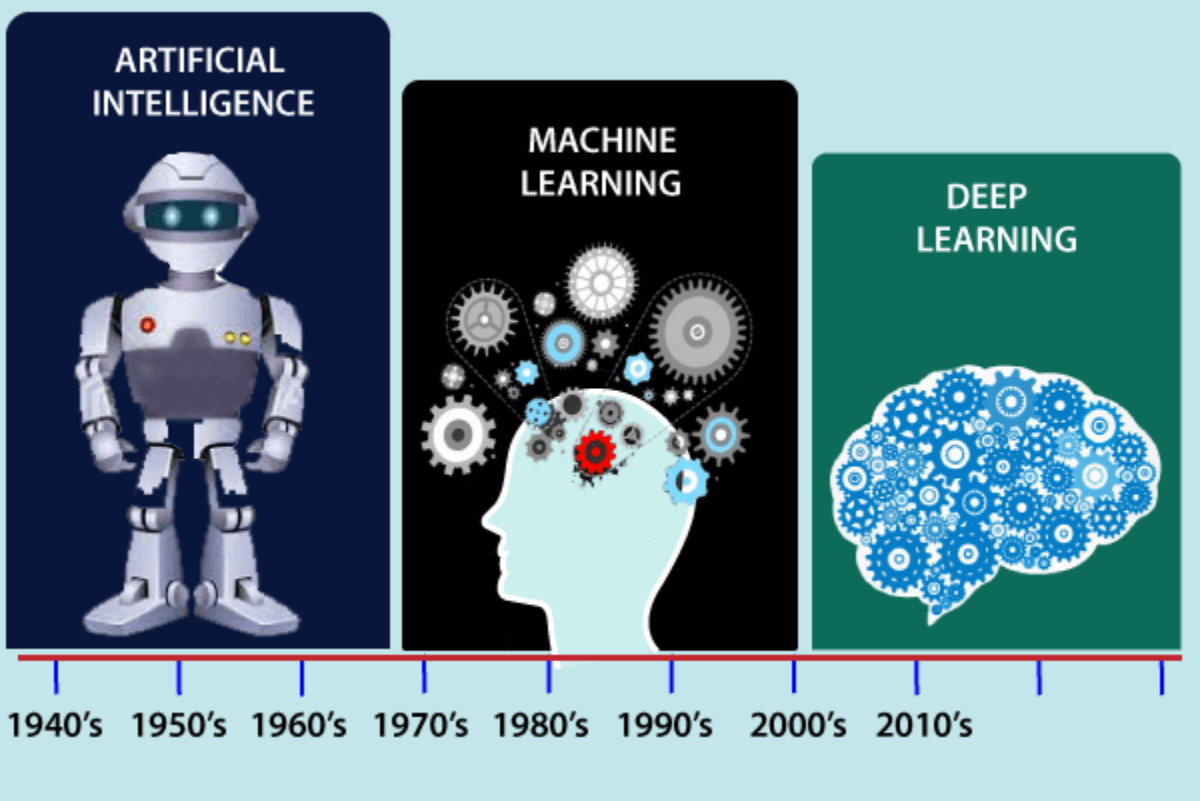
The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

Note: We will learn about the above types of machine learning in detail in later chapters.

History of Machine Learning

Before some years (about 40-50 years), machine learning was science fiction, but today it is the part of our daily life. Machine learning is making our day to day life easy from **self-driving cars** to **Amazon virtual assistant**

"**Alexa**". However, the idea behind machine learning is so old and has a long history. Below some milestones are given which have occurred in the history of machine learning:



The early history of Machine Learning (Pre-1940):

- **1834:** In 1834, Charles Babbage, the father of the computer, conceived a device that could be programmed with punch cards. However, the machine was never built, but all modern computers rely on its logical structure.
- **1936:** In 1936, Alan Turing gave a theory that how a machine can determine and execute a set of instructions.

The era of stored program computers:

- **1940:** In 1940, the first manually operated computer, "ENIAC" was invented, which was the first electronic general-purpose computer. After that stored program computer such as EDSAC in 1949 and EDVAC in 1951 were invented.
- **1943:** In 1943, a human neural network was modeled with an electrical circuit. In 1950, the scientists started applying their idea to work and analyzed how human neurons might work.

Computer machinery and intelligence:

- **1950:** In 1950, Alan Turing published a seminal paper, "**Computer Machinery and Intelligence**," on the topic of artificial intelligence. **In his paper, he asked, "Can machines think?"**

Machine intelligence in Games:

- **1952:** Arthur Samuel, who was the pioneer of machine learning, created a program that helped an IBM computer to play a checkers game. It performed better more it played.
- **1959:** In 1959, the term "Machine Learning" was first coined by **Arthur Samuel**.

The first "AI" winter:

- The duration of 1974 to 1980 was the tough time for AI and ML researchers, and this duration was called as **AI winter**.
- In this duration, failure of machine translation occurred, and people had reduced their interest from AI, which led to reduced funding by the government to the researches.

Machine Learning from theory to reality

- **1959:** In 1959, the first neural network was applied to a real-world problem to remove echoes over phone lines using an adaptive filter.
- **1985:** In 1985, Terry Sejnowski and Charles Rosenberg invented a neural network **NETtalk**, which was able to teach itself how to correctly pronounce 20,000 words in one week.
- **1997:** The IBM's **Deep blue** intelligent computer won the chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

Machine Learning at 21st century

- **2006:** In the year 2006, computer scientist Geoffrey Hinton has given a new name to neural net research as "**deep learning**," and nowadays, it has become one of the most trending technologies.
- **2012:** In 2012, Google created a deep neural network which learned to recognize the image of humans and cats in YouTube videos.
- **2014:** In 2014, the Chabot "**Eugen Goostman**" cleared the Turing Test. It was the first Chabot who convinced the 33% of human judges that it was not a machine.

- **2014: DeepFace** was a deep neural network created by Facebook, and they claimed that it could recognize a person with the same precision as a human can do.
- **2016: AlphaGo** beat the world's number second player **Lee sedol** at **Go game**. In 2017 it beat the number one player of this game **Ke Jie**.
- **2017:** In 2017, the Alphabet's Jigsaw team built an intelligent system that was able to learn the **online trolling**. It used to read millions of comments of different websites to learn to stop online trolling.

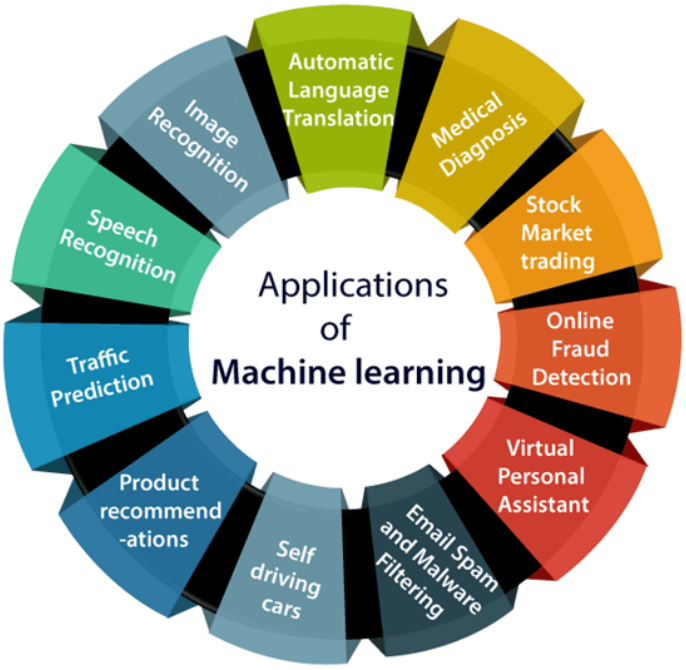
Machine learning at present:

Now machine learning has got a great advancement in its research, and it is present everywhere around us, such as **self-driving cars**, **Amazon Alexa**, **Catboats**, **recommender system**, and many more. It includes **Supervised**, **unsupervised**, and **reinforcement learning with clustering**, **classification**, **decision tree**, **SVM algorithms**, etc.

Modern machine learning models can be used for making various predictions, including **weather prediction**, **disease prediction**, **stock market analysis**, etc.

Applications of Machine learning

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:



1. Image Recognition:

Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, **Automatic friend tagging suggestion**:

Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's **face detection** and **recognition algorithm**.

It is based on the Facebook project named "**Deep Face**," which is responsible for face recognition and person identification in the picture.

2. Speech Recognition

While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google assistant**, **Siri**, **Cortana**, and **Alexa** are using speech recognition technology to follow the voice instructions.

3. Traffic prediction:

If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

- **Real Time location** of the vehicle form Google Map app and sensors
- **Average time has taken** on past days at the same time.

Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4. Product recommendations:

Machine learning is widely used by various e-commerce and entertainment companies such as **Amazon**, **Netflix**, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.

As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

5. Self-driving cars:

One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering:

Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

Some machine learning algorithms such as **Multi-Layer Perceptron**, **Decision tree**, and **Naïve Bayes classifier** are used for email spam filtering and malware detection.

7. Virtual Personal Assistant:

We have various virtual personal assistants such as **Google assistant**, **Alexa**, **Cortana**, **Siri**. As the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.

These virtual assistants use machine learning algorithms as an important part.

These assistant record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

8. Online Fraud Detection:

Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as **fake accounts**, **fake ids**, and **steal money** in the middle of a transaction. So to detect this, **Feed Forward Neural network** helps us by checking whether it is a genuine transaction or a fraud transaction.

For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

9. Stock Market trading:

Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's **long short term memory neural network** is used for the prediction of stock market trends.

10. Medical Diagnosis:

In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain.

It helps in finding brain tumors and other brain-related diseases easily.

11. Automatic Language Translation:

Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.

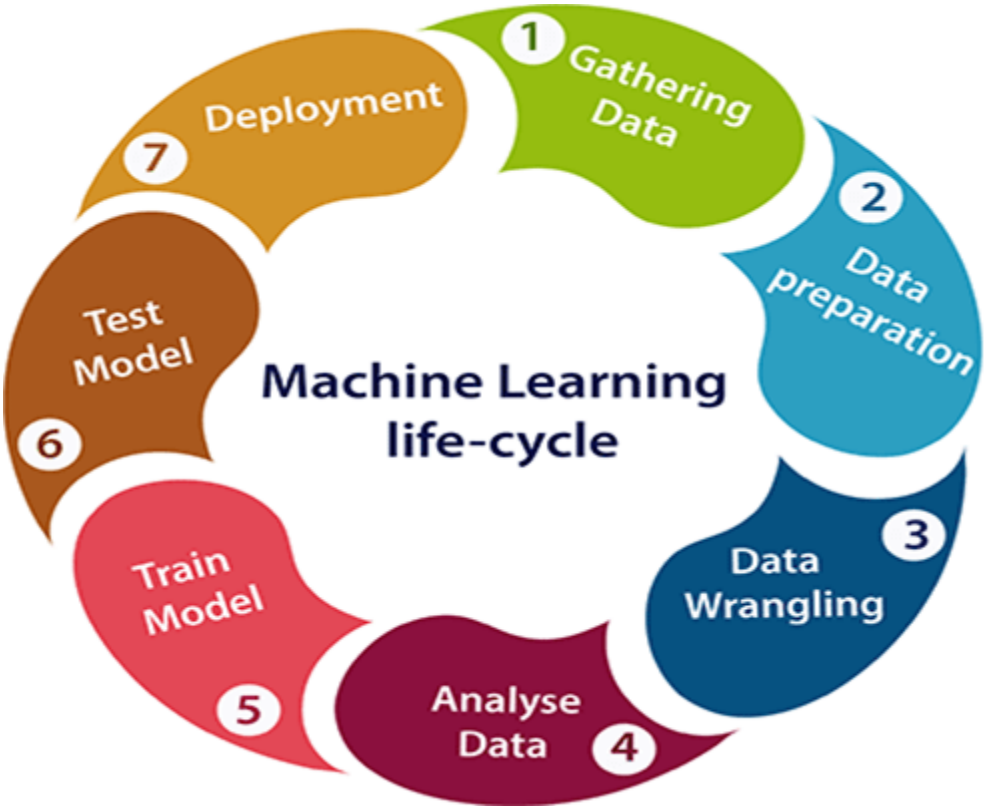
The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

Machine learning Life cycle

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps, which are given below:

- **Gathering Data**
- **Data preparation**
- **Data Wrangling**
- **Analyse Data**
- **Train the model**
- **Test the model**
- **Deployment**



The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

---

1. Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**

- **Integrate the data obtained from different sources**

By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

---

2. Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

- **Data exploration:**

  It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.

  A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- **Data pre-processing:**

  Now the next step is preprocessing of data for its analysis.

---

3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

---

4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- **Selection of analytical techniques**
- **Building models**
- **Review the result**

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification**, **Regression**, **Cluster analysis**, **Association**, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

---

5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

---

6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

---

7. Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

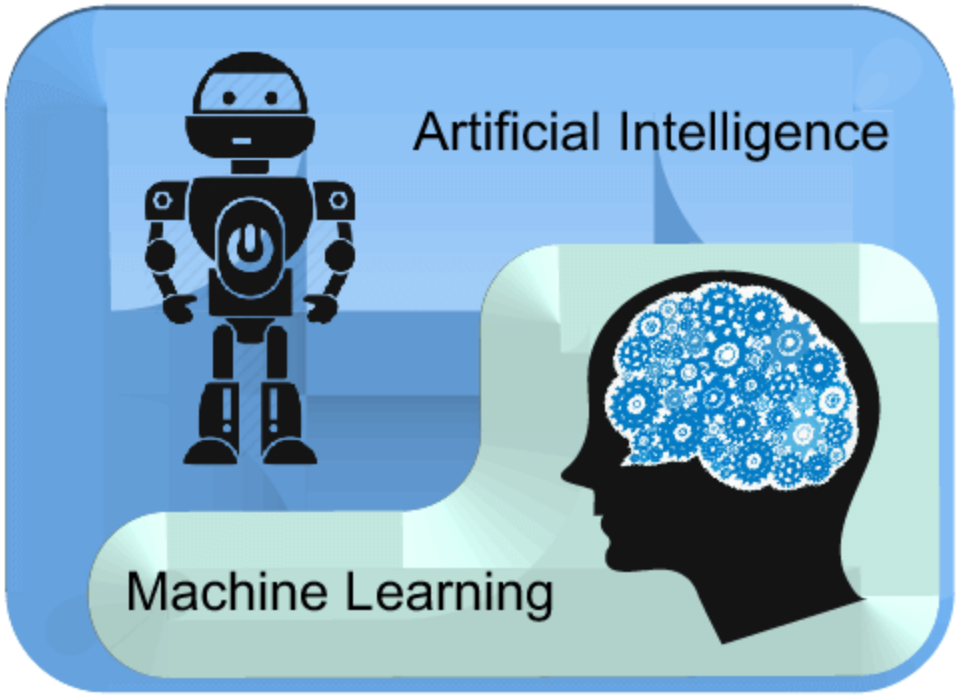
Difference between Artificial intelligence and Machine learning

Artificial intelligence and machine learning are the part of computer science that are correlated with each other. These two technologies are the most trending technologies which are used for creating intelligent systems.

Although these are two related technologies and sometimes people use them as a synonym for each other, but still both are the two different terms in various cases.

On a broad level, we can differentiate both AI and ML as:

AI is a bigger concept to create intelligent machines that can simulate human thinking capability and behavior, whereas, machine learning is an application or subset of AI that allows machines to learn from data without being programmed explicitly.



Below are some main differences between AI and machine learning along with the overview of Artificial intelligence and machine learning.

Artificial Intelligence

Artificial intelligence is a field of computer science which makes a computer system that can mimic human intelligence. It is comprised of two words "**Artificial**" and "**intelligence**", which means "a human-made thinking power." Hence we can define it as,

> *Artificial intelligence is a technology using which we can create intelligent systems that can simulate human intelligence.*

The Artificial intelligence system does not require to be pre-programmed, instead of that, they use such algorithms which can work with their own intelligence. It involves machine learning algorithms such as Reinforcement learning algorithm and deep learning neural networks. AI is being used in multiple places such as Siri, Google?s AlphaGo, AI in Chess playing, etc.

Based on capabilities, AI can be classified into three types:

- **Weak AI**
- **General AI**
- **Strong AI**

Currently, we are working with weak AI and general AI. The future of AI is Strong AI for which it is said that it will be intelligent than humans.

Machine learning

Machine learning is about extracting knowledge from the data. It can be defined as,

> *Machine learning is a subfield of artificial intelligence, which enables machines to learn from past data or experiences without being explicitly programmed.*

Machine learning enables a computer system to make predictions or take some decisions using historical data without being explicitly programmed. Machine learning uses a massive amount of structured and semi-structured data so that a machine learning model can generate accurate result or give predictions based on that data.

Machine learning works on algorithm which learn by it?s own using historical data. It works only for specific domains such as if we are creating a machine learning model to detect pictures of dogs, it will only give result

for dog images, but if we provide a new data like cat image then it will become unresponsive. Machine learning is being used in various places such as for online recommender system, for Google search algorithms, Email spam filter, Facebook Auto friend tagging suggestion, etc.

It can be divided into three types:

- **Supervised learning**
- **Reinforcement learning**
- **Unsupervised learning**

Key differences between Artificial Intelligence (AI) and Machine learning (ML):

Artificial Intelligence	Machine learning
Artificial intelligence is a technology which enables a machine to simulate human behavior.	Machine learning is a subset of AI which allows a machine to automatically learn from past data without programming explicitly.
The goal of AI is to make a smart computer system like humans to solve complex problems.	The goal of ML is to allow machines to learn from data so that they can give accurate output.
In AI, we make intelligent systems to perform any task like a human.	In ML, we teach machines with data to perform a particular task and give an accurate result.
Machine learning and deep learning are the two main subsets of AI.	Deep learning is a main subset of machine learning.
AI has a very wide range of scope.	Machine learning has a limited scope.
AI is working to create an intelligent system which can perform various complex tasks.	Machine learning is working to create machines that can perform only those specific tasks for which they are trained.
AI system is concerned about maximizing the chances of success.	Machine learning is mainly concerned about accuracy and patterns.
The main applications of AI are **Siri, customer support using catboats**, Expert System, Online game playing, intelligent humanoid robot, etc.	The main applications of machine learning are **Online recommender system**, **Google search algorithms**, **Facebook auto friend tagging suggestions**, etc.

On the basis of capabilities, AI can be divided into three types, which are, **Weak AI**, **General AI**, and **Strong AI**.	Machine learning can also be divided into mainly three types that are **Supervised learning**, **Unsupervised learning**, and **Reinforcement learning**.
It includes learning, reasoning, and self-correction.	It includes learning and self-correction when introduced with new data.
AI completely deals with Structured, semi-structured, and unstructured data.	Machine learning deals with Structured and semi-structured data.

Supervised Machine Learning

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

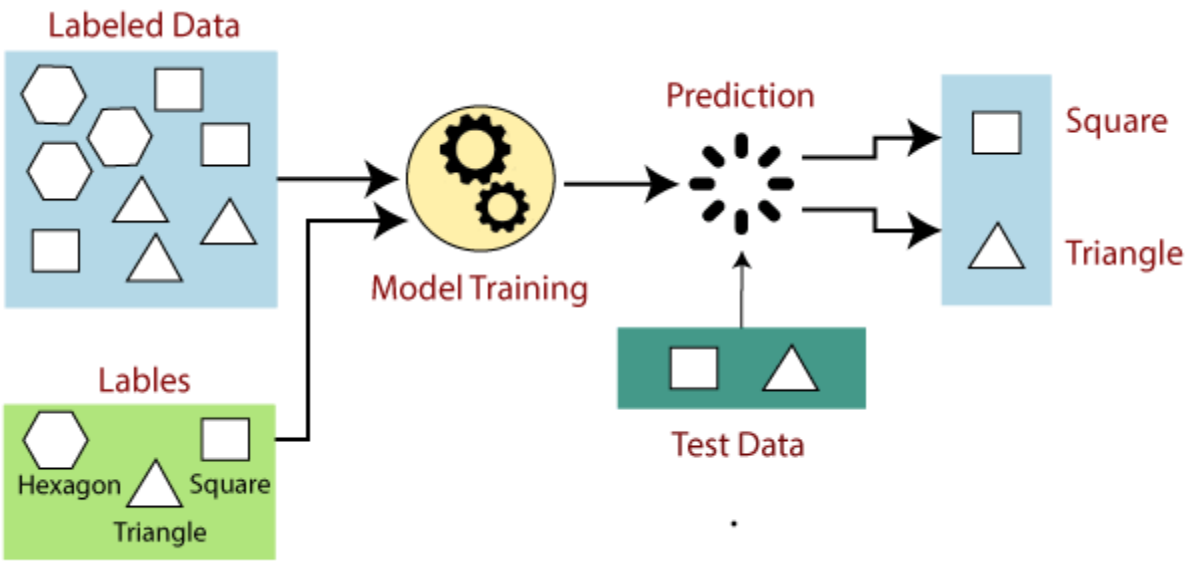
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

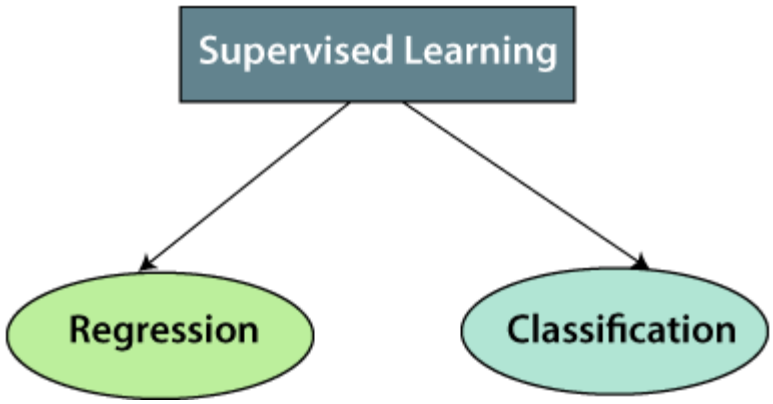
Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.

- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:



1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

Spam Filtering,

- Random Forest
- Decision Trees
- Logistic Regression

- Support vector Machines

Note: We will discuss these algorithms in detail in later chapters.

Advantages of Supervised learning:

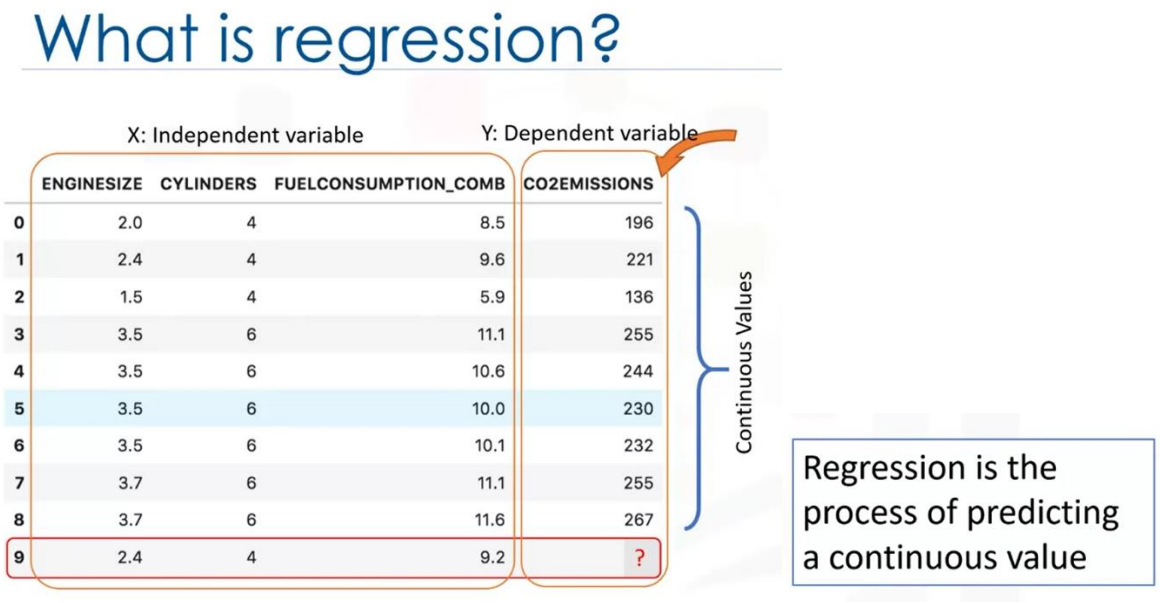
- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

Disadvantages of supervised learning:

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

Introduction to Regression:

What is Regression?



In this lecture we will see a brief introduction to regression. So let's get started. Look at this data set. It's related to co2 emissions from different cars. It includes engine size, number of cylinders, fuel consumption, and co2 emission from various automobile models. The question is: given this data set can we predict the co2 emission of a car using other fields such as engine size or cylinders?

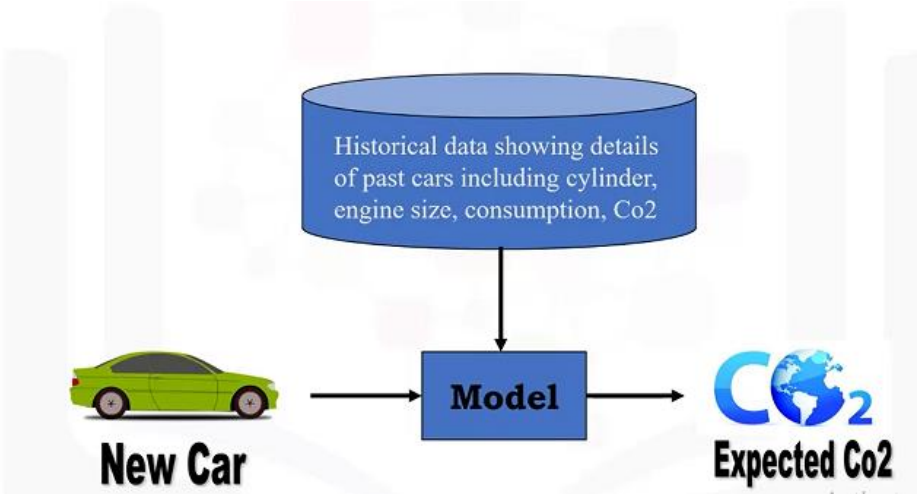
Let's assume we have some historical data from different cars and assume that a car such as in row 9 has not been manufactured yet, but we're interested in estimating its approximate co2 emission after production.

Is it possible?

We can use regression methods to predict a continuous value such as co2 emission using some other variables. Indeed regression is the process of predicting a continuous value. In regression there are two types of variables: a dependent variable and one or more independent variables.

The dependent variable can be seen as the state, target, or final goal we study and try to predict. And the independent variables, also known as explanatory variables, can be seen as the causes of those states.

The independent variables are shown conventionally by X and the dependent variable is notated by Y. A regression model relates Y or the dependent variable to a function of X i.e. the independent variables. The key point in the regression is that our dependent value should be continuous and cannot be a discrete value. However, the independent variable, or variables, can be measured on either a categorical or continuous measurement scale.



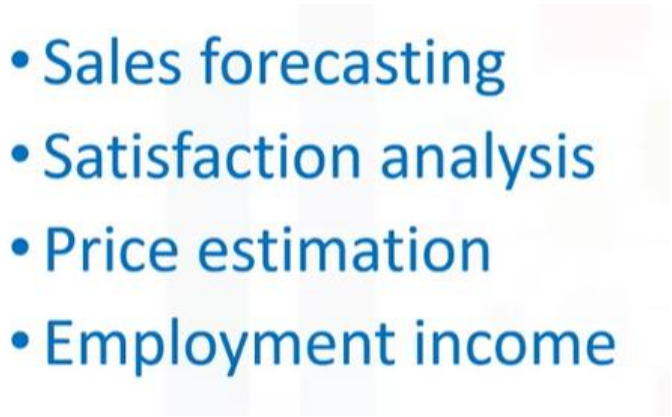
So, what we want to do here is to use the historical data of some cars using one or more of their features and from that data make a model. We use regression to build such a regression estimation model; then the model is used to predict the expected co2 emission for a new or unknown car.

- Simple Regression:
  - Simple Linear Regression
  - Simple Non-linear Regression

  Predict co2emission vs EngineSize of all cars
- Multiple Regression:
  - Multiple Linear Regression
  - Multiple Non-linear Regression

  Predict co2emission vs EngineSize and Cylinders of all cars

Basically there are two types of regression models simple regression and multiple regression. Simple regression is when one independent variable is used to estimate a dependent variable. It can be either linear or non-linear. For example, predicting co2 emission using the variable of engine size. Linearity of regression is based on the nature of relationship between independent and dependent variables. When more than one independent variable is present the process is called multiple linear regression. For example, predicting co2 emission using engine size and the number of cylinders in any given car. Again, depending on the relation between dependent and independent variables it can be either linear or non-linear regression.



Let's examine some sample applications of regression.

Essentially we use regression when we want to estimate a continuous value. For instance, one of the applications of regression analysis could be in the area of sales forecasting. You can try to predict a sales person's total yearly sales from independent variables such as age, education, and years of experience. It can also be used in the field of psychology, for example, to determine individual satisfaction, based on demographic and psychological factors. We can use regression analysis to predict the price of a house in an area, based on its size number of bedrooms, and so on. We can even use it to predict employment income for independent variables such as hours of work, education, occupation, sex, age, years of experience, and so on. Indeed, you can find many examples of the usefulness of regression analysis in these and many other fields or domains, such as finance, healthcare, retail, and more.

Simple Linear Regression:

Here, we'll be covering linear regression. You don't need to know any linear algebra to understand topics in linear regression. This high-level introduction will give you enough background information on linear regression to be able to use it effectively on your own problems.

Let's take a look at this data set. It's related to the Co2 emission of different cars. It includes engine size, cylinders, fuel consumption and Co2 emissions for various car models. The question is, given this data set, can we predict the Co2 emission of a car using another field such as engine size? Quite simply, yes. We can use linear regression to predict a continuous value such as Co2 emission by using other variables. Linear regression is the approximation of a linear model used to describe the relationship between two or more variables.

X: Independent variable

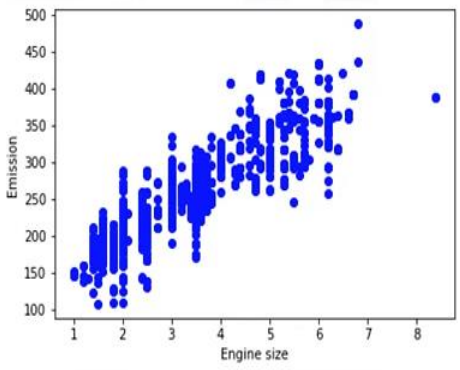
Y: Dependent variable

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

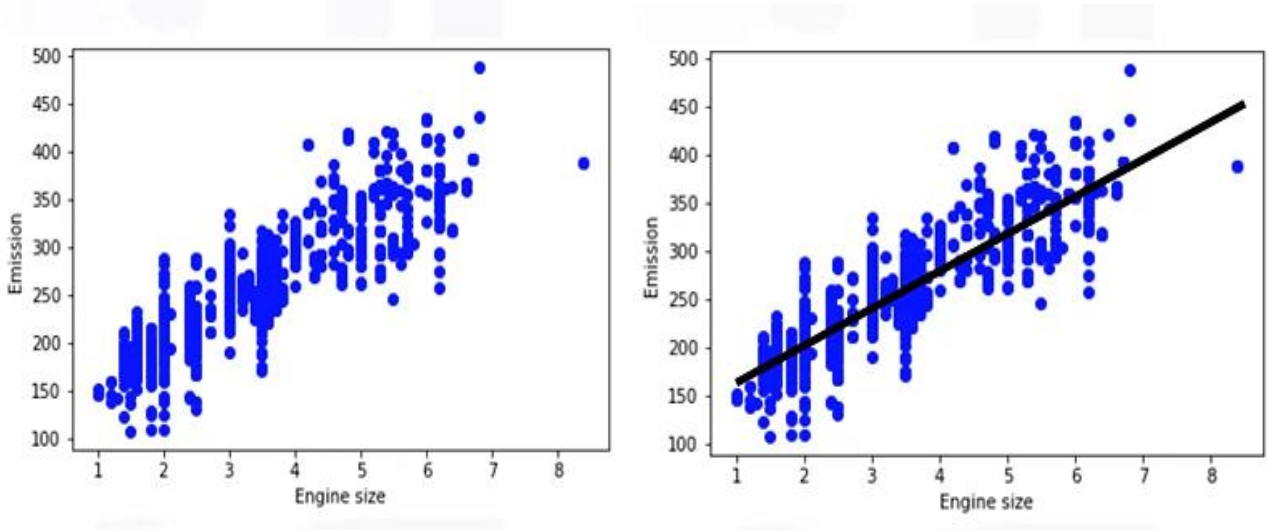
Continuous Values

In simple linear regression, there are two variables, a dependent variable and an independent variable. The key point in the linear regression is that our dependent value should be continuous and cannot be a discrete value. However, the independent variables can be measured on either a categorical or continuous measurement scale. There are two types of linear regression models. They are simple regression and multiple regression. Simple linear regression is when one independent variable is used to estimate a dependent variable. For example, predicting Co2 emission using the engine size variable. When more than one independent variable is present the process is called multiple linear regression, for example, predicting Co2 emission using engine size and cylinders of cars. Our focus in this lecture is on simple linear regression.

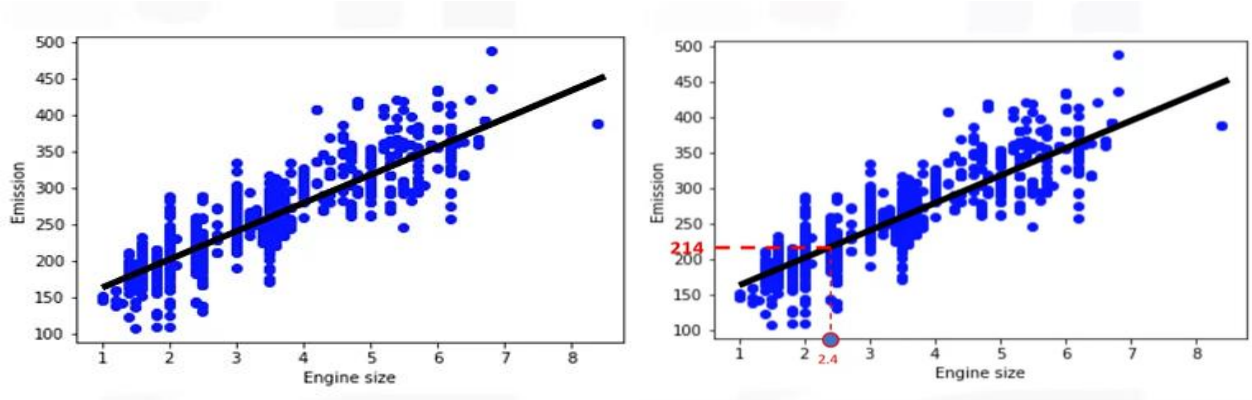
	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?



Now let's see how linear regression works. Okay, so let's look at our data set again. To understand linear regression, we can plot our variables here. We show engine size as an independent variable and emission as the target value that we would like to predict. A scatter plot clearly shows the relation between variables where changes in one variable explain or possibly cause changes in the other variable. Also, it indicates that these variables are linearly related.

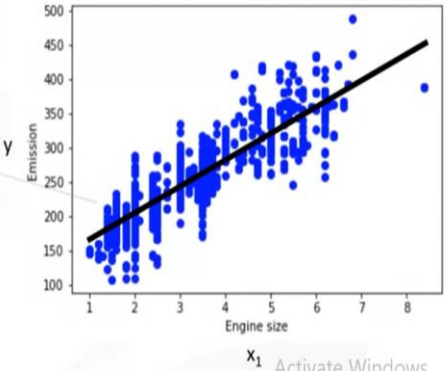


With linear regression you can fit a line through the data. For instance, as the engine size increases, so do the emissions. With linear regression you can model the relationship of these variables. A good model can be used to predict what the approximate emission of each car is. How do we use this line for prediction now? Let us assume for a moment that the line is a good fit of the data. We can use it to predict the emission of an unknown car. For example, for a sample car with engine size 2.4, you can find the emission is 214.



Now, let's talk about what the fitting line actually is. We're going to predict the target value y. In our case using the independent variable engine size represented by x1. The fit line is shown traditionally as a polynomial. In a simple regression problem, a single x, the form of the model would be theta 0 plus theta 1 x1. In this equation, y hat is the dependent variable of the predicted value. And x1 is the independent variable. Theta 0 and theta 1 are the parameters of the line that we must adjust. Theta 1 is known as the slope or gradient of the fitting line and theta 0 is known as the intercept. Theta 0 and theta 1 are also called the coefficients of the linear equation.

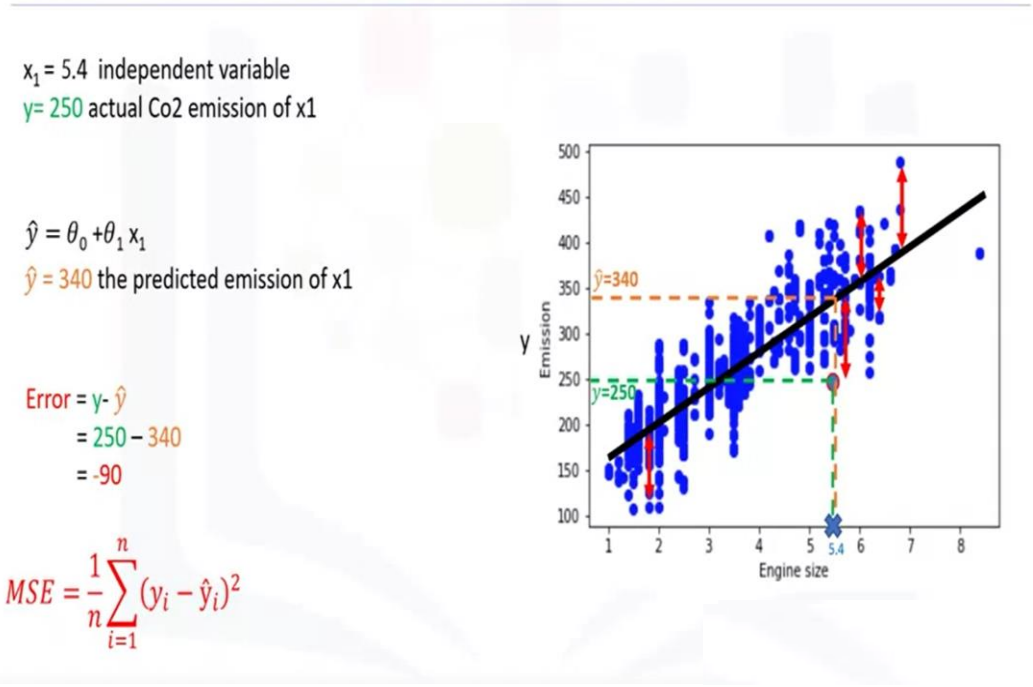
$$\hat{y} = \theta_0 + \theta_1 x_1$$



- **y hat** is the dependent variable
- **x1** is the independent variable
- **Theta 0** and **Theta 1** are the parameters or coefficients of the line
- **Theta 1** is known as the slope or gradient
- **Theta 0** is known as the intercept
-

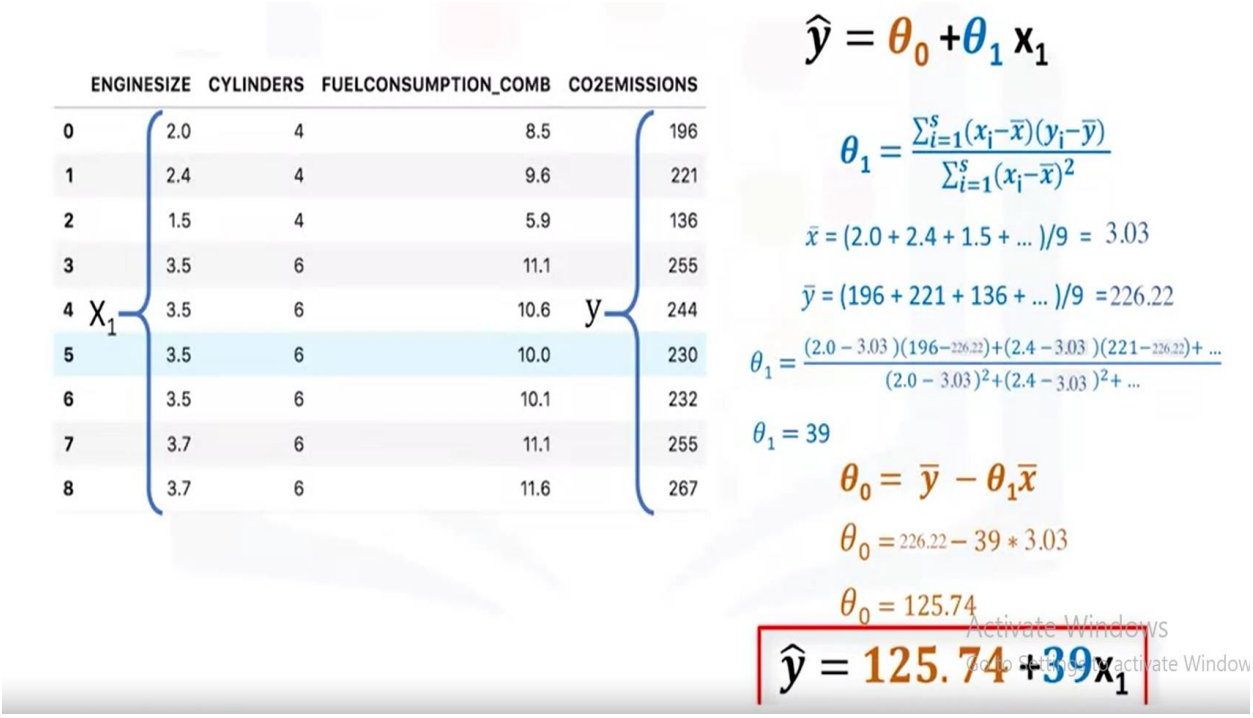
You can interpret this equation as y hat being a function of x1, or y hat being dependent of x1. How would you draw a line through the points? And how do you determine which line fits best? Linear regression estimates the coefficients of the line. This means we must calculate theta 0 and theta 1 to find the best line to fit the data. This line would best estimate the emission of the unknown data points. Let's see how we can find this line or, to be more precise, how we can adjust the parameters to make the line the best fit for the data. For a moment, let's assume we've already found the best fit line for our data. Now, let's go through all the points and check how well they align with this line. Best fit here means that if we have, for instance, a car with engine size x1 = 5.4 and actual Co2 = 250, its Co2 should be predicted very close to the actual value, which is y = 250 based on historical data. But if we use the fit line, or better to say using our polynomial with known parameters to predict the Co2 emission, it will return y hat = 340. Now if you compare the actual value of the emission of the car with what we've predicted using our model, you will find out that we have a 90 unit error. This means our prediction line is not accurate. This error is also

called the residual error. So we can say the error is the distance from the data point to the fitted regression line.



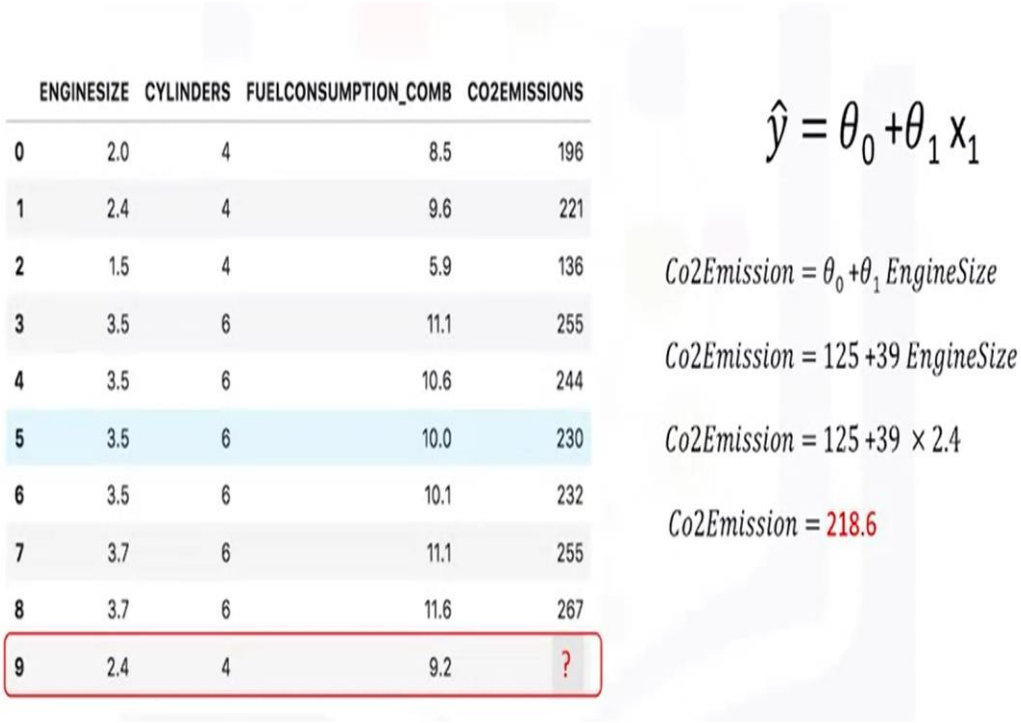
The mean of all residual errors shows how poorly the line fits with the whole data set. Mathematically it can be shown by the equation Mean Squared Error, shown as MSE. Our objective is to find a line where the mean of all these errors is minimized. In other words, the mean error of the prediction using the fit line should be minimized. Let's reword it more technically. The objective of linear regression, is to minimize this MSE equation and to minimize it, we should find the best parameters theta 0 and theta 1. Now the question is how to find theta 0 and theta 1 in such a way that it minimizes this error? How can we find such a perfect line? Or said another way, how should we find the best parameters for our line? Should we move the line a lot randomly and calculate the MSE value every time and choose the minimum one? Not really. Actually, we have two options here. Option one, we can use a mathematic approach, or option two, we can use an optimization approach. Let's see how we could easily use a mathematic formula to find the theta 0 and as mentioned before, theta 0 and theta 1 in the simple linear regression are the coefficients of the fit line. We can use a simple equation to estimate these coefficients. That is, given that it's a simple linear regression with only two parameters, and knowing that theta 0 and theta 1 are the intercept and slope of the line, we can estimate them directly from our data. It requires that we

calculate the mean of the independent and dependent or target columns from the data set. Notice that all of the data must be available to traverse and calculate the parameters.



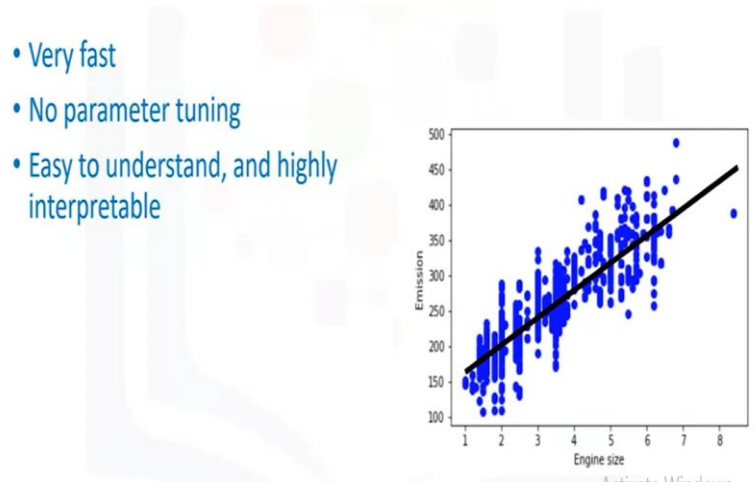
It can be shown that the intercept and slope can be calculated using these equations. We can start off by estimating the value for theta 1. This is how you can find the slope of a line based on the data. X bar is the average value for the engine size in our data set. Please consider that we have nine rows here, rows 0 to 8. First we calculate the average of x1 and of y, then we plug it into the slope equation to find theta 1. The xi and yi in the equation refer to the fact that we need to repeat these calculations across all values in our data set. And i refers to the ith value of x or y. Applying all values, we find theta 1 equals 39. It is our second parameter. It is used to calculate the first parameter which is the intercept of the line. Now we can plug theta 1 into the line equation to find theta 0. It is easily calculated hat theta 0 equals 125.74. So these are the two parameters for the line, where theta 0 is also called the bias coefficient, and theta 1 is the coefficient for the Co2 emission column. As a side note, you really don't need to remember the formula for calculating these parameters, as most of the libraries used for machine learning in Python, R and Scala can easily find these parameters for you. But it's always good to understand how it works. Now, we can write down the polynomial of the line. So we know how to find the best fit for our data and its

equation. Now the question is how can we use it to predict the emission of a new car based on its engine size?



	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

After we found the parameters of the linear equation, making predictions is as simple as solving the equation for a specific set of inputs. Imagine we are predicting Co2 emission, or y, from engine size, or x for the automobile in record number 9. Our linear regression model representation for this problem would be y hat= theta 0 + theta 1 x1. Or if we map it to our data set, it would be Co2Emission =theta 0 + theta 1 Engine Size. As we saw, we can find theta 0, theta 1 using the equations that we just talked about. Once found, we can plug in the equation of the linear model. For example, let's use theta 0 = 125 and theta 1 = 39. So we can rewrite the linear model as Co2 Emission equals 125 plus 39 Engine Size. Now let's plug in the 9th row of our data set and calculate the Co2 emission for a car with an engine size of 2.4. So Co2Emission = 125 + 39 x 2.4. Therefore, we can predict that the Co2Emission for this specific car would be 218.6.

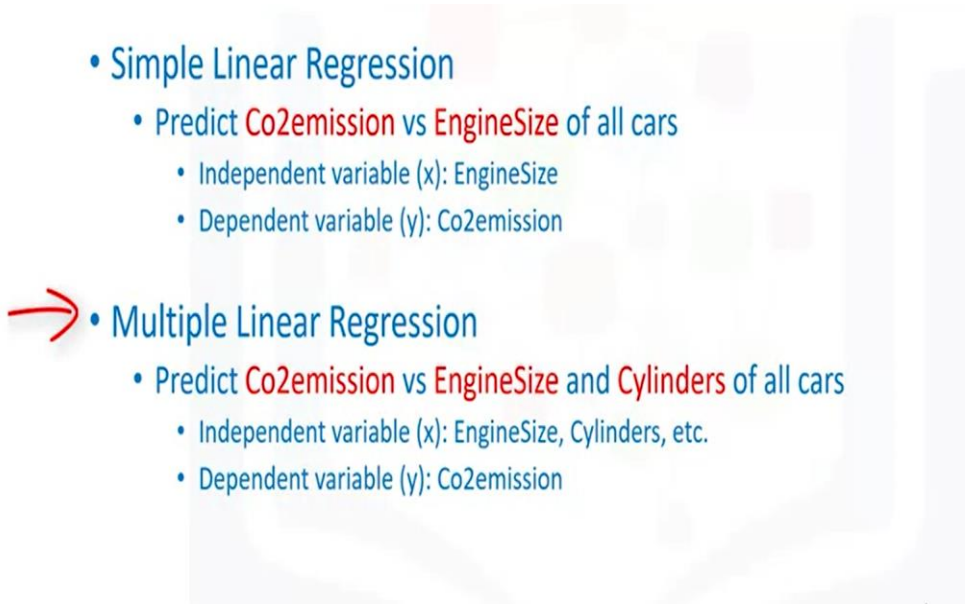


Let's talk a bit about why linear regression is so useful. Quite simply, it is the most basic regression to use and understand. In fact, one reason why linear regression is so useful is that it's fast. It also doesn't require tuning of parameters. So something like tuning the K parameter and K nearest neighbors, or the learning rate in neural networks isn't something to worry about. Linear regression is also easy to understand, and highly interpretable.

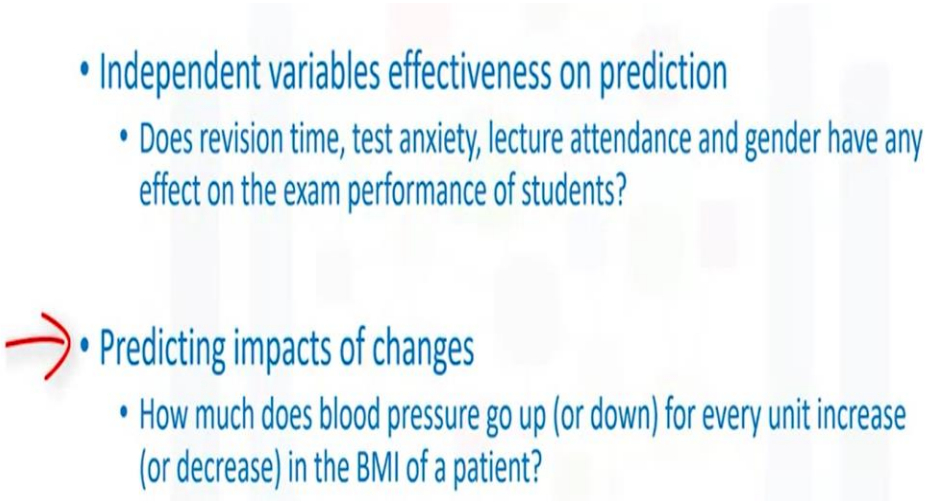
Multiple Linear Regression:

Here, we'll be covering multiple linear regression. As you know, there are two types of linear regression models, simple regression and multiple regression. Simple linear regression is when one independent variable is used to estimate a dependent variable. For example, predicting CO_2 emission using the variable of engine size. In reality, there are multiple variables that predict the CO_2 emission. When multiple independent variables are present, the process is called multiple linear regression. For example, predicting CO_2 emission using engine size and the number of cylinders in the car's engine. Our focus in this lecture is on multiple linear regression.

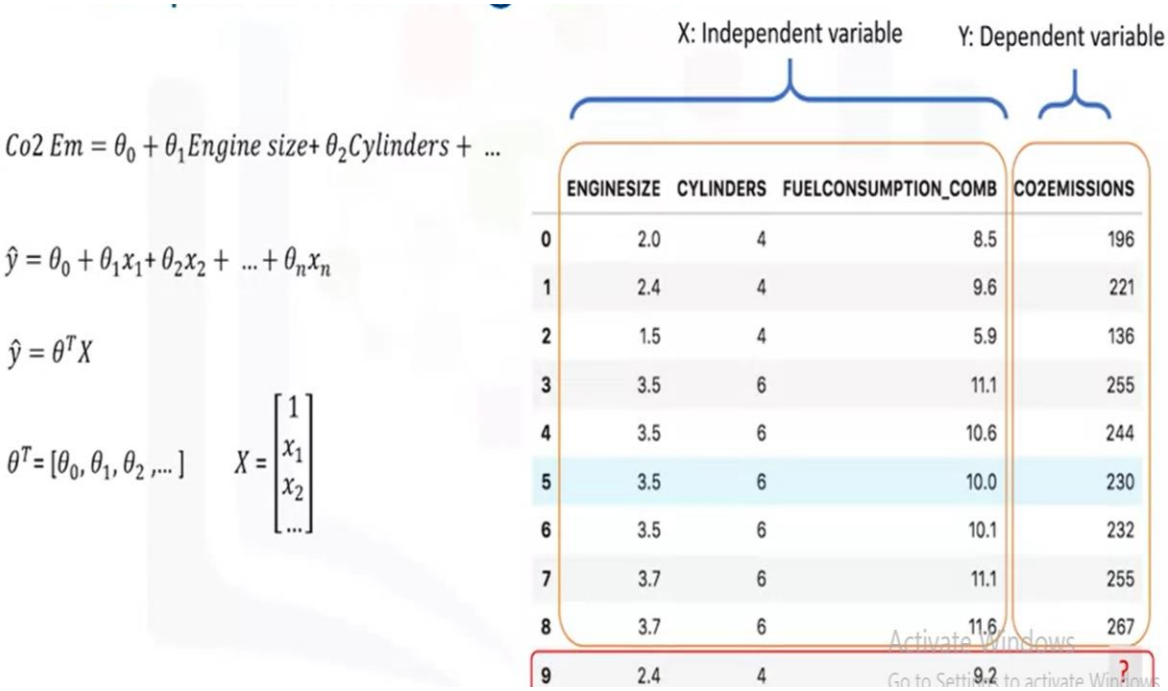
The good thing is that multiple linear regression is the extension of the simple linear regression model. So, I suggest you go through the simple linear regression first. Before we dive into a sample dataset and see how multiple linear regression works, I want to tell you what kind of problems it can solve, when we should use it, and specifically, what kind of questions we can answer using it. Basically, there are two applications for multiple linear regression.



First, it can be used when we would like to identify the strength of the effect that the independent variables have on the dependent variable. For example, does revision time, test anxiety, lecture attendance and gender have any effect on exam performance of students? Second, it can be used to predict the impact of changes, that is, to understand how the dependent variable changes when we change the independent variables. For example, if we were reviewing a person's health data, a multiple linear regression can tell you how much that person's blood pressure goes up or down for every unit increase or decrease in a patient's body mass index holding other factors constant. As is the case with simple linear regression, multiple linear regression is a method of predicting a continuous variable.

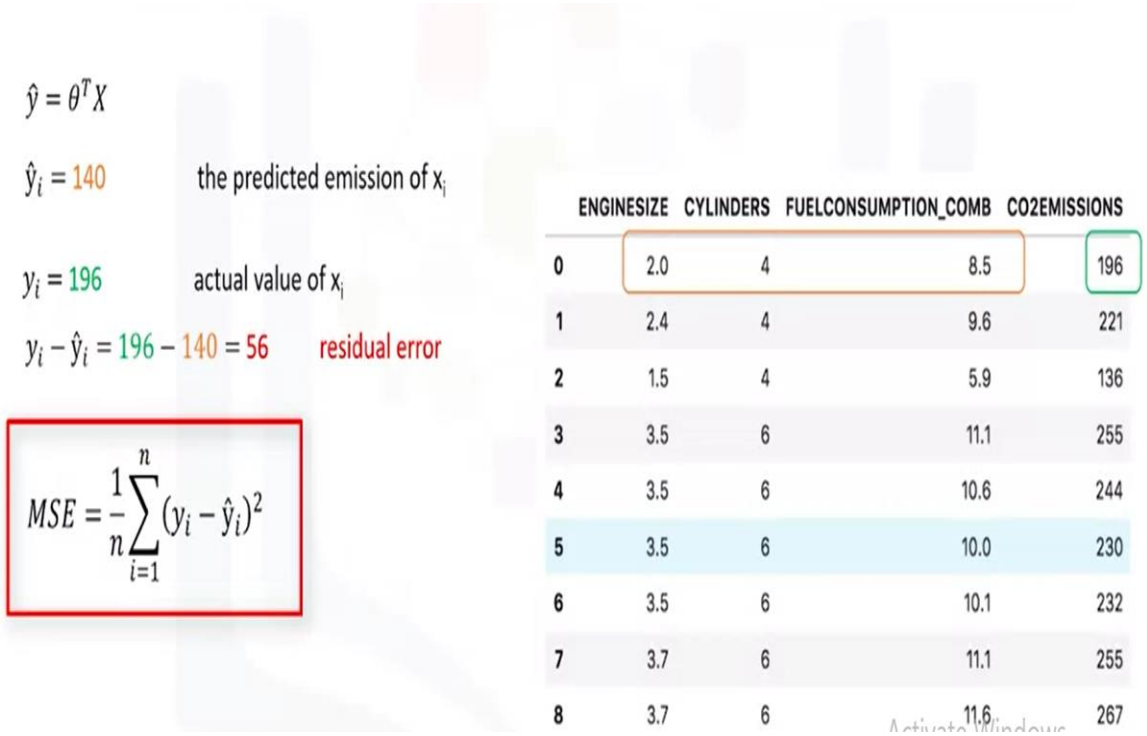


It uses multiple variables called independent variables or predictors that best predict the value of the target variable which is also called the dependent variable. In multiple linear regression, the target value Y, is a linear combination of independent variables X. For example, you can predict how much CO_2 a car might admit due to independent variables such as the car's engine size, number of cylinders, and fuel consumption. Multiple linear regression is very useful because you can examine which variables are significant predictors of the outcome variable. Also, you can find out how each feature impacts the outcome variable. Again, as is the case in simple linear regression, if you manage to build such a regression model, you can use it to predict the emission amount of an unknown case such as record number nine. Generally, the model is of the form y hat equals theta zero, plus theta one x_1, plus theta two x_2 and so on, up to theta n x_n. mathematically, we can show it as a vector form as well. This means it can be shown as a dot product of two vectors; the parameters vector and the feature set vector. Generally, we can show the equation for a multidimensional space as theta transpose x, where theta is an n by one vector of unknown parameters in a multi-dimensional space, and x is the vector of the featured sets, as theta is a vector of coefficients and is supposed to be multiplied by x.



Conventionally, it is shown as transpose theta. Theta is also called the parameters or weight vector of the regression equation. Both these terms can be used interchangeably, and x is the feature set which represents a car. For example, x_1 for engine size or x_2 for cylinders, and so on. The first

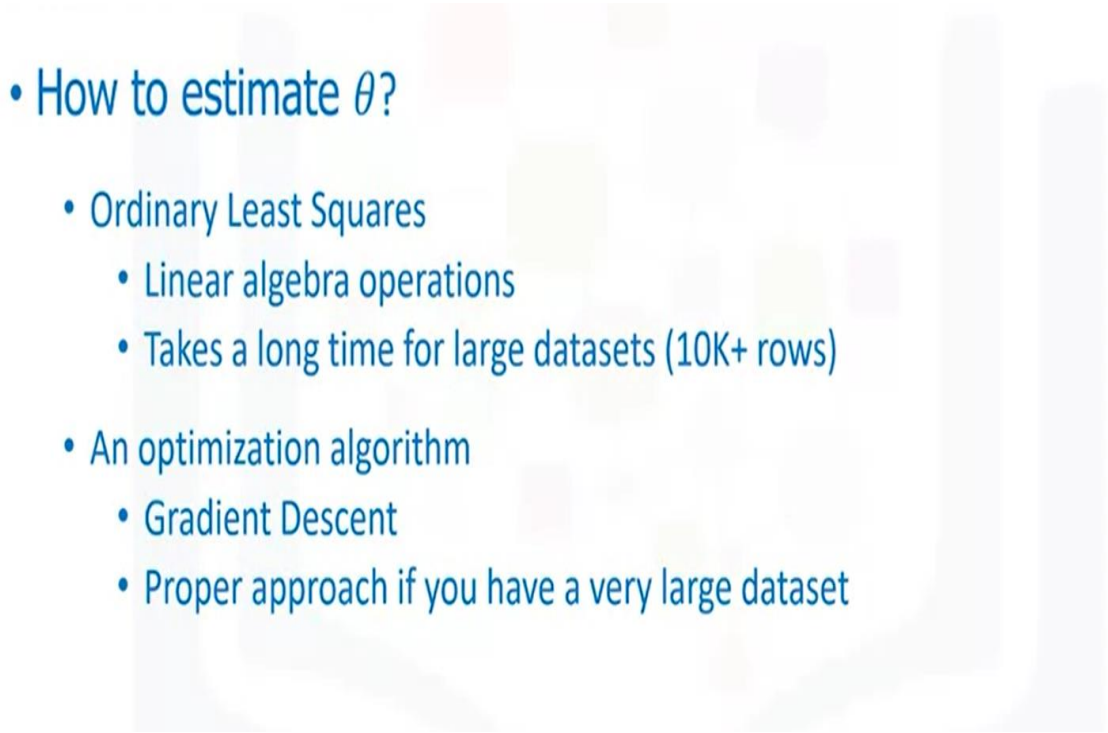
element of the feature set would be set to one, because it turns that theta zero into the intercept or biased parameter when the vector is multiplied by the parameter vector. Please notice that theta transpose x in a one-dimensional space is the equation of a line, it is what we use in simple linear regression. In higher dimensions when we have more than one input or x the line is called a plane or a hyperplane, and this is what we use for multiple linear regression. So, the whole idea is to find the best fit hyperplane for our data. To this end and as is the case in linear regression, we should estimate the values for theta vector that best predict the value of the target field in each row. To achieve this goal, we have to minimize the error of the prediction. Now, the question is, how do we find the optimized parameters?



	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

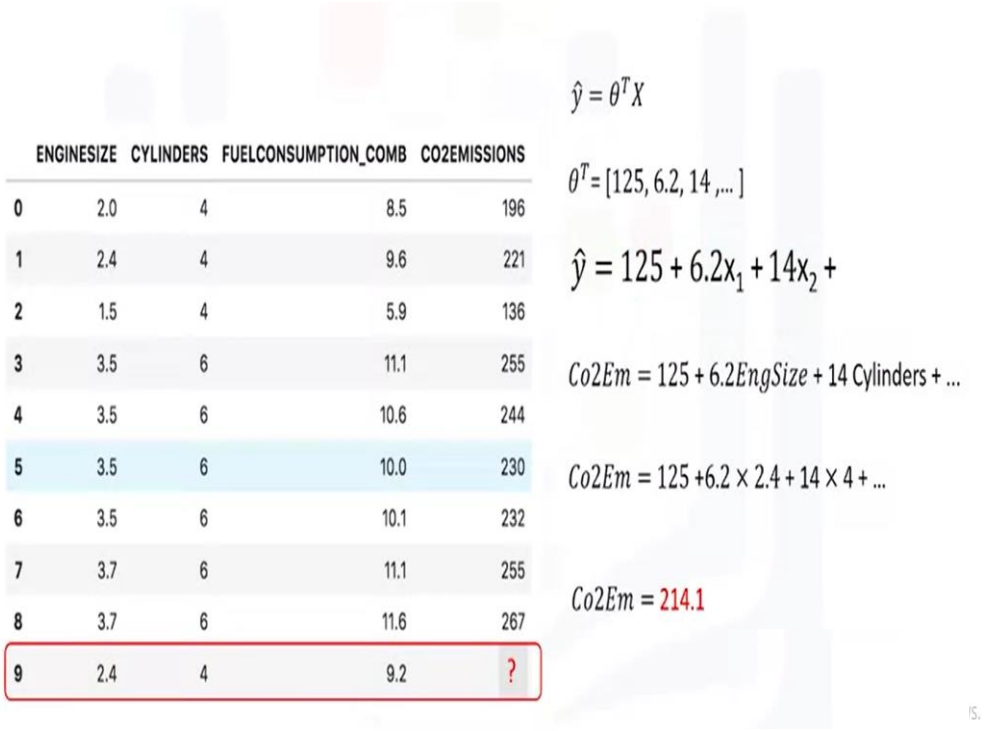
To find the optimized parameters for our model, we should first understand what the optimized parameters are, then we will find a way to optimize the parameters. In short, optimized parameters are the ones which lead to a model with the fewest errors. Let's assume for a moment that we have already found the parameter vector of our model, it means we already know the values of theta vector. Now we can use the model and the feature set of the first row of our dataset to predict the CO_2 emission for the first car, correct? If we plug the feature set values into the model equation, we find y hat. Let's say for example, it returns 140 as the predicted value for this specific row,

what is the actual value? Y equals 196. How different is the predicted value from the actual value of 196? Well, we can calculate it quite simply as 196 subtract 140, which of course equals 56. This is the error of our model only for one row or one car in our case. As is the case in linear regression, we can say the error here is the distance from the data point to the fitted regression model. The mean of all residual errors shows how bad the model is representing the data set, it is called the mean squared error, or MSE. Mathematically, MSE can be shown by an equation. While this is not the only way to expose the error of a multiple linear regression model, it is one of the most popular ways to do so. The best model for our data set is the one with minimum error for all prediction values. So, the objective of multiple linear regression is to minimize the MSE equation.



To minimize it, we should find the best parameters theta, but how? Okay, how do we find the parameter or coefficients for multiple linear regression? There are many ways to estimate the value of these coefficients. However, the most common methods are the ordinary least squares and optimization approach. Ordinary least squares tries to estimate the values of the coefficients by minimizing the mean square error. This approach uses the data as a matrix and uses linear algebra operations to estimate the optimal values for the theta. The problem with this technique is the time complexity of calculating matrix operations as it can take a very long time to finish. When the number of rows in your data set is less than 10,000, you can think of this technique as an option.

However, for greater values, you should try other faster approaches. The second option is to use an optimization algorithm to find the best parameters. That is, you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data. For example, you can use gradient descent which starts optimization with random values for each coefficient, then calculates the errors and tries to minimize it through y's changing of the coefficients in multiple iterations. Gradient descent is a proper approach if you have a large data set. Please understand however, that there are other approaches to estimate the parameters of the multiple linear regression that you can explore on your own. After you find the best parameters for your model, you can go to the prediction phase.



	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

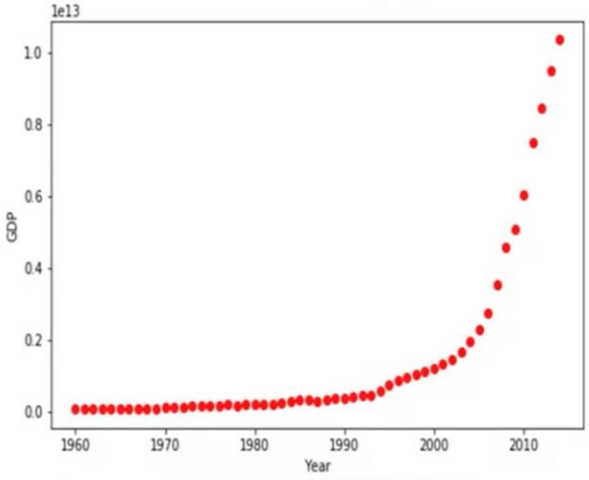
After we found the parameters of the linear equation, making predictions is as simple as solving the equation for a specific set of inputs. Imagine we are predicting CO_2 emission or Y from other variables for the automobile in record number nine. Our linear regression model representation for this problem would be y hat equals theta transpose x. Once we find the parameters, we can plug them into the equation of the linear model. For example, let's use theta zero equals 125, theta one equals 6.2, theta two equals 14, and so on. If we map it to our data set, we can rewrite the linear model as CO_2 emissions equals 125 plus 6.2 multiplied by engine size, plus 14 multiplied by cylinder, and so on. As you can see, multiple linear regression estimates the relative importance of

predictors. For example, it shows cylinder has higher impact on CO_2 emission amounts in comparison with engine size. Now, let's plug in the ninth row of our data set and calculate the CO_2 emission for a car with the engine size of 2.4. So, CO_2 emission equals 125 plus 6.2 times 2.4, plus 14 times four, and so on. We can predict the CO_2 emission for this specific car would be 214.1. Now, let me address some concerns that you might already be having regarding multiple linear regression. As you saw, you can use multiple independent variables to predict a target value in multiple linear regression. It sometimes results in a better model compared to using a simple linear regression which uses only one independent variable to predict the dependent variable. Now the question is how, many independent variable should we use for the prediction? Should we use all the fields in our data set? Does adding independent variables to a multiple linear regression model always increase the accuracy of the model? Basically, adding too many independent variables without any theoretical justification may result in an overfit model. An overfit model is a real problem because it is too complicated for your data set and not general enough to be used for prediction. So, it is recommended to avoid using many variables for prediction. There are different ways to avoid overfitting a model in regression, however that is outside the scope of this lecture. The next question is, should independent variables be continuous? Basically, categorical independent variables can be incorporated into a regression model by converting them into numerical variables. For example, given a binary variables such as car type, the code dummy zero for manual and one for automatic cars. As a last point, remember that multiple linear regression is a specific type of linear regression. So, there needs to be a linear relationship between the dependent variable and each of your independent variables. There are a number of ways to check for linear relationship. For example, you can use scatter plots and then visually checked for linearity. If the relationship displayed in your scatter plot is not linear, then you need to use non-linear regression.

Non-Linear Regression:

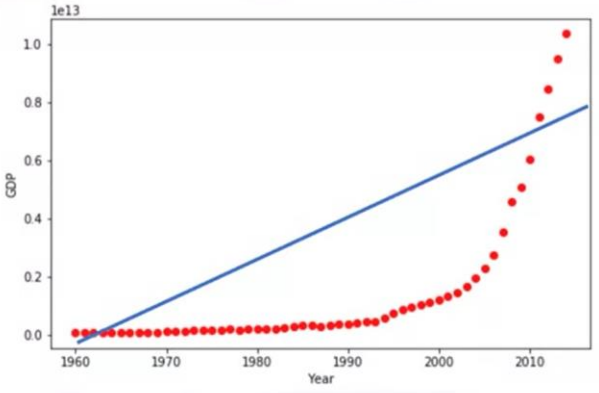
Here, we'll be discussing non-linear regression basics. So, let's get started. These data points correspond to China's gross domestic product or GDP from 1960-2014. The first column is the years and the second is China's corresponding annual gross domestic income in US dollars for that year. This is what the data points look like. Now, we have a couple of interesting questions.

	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10
...	.....	.....



Should we use linear regression?

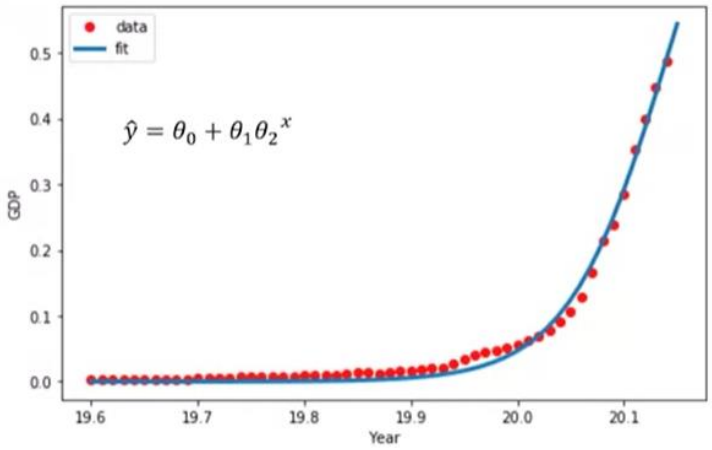
	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10
...	.....	.....



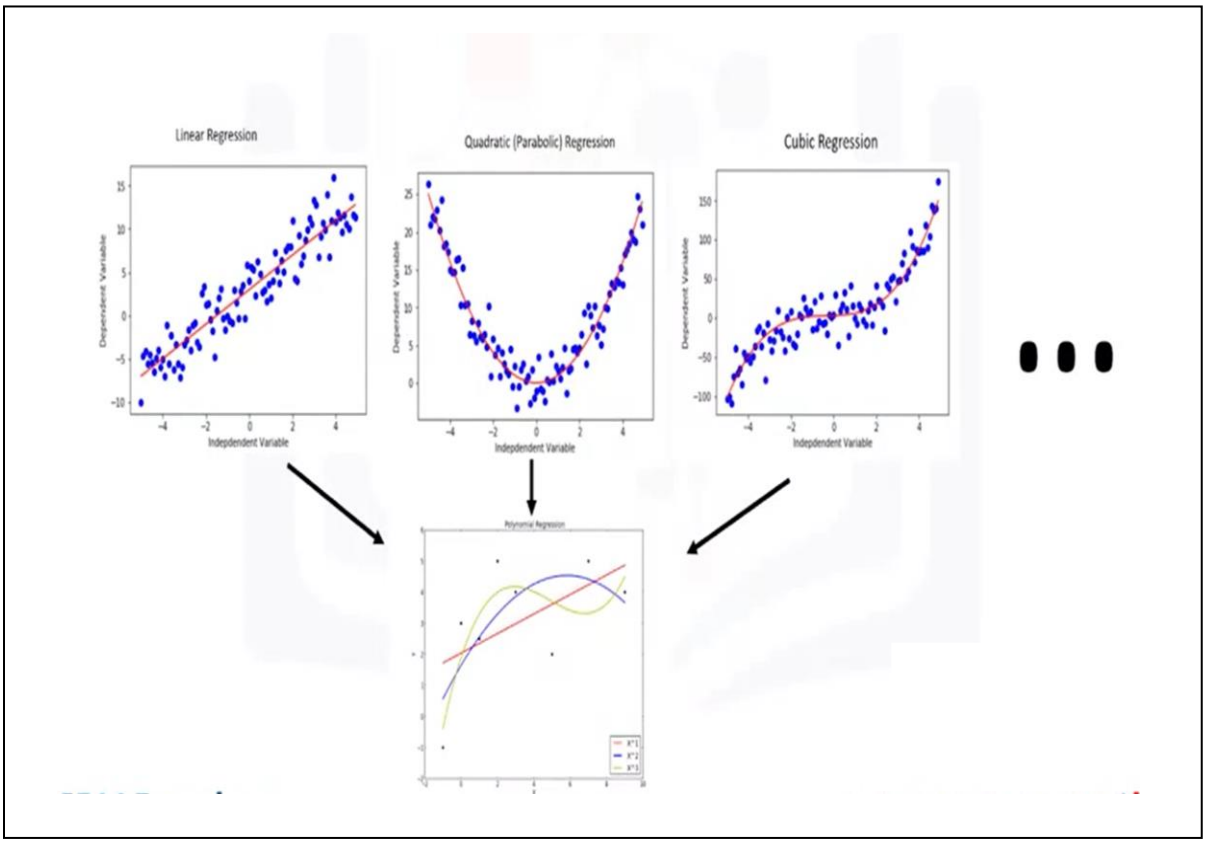
China GDP

First, can GDP be predicted based on time? Second, can we use a simple linear regression to model it? Indeed. If the data shows a curvy trend, then linear regression would not produce very accurate results when compared to a non-linear regression. Simply because, as the name implies, linear regression presumes that the data is linear. The scatter plot shows that there seems to be a strong relationship between GDP and time, but the relationship is not linear. As you can see, the growth starts off slowly, then from 2005 onward, the growth is very significant. Finally, it decelerates slightly in the 2010s. It looks like either a logistical or exponential function.

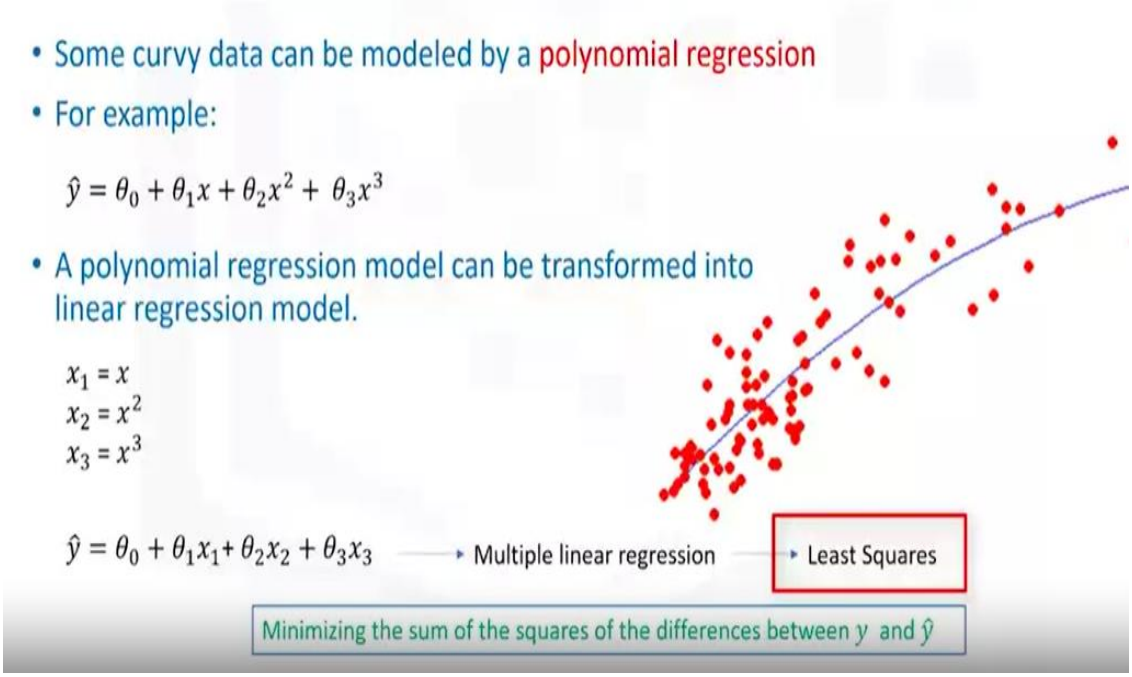
	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10
...	.....	.....



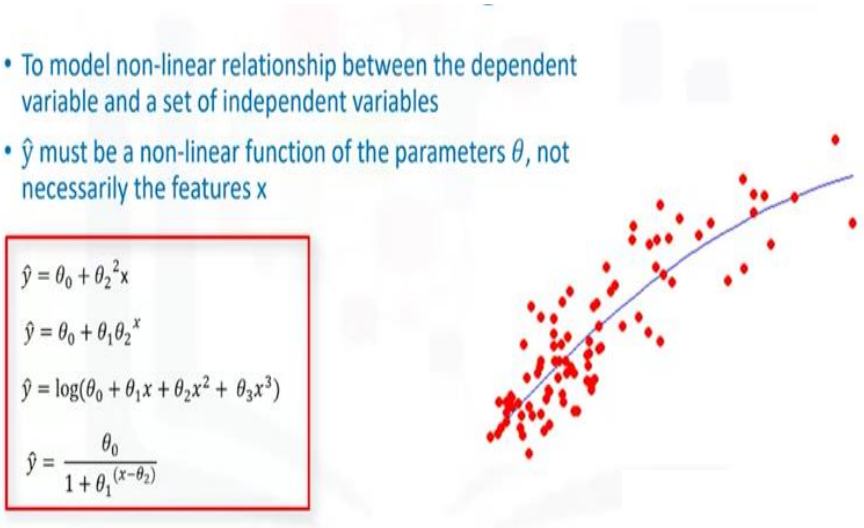
So, it requires a special estimation method of the non-linear regression procedure. For example, if we assume that the model for these data points are exponential functions, such as Y hat equals Theta zero plus Theta one Theta two transpose X or to the power of X, our job is to estimate the parameters of the model, i.e., Thetas, and use the fitted model to predict GDP for unknown or future cases. In fact, many different regressions exists that can be used to fit whatever the dataset looks like. You can see a quadratic and cubic regression lines here, and it can go on and on to infinite degrees. In essence, we can call all of these polynomial regression, where the relationship between the independent variable X and the dependent variable Y is modeled as an nth degree polynomial in X. With many types of regression to choose from, there's a good chance that one will fit your dataset well. Remember, it's important to pick a regression that fits the data the best. So, what is polynomial regression? Polynomial regression fits a curve line to your data. A simple example of polynomial with degree three is shown as Y hat equals Theta zero plus Theta 1_X plus Theta 2_X squared plus Theta 3_X cubed or to the power of three, where Thetas are parameters to be estimated that makes the model fit perfectly to the underlying data. Though the relationship between X and Y is non-linear here and polynomial regression can't fit them, a polynomial regression model can still be expressed as linear regression.



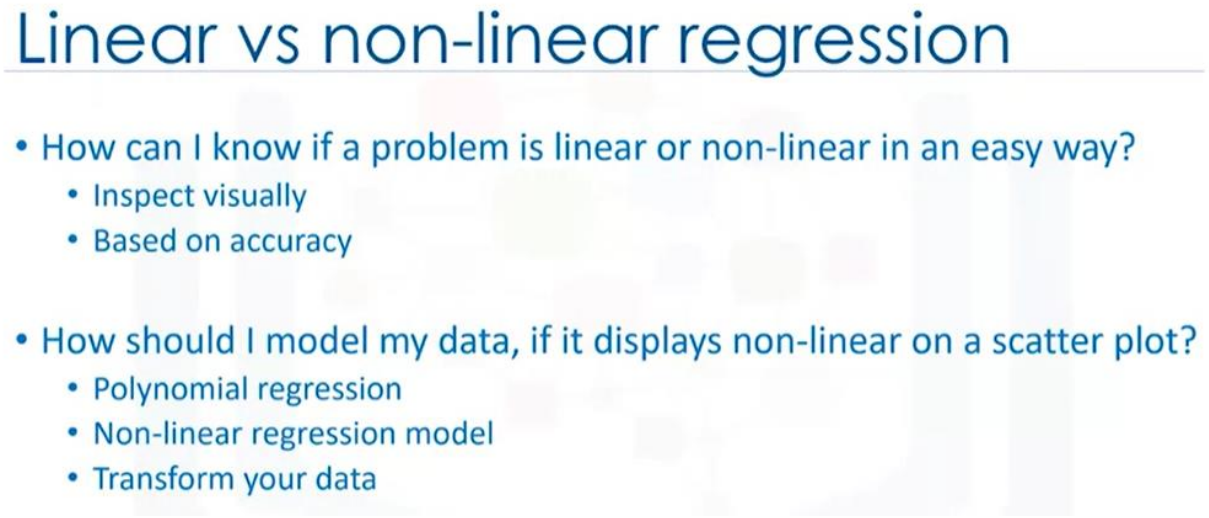
I know it's a bit confusing, but let's look at an example. Given the third degree polynomial equation, by defining X_1 equals X and X_2 equals X squared or X to the power of two and so on, the model is converted to a simple linear regression with new variables as Y hat equals Theta zero plus Theta one X_1 plus Theta two X_2 plus Theta three X_3. This model is linear in the parameters to be estimated, right? Therefore, this polynomial regression is considered to be a special case of traditional multiple linear regression. So, you can use the same mechanism as linear regression to solve such a problem. Therefore, polynomial regression models can fit using the model of least squares. Least squares is a method for estimating the unknown parameters in a linear regression model by minimizing the sum of the squares of the differences between the observed dependent variable in the given dataset and those predicted by the linear function. So, what is non-linear regression exactly?



First, non-linear regression is a method to model a non-linear relationship between the dependent variable and a set of independent variables. Second, for a model to be considered non-linear, Y hat must be a non-linear function of the parameters Theta, not necessarily the features X. When it comes to non-linear equation, it can be the shape of exponential, logarithmic, and logistic, or many other types. As you can see in all of these equations, the change of Y hat depends on changes in the parameters Theta, not necessarily on X only.



That is, in non-linear regression, a model is non-linear by parameters. In contrast to linear regression, we cannot use the ordinary least squares method to fit the data in non-linear regression. In general, estimation of the parameters is not easy. Let me answer two important questions here. First, how can I know if a problem is linear or non-linear in an easy way? To answer this question, we have to do two things. The first is to visually figure out if the relation is linear or non-linear. It's best to plot bivariate plots of output variables with each input variable. Also, you can calculate the correlation coefficient between independent and dependent variables, and if, for all variables, it is 0.7 or higher, there is a linear tendency and thus, it's not appropriate to fit a non-linear regression.



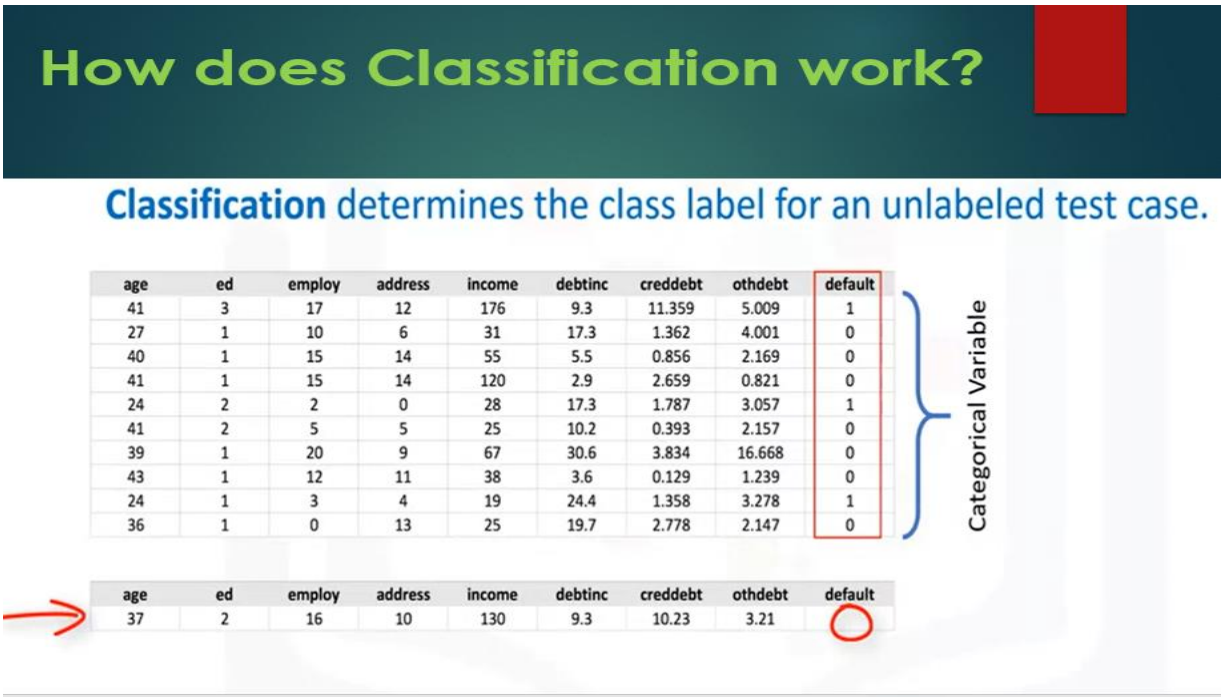
The second thing we have to do is to use non-linear regression instead of linear regression when we cannot accurately model the relationship with linear parameters. The second important question is, how should I model my data if it displays non-linear on a scatter plot? Well, to address this, you have to use either a polynomial regression, use a non-linear regression model, or transform your data, which is not in scope for this course.

What is Classification?

Here, we'll give you an introduction to classification. So let's get started. In machine learning classification is a supervised learning approach which can be thought of as a means of categorizing or classifying some unknown items into a discrete set of classes. Classification attempts to learn

the relationship between a set of feature variables and a target variable of interest. The target attribute in classification is a categorical variable with discrete values.

- A supervised learning approach
- Categorizing some unknown items into a discrete set of categories or "classes"
- The target attribute is a categorical variable

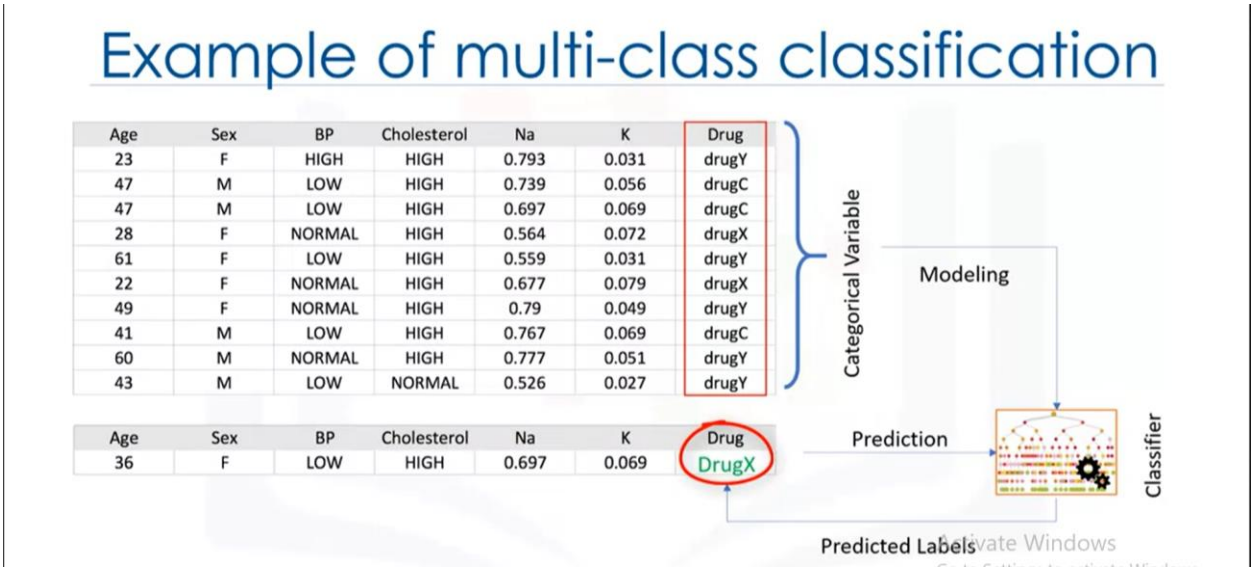


age	ed	employ	address	income	debtinc	creddebt	othdebt	default
41	3	17	12	176	9.3	11.359	5.009	1
27	1	10	6	31	17.3	1.362	4.001	0
40	1	15	14	55	5.5	0.856	2.169	0
41	1	15	14	120	2.9	2.659	0.821	0
24	2	2	0	28	17.3	1.787	3.057	1
41	2	5	5	25	10.2	0.393	2.157	0
39	1	20	9	67	30.6	3.834	16.668	0
43	1	12	11	38	3.6	0.129	1.239	0
24	1	3	4	19	24.4	1.358	3.278	1
36	1	0	13	25	19.7	2.778	2.147	0

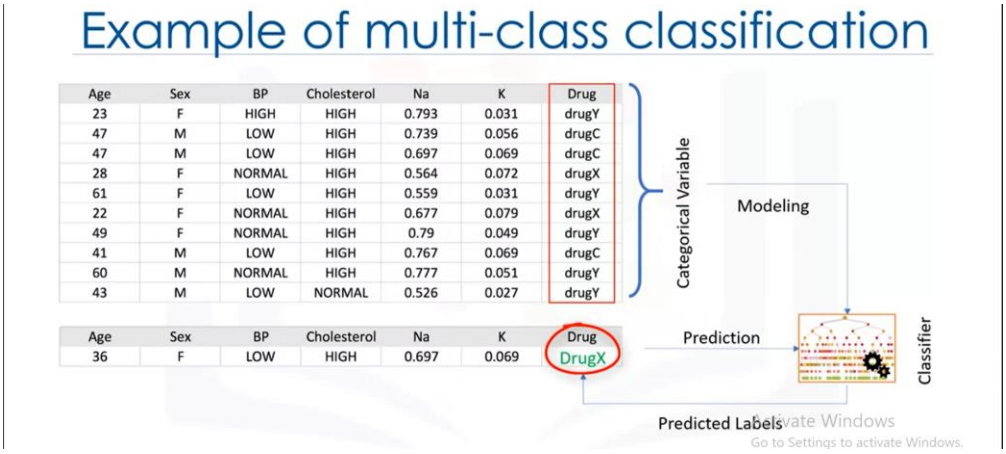
age	ed	employ	address	income	debtinc	creddebt	othdebt	default
37	2	16	10	130	9.3	10.23	3.21	

So, how does classification and classifiers work? Given a set of training data points along with the target labels, classification determines the class label for an unlabeled test case. Let's explain this with an example. A good sample of classification is the loan default prediction. Suppose a bank is concerned about the potential for loans not to be repaid? If previous loan default data can be used to predict which customers are likely to have problems repaying loans, these bad risk customers can either have their loan application declined or offered alternative products. The goal of a loan default predictor is to use existing loan default data which has information about the customers such as age, income, education et cetera, to build a classifier, pass a new customer or potential

future default to the model, and then label it, i.e. the data points as defaulter or not defaulter. Or for example zero or one. This is how a classifier predicts an unlabeled test case. Please notice that this specific example was about a binary classifier with two values. We can also build classifier models for both binary classification and multi-class classification.



For example, imagine that you've collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of three medications. You can use this labeled dataset with a classification algorithm to build a classification model. Then you can use it to find out which drug might be appropriate for a future patient with the same illness. As you can see, it is a sample of multi-class classification.

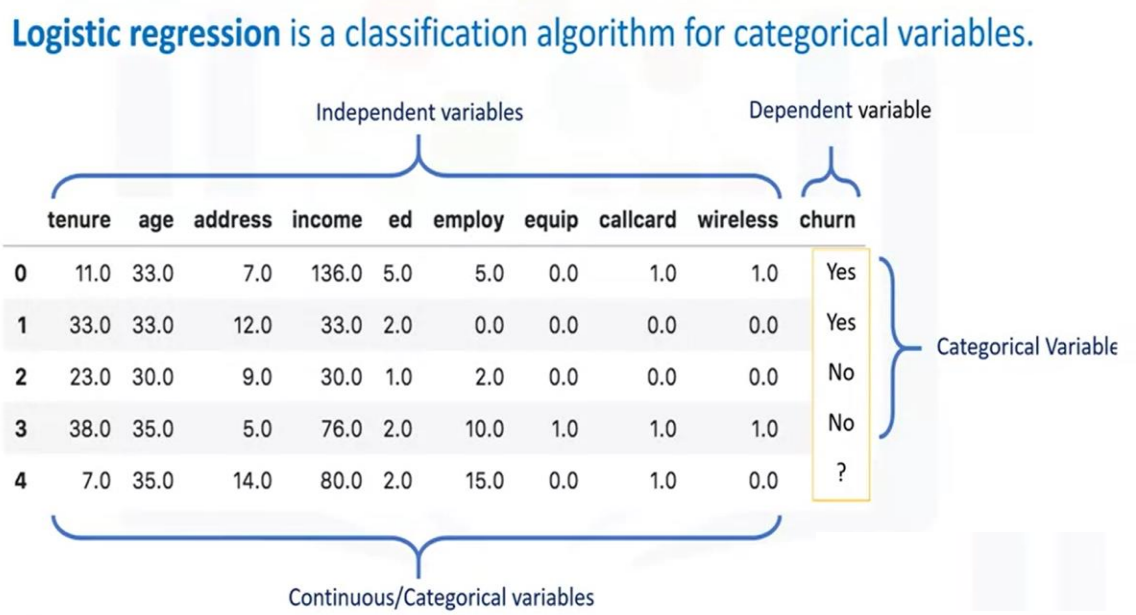


Classification has different business use cases as well. For example, to predict the category to which a customer belongs, for churn detection where we predict whether a customer switches to another provider or brand, or to predict whether or not a customer responds to a particular advertising campaign.

Data classification has several applications in a wide variety of industries. Essentially, many problems can be expressed as associations between feature and target variables, especially when labelled data is available. This provides a broad range of applicability for classification. For example, classification can be used for email filtering, speech recognition, handwriting recognition, biometric identification, document classification and much more. Here we have the types of classification algorithms and machine learning. They include decision trees, naive bayes, linear discriminant analysis, k-nearest neighbor, logistic regression, neural networks, and support vector machines. There are many types of classification algorithms. We will only cover a few in this course.

Logistic Regression:

In this Lecture, we'll learn a machine learning method called Logistic Regression which is used for classification. In examining this method, we'll specifically answer these three questions. What is logistic regression? What kind of problems can be solved by logistic regression?

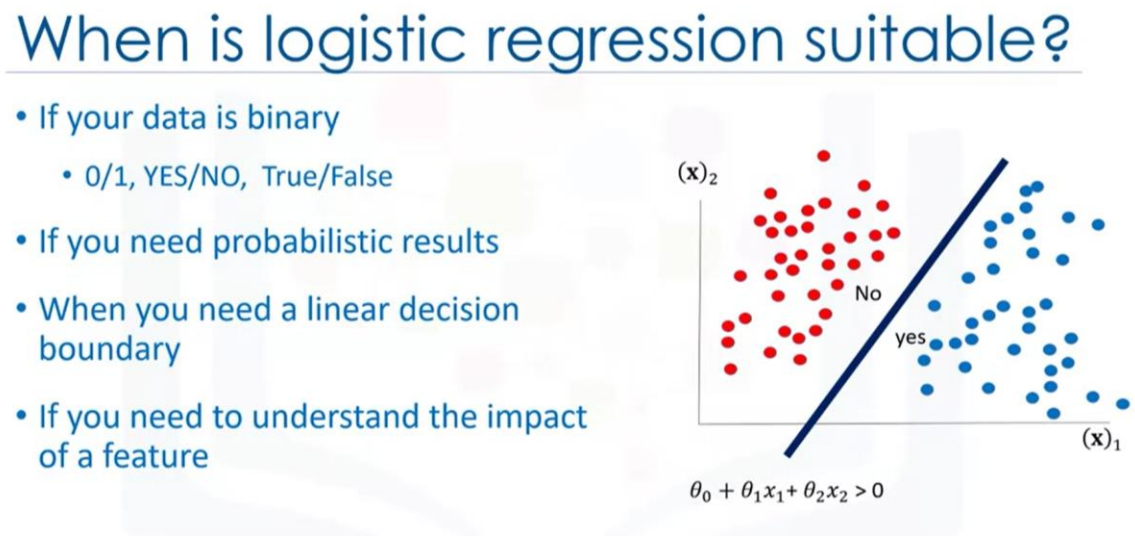


In which situations do we use logistic regression? Logistic regression is a statistical and machine learning technique for classifying records of a dataset based on the values of the input fields. Let's say we have a telecommunication dataset that we'd like to analyze in order to understand which customers might leave us next month. This is historical customer data where each row represents one customer. Imagine that you're an analyst at this company and you have to find out who is leaving and why? You'll use the dataset to build a model based on historical records and use it to predict the future churn within the customer group. The dataset includes information about services that each customer has signed up for, customer account information, demographic information about customers like gender and age range and also customers who've left the company within the last month. The column is called churn. We can use logistic regression to build a model for predicting customer churn using the given features. In logistic regression, we use one or more independent variables such as tenure, age, and income to predict an outcome, such as churn, which we call the dependent variable representing whether or not customers will stop using the service. Logistic regression is analogous to linear regression but tries to predict a categorical or discrete target field instead of a numeric one. In linear regression, we might try to predict a continuous value of variables such as the price of a house, blood pressure of a patient, or fuel consumption of a car. But in logistic regression, we predict a variable which is binary such as yes/no, true/false,

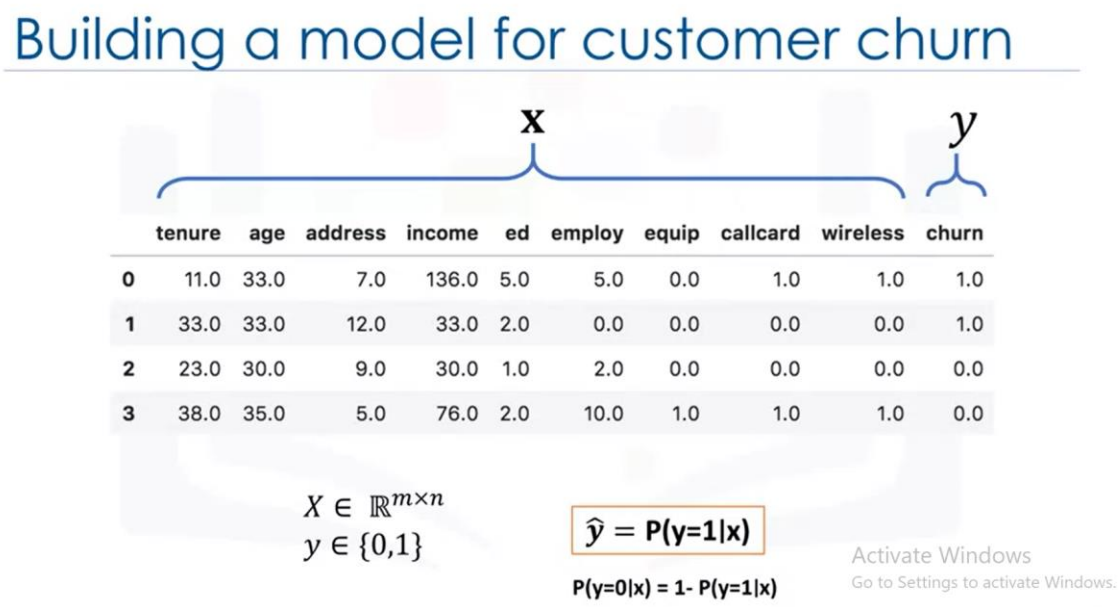
successful or not successful, pregnant/not pregnant, and so on, all of which can be coded as zero or one.

In logistic regression independent variables should be continuous. If categorical, they should be dummy or indicator coded. This means we have to transform them to some continuous value. Please note that logistic regression can be used for both binary classification and multi-class classification. But for simplicity in this lecture, we'll focus on binary classification. Let's examine some applications of logistic regression before we explain how they work. As mentioned, logistic regression is a type of classification algorithm, so it can be used in different situations. For example, to predict the probability of a person having a heart attack within a specified time period, based on our knowledge of the person's age, sex, and body mass index. Or to predict the chance of mortality in an injured patient or to predict whether a patient has a given disease such as diabetes based on observed characteristics of that patient such as weight, height, blood pressure, and results of various blood tests and so on. In a marketing context, we can use it to predict the likelihood of a customer purchasing a product or halting a subscription as we've done in our churn example. We can also use logistic regression to predict the probability of failure of a given process, system or product. We can even use it to predict the likelihood of a homeowner defaulting on a mortgage. These are all good examples of problems that can be solved using logistic regression. Notice that in all these examples not only do we predict the class of each case, we also measure the probability of a case belonging to a specific class. There are different machine algorithms which can classify or estimate a variable. The question is, when should we use logistic regression? Here are four situations in which logistic regression is a good candidate. First, when the target field in your data

is categorical or specifically is binary. Such as zero/one, yes/no, churn or no churn, positive/negative and so on.



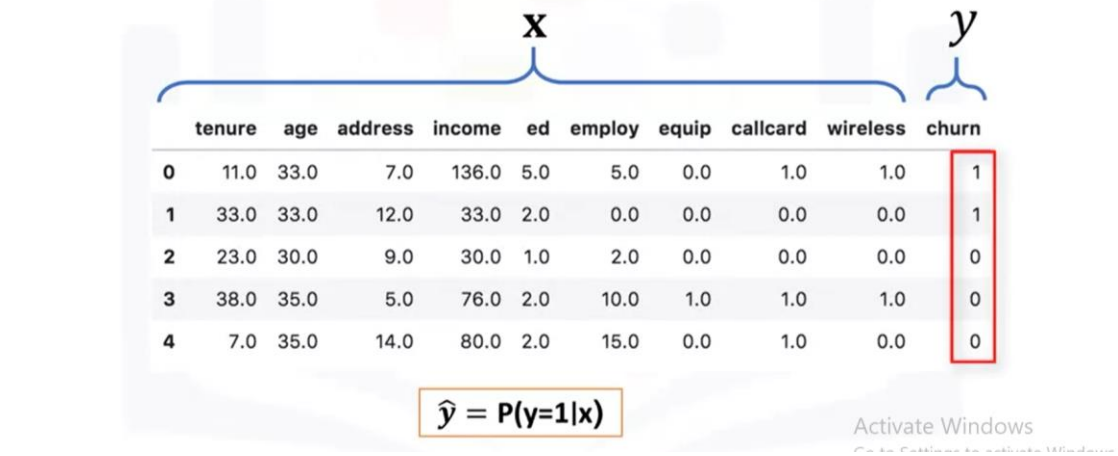
Second, you need the probability of your prediction. For example, if you want to know what the probability is of a customer buying a product. Logistic regression returns a probability score between zero and one for a given sample of data. In fact, logistic regression predicts the probability of that sample and we map the cases to a discrete class based on that probability. Third, if your data is linearly separable. The decision boundary of logistic regression is a line or a plane or a hyper plane. A classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class. For example, if we have just two features and are not applying any polynomial processing we can obtain an inequality like Theta zero plus Theta 1x1 plus theta 2x2 is greater than zero, which is a half-plane easily plottable. Please note that in using logistic regression, we can also achieve a complex decision boundary using polynomial processing as well, which is out of scope here. You'll get more insight from decision boundaries when you understand how logistic regression works. Fourth, you need to understand the impact of a feature. You can select the best features based on the statistical significance of the logistic regression model coefficients or parameters. That is, after finding the optimum parameters, a feature X with the weight Theta one close to zero has a smaller effect on the prediction than features with large absolute values of Theta one. Indeed, it allows us to understand the impact an independent variable has on the dependent variable while controlling other independent variables. Let's look at our dataset again.



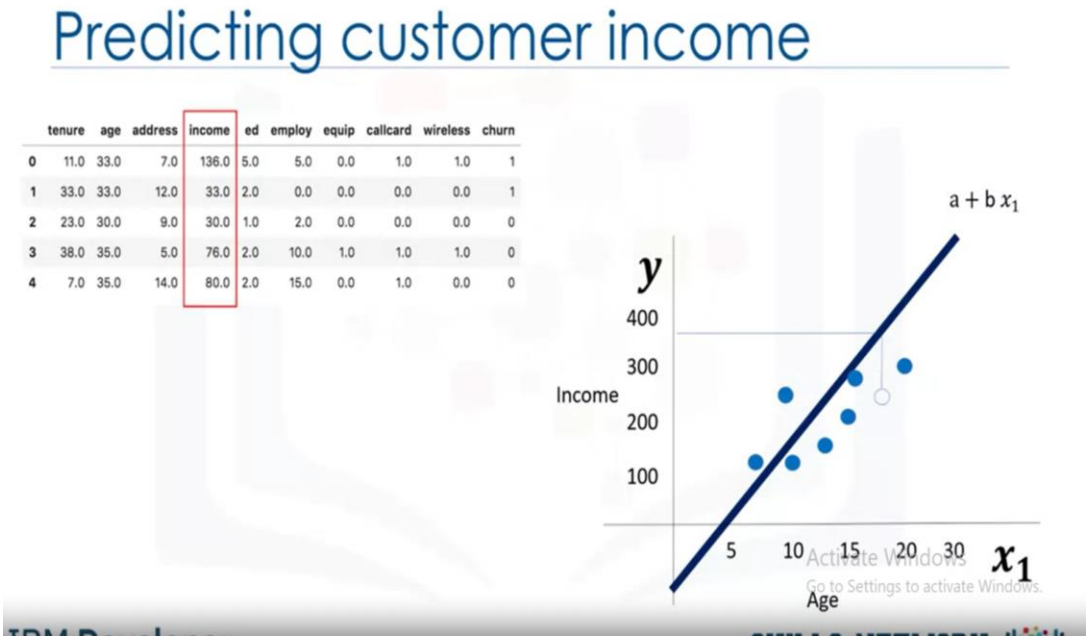
We defined the independent variables as X and dependent variable as Y. Notice, that for the sake of simplicity we can code the target or dependent values to zero or one. The goal of logistic regression is to build a model to predict the class of each sample which in this case is a customer, as well as the probability of each sample belonging to a class. Given that, let's start to formalize the problem. X is our dataset in the space of real numbers of m by n. That is, of m dimensions or features and n records, and Y is the class that we want to predict, which can be either zero or one. Ideally, a logistic regression model, so-called Y hat, can predict that the class of the customer is one, given its features X. It can also be shown quite easily that the probability of a customer being in class zero can be calculated as one minus the probability that the class of the customer is one.

Linear Regression Vs Logistic Regression:

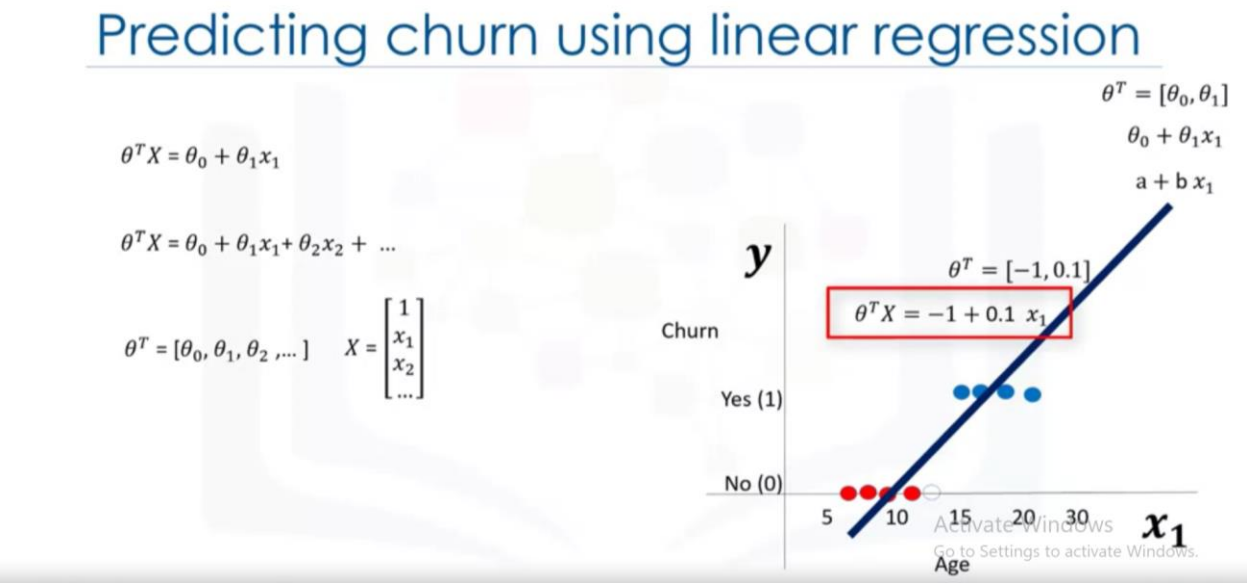
Here, we will learn the difference between linear regression and logistic regression. We go over linear regression and see why it cannot be used properly for some binary classification problems. We also look at the sigmoid function, which is the main part of logistic regression. Let's look at the telecommunication dataset again. The goal of logistic regression is to build a model to predict the class of each customer and also the probability of each sample belonging to a class. Ideally, we want to build a model, y hat, that can estimate that the class of a customer is one given its feature is x.



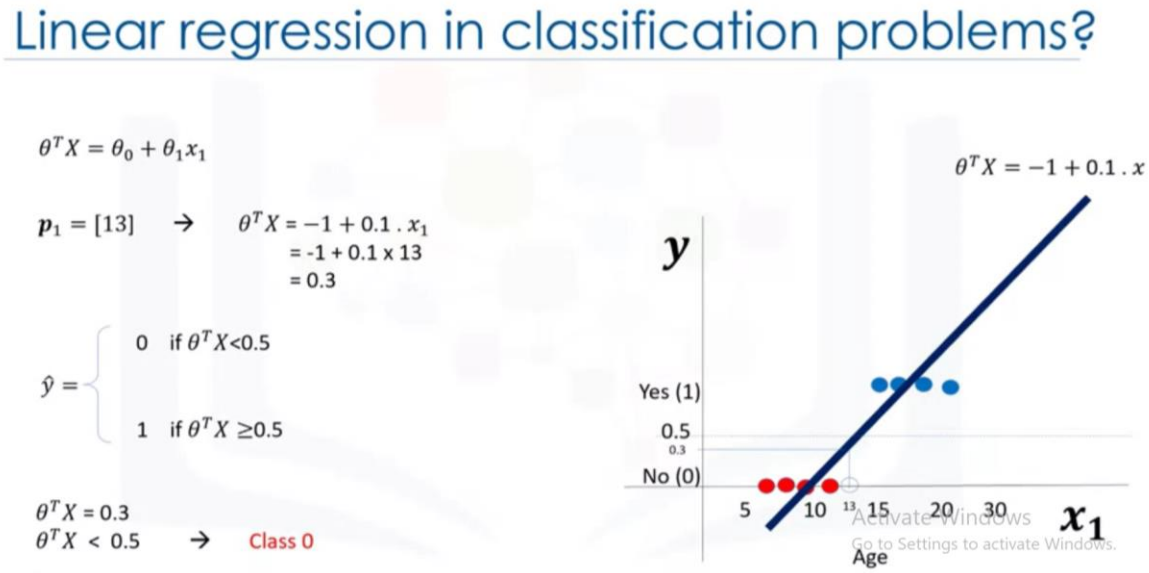
I want to emphasize that y is the label's vector, also called actual values, that we would like to predict, and y hat is the vector of the predicted values by our model. Mapping the class labels to integer numbers, can we use linear regression to solve this problem? First, let's recall how linear regression works to better understand logistic regression. Forget about the churn prediction for a minute and assume our goal is to predict the income of customers in the dataset. This means that instead of predicting churn, which is a categorical value, let's predict income, which is a continuous value. So, how can we do this? Let's select an independent variable such as customer age and predict the dependent variable such as income.



Of course, we can have more features but for the sake of simplicity, let's just take one feature here. We can plot it and show age as an independent variable and income as the target value we would like to predict. With linear regression, you can fit a line or polynomial through the data. We can find this line through training our model or calculating it mathematically based on the sample sets. We'll say, this is a straight line through the sample set. This line has an equation shown as a plus bx1. Now, use this line to predict the continuous value, y. That is, use this line to predict the income of an unknown customer based on his or her age, and it is done. What if we want to predict churn?

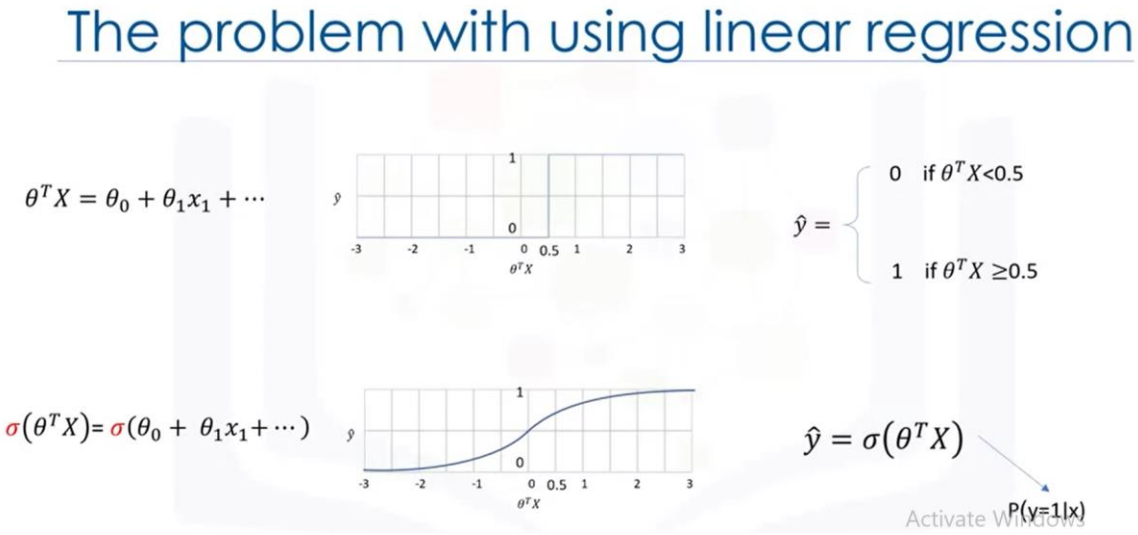


Can we use the same technique to predict a categorical field such as churn? Okay, let's see. Say, we're given data on customer churn and our goal this time is to predict the churn of customers based on their age. We have a feature, age denoted as x1, and a categorical feature, churn, with two classes, churn is yes and churn is no. As mentioned, we can map yes and no to integer values zero and one. How can we model it now? Well, graphically, we could represent our data with a scatterplot, but this time, we have only two values for the y-axis. In this plot, class zero is denoted in red, and class one is denoted in blue. Our goal here is to make a model based on existing data to predict if a new customer is red or blue. Let's do the same technique that we used for linear regression here to see if we can solve the problem for a categorical attribute such as churn. With linear regression, you again can fit a polynomial through the data, which is shown traditionally as a plus bx. This polynomial can also be shown traditionally as Theta0 plus Theta1 x1.



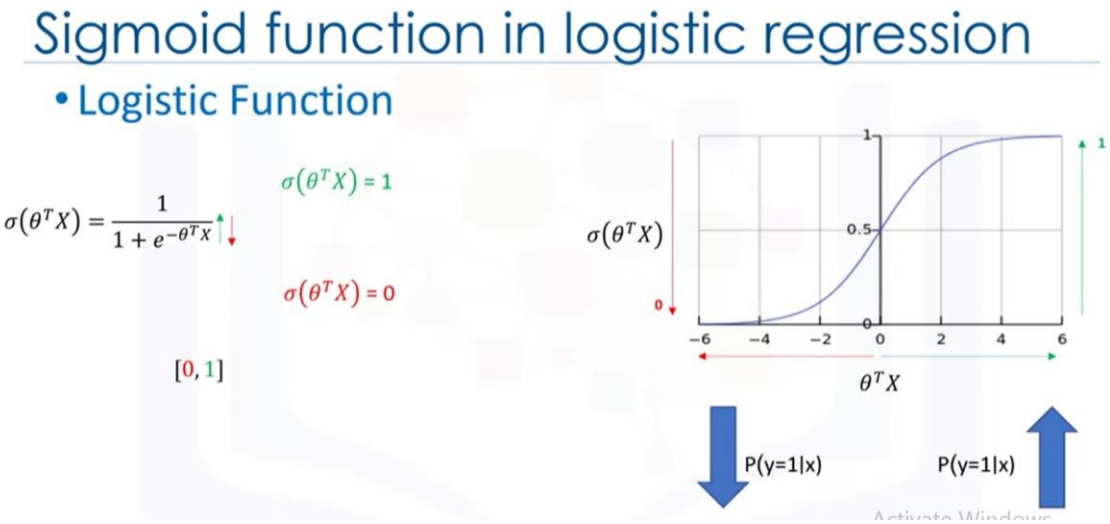
This line has two parameters which are shown with vector Theta where the values of the vector are Theta0 and Theta1. We can also show the equation of this line formally as Theta transpose x. Generally, we can show the equation for a multidimensional space as Theta transpose x, where Theta is the parameters of the line in two-dimensional space or parameters of a plane in three-dimensional space, and so on. As Theta is a vector of parameters and is supposed to be multiplied by x, it is shown conventionally as transpose Theta. Theta is also called the weights factor or confidences of the equation, with both these terms used interchangeably, and X is the feature set which represents a customer. Anyway, given a dataset, all the feature sets x Theta parameters can be calculated through an optimization algorithm or mathematically, which results in the equation of the fitting line. For example, the parameters of this line are minus one and 0.1, and the equation for the line is minus one plus 0.1 x1. Now, we can use this regression line to predict the churn of a new customer. For example, for our customer or, let's say, a data point with x value of age equals 13, we can plug the value into the line formula, and the y value is calculated and returns a number. For instance, for p1 point, we have Theta transpose x equals minus 1 plus 0.1 times x1, equals minus 1 plus 0.1 times 13, equals 0.3. We can show it on our graph. Now, we can define a threshold here. For example, at 0.5 to define the class. So, we write a rule here for our model, y hat, which allows us to separate class zero from class one. If the value of Theta transpose x is less than 0.5, then the class is zero. Otherwise, if the value of Theta transpose x is more than 0.5, then the class is one, and because our customers y value is less than the threshold, we can say it belongs to class

zero based on our model. But there is one problem here. What is the probability that this customer belongs to class zero? As you can see, it's not the best model to solve this problem. Also, there are some other issues which verify that linear regression is not the proper method for classification problems. So, as mentioned, if we use the regression line to calculate the class of a point, it always returns a number such as three or negative two, and so on. Then, we should use a threshold, for example, 0.5, to assign that point to either class of zero or one. This threshold works as a step function that outputs zero or one regardless of how big or small, positive or negative the input is.

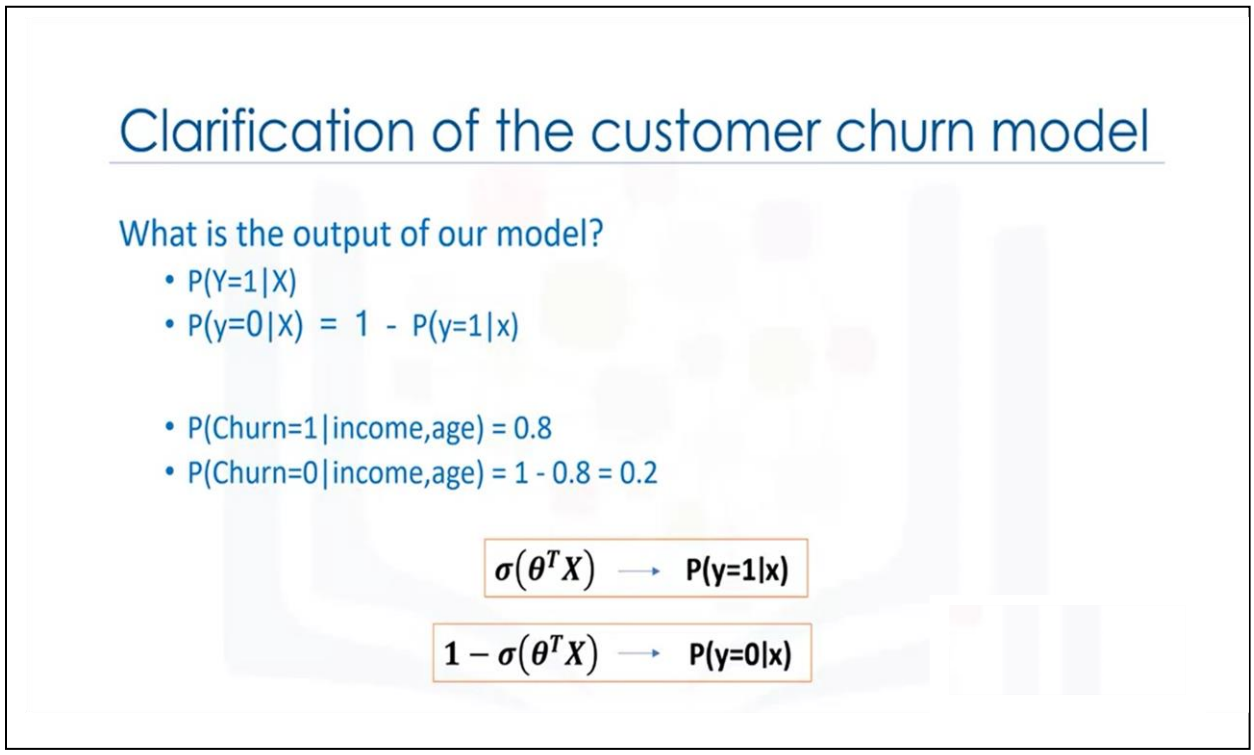


So, using the threshold, we can find the class of a record. Notice that in the step function, no matter how big the value is, as long as it's greater than 0.5, it simply equals one and vice versa. Regardless of how small the value y is, the output would be zero if it is less than 0.5. In other words, there is no difference between a customer who has a value of one or 1,000. The outcome would be one. Instead of having this step function, wouldn't it be nice if we had a smoother line, one that would project these values between zero and one? Indeed, the existing method does not really give us the probability of a customer belonging to a class, which is very desirable. We need a method that can give us the probability of falling in the class as well. So, what is the scientific solution here? Well, if instead of using Theta transpose x, we use a specific function called sigmoid, then sigmoid of Theta transpose x gives us the probability of a point belonging to a class instead of the value of y directly. I'll explain this sigmoid function in a second, but for now, please except that it will do the trick. Instead of calculating the value of Theta transpose x directly, it returns the probability that a

Theta transpose x is very big or very small. It always returns a value between 0 and 1, depending on how large the Theta transpose x actually is.

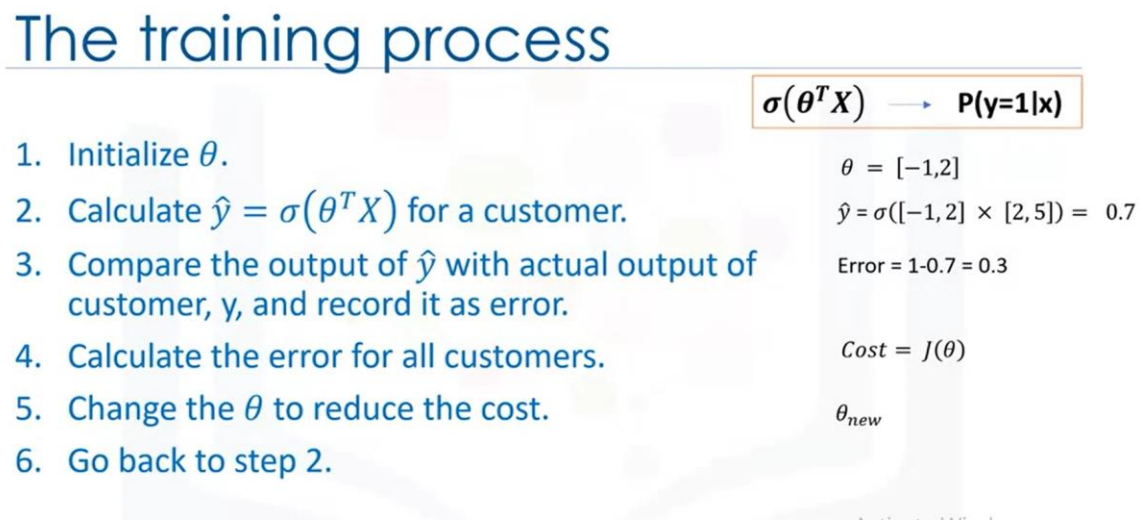


Now, our model is sigmoid of Theta transpose x, which represents the probability that the output is 1 given x. Now, the question is, what is the sigmoid function? Let me explain in detail what sigmoid really is. The sigmoid function, also called the logistic function, resembles the step function and is used by the following expression in the logistic regression. The sigmoid function looks a bit complicated at first, but don't worry about remembering this equation, it'll make sense to you after working with it. Notice that in the sigmoid equation, when Theta transpose x gets very big, the e power minus Theta transpose x in the denominator of the fraction becomes almost 0, and the value of the sigmoid function gets closer to 1. If Theta transpose x is very small, the sigmoid function gets closer to 0. Depicting on the in sigmoid plot, when Theta transpose x gets bigger, the value of the sigmoid function gets closer to 1, and also, if the Theta transpose x is very small, the sigmoid function gets closer to 0. So, the sigmoid functions output is always between 0 and 1, which makes it proper to interpret the results as probabilities. It is obvious that when the outcome of the sigmoid function gets closer to 1, the probability of y equals 1 given x goes up. In contrast, when the sigmoid value is closer to 0, the probability of y equals 1 given x is very small. So what is the output of our model when we use the sigmoid function? In logistic regression, we model the probability that an input, x, belongs to the default class y equals 1, and we can write this formally as probability of y equals 1 given x. We can also write probability of y belongs to class 0 given x is 1 minus probability of y equals 1 given x.



For example, the probability of a customer staying with the company can be shown as probability of churn equals 1 given a customer's income and age, which can be, for instance, 0.8, and the probability of churn is 0 for the same customer given a customer's income and age can be calculated as 1 minus 0.8 equals 0.2. So, now our job is to train the model to set its parameter values in such a way that our model is a good estimate of probability of y equals 1 given x. In fact, this is what a good classifier model built by logistic regression is supposed to do for us. Also, it should be a good estimate of probability of y belongs to class 0 given x that can be shown as 1 minus sigmoid of Theta transpose x. Now, the question is, how can we achieve this? We can find Theta through the training process. So, let's see what the training process is. Step one, initialize Theta vector with random values as with most machine learning algorithms. For example, minus 1 or 2. Step two, calculate the model output, which is sigmoid of Theta transpose x. For example, customer in your training set. X and Theta transpose x is the feature vector values. For example, the age and income of the customer, for instance, 2 and 5, and Theta is the confidence or weight that you've set in the previous step. The output of this equation is the prediction value, in other words, the probability that the customer belongs to class 1. Step three, compare the output of our model, y hat, which could be a value of, let's say, 0.7, with the actual label of the customer, which is for example, 1, for churn. Then, record the difference as our model's error for this customer,

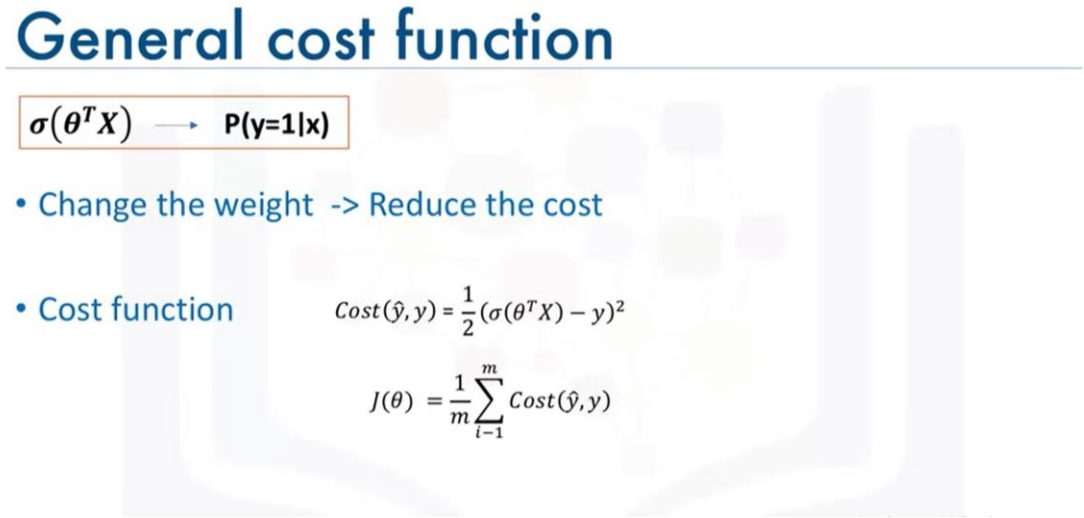
which would be 1 minus 0.7, which of course, equals 0.3. This is the error for only one customer out of all the customers in the training set. Step four, calculate the error for all customers as we did in the previous steps and add up these errors. The total error is the cost of your model and is calculated by the models cost function. The cost function, by the way, basically represents how to calculate the error of the model which is the difference between the actual and the models predicted values.



So, the cost shows how poorly the model is estimating the customers labels. Therefore, the lower the cost, the better the model is at estimating the customers labels correctly. So, what we want to do is to try to minimize this cost. Step five, but because the initial values for Theta were chosen randomly, it's very likely that the cost function is very high, so we change the Theta in such a way to hopefully reduce the total cost. Step six, after changing the values of Theta, we go back to step two, then we start another iteration and calculate the cost of the model again. We keep doing those steps over and over, changing the values of Theta each time until the cost is low enough. So, this brings up two questions. First, how can we change the values of Theta so that the cost is reduced across iterations? Second, when should we stop the iterations? There are different ways to change the values of Theta, but one of the most popular ways is gradient descent. Also, there are various ways to stop iterations, but essentially you stop training by calculating the accuracy of your model and stop it when it's satisfactory.

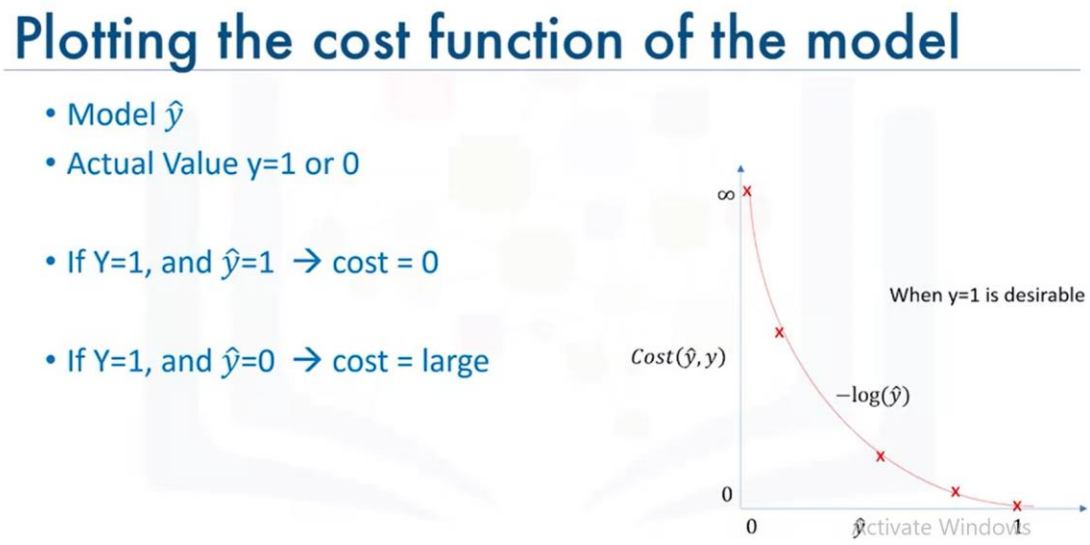
Logistic Regression Training Process:

Here, we will learn more about training a logistic regression model. Also, we will be discussing how to change the parameters of the model to better estimate the outcome. Finally, we talk about the cost function and gradient descent in logistic regression as a way to optimize the model. So, let's start. The main objective of training and logistic regression is to change the parameters of the model, so as to be the best estimation of the labels of the samples in the dataset. For example, the customer churn. How do we do that? In brief, first we have to look at the cost function, and see what the relation is between the cost function and the parameters theta.



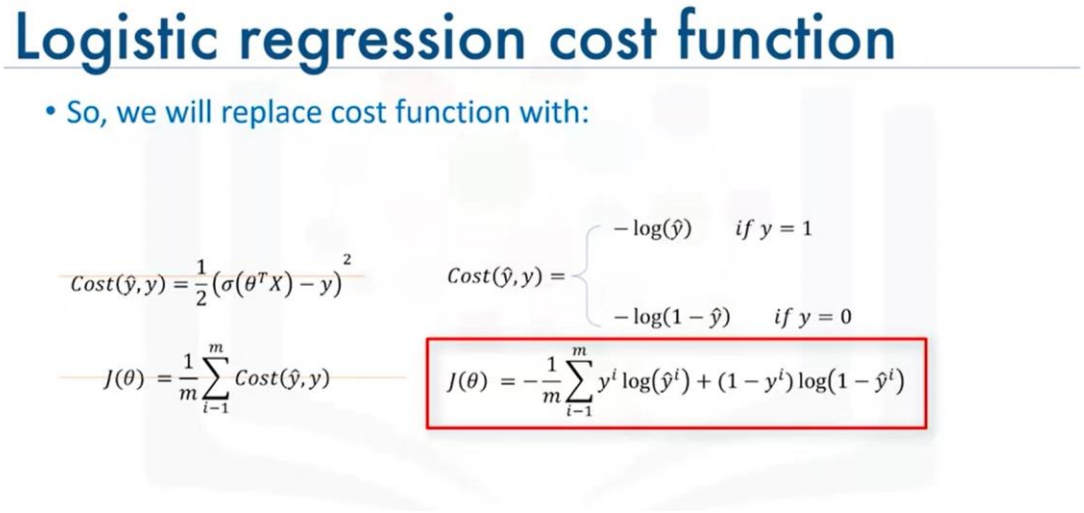
So, we should formulate the cost function. Then, using the derivative of the cost function we can find how to change the parameters to reduce the cost or rather the error. Let's dive into it to see how it works. But before I explain it, I should highlight for you that it needs some basic mathematical background to understand it. However, you shouldn't worry about it as most data science languages like Python, R, and Scala have some packages or libraries that calculate these parameters for you. So, let's take a look at it. Let's first find the cost function equation for a sample case. To

do this, we can use one of the customers in the churn problem. There's normally a general equation for calculating the cost.



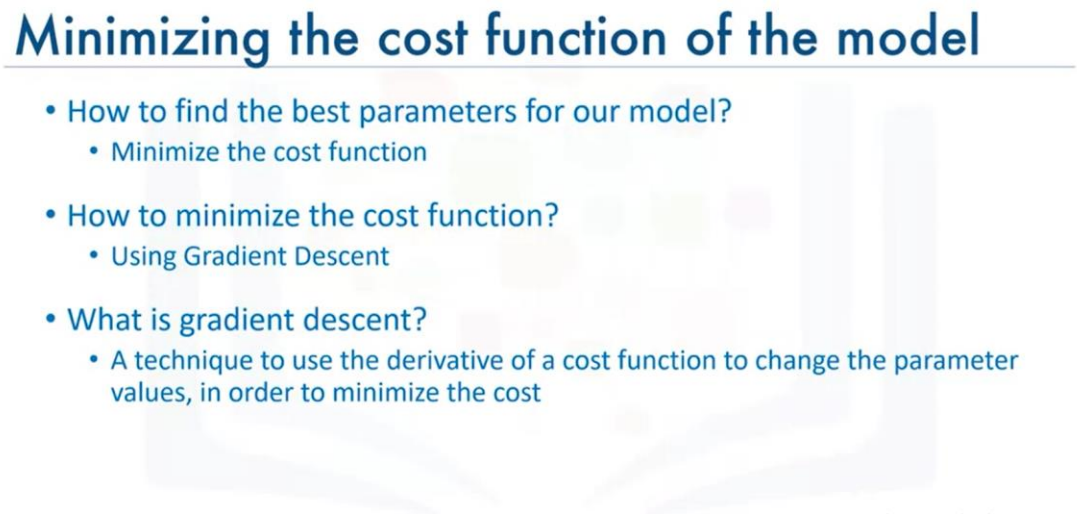
The cost function is the difference between the actual values of y and our model output y hat. This is a general rule for most cost functions in machine learning. We can show this as the cost of our model comparing it with actual labels, which is the difference between the predicted value of our model and actual value of the target field, where the predicted value of our model is sigmoid of theta transpose x. Usually the square of this equation is used because of the possibility of the negative result and for the sake of simplicity, half of this value is considered as the cost function through the derivative process. Now, we can write the cost function for all the samples in our training set. For example, for all customers we can write it as the average sum of the cost functions of all cases. It is also called the mean squared error and as it is a function of a parameter vector theta, it is shown as J of theta. Okay good, we have the cost function. Now, how do we find or set the best weights or parameters that minimize this cost function? The answer is, we should calculate the minimum point of this cost function and it will show us the best parameters for our

model. Although we can find the minimum point of a function using the derivative of a function, there's not an easy way to find the global minimum point for such an equation.



Given this complexity, describing how to reach the global minimum for this equation is outside the scope of this course. So, what is the solution? Well we should find another cost function instead, one which has the same behavior but is easier to find its minimum point. Let's plot the desirable cost function for our model. Recall that our model is y hat. Our actual value is y which equals zero or one, and our model tries to estimate it as we want to find a simple cost function for our model. For a moment assume that our desired value for y is one. This means our model is best if it estimates y equals one. In this case, we need a cost function that returns zero if the outcome of our model is one, which is the same as the actual label. And the cost should keep increasing as the outcome of our model gets farther from one. And cost should be very large if the outcome of our model is close to zero. We can see that the minus log function provides such a cost function for us. It means if the actual value is one and the model also predicts one, the minus log function returns zero cost. But if the prediction is smaller than one, the minus log function returns a larger

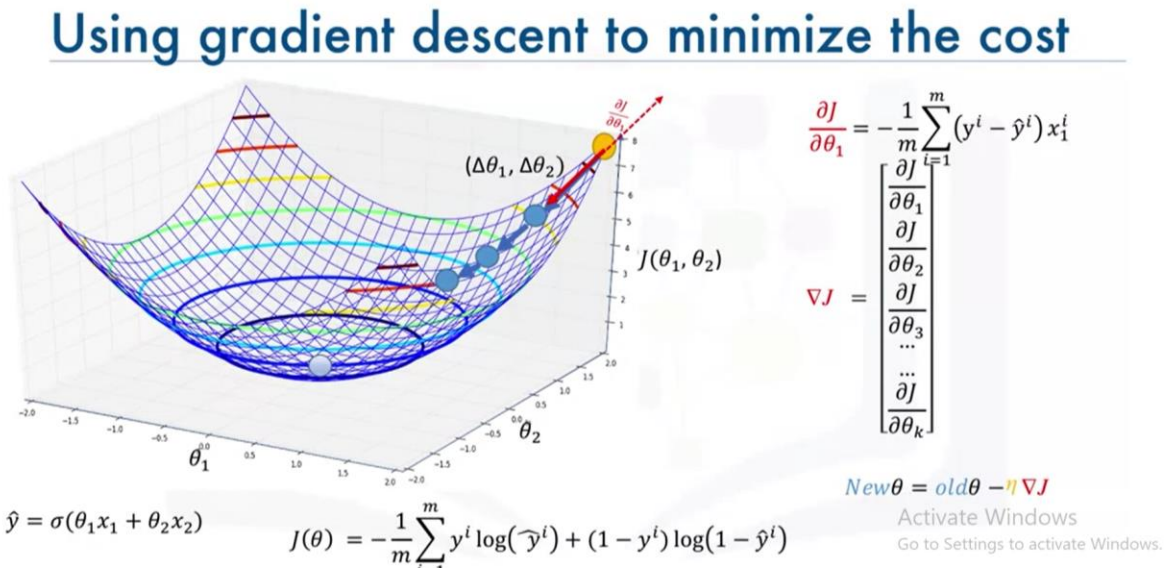
cost value. So, we can use the minus log function for calculating the cost of our logistic regression model.



So, if you recall, we previously noted that in general it is difficult to calculate the derivative of the cost function. Well, we can now change it with the minus log of our model. We can easily prove that in the case that desirable y is one, the cost can be calculated as minus log y hat, and in the case that desirable y is zero the cost can be calculated as minus log one minus y hat. Now, we can plug it into our total cost function and rewrite it as this function. So, this is the logistic regression cost function. As you can see for yourself it penalizes situations in which the class is zero and the model output is one, and vice versa. Remember, however, that y hat does not return a class as output but it's a value of zero or one which should be assumed as a probability. Now, we can easily use this function to find the parameters of our model in such a way as to minimize the cost. Okay, let's recap what we have done. Our objective was to find a model that best estimates the actual labels. Finding the best model means finding the best parameters theta for that model. So, the first question was, how do we find the best parameters for our model? Well, by finding and minimizing the cost function of our model. In other words, to minimize the J of theta we just defined. The next question is, how do we minimize the cost function?

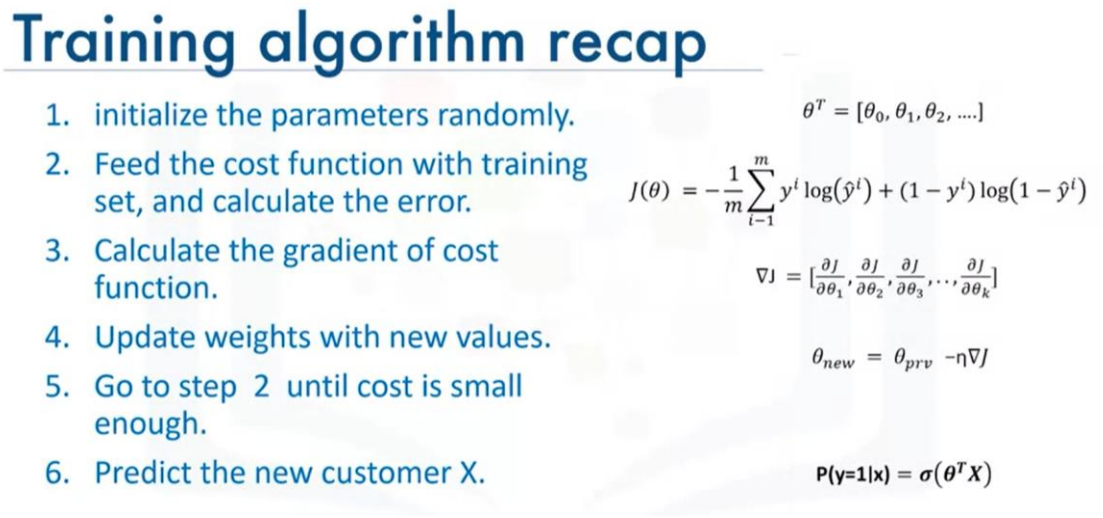
The answer is, using an optimization approach. There are different optimization approaches, but we use one of the most famous and effective approaches here, gradient descent. The next question is, what is gradient descent? Generally, gradient descent is an iterative approach to finding the minimum of a function. Specifically in our case gradient descent is a technique to use the derivative of a cost function to change the parameter values to minimize the cost or error. Let's see how it works. The main objective of gradient descent is to change the parameter values so as to minimize the cost. How can gradient descent do that? Think of the parameters or weights in our model to be in a two-dimensional space. For example, theta one, theta two for two feature sets, age and income. Recall the cost function, J, that we discussed in the previous slides. We need to minimize the cost function J which is a function of variables theta one and theta two. So, let's add a dimension for the observed cost, or error, J function. Let's assume that if we plot the cost function based on all possible values of theta one, theta two, we can see something like this. It represents the error value for different values of parameters that is error which is a function of the parameters. This is called your error curve or error bowl of your cost function. Recall that we want to use this error bowl to find the best parameter values that result in minimizing the cost value. Now, the question is, which point is the best point for your cost function? Yes, you should try to minimize your position on the error curve. So, what should you do? You have to find the minimum value of the cost by changing the parameters. But which way? Will you add some value to your weights or deduct some value? And how much would that value be? You can select random parameter values that locate a point on the bowl. You can think of our starting point being the yellow point. You change the parameters by delta theta one and delta theta two, and take one step on the surface. Let's assume we go down one step in the bowl. As long as we are going downwards we can go one more step. The steeper the slope the further we can step, and we can keep taking steps. As we

approach the lowest point the slope diminishes, so we can take smaller steps until we reach a flat surface. This is the minimum point of our curve and the optimum theta one, theta two.



What are these steps really? I mean in which direction should we take these steps to make sure we descend, and how big should the steps be? To find the direction and size of these steps, in other words to find how to update the parameters, you should calculate the gradient of the cost function at that point. The gradient is the slope of the surface at every point and the direction of the gradient is the direction of the greatest uphill. Now, the question is, how do we calculate the gradient of a cost function at a point? If you select a random point on this surface, for example the yellow point, and take the partial derivative of J of theta with respect to each parameter at that point, it gives you the slope of the move for each parameter at that point. Now, if we move in the opposite direction of that slope, it guarantees that we go down in the error curve. For example, if we calculate the derivative of J with respect to theta one, we find out that it is a positive number. This indicates that function is increasing as theta one increases. So, to decrease J we should move in the opposite direction. This means to move in the direction of the negative derivative

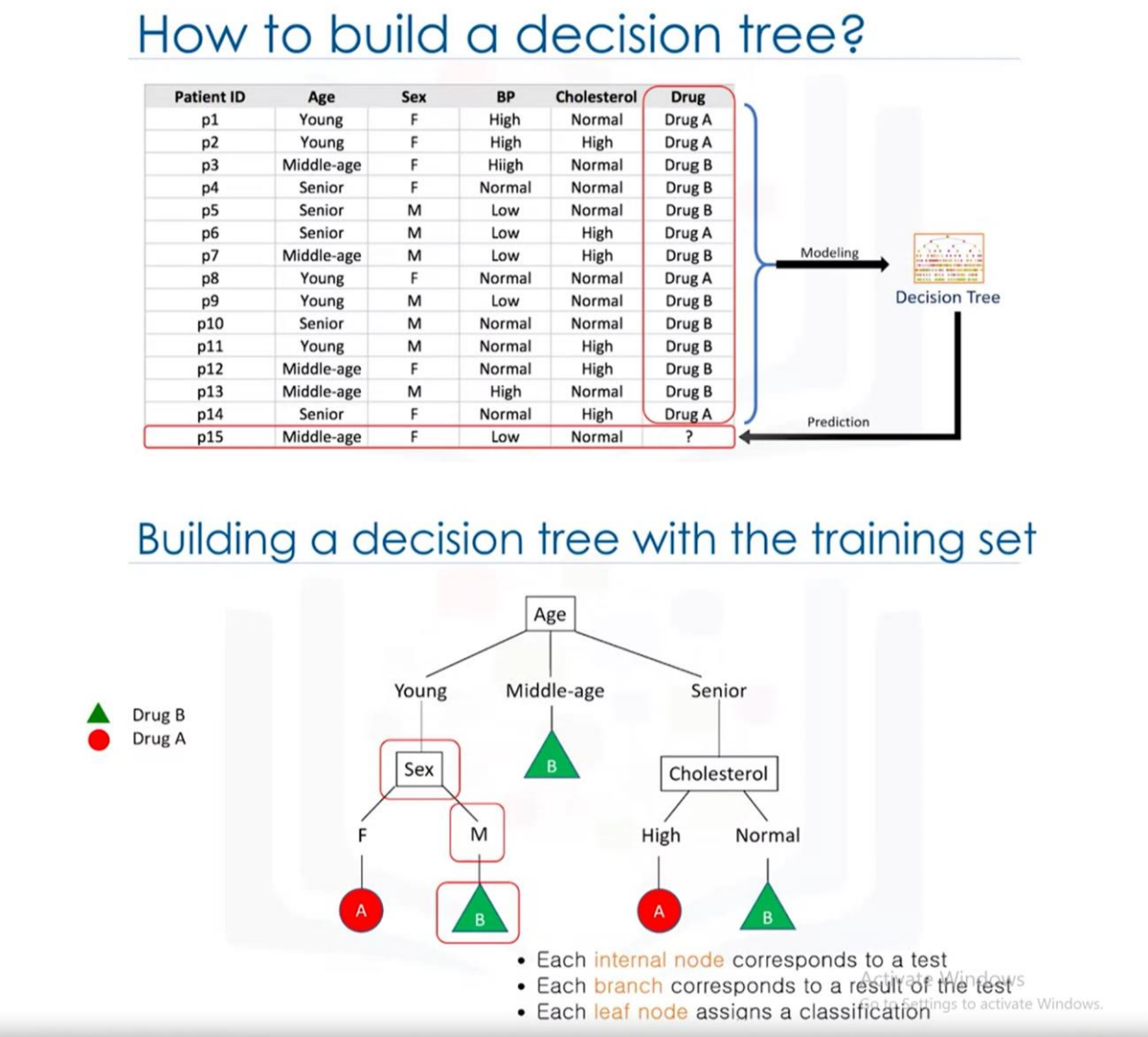
for theta one, i.e. slope. We have to calculate it for other parameters as well at each step. The gradient value also indicates how big of a step to take. If the slope is large we should take a large step because we are far from the minimum. If the slope is small we should take a smaller step. Gradient descent takes increasingly smaller steps towards the minimum with each iteration. The partial derivative of the cost function J is calculated using this expression. If you want to know how the derivative of the J function is calculated, you need to know the derivative concept which is beyond our scope here. But to be honest you don't really need to remember all the details about it as you can easily use this equation to calculate the gradients. So, in a nutshell, this equation returns the slope of that point and we should update the parameter in the opposite direction of the slope. A vector of all these slopes is the gradient vector, and we can use this vector to change or update all the parameters. We take the previous values of the parameters and subtract the error derivative. This results in the new parameters for theta that we know will decrease the cost. Also we multiply the gradient value by a constant value mu, which is called the learning rate. Learning rate, gives us additional control on how fast we move on the surface. In sum, we can simply say, gradient descent is like taking steps in the current direction of the slope, and the learning rate is like the length of the step you take. So, these would be our new parameters. Notice that it's an iterative operation and in each iteration we update the parameters and minimize the cost until the algorithm converge is on an acceptable minimum. Okay, let's recap what we have done to this point by going through the training algorithm again, step-by-step. Step one, we initialize the parameters with random values. Step two, we feed the cost function with the training set and calculate the cost. We expect a high error rate as the parameters are set randomly. Step three, we calculate the gradient of the cost function keeping in mind that we have to use a partial derivative. So, to calculate the gradient vector we need all the training data to feed the equation for each parameter.



Of course, this is an expensive part of the algorithm, but there are some solutions for this. Step four, we update the weights with new parameter values. Step five, here we go back to step two and feed the cost function again, which has new parameters. As was explained earlier, we expect less error as we are going down the error surface. We continue this loop until we reach a short value of cost or some limited number of iterations. Step six, the parameter should be roughly found after some iterations. This means the model is ready and we can use it to predict the probability of a customer staying or leaving.

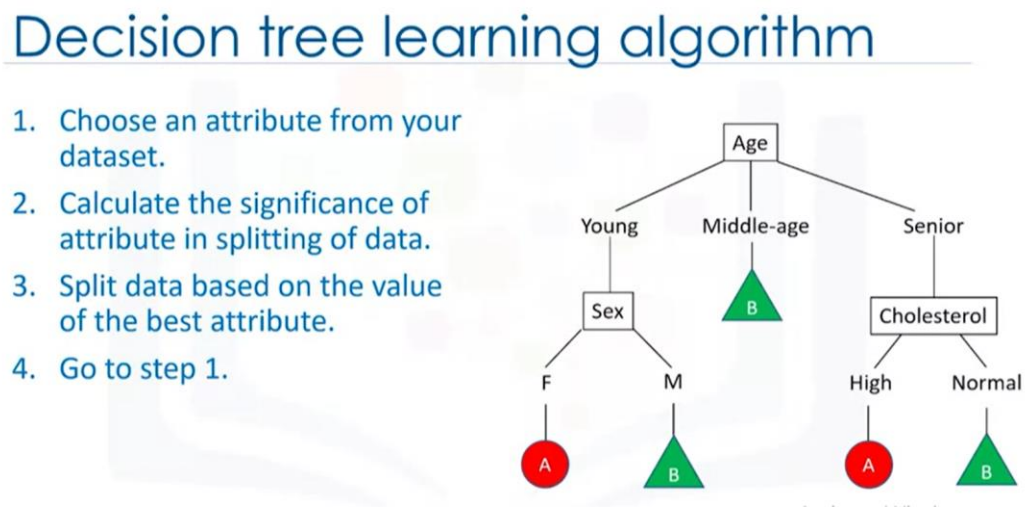
Decision Tree:

Here, we're going to introduce an examine decision trees. What exactly is a decision tree? How do we use them to help us classify? How can I grow my own decision tree? These may be some of the questions that you have in mind from hearing the term decision tree. Hopefully, you'll soon be able to answer these questions and many more after this lecture. Imagine that you're a medical researcher compiling data for a study. You've already collected data about a set of patients all of whom suffered from the same illness. During their course of treatment, each patient responded to one of two medications. We call them drug A and drug B. Part of your job is to build a model to find out which drug might be appropriate for a future patient with the same illness. The feature sets of this dataset are age, gender, blood pressure, and cholesterol of our group of patients and the target is the drug that each patient responded to. It is a sample of binary classifiers, and you can use the training part of the data set to build a decision tree and then use it to predict the class of an unknown patient. In essence, to come up with a decision on which drug to prescribe to a new patient. Let's see how a decision tree is built for this dataset. Decision trees are built by splitting the training set into distinct nodes, where one node contains all of or most of one category of the data. If we look at the diagram here, we can see that it's a patient's classifier. So as mentioned, we want to prescribe a drug to a new patient, but the decision to choose drug A or B will be influenced by the patient's situation. We start with age, which can be young, middle aged or senior. If the patient is middle aged, then we'll definitely go for drug B. On the other hand, if he has a young or a senior patient, will need more details to help us determine which drug to prescribe. The additional decision variables can be things such as cholesterol levels, gender or blood pressure. For example, if the patient is female, then we will recommend drug A, but if the patient is male, then will go for drug B.

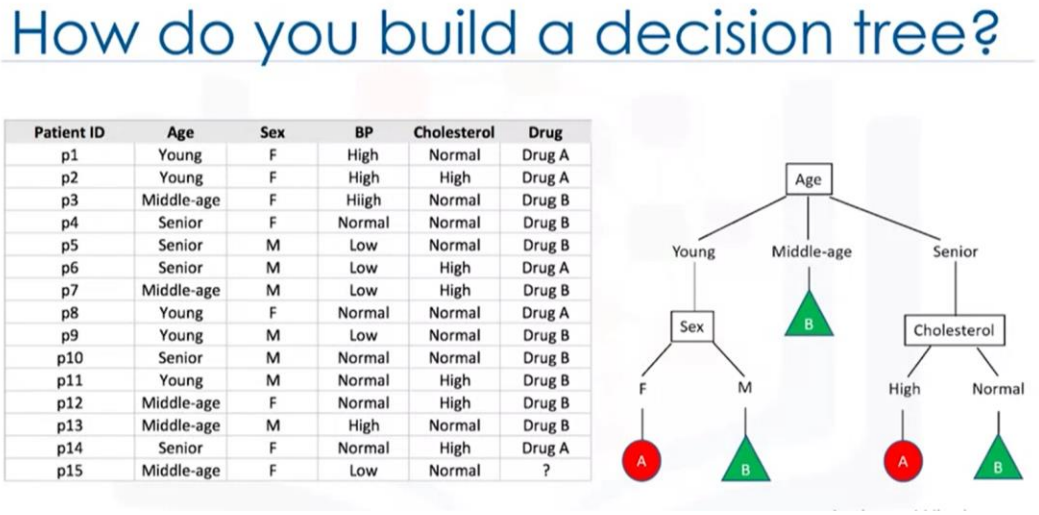


As you can see, decision trees are about testing an attribute and branching the cases based on the result of the test. Each internal node corresponds to a test, and each branch corresponds to a result of the test, and each leaf node assigns a patient to a class. Now the question is, how can we build such a decision tree? Here is the way that a decision tree is build. A decision tree can be constructed by considering the attributes one by one. First, choose an attribute from our dataset. Calculate the significance of the attribute in the splitting of the data. In the next lecture, we will

explain how to calculate the significance of an attribute to see if it's an effective attribute or not.

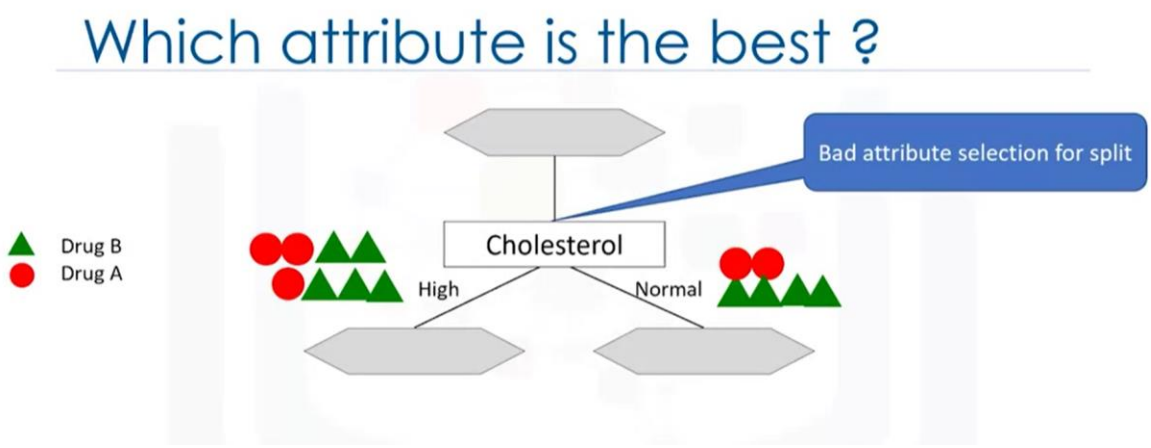


Next, split the data based on the value of the best attribute, then go to each branch and repeat it for the rest of the attributes. After building this tree, you can use it to predict the class of unknown cases; or in our case, the proper drug for a new patient based on his or her characteristics.

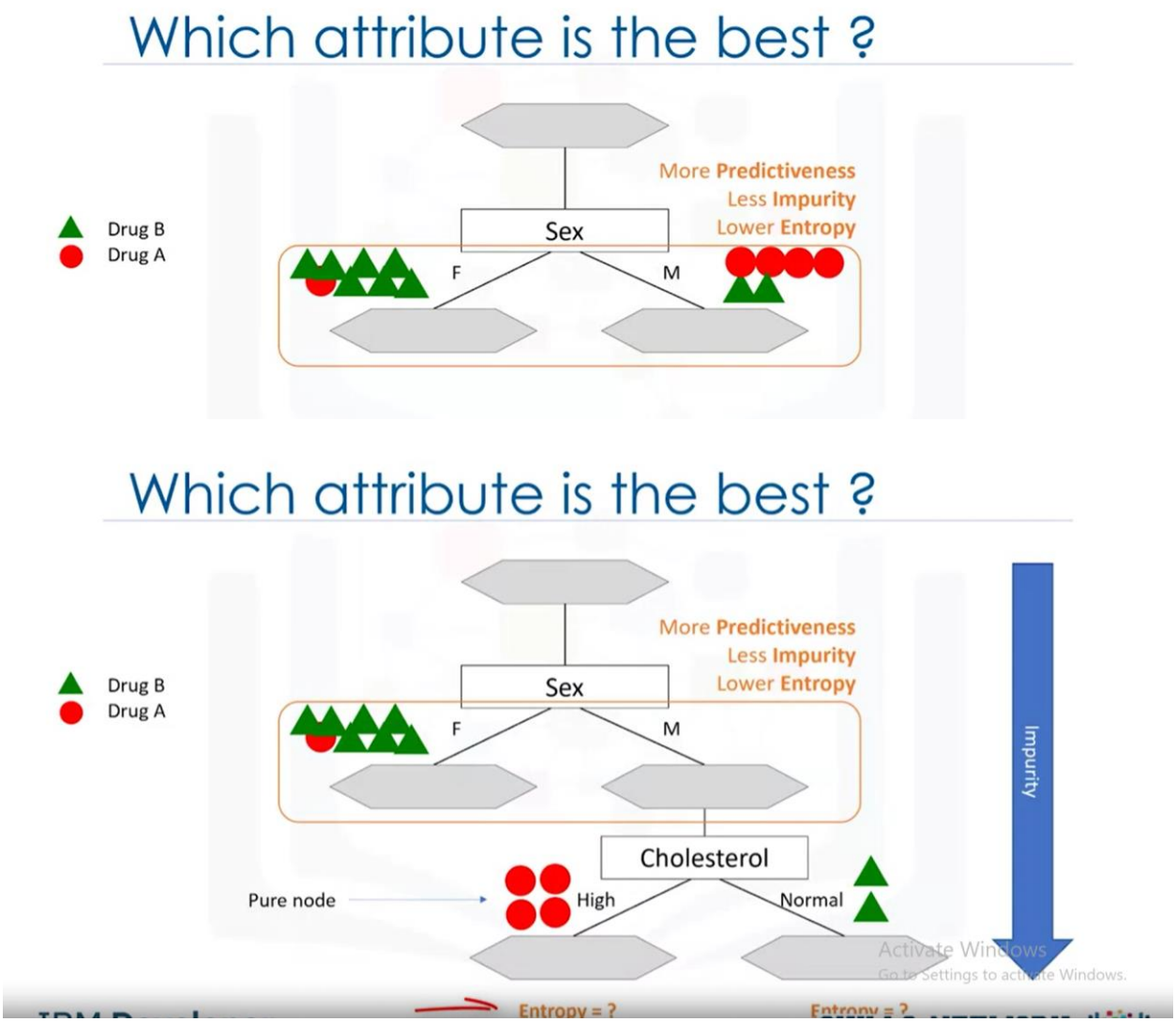


Building Decision Trees:

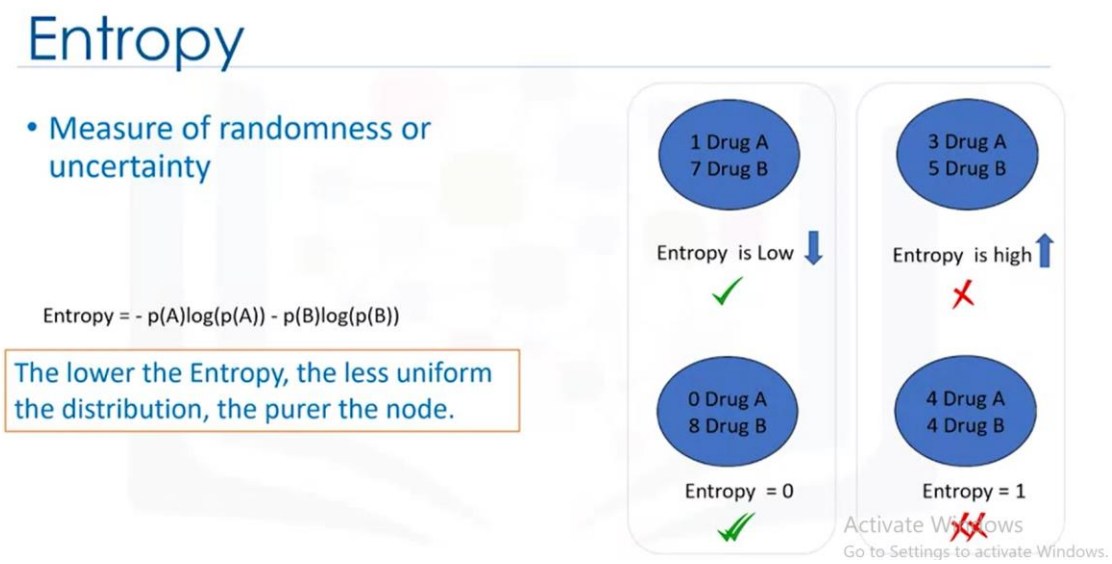
Here, we'll be covering the process of building decision trees. Consider the drug data set again. The question is, how do we build a decision tree based on that data set?



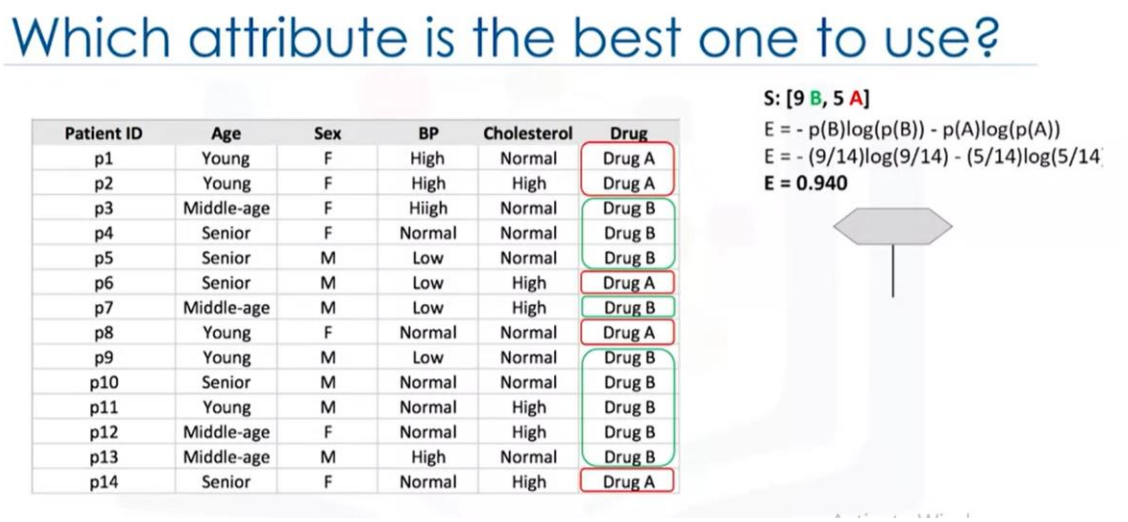
Decision trees are built using recursive partitioning to classify the data. Let's say we have 14 patients in our data set, the algorithm chooses the most predictive feature to split the data on. What is important in making a decision tree, is to determine which attribute is the best or more predictive to split data based on the feature. Let's say we pick cholesterol as the first attribute to split data, it will split our data into two branches. As you can see, if the patient has high cholesterol we cannot say with high confidence that drug B might be suitable for him. Also, if the patient's cholesterol is normal, we still don't have sufficient evidence or information to determine if either drug A or drug B is in fact suitable. It is a sample of bad attributes selection for splitting data. So, let's try another attribute. Again, we have our 14 cases, this time we picked the sex attribute of patients. It will split our data into two branches, male and female. As you can see, if the patient is female, we can say drug B might be suitable for her with high certainty. But if the patient is male, we don't have sufficient evidence or information to determine if drug A or drug B is suitable. However, it is still a better choice in comparison with the cholesterol attribute because the result in the nodes are more pure. It means nodes that are either mostly drug A or drug B. So, we can say the sex attribute is more significant than cholesterol, or in other words it's more predictive than the other attributes. Indeed, predictiveness is based on decrease in impurity of nodes.



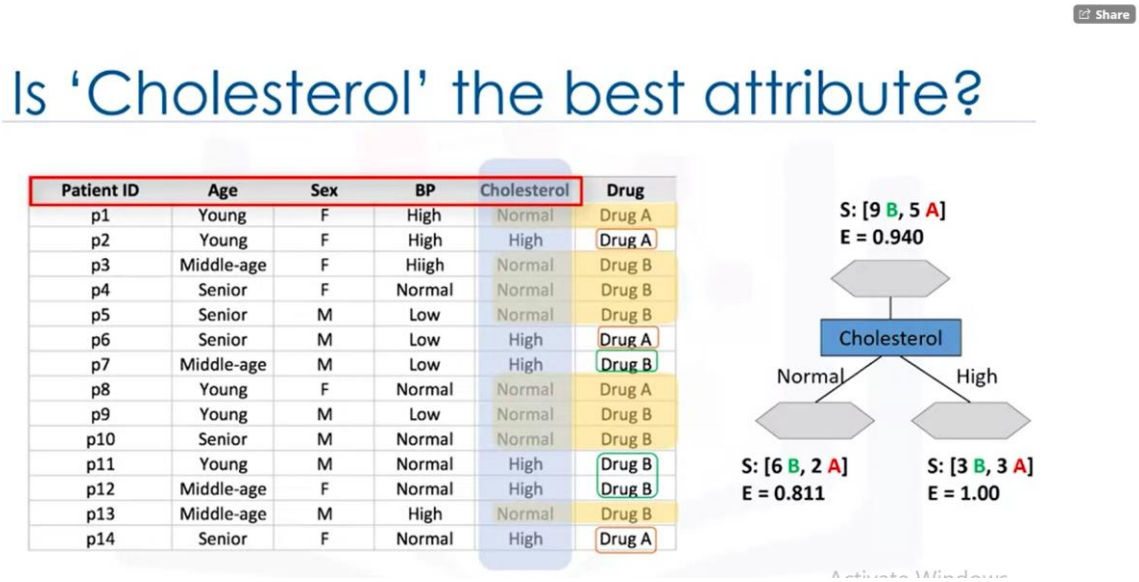
We're looking for the best feature to decrease the impurity of patients in the leaves, after splitting them up based on that feature. So, the sex feature is a good candidate in the following case because it almost found the pure patients. Let's go one step further. For the male patient branch, we again test other attributes to split the sub-tree. We test cholesterol again here, as you can see it results in even more pure leaves. So we can easily make a decision here. For example, if a patient is male and his cholesterol is high, we can certainly prescribe drug A, but if it is normal, we can prescribe drug B with high confidence. As you might notice, the choice of attribute to split data is very important and it is all about purity of the leaves after the split.



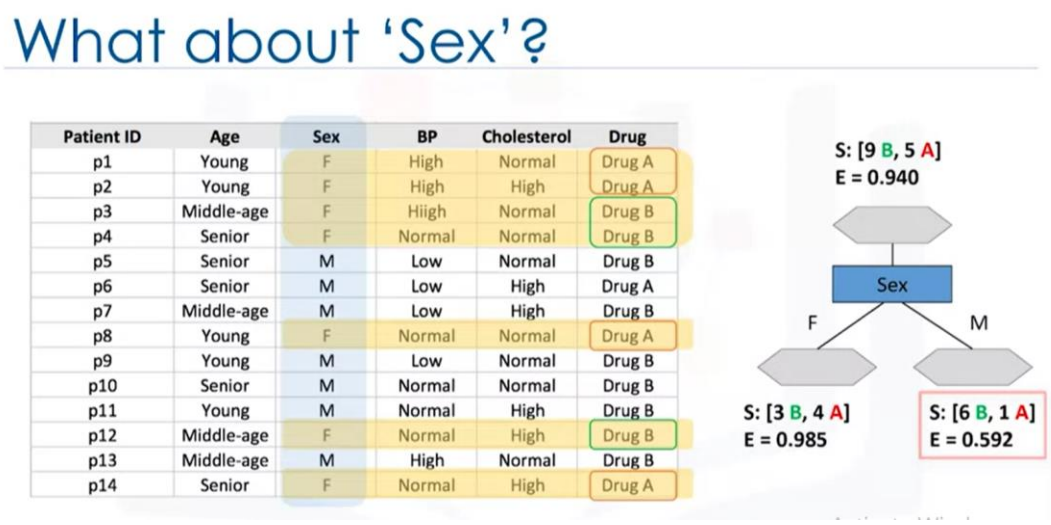
A node in the tree is considered pure if in 100 percent of the cases, the nodes fall into a specific category of the target field. In fact, the method uses recursive partitioning to split the training records into segments by minimizing the impurity at each step. Impurity of nodes is calculated by entropy of data in the node. So, what is entropy? Entropy is the amount of information disorder or the amount of randomness in the data. The entropy in the node depends on how much random data is in that node and is calculated for each node. In decision trees, we're looking for trees that have the smallest entropy in their nodes. The entropy is used to calculate the homogeneity of the samples in that node. If the samples are completely homogeneous, the entropy is zero and if the samples are equally divided it has an entropy of one. This means if all the data in a node are either drug A or drug B, then the entropy is zero, but if half of the data are drug A and other half are B then the entropy is one. You can easily calculate the entropy of a node using the frequency table of the attribute through the entropy formula where P is for the proportion or ratio of a category, such as drug A or B. Please remember though that you don't have to calculate these as it's easily calculated by the libraries or packages that you use. As an example, let's calculate the entropy of the data set before splitting it.

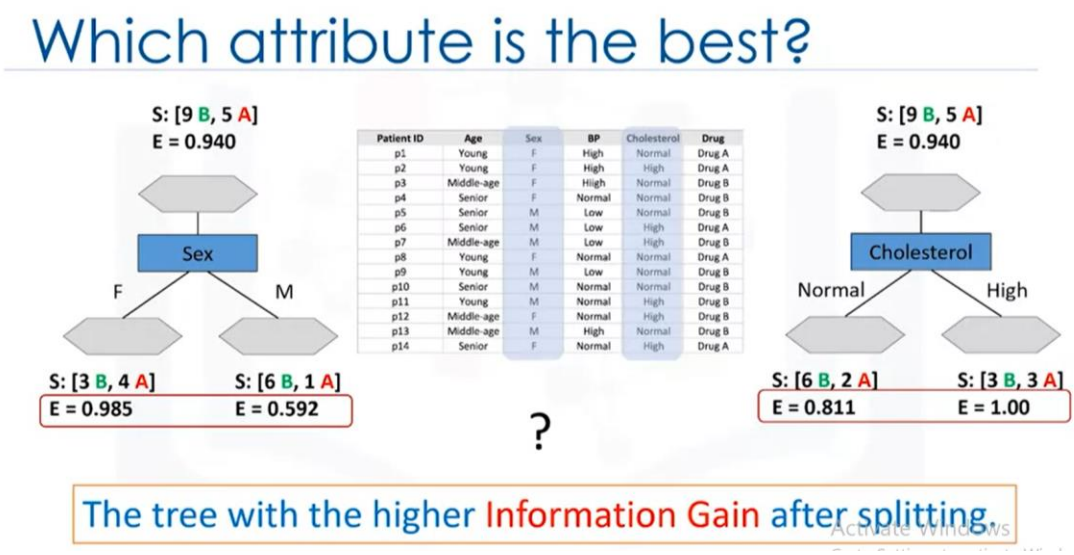


We have nine occurrences of drug B and five of drug A. You can embed these numbers into the entropy formula to calculate the impurity of the target attribute before splitting it. In this case, it is 0.94. So, what is entropy after splitting? Now, we can test different attributes to find the one with the most predictiveness, which results in two more pure branches. Let's first select the cholesterol of the patient and see how the data gets split based on its values. For example, when it is normal we have six for drug B, and two for drug A. We can calculate the entropy of this node based on the distribution of drug A and B which is 0.8 in this case. But, when cholesterol is high, the data is split into three for drug B and three for drug A. Calculating it's entropy, we can see it would be 1.0. We should go through all the attributes and calculate the entropy after the split and then choose the best attribute.

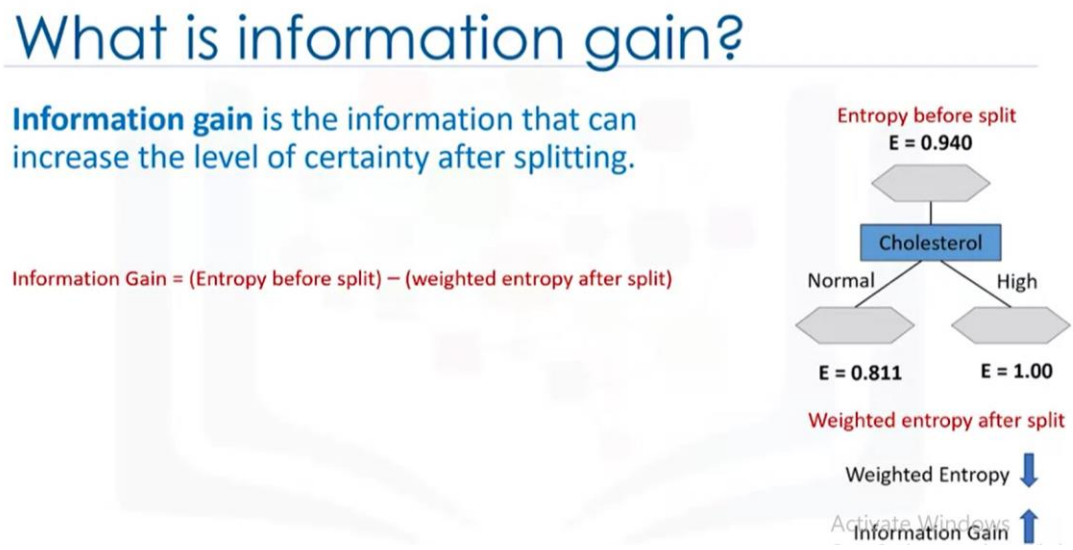


Okay. Let's try another field. Let's choose the sex attribute for the next check. As you can see, when we use the sex attribute to split the data, when its value is female, we have three patients that responded to drug B and four patients that responded to drug A. The entropy for this node is 0.98 which is not very promising. However, on the other side of the branch, when the value of the sex attribute is male, the result is more pure with sex for drug B and only one for drug A. The entropy for this group is 0.59. Now, the question is between the cholesterol and sex attributes which one is a better choice? Which one is better at the first attribute to divide the data-set into two branches? Or in other words, which attribute results in more pure nodes for our drugs? Or in which tree do we have less entropy after splitting rather than before splitting? The sex attribute with entropy of 0.98 and 0.59 or the cholesterol attribute with entropy of 0.81 and 1.0 in its branches. The answer is the tree with the higher information gain after splitting. So, what is information gain? Information gain is the information that can increase the level of certainty after splitting. It is the entropy of a tree before the split minus the weighted entropy after the split by an attribute. We can think of information gain and entropy as opposites.

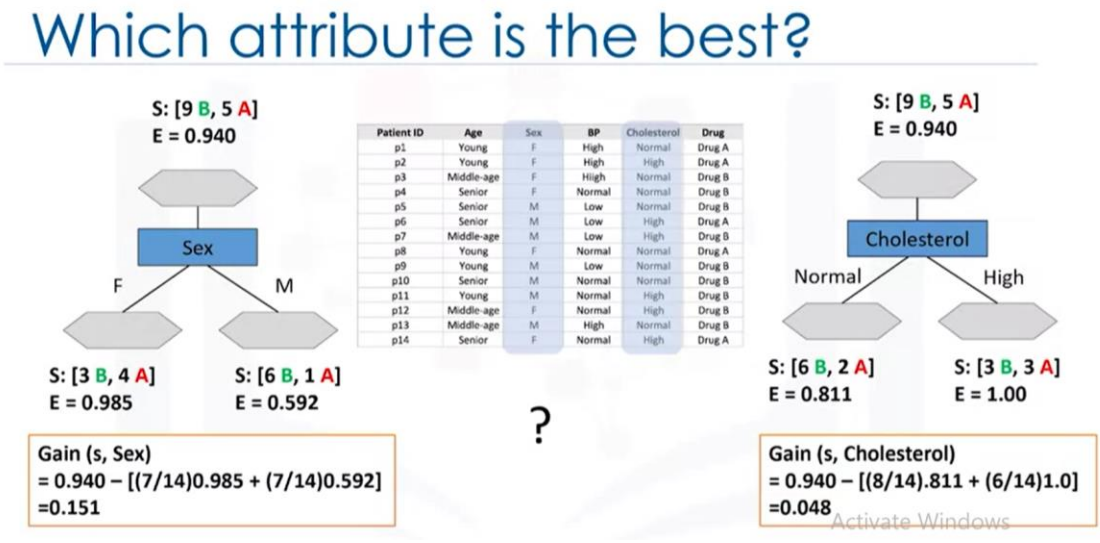




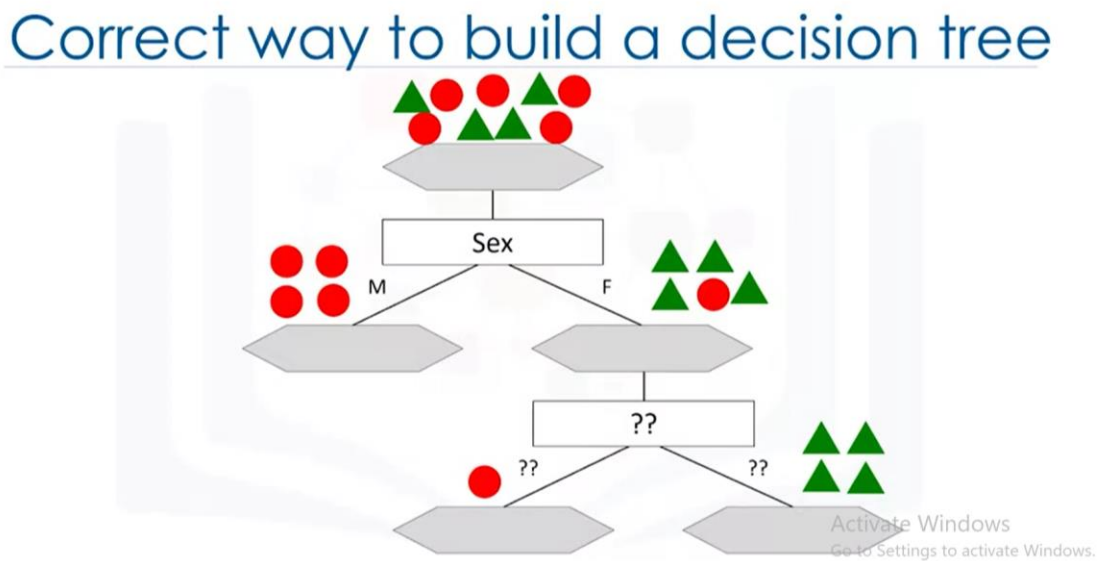
As entropy or the amount of randomness decreases, the information gain or amount of certainty increases and vice versa. So, constructing a decision tree is all about finding attributes that return the highest information gain. Let's see how information gain is calculated for the sex attribute. As mentioned, the information gained is the entropy of the tree before the split minus the weighted entropy after the split. The entropy of the tree before the split is 0.94, the portion of female patients is seven out of 14 and its entropy is 0.985. Also, the portion of men is seven out of 14 and the entropy of the male node is 0.592. The result of a square bracket here is the weighted entropy after the split.



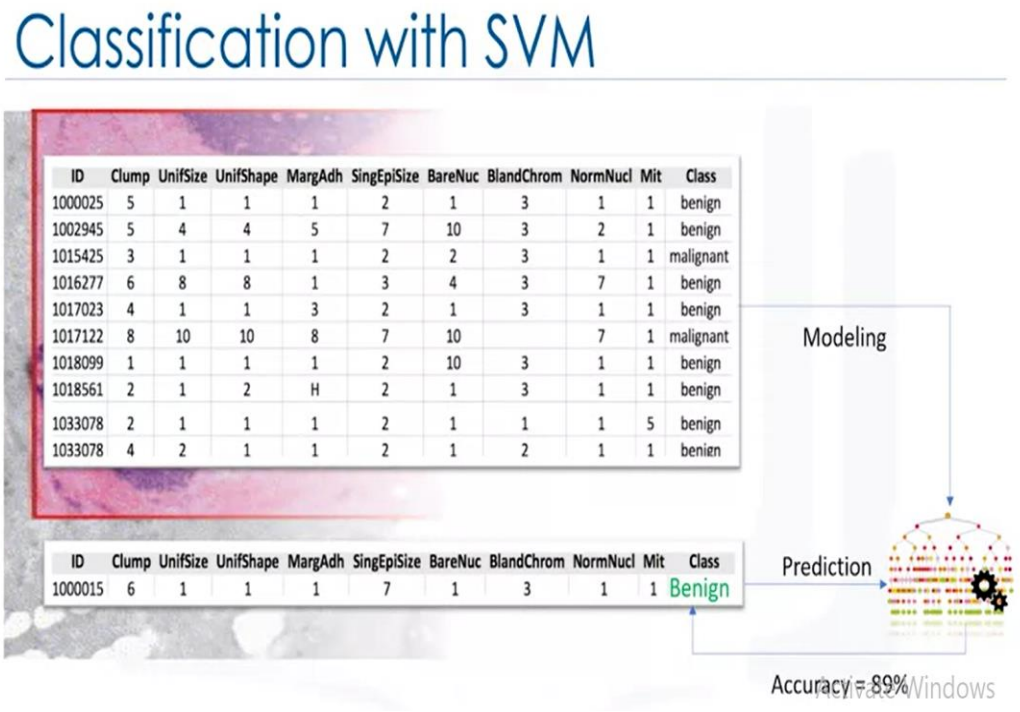
So, the information gain of the tree if we use the sex attribute to split the data set is 0.151. As you could see, we will consider the entropy over the distribution of samples falling under each leaf node and we'll take a weighted average of that entropy weighted by the proportion of samples falling under that leave. We can calculate the information gain of the tree if we use cholesterol as well. It is 0.48. Now, the question is, which attribute is more suitable?



Well, as mentioned, the tree with the higher information gained after splitting, this means the sex attribute. So, we select the sex attribute as the first splitter.

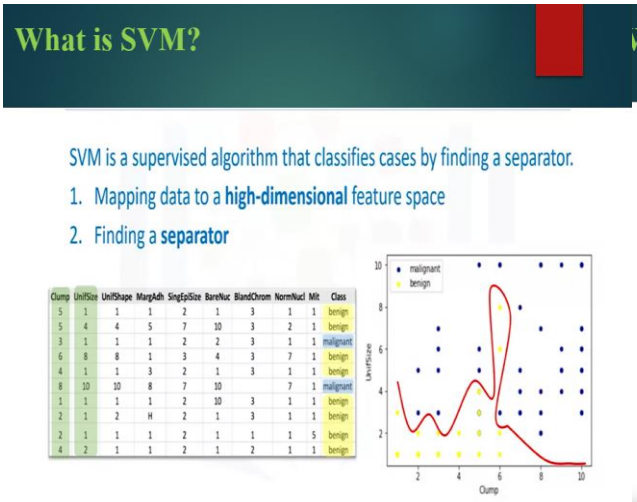


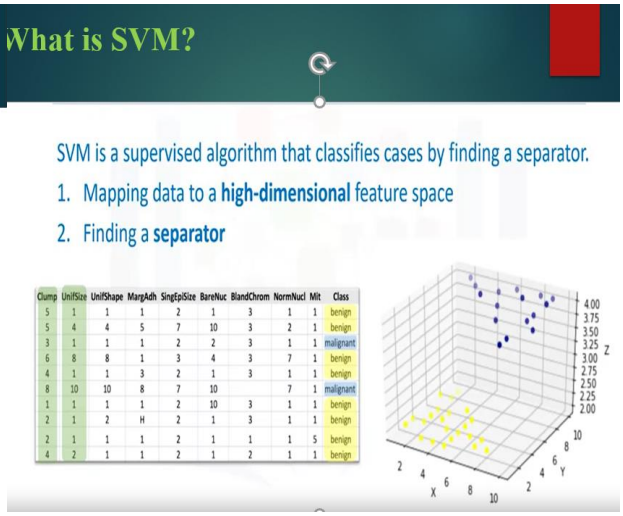
Now, what is the next attribute after branching by the sex attribute? Well, as you can guess, we should repeat the process for each branch and test each of the other attributes to continue to reach the most pure leaves. This is the way you build a decision tree.

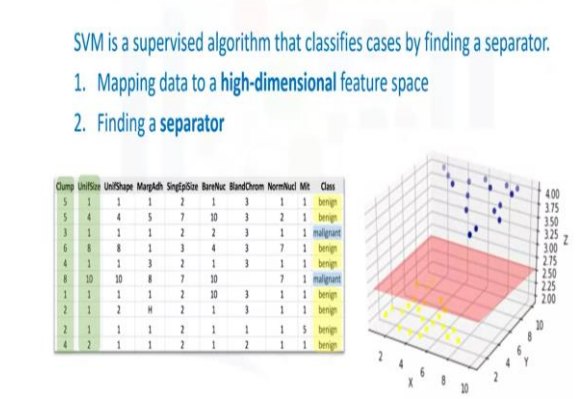


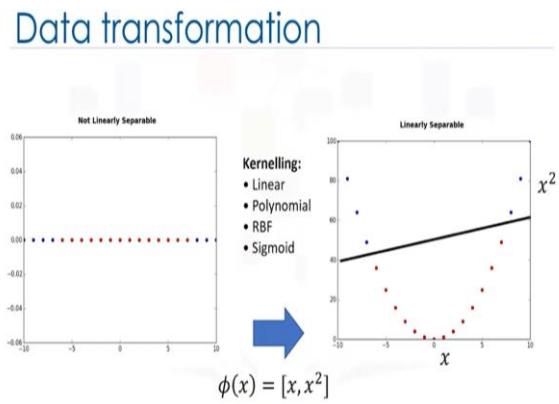
Support Vector Machine (SVM):

Here, we will learn a machine learning method called, Support Vector Machine, or SVM, which is used for classification. So let's get started. Imagine that you've obtained a dataset containing characteristics of thousands of human cell samples extracted from patients who were believed to be at risk of developing cancer. Analysis of the original data showed that many of the characteristics differed significantly between benign and malignant samples. You can use the values of these cell characteristics in samples from other patients, to give an early indication of whether a new sample might be benign or malignant. You can use Support Vector Machine, or SVM, as a classifier to train your model to understand patterns within the data that might show, benign or malignant cells. Once the model has been trained, it can be used to predict your new or unknown cell with rather high accuracy. Now, let me give you a formal definition of SVM. A Support Vector Machine is a supervised algorithm that can classify cases by finding a separator. SVM works by first mapping data to a high dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. Then, a separator is estimated for the data. The data should be transformed in such a way that a separator could be drawn as a hyperplane. For example, consider the following figure, which shows the distribution of a small set of cells only based on their unit size and clump thickness. As you can see, the data points fall into two different categories. It represents a linearly non separable data set. The two categories can be separated with a curve but not a line. That is, it represents a linearly non separable data set, which is the case for most real world data sets. We can transfer this data to a higher-dimensional space, for example, mapping it to a three-dimensional space. After the transformation, the boundary between the two categories can be defined by a hyperplane.

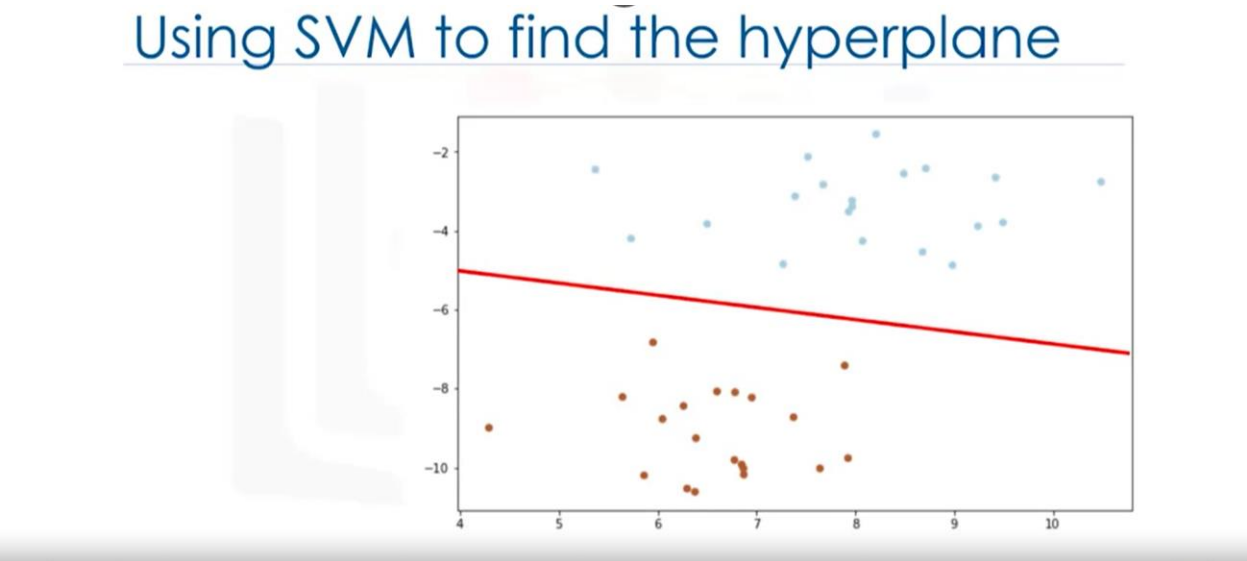




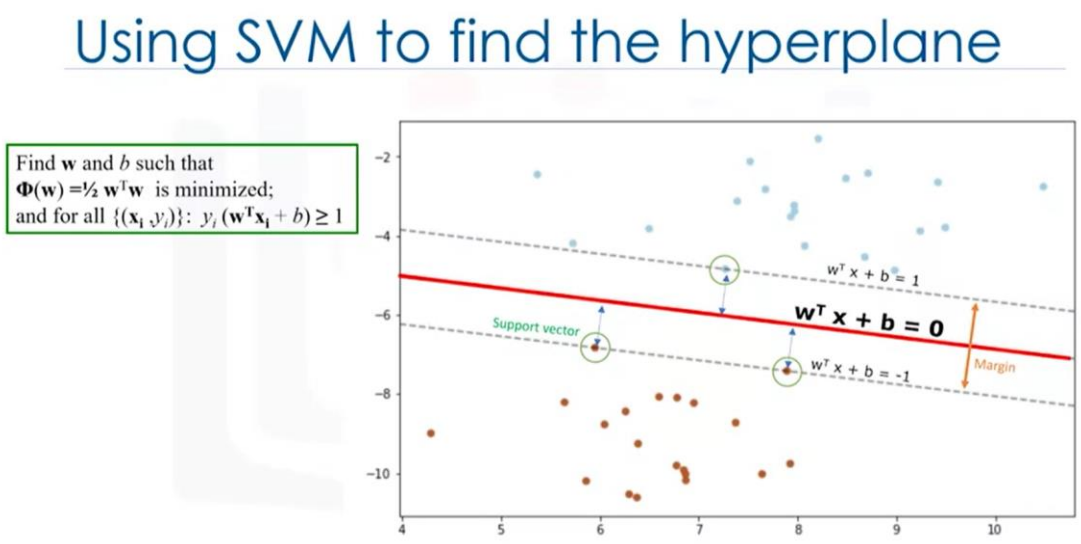




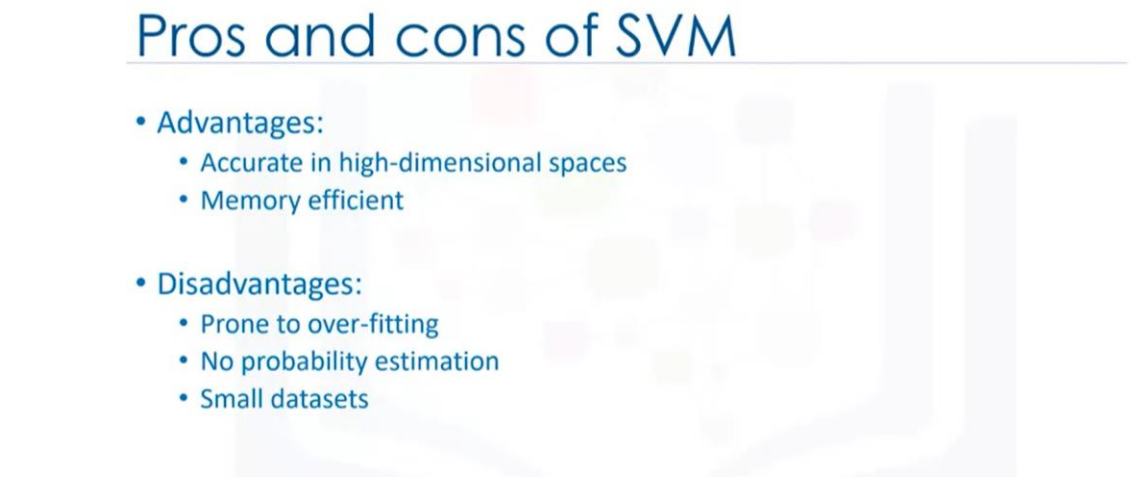
As we are now in three-dimensional space, the separator is shown as a plane. This plane can be used to classify new or unknown cases. Therefore, the SVM algorithm outputs an optimal hyperplane that categorizes new examples. Now, there are two challenging questions to consider. First, how do we transfer data in such a way that a separator could be drawn as a hyperplane? And two, how can we find the best or optimized hyperplane separator after transformation? Let's first look at transforming data to see how it works. For the sake of simplicity, imagine that our dataset is one-dimensional data. This means we have only one feature x. As you can see, it is not linearly separable. So what can we do here? Well, we can transfer it into a two-dimensional space. For example, you can increase the dimension of data by mapping x into a new space using a function with outputs x and x squared.



Now the data is linearly separable, right? Notice that as we are in a two-dimensional space, the hyperplane is a line dividing a plane into two parts where each class lays on either side. Now we can use this line to classify new cases. Basically, mapping data into a higher-dimensional space is called, kernelling. The mathematical function used for the transformation is known as the kernel function, and can be of different types, such as linear, polynomial, Radial Basis Function,or RBF, and sigmoid. Each of these functions has its own characteristics, its pros and cons, and its equation. But the good news is that you don't need to know them as most of them are already implemented in libraries of data science programming languages. Also, as there's no easy way of knowing which function performs best with any given dataset, we usually choose different functions in turn and compare the results. Now we get to another question. Specifically, how do we find the right or optimized separator after transformation? Basically, SVMs are based on the idea of finding a hyperplane that best divides a data set into two classes as shown here. As we're in a two-dimensional space, you can think of the hyperplane as a line that linearly separates the blue points from the red points. One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes. So the goal is to choose a hyperplane with as big a margin as possible.



Examples closest to the hyperplane are support vectors. It is intuitive that only support vectors matter for achieving our goal. And thus, other trending examples can be ignored. We tried to find the hyperplane in such a way that it has the maximum distance to support vectors. Please note that the hyperplane and boundary decision lines have their own equations. So finding the optimized hyperplane can be formalized using an equation which involves quite a bit more math, so I'm not going to go through it here in detail. That said, the hyperplane is learned from training data using an optimization procedure that maximizes the margin. And like many other problems, this optimization problem can also be solved by gradient descent, which is out of scope of this lecture. Therefore, the output of the algorithm is the values w and b for the line. You can make classifications using this estimated line. It is enough to plug in input values into the line equation. Then, you can calculate whether an unknown point is above or below the line. If the equation returns a value greater than 0, then the point belongs to the first class which is above the line, and vice-versa. The two main advantages of support vector machines are that they're accurate in high-dimensional spaces. And they use a subset of training points in the decision function called, support vectors, so it's also memory efficient.



The disadvantages of Support Vector Machines include the fact that the algorithm is prone for over-fitting if the number of features is much greater than the number of samples. Also, SVMs do not directly provide probability estimates, which are desirable in most classification problems. And finally, SVMs are not very efficient computationally if your dataset is very big, such as when you have more than 1,000 rows. And now our final question is, in which situation should I use SVM? Well, SVM is good for image analysis tasks, such as image classification and hand written digit recognition. Also, SVM is very effective in text mining tasks, particularly due to its effectiveness in dealing with high-dimensional data. For example, it is used for

detecting spam, text category assignment and sentiment analysis. Another application of SVM is in gene expression data classification, again, because of its power in high-dimensional data classification. SVM can also be used for other types of machine learning problems, such as regression, outlier detection and clustering. I'll leave it to you to explore more about these particular problems.

Unsupervised Machine Learning

In the previous topic, we learned supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

> *Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.*

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

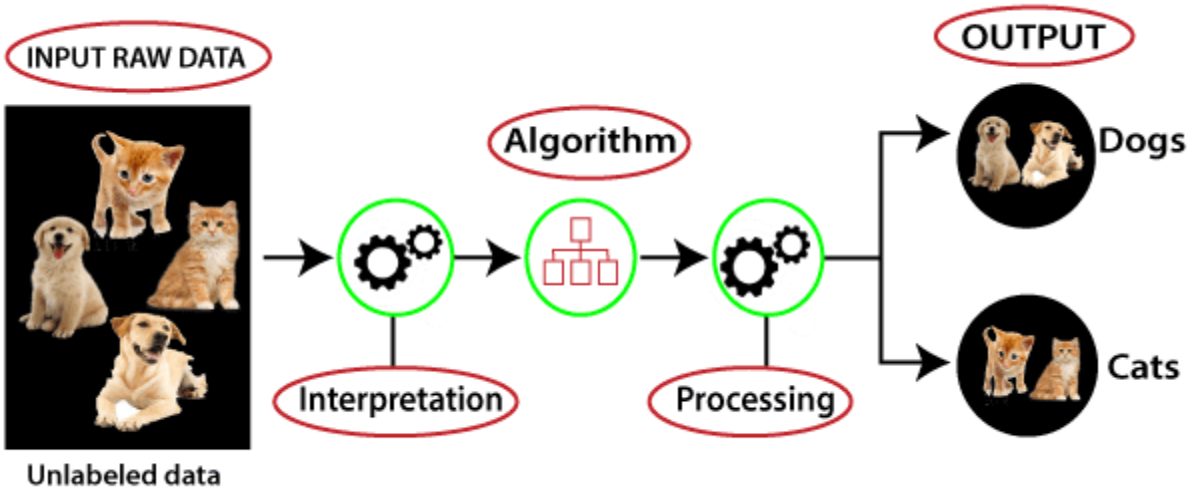
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

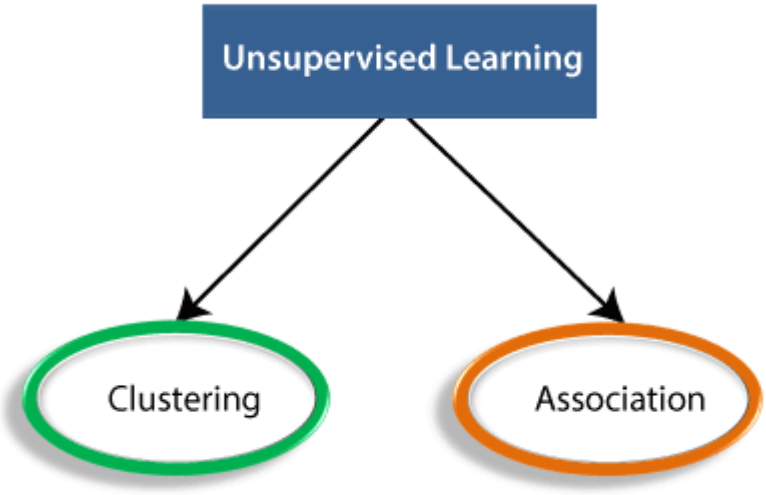


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Note: We will learn these algorithms in later chapters.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

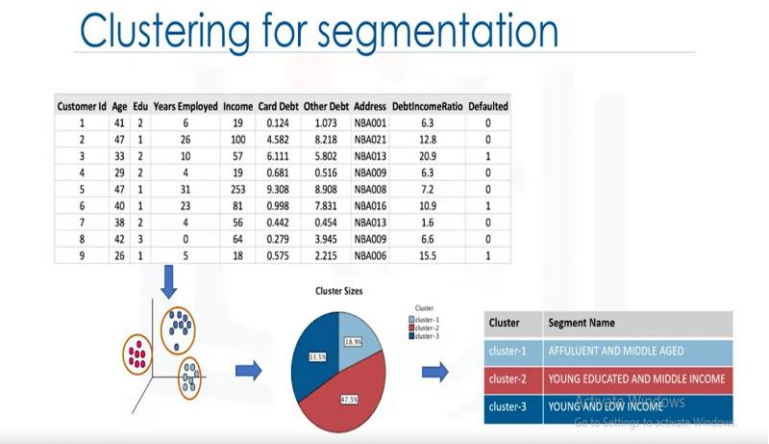
- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchal clustering**
- **Anomaly detection**
- **Neural Networks**
- **Principle Component Analysis**
- **Independent Component Analysis**
- **Apriori algorithm**
- **Singular value decomposition**

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

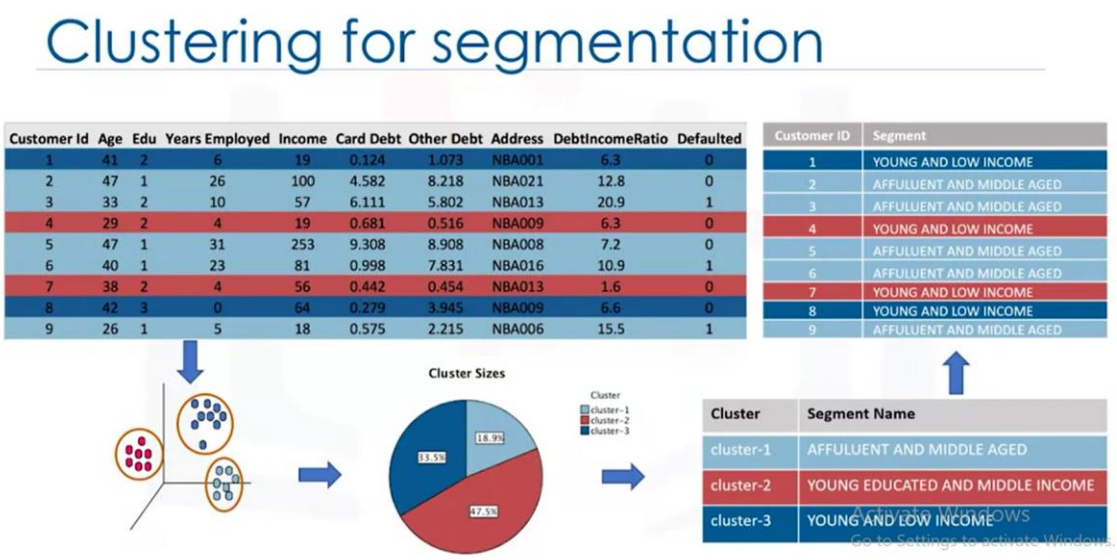
- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.



Intro to Clustering:

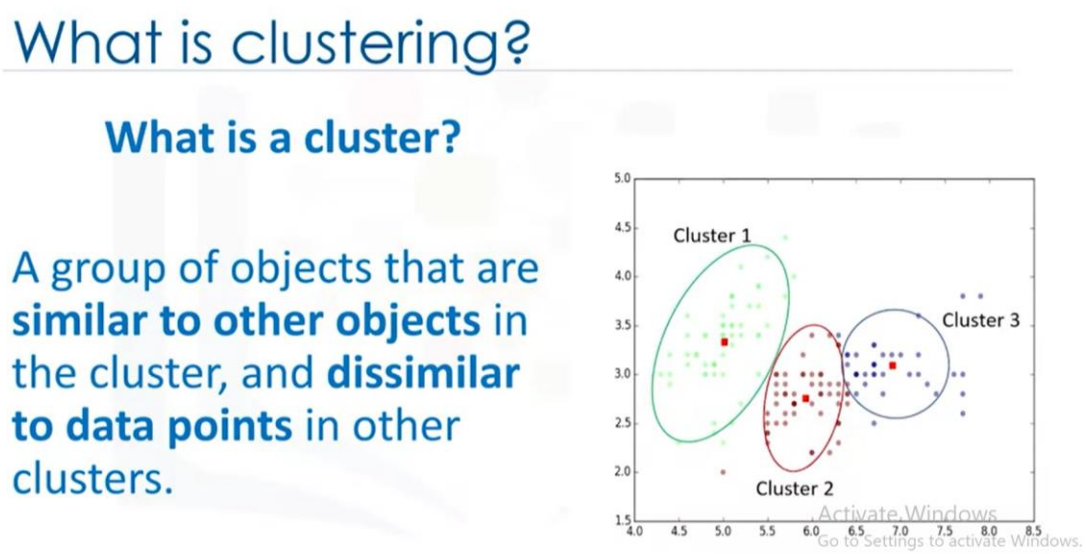
Now, we'll see a high level introduction to clustering, its applications, and different types of clustering algorithms. Imagine that you have a customer dataset and you need to apply customer segmentation on this historical data. Customer segmentation is the practice of partitioning a customer base into groups of individuals that have similar characteristics. It is a significant strategy, as it allows the business to target specific groups of customers, so as to more effectively allocate marketing resources. For example, one group might contain customers who are high profit and low risk. That is, more likely to purchase products or subscribe for a service. Knowing this information allows a business to devote more time and attention to retaining these customers. Another group might include customers from nonprofit organizations and so on. A general segmentation process is not usually feasible for large volumes

of varied data, therefore you need an analytical approach to deriving segments and groups from large datasets.



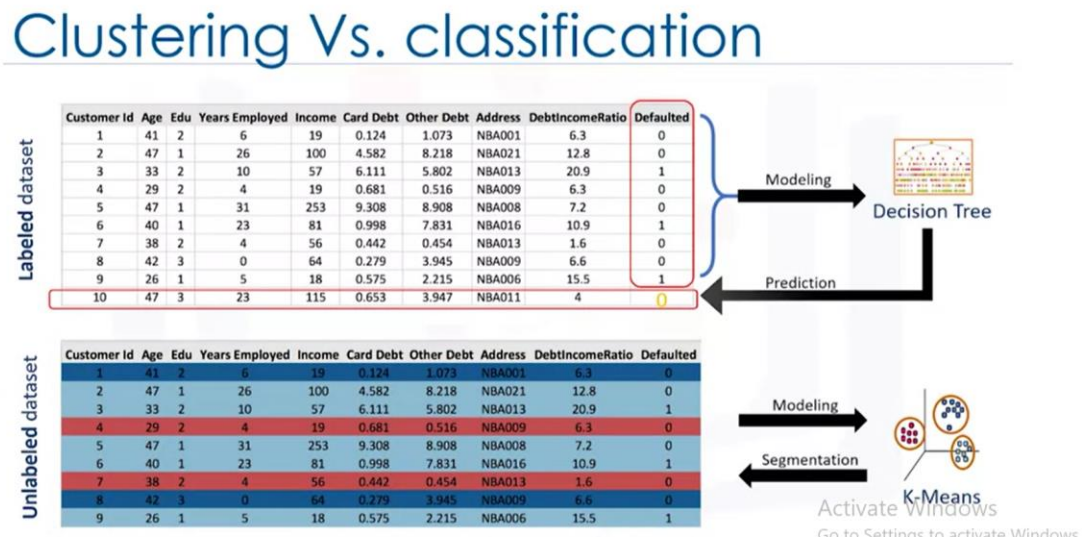
Customers can be grouped based on several factors, including age, gender, interests, spending habits and so on. The important requirement is to use the available data to understand and identify how customers are similar to each other. Let's learn how to divide a set of customers into categories, based on characteristics they share. One of the most adopted approaches that can be used for customer segmentation is clustering. Clustering can group data only unsupervised, based on the similarity of customers to each other. It will partition your customers into mutually exclusive groups. For example, into three clusters. The customers in each cluster are similar to each other demographically. Now we can create a profile for each group, considering the common characteristics of each cluster. For example, the first group made up of affluent and middle aged customers. The second is made up of young, educated and middle income customers, and the third group includes young and low income customers. Finally, we can assign each individual in our dataset to one of these groups or segments of customers. Now imagine that you cross join this segmented

dataset with the dataset of the product or services that customer's purchase from your company.



This information would really help to understand and predict the differences and individual customers' preferences and their buying behaviors across various products. Indeed, having this information would allow your company to develop highly personalized experiences for each segment. Customer segmentation is one of the popular usages of clustering. Cluster analysis also has many other applications in different domains. So let's first define clustering and then we'll look at other applications. Clustering means finding clusters in a dataset, unsupervised. So what is a cluster? A cluster is a group of data points or objects in a dataset that are similar to other objects in the group, and dissimilar to datapoints in other clusters. Now the question is," What is different between clustering and classification?" Let's look at our customer dataset again. Classification algorithms predict categorical classed labels. This means assigning instances to predefined classes such as defaulted or not defaulted. For example, if an analyst wants to analyze customer data in order to know which customers might default on their payments, she uses a labeled dataset as

training data and uses classification approaches such as a decision tree, Support Vector Machines or SVM, or logistic regression, to predict the default value for a new or unknown customer.

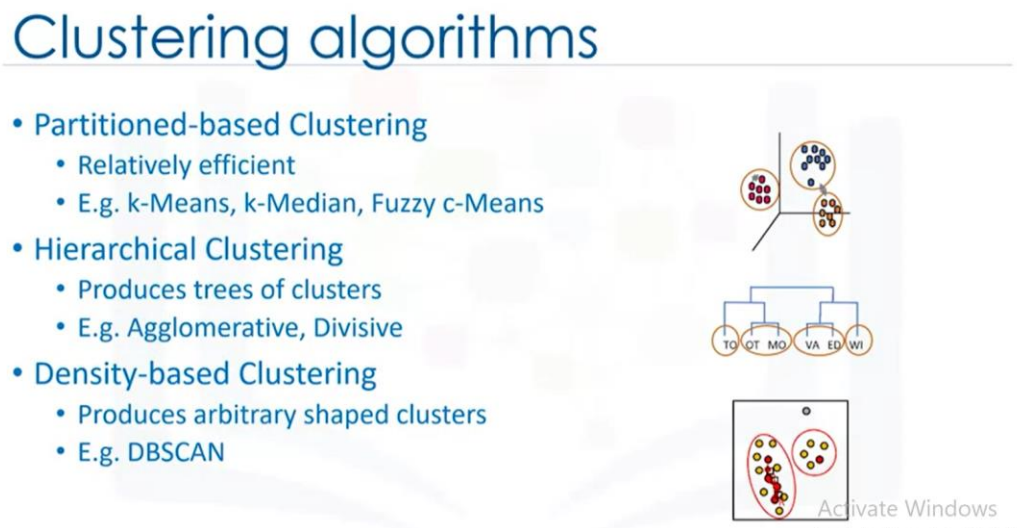


Generally speaking, classification is a supervised learning where each training data instance belongs to a particular class. In clustering however, the data is unlabeled and the process is unsupervised. For example, we can use a clustering algorithm such as k-means to group similar customers as mentioned, and assign them to a cluster, based on whether they share similar attributes, such as; age, education, and so on. While I'll be giving you some examples in different industries, I'd like you to think about more samples of clustering.

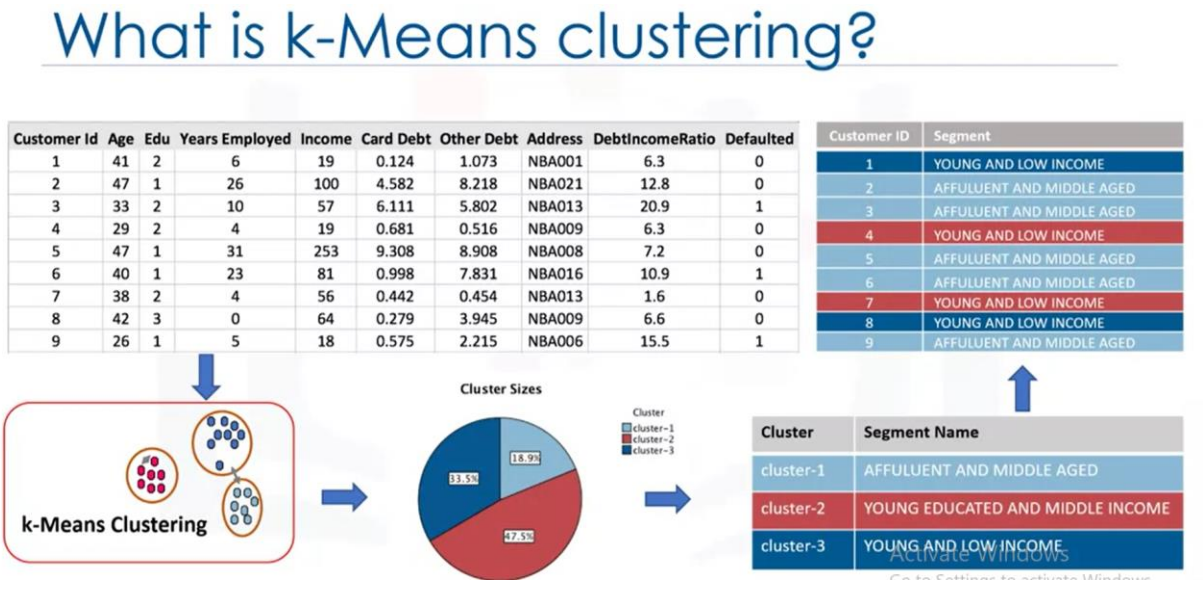
In the retail industry, clustering is used to find associations among customers based on their demographic characteristics and use that information to identify buying patterns of various customer groups. Also, it can be used in recommendation systems to find a group of similar items or similar users and use it for collaborative filtering, to recommend things like books or movies to customers. In banking, analysts find clusters of normal transactions to find the patterns of fraudulent credit card usage. Also they use clustering to identify clusters of customers. For instance, to find loyal customers versus churned customers. In the insurance industry, clustering is used for fraud detection in claims analysis, or to evaluate the insurance risk of certain customers based on their segments. In publication media, clustering is used to auto categorize news based on his content or to tag news, then cluster it so as to recommend similar news articles to readers. In medicine, it can be used to characterize patient behavior, based on their similar characteristics. So as to identify successful medical therapies for different illnesses or in biology, clustering is used to group genes with similar expression patterns or to cluster genetic markers to identify family ties.

If you look around you can find many other applications of clustering, but generally clustering can be used for one of the following purposes: exploratory data analysis, summary generation or reducing the scale, outlier detection- especially to be used for fraud detection or noise removal, finding duplicates and datasets or as a pre-processing step for either prediction, other data mining tasks or as part of a complex system. Let's briefly look at different clustering algorithms and their characteristics. Partition-based clustering is a group of clustering algorithms that produces sphere-like clusters, such as; K-Means, K-Medians or Fuzzy c-Means. These algorithms are relatively efficient and are used for medium and large sized databases. Hierarchical

clustering algorithms produce trees of clusters, such as agglomerative and divisive algorithms.

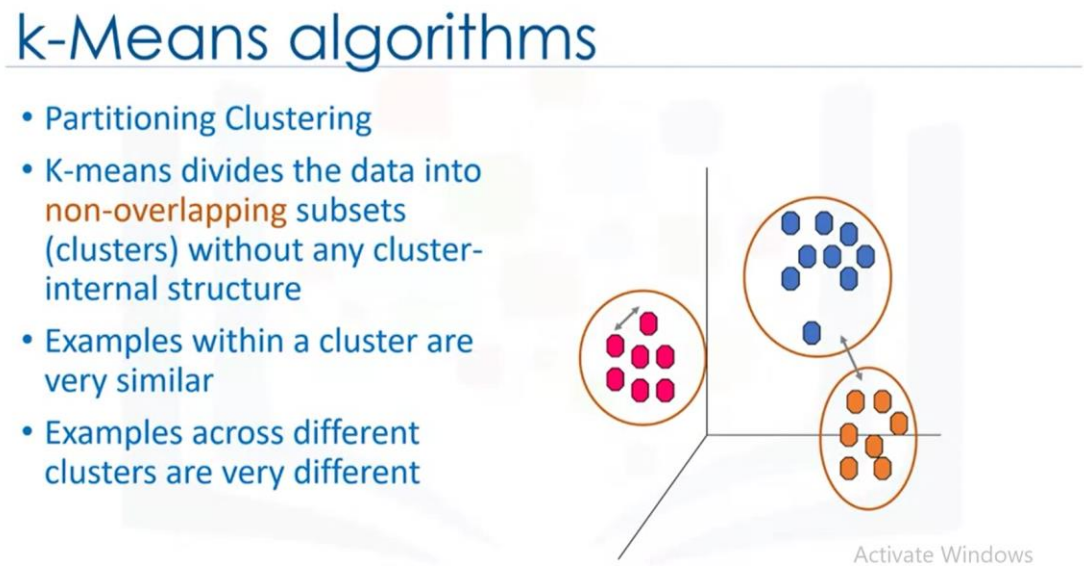


This group of algorithms are very intuitive and are generally good for use with small size datasets. Density-based clustering algorithms produce arbitrary shaped clusters. They are especially good when dealing with spatial clusters or when there is noise in your data set. For example, the DB scan algorithm.

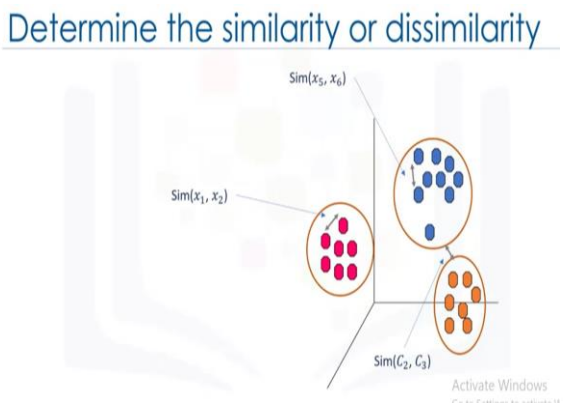


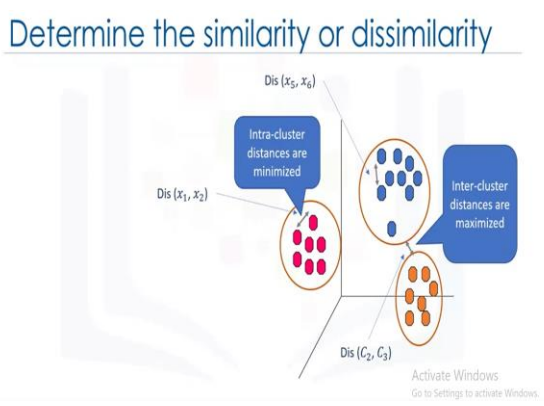
Introduction to K-means clustering:

Here, we will be covering K-Means Clustering. Imagine that you have a customer dataset and you need to apply customer segmentation on this historical data. Customer segmentation is the practice of partitioning a customer base into groups of individuals that have similar characteristics. One of the algorithms that can be used for customer segmentation is K-Means clustering. K-Means can group data only unsupervised based on the similarity of customers to each other. Let's define this technique more formally. There are various types of clustering algorithms such as partitioning, hierarchical or density-based clustering. K-Means is a type of partitioning clustering, that is, it divides the data into K non-overlapping subsets or clusters without any cluster internal structure or labels. This means, it's an unsupervised algorithm. Objects within a cluster are very similar, and objects across different clusters are very different or dissimilar.

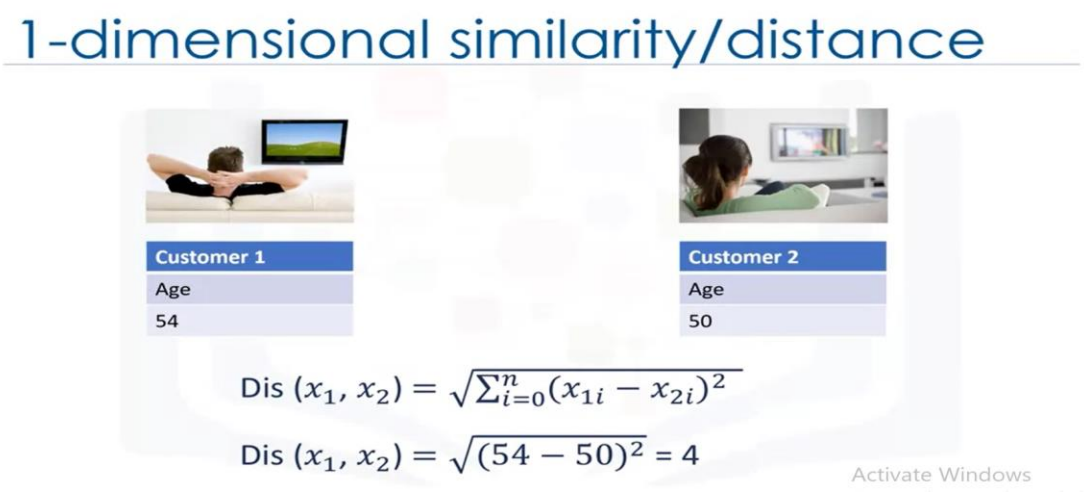


As you can see, for using K-Means we have to find similar samples: for example, similar customers. Now, we face a couple of key questions. First, how can we find the similarity of samples in clustering, and then how do we measure how similar two customers are with regard to their demographics?



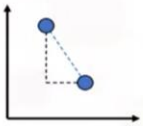


Though the objective of K-Means is to form clusters in such a way that similar samples go into a cluster, and dissimilar samples fall into different clusters, it can be shown that instead of a similarity metric, we can use dissimilarity metrics. In other words, conventionally the distance of samples from each other is used to shape the clusters. So we can say K-Means tries to minimize the intra-cluster distances and maximize the inter-cluster distances.



2-dimensional similarity/distance

Customer 1	
Age	Income
54	190



Customer 2	
Age	Income
50	200

$$\text{Dis}(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

$$= \sqrt{(54-50)^2+(190-200)^2} = 10.77$$

Multi-dimentional similarity/distance

Customer 1		
Age	Income	education
54	190	3

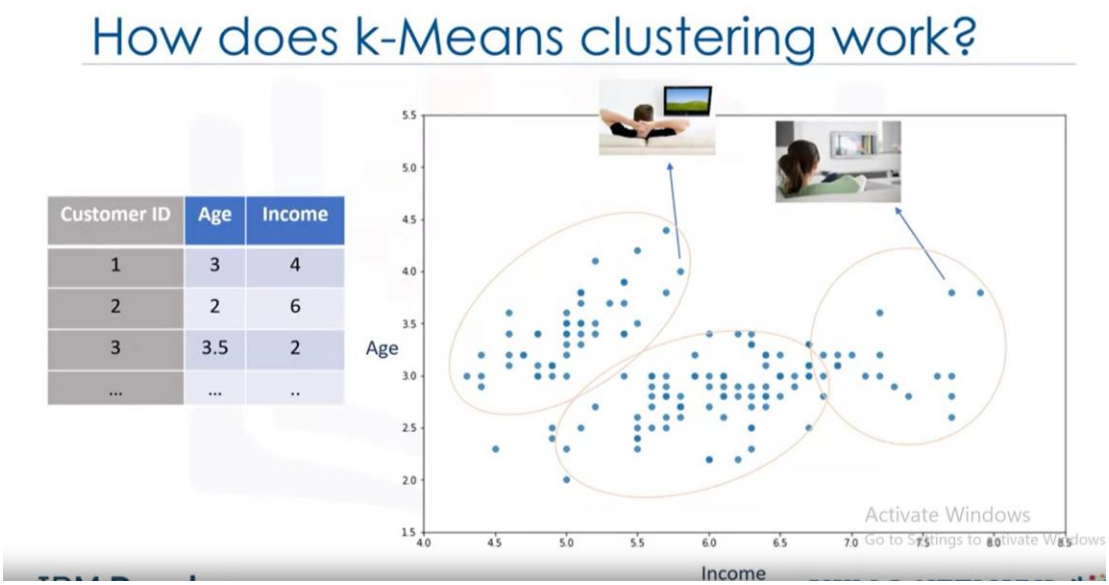
Customer 2		
Age	Income	education
50	200	8

$$\text{Dis}(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

$$= \sqrt{(54-50)^2+(190-200)^2+(3-8)^2} = 11.87$$

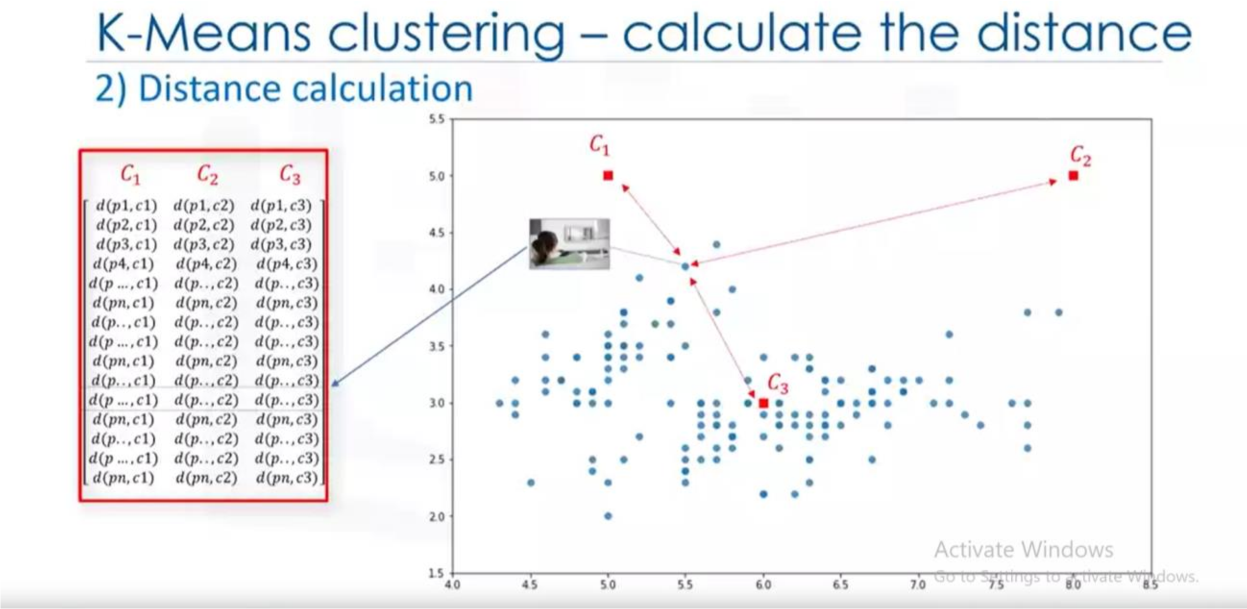
Now, the question is, how can we calculate the dissimilarity or distance of two cases such as two customers? Assume that we have two customers, we will call them Customer one and two. Let's also assume that we have only one feature for each of these two customers and that feature is age. We can easily use a specific type of Minkowski distance to calculate the distance of these two customers. Indeed, it is the Euclidean distance. Distance of x1 from x2 is root of 34 minus 30_2 which is four. What about if we have more than one feature, for example age and income. For example, if we have income and age for each customer, we can still use the same formula but this time in a two-dimensional space. Also, we can use the same distance

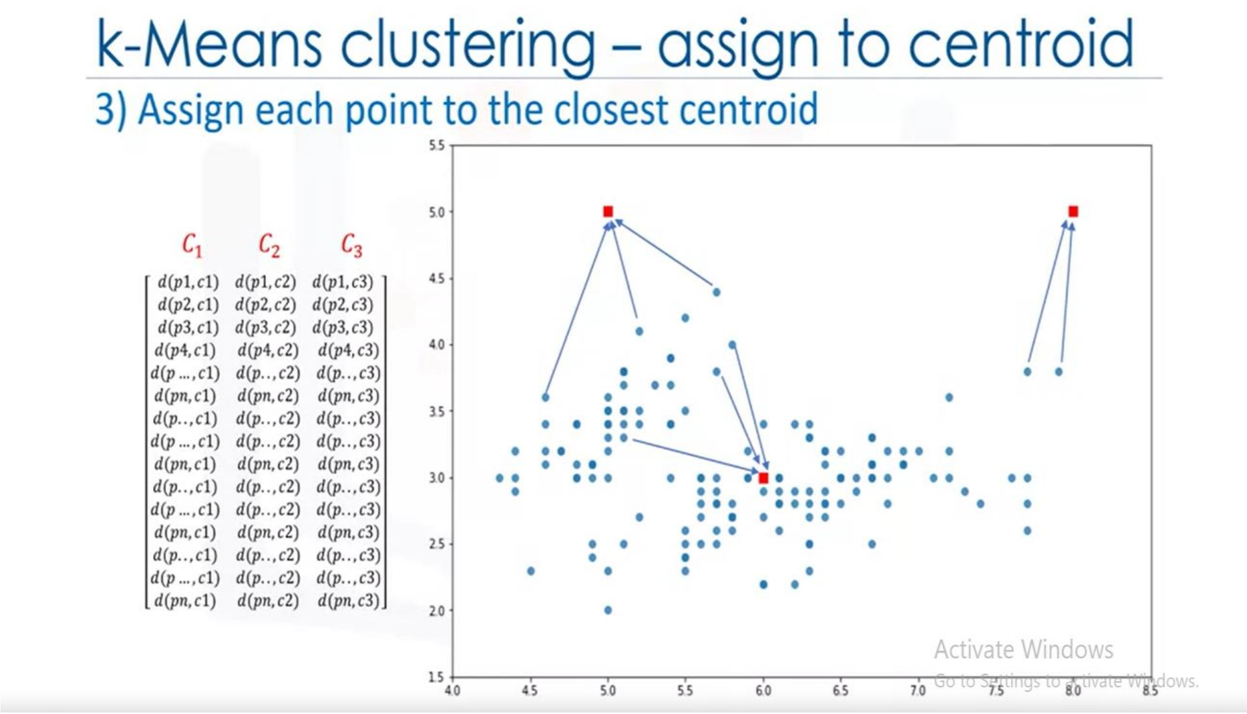
matrix for multidimensional vectors. Of course, we have to normalize our feature set to get the accurate dissimilarity measure. There are other dissimilarity measures as well that can be used for this purpose, but it is highly dependent on datatype and also the domain that clustering is done for it. For example you may use Euclidean distance, Cosine similarity, Average distance, and so on. Indeed, the similarity measure highly controls how the clusters are formed, so it is recommended to understand the domain knowledge of your dataset and datatype of features and then choose the meaningful distance measurement. Now, let's see how K-Means clustering works. For the sake of simplicity, let's assume that our dataset has only two features: the age and income of customers. This means, it's a two-dimensional space. We can show the distribution of customers using a scatter plot: The Y-axis indicates age and the X-axis shows income of customers. We try to cluster the customer dataset into distinct groups or clusters based on these two dimensions. In the first step, we should determine the number of clusters. The key concept of the K-Means algorithm is that it randomly picks a center point for each cluster. It means we must initialize K which represents number of clusters. Essentially, determining the number of clusters in a dataset or K is a hard problem in K-Means that we will discuss later. For now, let's put K equals three here for our sample dataset.

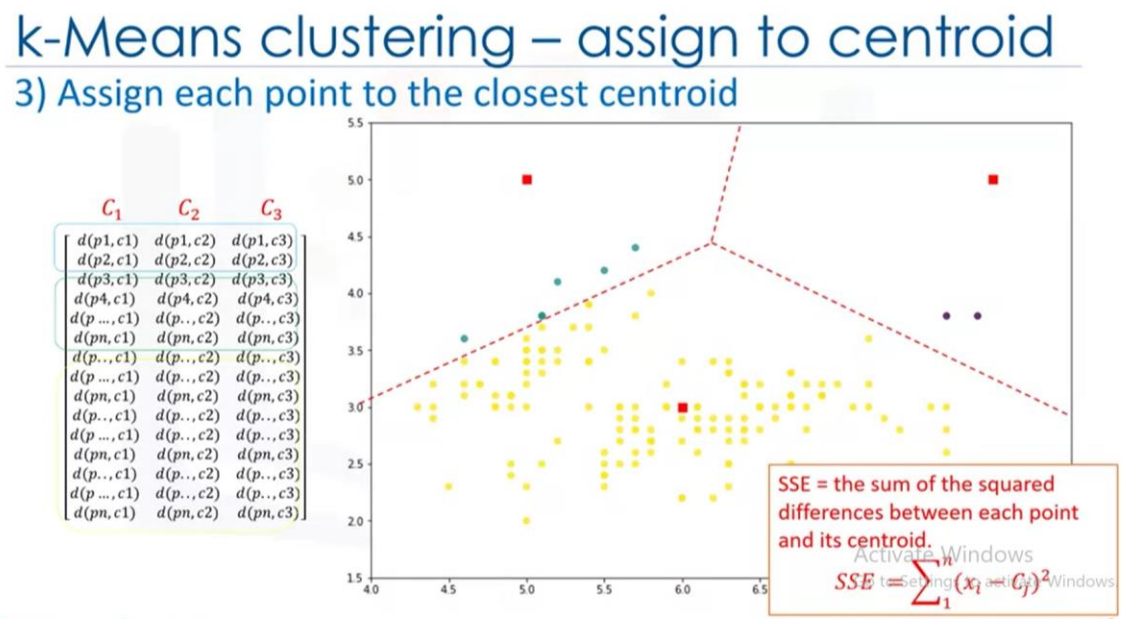


It is like we have three representative points for our clusters. These three data points are called centroids of clusters and should be of same feature size of our customer feature set. There are two approaches to choose these centroids. One, we can randomly choose three observations out of the dataset and use these observations as the initial means. Or two, we can create three random points as centroids of the clusters which is our choice that is shown in the plot with red color. After the

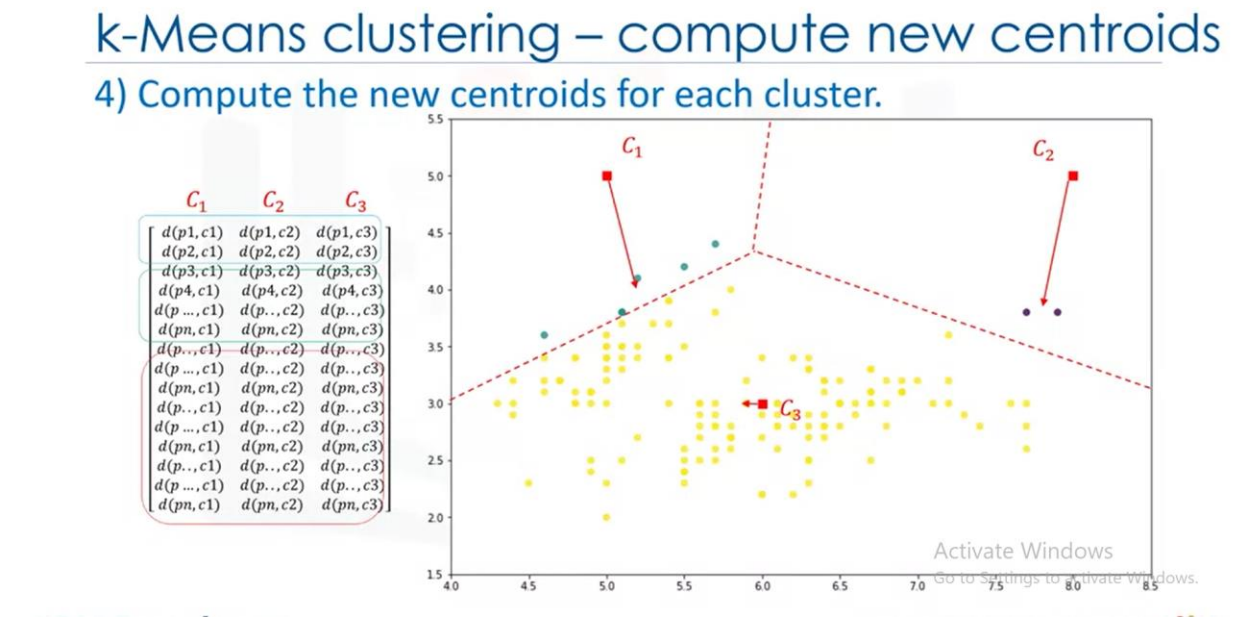
initialization step which was defining the centroid of each cluster, we have to assign each customer to the closest center.

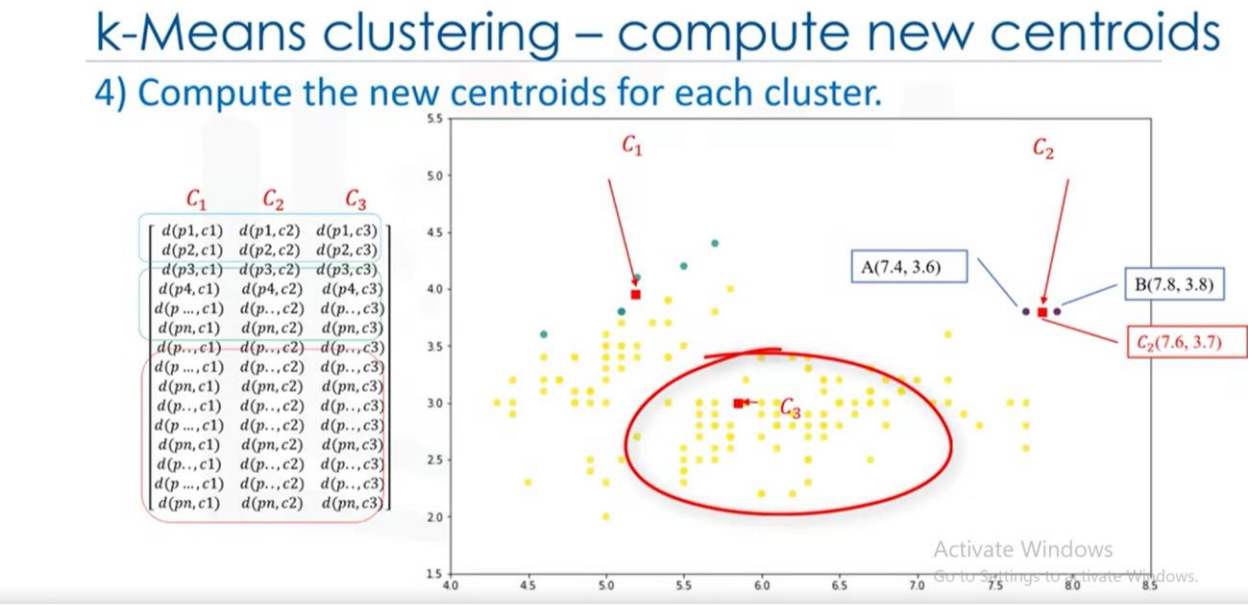






For this purpose, we have to calculate the distance of each data point or in our case each customer from the centroid points. As mentioned before, depending on the nature of the data and the purpose for which clustering is being used different measures of distance may be used to place items into clusters. Therefore, you will form a matrix where each row represents the distance of a customer from each centroid. It is called the Distance Matrix. The main objective of K-Means clustering is to minimize the distance of data points from the centroid of this cluster and maximize the distance from other cluster centroids. So, in this step, we have to find the closest centroid to each data point. We can use the distance matrix to find the nearest centroid to datapoints. Finding the closest centroids for each data point, we assign each data point to that cluster. In other words, all the customers will fall to a cluster based on their distance from centroids. We can easily say that it does not result in good clusters because the centroids were chosen randomly from the first. Indeed, the model would have a high error. Here, error is the total distance of each point from its centroid. It can be shown as within-cluster sum of squares error. Intuitively, we try to reduce this error. It means we should shape clusters in such a way that the total distance of all members of a cluster from its centroid be minimized.



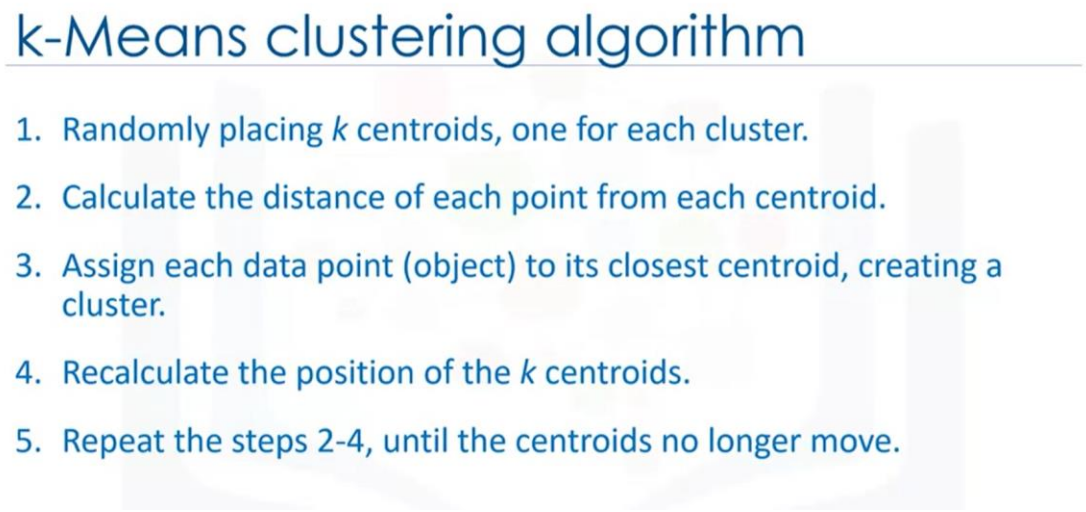


Now, the question is, how can we turn it into better clusters with less error? Okay, we move centroids. In the next step, each cluster center will be updated to be the mean for datapoints in its cluster. Indeed, each centroid moves according to their cluster members. In other words the centroid of each of the three clusters becomes the new mean. For example, if point A coordination is 7.4 and 3.6, and B point

features are 7.8 and 3.8, the new centroid of this cluster with two points would be the average of them, which is 7.6 and 3.7.

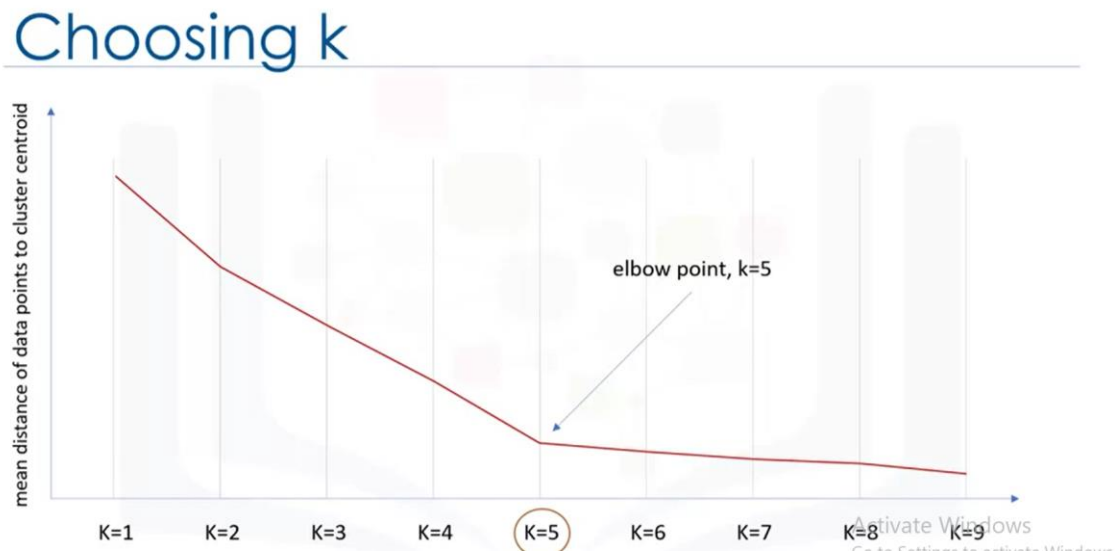
Now, we have new centroids. As you can guess, once again we will have to calculate the distance of all points from the new centroids. The points are reclustered and the centroids move again. This continues until the centroids no longer move. Please note that whenever a centroid moves, each points distance to the centroid needs to be measured again. Yes, K-Means is an iterative algorithm and we have to repeat steps two to four until the algorithm converges. In each iteration, it will move the centroids, calculate the distances from new centroids and assign data points to the nearest centroid. It results in the clusters with minimum error or the most dense clusters. However, as it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum and the result may depend on the initial clusters. It means, this algorithm is guaranteed to converge to a result, but the result may be a local optimum i.e. not necessarily the best possible outcome. To solve this problem, it is common to run the whole process multiple times with different starting conditions. This means with randomized starting centroids, it may give a better

outcome. As the algorithm is usually very fast, it wouldn't be any problem to run it multiple times.



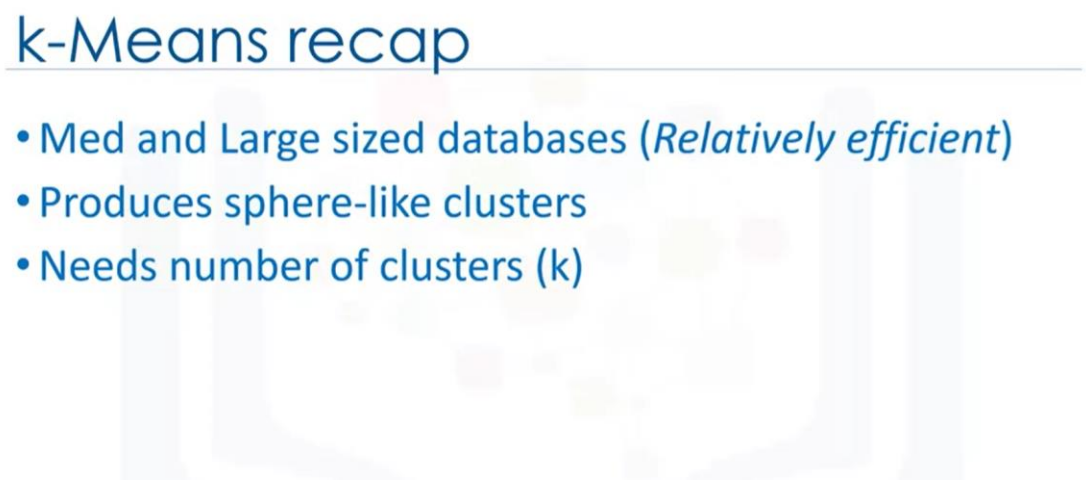
More on k-Means:

Here, we'll look at k-Means accuracy and characteristics. Let's define the algorithm more concretely, before we talk about its accuracy. A k-Means algorithm works by randomly placing k centroids, one for each cluster. The farther apart the clusters are placed, the better. The next step is to calculate the distance of each data point or object from the centroids. Euclidean distance is used to measure the distance from the object to the centroid. Please note, however, that you can also use different types of distance measurements, not just Euclidean distance. Euclidean distance is used because it's the most popular. Then, assign each data point or object to its closest centroid creating a group. Next, once each data point has been classified to a group, recalculate the position of the k centroids. The new centroid position is determined by the mean of all points in the group. Finally, this continues until the centroids no longer move. Now, the questions is, how can we evaluate the goodness of the clusters formed by k-Means? In other words, how do we calculate the accuracy of k-Means clustering? One way is to compare the clusters with the ground truth, if it's available.



However, because k-Means is an unsupervised algorithm we usually don't have ground truth in real world problems to be used. But there is still a way to say how bad each cluster is, based on the objective of the k-Means. This value is the average distance between data points within a cluster. Also, average of the distances of data points from their cluster centroids can be used as a metric of error for the clustering algorithm. Essentially, determining the number of clusters in a data set, or k as in the k-Means algorithm, is a frequent problem in data clustering. The correct choice of K is often ambiguous because it's very dependent on the shape and scale of the distribution of points in a dataset. There are some approaches to address this problem, but one of the techniques that is commonly used is to run the clustering across the different values of K and looking at a metric of accuracy for clustering. This metric can be mean, distance between data points and their cluster's centroid, which indicate how dense our clusters are or, to what extent we minimize the error of clustering. Then, looking at the change of this metric, we can find the best value for K. But the problem is that with increasing the number of clusters, the distance of centroids to data points will always reduce. This means increasing K will always decrease the error. So, the value of the metric as a function of K is plotted and the

elbow point is determined where the rate of decrease sharply shifts. It is the right K for clustering.



This method is called the elbow method. So let's recap k-Means clustering: k-Means is a partition-based clustering which is A, relatively efficient on medium and large sized data sets; B, produces sphere-like clusters because the clusters are shaped around the centroids; and C, its drawback is that we should pre-specify the number of clusters, and this is not an easy task.