upGrad

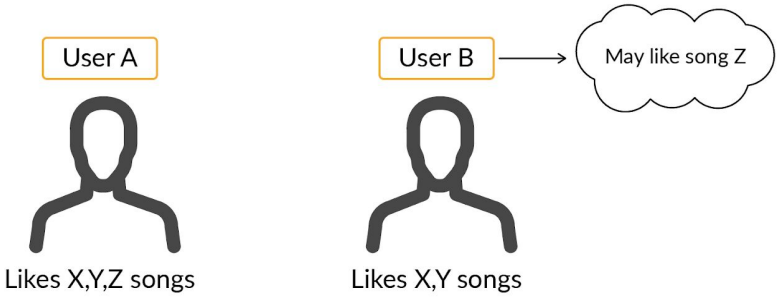
Summary

Understanding Machine Learning

In this module, you have gained an understanding of the broad classification of Machine Learning. You have also learnt about the implementation of the Apriori algorithm, which falls under the association rule mining category of algorithms. Besides that, you have learnt about the various steps involved in putting any machine learning model into production, i.e., setting up a **machine learning pipeline**.

What is Machine Learning?

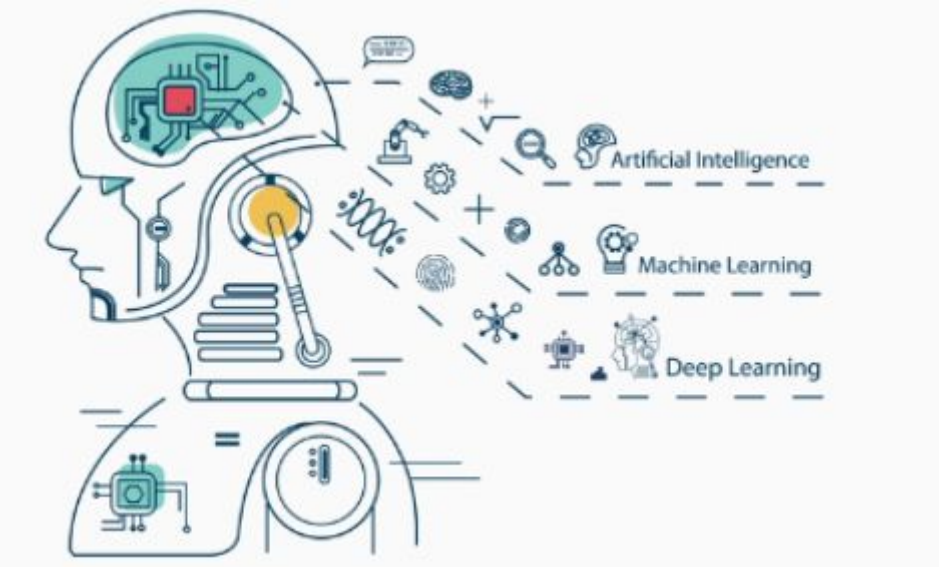
Machine Learning is what allows users to look for patterns in a data set and use that information to make future predictions. How else would YouTube know what music you like and create an entire playlist for you; or how else would Amazon predict what products you would likely want to buy next? Machine learning is used very commonly in the finance industry, for instance, enabling banks to detect suspicious transactions, stock prediction, etc.



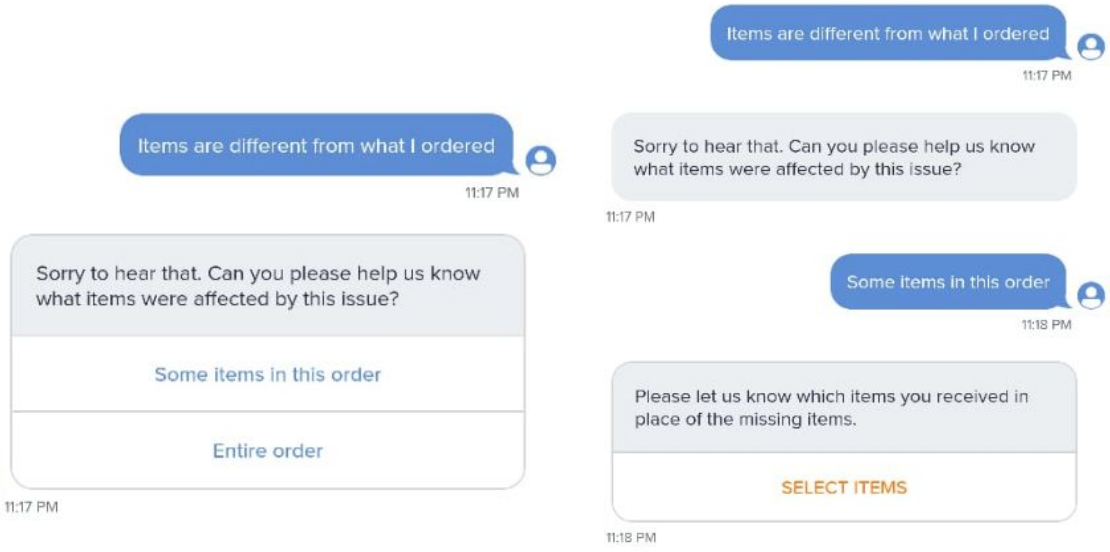
To be able to perform these tasks, we have to train our machine learning model on a set of features within the data. In the case of banks detecting fraudulent transactions, the various features would include vendor information, time and date of transactions, transaction details, etc. Such a model would look at all past transactions (which would include both fraudulent and non-fraudulent transactions) and analyse the likely values for each feature such that a fraudulent transaction is most likely. Thus, when we feed new transaction data into the model, it uses the information learned from the previous transactions to predict whether the current transaction is more likely to be fraudulent or not and classify it accordingly.

upGrad

Difference between AI, ML and DL



Artificial Intelligence is a broader term that includes both **Machine Learning and Deep Learning**. The chatbot that you use to interact with your phone is a practical application of Artificial Intelligence.



Another example of Artificial Learning is when your system automatically classifies your emails as spam or not spam. This model is based on a Machine Learning algorithm that constantly updates its rules by looking at past spam and non-spam emails and also learning from spam emails that you receive from time to time. On the other hand, the Google Assistant on your phone, which needs to understand what you speak and consequently respond by performing the task that you ask it to perform, is based on deep models, which analyse the input at multiple layers and perform the task accordingly.

Application of Machine Learning

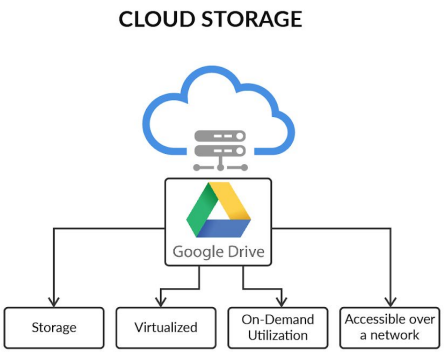
All of us use machine learning directly or indirectly without even realising it. Be it the products that we use in our daily lives, the various services that we use, the movie or song recommendations that we follow, and so much more. The moment you log in to your online retail store account, the system knows exactly what products you prefer and shows offers on those products. Even the arrangement of products inside a supermarket is done by studying the behaviour of the customers, so that the sales of the market can be maximised. From the offers that you come across at a nearby store to the planning involved in running a supply chain, you can find elements of machine learning being used.

Now, let's take a look at a few aspects of machine learning applications.

1) E-commerce websites: All e-commerce websites use a recommendation engine to maximise their sales and customer share of the wallet.
2) Manufacturing sector: Various manufacturing operations involve processes that are governed by specific levels of components and require maintaining specific environmental conditions. The conditions need to be precise in order to get the perfect product. In order to monitor these conditions, companies use anomaly detection systems, which warn the engineers if any process goes out of control.
3) Finance & banking: You often get phone calls from companies asking whether you would be interested in availing loans. Now, how do they target you specifically as a customer? Of course, it is by virtue of using machine learning algorithms.
4) Healthcare: Machine learning has also worked wonders in the healthcare industry. Be it identifying optimal treatment options or providing accurate diagnostic reports, machine learning has been involved, and it is intimately associated with our lives.

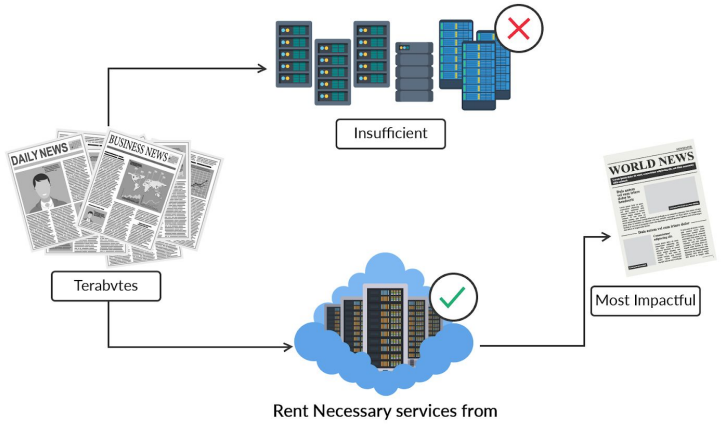
Machine Learning and Cloud

Many of us have shared documents routinely via Google Drive without even realising where the server stores these documents. Well, the storage space is nothing but the cloud, which is an external location different from the local storage. This external location involves servers or a data centre, which you can access over a network. In technical terms, cloud refers to virtualisation of services, such as storage, computing power, etc., and delivering these services over a network.



Similarly, let's say you are visiting different websites; the traffic across these websites may not be the same at all times. At times during the day, the traffic might be, let's say, 500 users/min while at nighttime, it might be 10 users/min. Also, during a sale, the traffic on an e-commerce website might be 10,000 users/min. Now, how will these websites handle such varying traffic? They cannot simply add more servers to handle their computational needs, which can vary every minute. Instead, they use hosting services. These provide the necessary infrastructure to meet their processing needs.

The two examples above are those of cloud solutions where necessary infrastructure such as storage, memory or computation power are provided over a network, thus allowing you to perform operations that might be difficult to carry out using your local server or personal devices.



There are many players in the market that provide these cloud services. Some of the major players include Amazon Web Services (AWS), the Google cloud platform (GCP), Microsoft Azure and IBM cloud. These cloud providers offer a wide range of services, which include solutions for archiving data, having your databases or for hosting your websites.

In the domain of machine learning, where you need a lot of computational power to train and build models, you need machines or servers with high computing power. Clouds make this possible. You can use

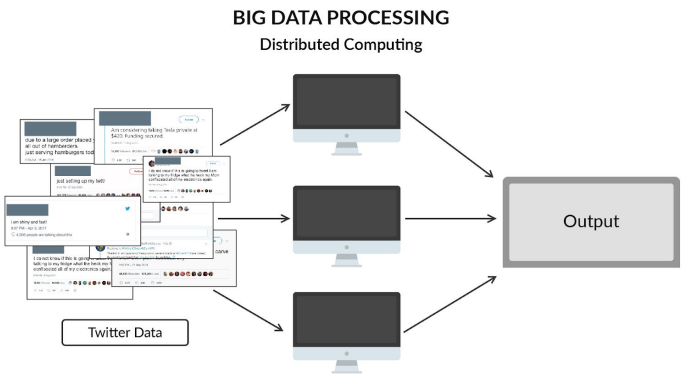
them right from storing huge chunks of data, which can include logs generated from a website or data from a streaming site, to deploying your solution.

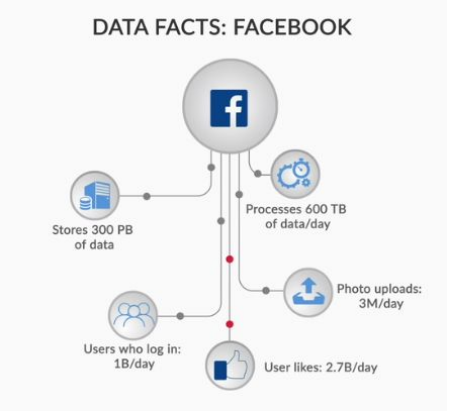
In recent times, the field of cloud computing has been witnessing a lot of advancements in order to build machine learning solutions at any scale. For instance, Amazon Sagemaker and Cloud AutoML are some of the readily available machine learning services provided by AWS and GCP, respectively, which enable building high-quality models specific to the business needs of organisations.

Machine Learning and Big Data

Big Data represents large data sets that **cannot be handled using a single machine**. The data that was handled traditionally is small in size, and so it was possible to store the data and perform computation using the available computer memory. Now, it is possible to increase the available computer memory or storage with increase in the size of the data, but this no longer helps when the data being handled is in the range of petabytes and terabytes. What you need now are multiple machines working together as a single machine. This set of machines working together to achieve a single task is a distributed system. The machines connected together need to run a set of services in order to communicate and coordinate with each other while performing an action.

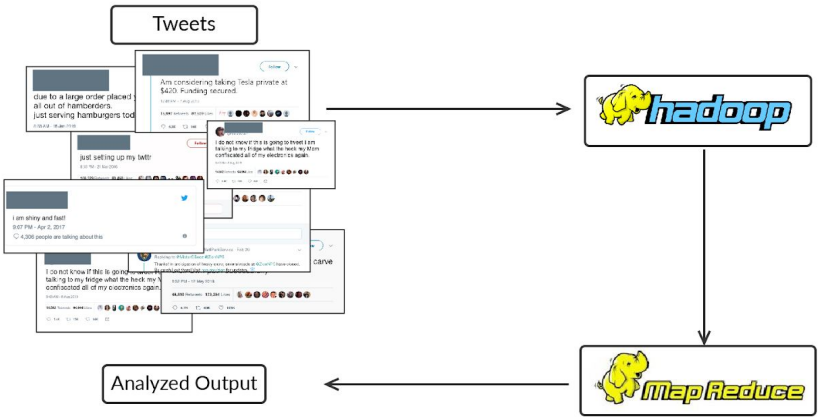
Let's say you have a huge file and are supposed to return the words and their corresponding counts from this file. Now, considering the file is huge, it would not be possible to handle it with a single machine. Hence, the first step to implement this word count would be to divide the data of this large file into chunks or, in other words, split the data. This data would be present on different nodes or machines. Now, you can return the words and respective counts for the data available on a single device. But having these words and their respective counts would not fetch the answer; what you need is a cumulative result that combines the individual results at each machine. To do this, you need to use a function that would return a value for a particular word. Based on these values, you can now send the list of words with specific values to a new machine and there you can combine the words with a common value; and, finally, combining the results at each of these machines would give you the final result.





Here, in the entire process, you never handled data at a single location, but were able to achieve the result using multiple devices. You can use this distributed framework to handle massive data sets. However, while doing so, you may encounter many problems, such as a machine failing during the execution or a particular machine taking too much time, making the others wait, or a single machine handling a huge amount of data. In the latter part of the course, you will learn how these problems are handled using frameworks such as the Hadoop File System (HDFS) and MapReduce.

TWITTER ANALYSIS



How is big data or distributed computing related to machine learning?

Machine learning is all about learning the underlying trends in the data. If the data generated is so huge that you need a scalable solution to process it without any latency or delays, then you need the concepts of big data.

Types of Machine Learning

Machine learning can be defined as follows:

"**A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.**"

- Tom Mitchell

EXAMPLES TO MACHINE LEARNING

Task T
Recognizing face in an image

Experience E
Set of images.

Performance P
Correct recognition of face

Task T
Assigning sentiment to a tweet

Experience E
Sample tweets

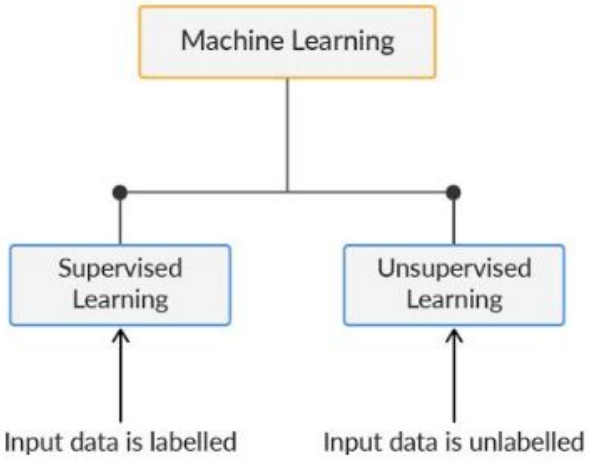
Performance P
Correct tweet classification

Task T
Grouping of photos

Experience E
Set of photos

Performance P
Photos with similar features grouped

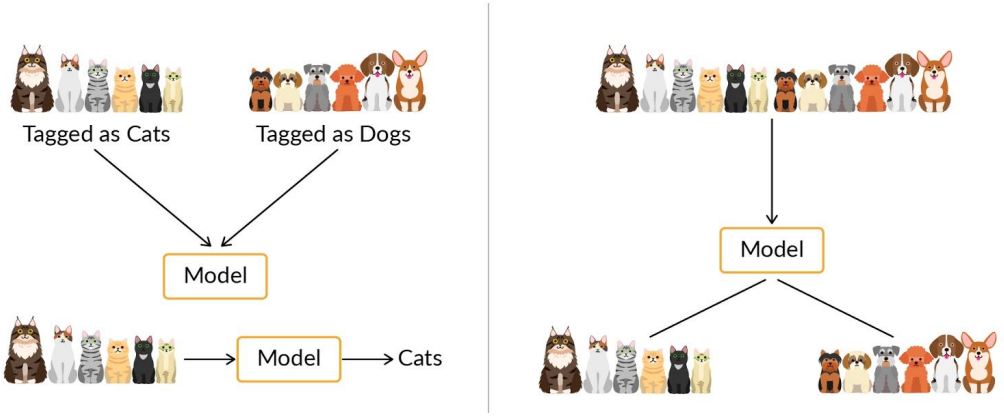
Machine learning algorithms can be **broadly classified into supervised learning and unsupervised learning.**



Supervised learning is the class of machine learning where you learn from **well-labelled** data; or, in other words, the data should have a well-defined input and output. For example, imagine you want to predict the future value of a stock based on its historical data; this would be supervised learning, because the learning here is taking place to predict something based on past trends.

Now, unlike supervised learning, in **unsupervised learning**, you do not have any information about the end outcome. Here, you work with **unlabelled data**. Consider the example of cats and dogs shown below; here, let's say, you aren't told anything about the images, but are asked to group them together. So, first, you start looking at the noses, eyes, ears and other features in the images and start grouping similar images together. Here you do not have any labelled data; what you have is unlabelled data and learning happens based on the patterns observed in this data.

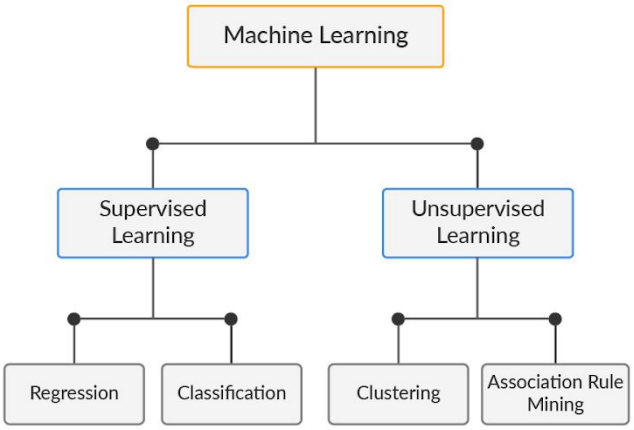
SUPERVISED VS UNSUPERVISED LEARNING



Supervised Learning Algorithms

Supervised learning can be of two types:

1. Regression
2. Classification

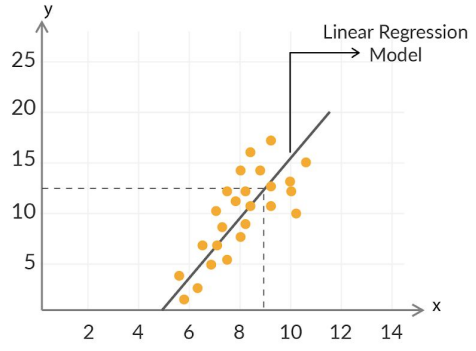


Regression is a supervised learning technique where machine learning happens based on labelled data, and you try to predict a continuous value, or, in other words, a numerical value.

For example, when you book a cab, you are shown an estimated time of arrival, or ETA. The system would not show a random number by merely observing the distance between two locations. Multiple factors, such as geographic location, the time of the day, driver information and traffic information, are involved in the actual calculation of the ETA. From the past data, the model finds the relationship between these parameters and the ETA; then based on the learned relationship, the model tries to predict the ETA in the current scenario.

LINEAR REGRESSION

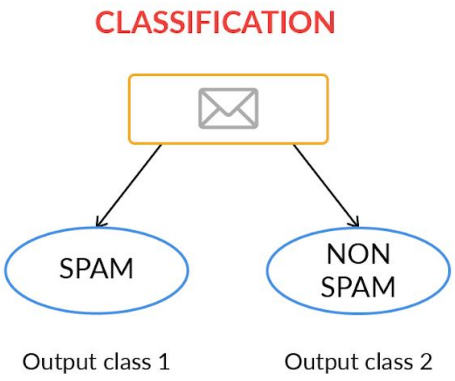
GPA	SALARY
9.2	20
8.4	15
6.4	8
... ...	
8.2	17



Classification is also a supervised learning technique where, as the name suggests, you try to classify your observations. The main difference from regression is that in linear regression, the output is a continuous value, whereas in classification, the output is a discrete value.

Now, have you ever wondered how your email service provider automatically classifies some of your emails as spam and moves them to your spam folder? Well, this is a classic example of classification. The

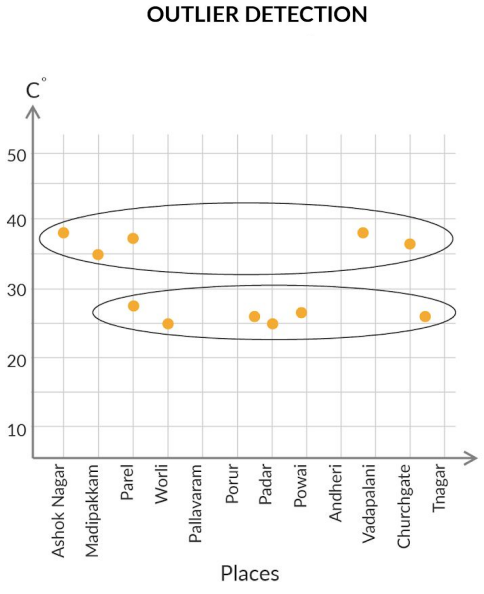
service provider observes the characteristics of the emails that are marked as spam by many users; they consider parameters such as sender information, the content of these mails, etc., it deploys models that will analyse an incoming email and categorise it as spam or non-spam. This prediction whether a particular email is spam or not is called classification. Classification involving text is known as **Text Classification.**



Unsupervised Learning Algorithms: I

Clustering is the process of organising objects into groups whose members are similar in one way or the other, and display some underlying pattern. Unlike supervised learning, unsupervised learning does not have a target variable and aims to organise objects into multiple segments based on their properties.

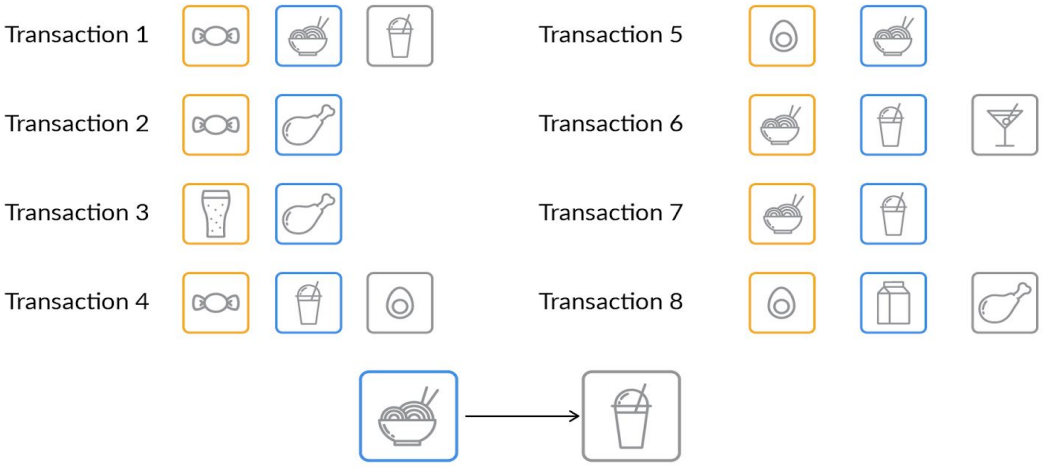
Now, assume that you are the head of marketing at a multinational retail chain. To boost sales in the coming quarter, you wish to understand your customers' preferences and scale up your business. Understanding each customer on an individual level is practically impossible. Instead, you can group them and identify target groups based on features such as age, buying frequency and geographical area. This is a classic example of clustering, where you find patterns in the data by grouping customers with similar characteristics.



Common clustering algorithms include DBSCAN, K-means clustering and bisecting K-means clustering.

Association rule mining is an unsupervised learning technique used to identify interesting patterns between seemingly independent relational databases or other data repositories.

When this algorithm was applied to the data collected from certain Walmart stores, it led to an interesting revelation, which was later termed '**Walmart's beer diaper parable**'. The analysis suggested that American males who came to the stores to buy diapers were also likely to buy beer. After seeing these results, Walmart stores started selling beer along with diapers and beer sales shot up.



Hence, we can form a rule of the form:
{Diaper} ----> {Beer}
This is an example of an association rule. Some common examples of association rules are as follows:
{Bread, Egg} ----> {Milk}
{illness symptom 1, illness symptom 2} ⟶ {illness symptom 3}

Every rule consists of:

a) an **antecedent** (if) and
b) a **consequent** (then).

An antecedent is something that is found in the data, and a consequent is an item that is found in combination with the antecedent. Have a look at this rule, for instance:

"If a customer buys bread, he's 70% likely to buy milk."

In the association rule above, bread is the antecedent and milk is the consequent.

Here are the key metrics for generating association rules:

1. **Support:** It denotes the frequency of occurrence of an item:
    - Support = (Number of times the item occurred)/(Total number of transactions)
2. **Confidence:** It denotes the likelihood of occurrence of an event given that another event has occurred:
    - Suppose we want to examine whether the given rule is an association rule:
      {A, B} ---> {C}

      Confidence = Support(A, B, C)/Support(A, B)

Suppose we have a Market Dataset consisting of N transactions, and we need to discover all rules that satisfy the following conditions:

Support >= min_support
Confidence >= min_confidence

Where min_support and min_confidence are the support and confidence thresholds, respectively.

Every association rule mining algorithm can be broken down into two subparts:

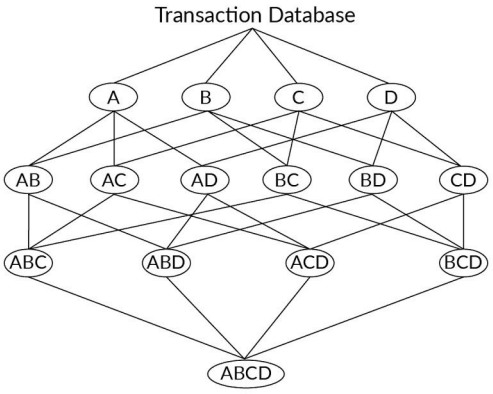
1. **Frequent Itemset Generation**:
    - This part of the algorithm returns all possible combinations of itemsets that have support values greater than their respective thresholds.
2. **Rule Generation**:
    - This part of the algorithm uses the frequent itemsets generated before to check for rules that satisfy the confidence threshold.

A brute force approach (considering all possible permutations and combinations for finding rules) would be computationally too expensive and, hence, we need a more systematic and less computationally expensive approach. The 'Apriori Algorithm' is a popular algorithm used for Rule Generation.

Apriori

Consider a store transaction database that consists of all the transactions of each customer. Assuming there are only four items in the store, {A, B, C, D}, and that each transaction consists of at least one of the four items, we can create a visual as shown below. We can generate frequent itemsets using the Apriori principle.

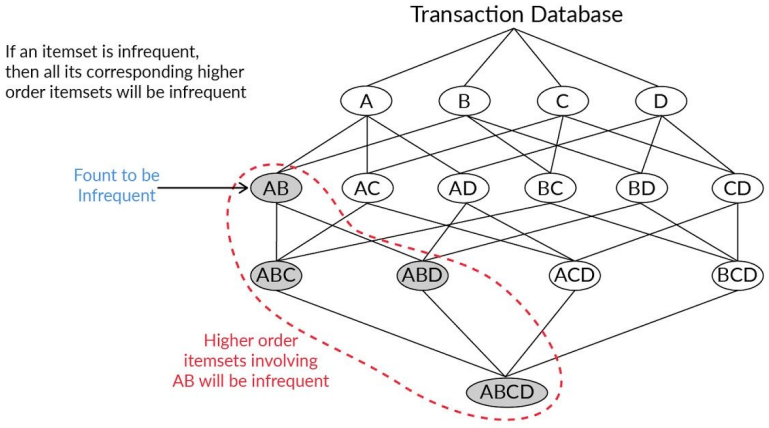
APRIORI ALGORITHM



The Apriori Principle

1. If an itemset is found to be frequent (the support value does not exceed the threshold), then all of its lower-order itemsets must be frequent.
2. Vice versa, if an itemset is found to be infrequent, then all of its consequent higher-order itemsets must be infrequent.

APRIORI ALGORITHM



Candidate Generation and Pruning for Apriori

1. Candidate Generation:
   - Merge itemsets of the kth order to create itemsets of order k+1
   - Two frequent itemsets of order k are merged only if their k-1 itemsets are identical.
   - **Example:** Suppose {A, B}, and {A, D} are frequent; they can be merged to form {A, B, D}.
   - **Note:** All unique items must be arranged in a particular order, and this order must be maintained while creating the higher-order itemsets.

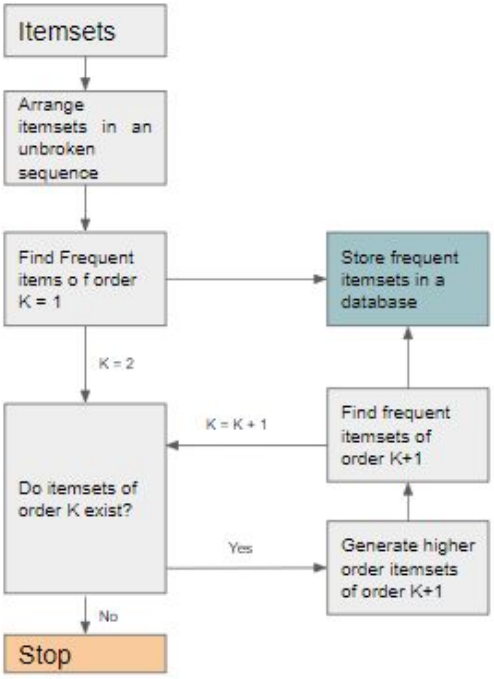
2. Candidate Pruning:
    - Eliminate candidates if their support values are less than the threshold
    - Confidence-based pruning:
        a. Let us consider the generating rules for the itemset {a, b, c, d}.
        b. Suppose the confidence for {b, c, d} ---> {a} is less than the threshold; we can eliminate all rules in which the rule consequent consists of higher-order itemsets containing {a}.
        c. Thus, we can eliminate {b, d} ⟶ {a, c}, {c, d} ⟶ {a, b}, {b, c} ⟶ {a, d}, {d} ⟶ {a, b, c}.

Apriori Algorithm

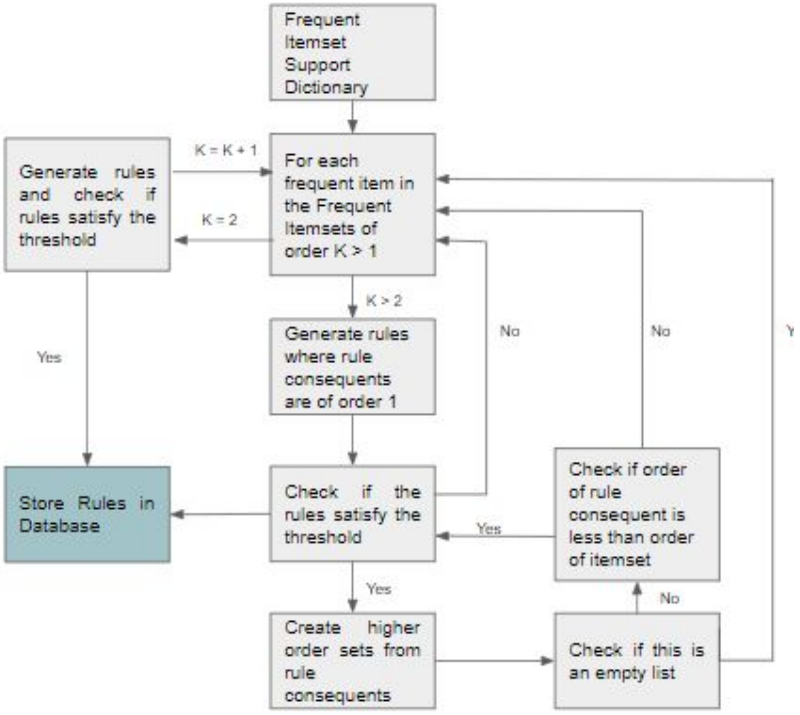
1. The first step of the algorithm is to identify distinct items in a given set of transactions.
2. Once you have different items, the next step would be to calculate the frequency of each of them. Dividing this frequency by the total number of transactions will give you the support values. Items with support values less than the threshold support need to be removed from our itemset.
3. The next step would be to form itemsets involving two items.
4. Again, you need to check and remove the items that do not qualify the minimum support criteria.
5. These two-item sets then become inputs for the generation of three-item sets, and, again, itemsets not satisfying the minimum support criteria are removed. This process continues until you can generate no new itemsets.

upGrad

1. Frequent Itemset Generation for Apriori - Flow Chart



2. Rule Generation for Apriori - Flow Chart



Disadvantages

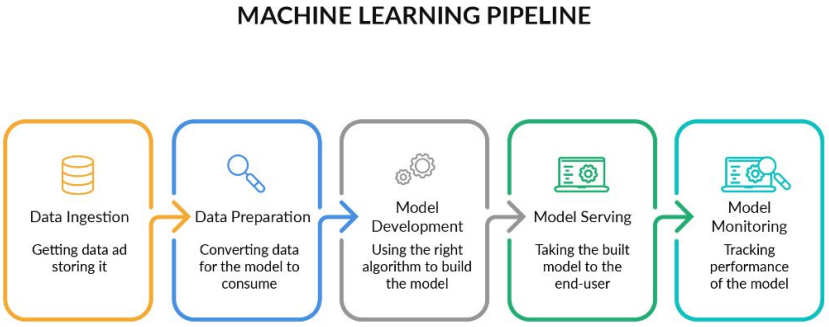
Although this algorithm is easy to understand and implement, it has its drawbacks. It is a computationally expensive algorithm. To compute the support for each itemset, you have to go through the entire database of transactions and check whether the itemset is a part of a transaction or not. Applying such an algorithm on a smaller data set having 20 items and a hundred purchases would not be a challenging task; but if you are dealing with the sales happening on Amazon or Flipkart, for instance, then you can no longer handle such jobs with regular infrastructure. You need cloud to store the massive amount of data and have multiple instances in the cloud to run the algorithm.

You need to create the required number of instances in the cloud and use distributed filesystem frameworks such as Hadoop File System (HDFS) to store the billions of transaction inputs that are fed to the algorithm in multiple nodes. We use frameworks such as MapReduce to write our distributed programs to run on these data sets; determine frequently occurring itemsets in the database; and use this information to discover the association rules present.

Summary

The Machine Learning Pipeline

There are many steps involved in building an end-to-end machine learning model. Applying the algorithm is one of the crucial steps of the entire machine learning pipeline, although there are other essential steps as well, such as how to get the data, where to store the data and how to get your solution to the end user.



You will broadly cover the following topics:

- Data Ingestion
- Data Preparation
- Model Development
- Model Serving
- Model Monitoring

upGrad

Data Ingestion

Data ingestion can be defined as the process of **absorbing data for immediate use or storage**. It is a bridge to transfer the data generated at a particular source to a destination for storage.

Some of the key aspects that need to be considered during the data ingestion phase include:

1. Type of data
2. Processing requirements of data
3. Storage

Based on its structure, data can be classified into three types:

1. **Structured data**: This type of data is organised and can be stored in SQL databases, i.e., in tables with rows and columns. Thus, its advantage is that it can be entered, stored, queried and analysed efficiently using Structured Query Language (SQL):
    - Examples include Aadhaar data, financial data and the metadata of files.

2. **Unstructured data**: We can naively describe unstructured data as being a complement to structured data, i.e., it cannot be organised easily and stored in the form of tables (columns and rows).
    - Examples of unstructured data include images, audio, video and chat messages, which are usually generated and consumed by humans. It is not surprising that the vast majority (~80%) of the data being generated in the world today is unstructured.

3. **Semi-structured data**: There is no predefined schema for semi-structured data. In terms of readability, it lies between structured and unstructured data:
    - XML and JSON files are examples of semi-structured data.

Emails are also examples of semi-structured data, since they contain fields such as ‘From’, ‘To’, ‘Subject’ and ‘Body’. Emails have internal tags and markings that identify separate data elements; this enables information-grouping and the creation of hierarchies among the elements. However, this schema does not constrain the data as in an RDBMS, e.g., the ‘Subject’ and ‘Body’ fields may contain text of any size.

Once you know the type of data being handled, the next step would be to identify the processing requirements of the data. Ingestion tools and storage components vary based on these processing requirements. Different types of processing include:

1. **Batch processing:** In batch processing, all of the data is collected and stored offline. Then a code is run on the complete data set to generate valuable insights. Usually, it takes a long time for batch processing to complete, but it also provides more accurate results.

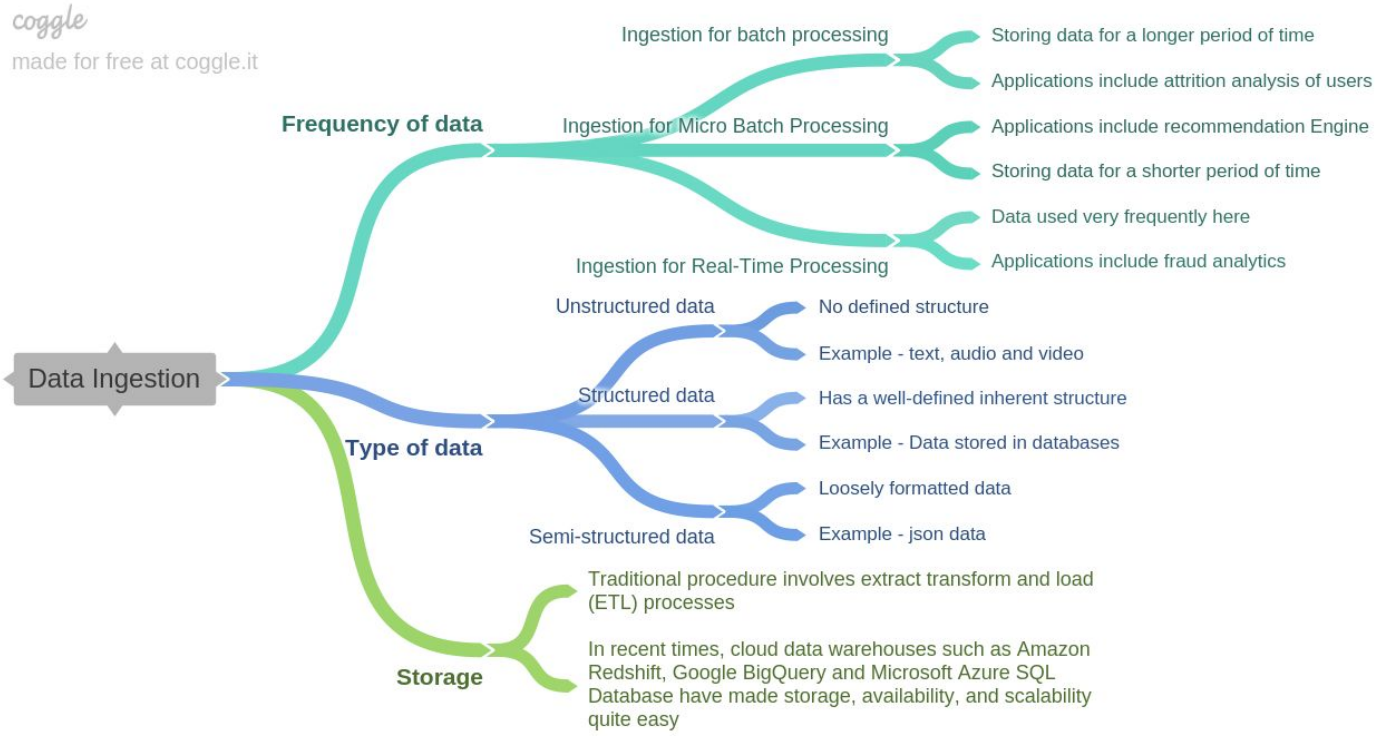
2. **Stream processing:** Unlike batch-processing systems, stream-processing systems handle all incoming data instantaneously. However, systems using stream processing handle data that is usually of the order of a few KBs. The results produced are available almost instantly after the data is received.
3. **Micro-batch processing:** This is very similar to batch processing, except in this case, the batches are very small. In micro-batch processing, the batch size generally depends on a few events gathered over a short span of time, say, a few milliseconds.

Batch processing is generally used for complex analytics, whereas stream processing is used to gain real-time insights.

The storage component for the collected data is decided based on the above factors:

1. **If the data is well structured, then a traditional database can be used.**
2. **If the data is unstructured, then NoSQL databases are needed to store it**.

Amazon provides multiple storage options such as **Amazon DynamoDB**, **Amazon RDS**, **Amazon Redshift and Amazon Cache**. The choice of these services depends on the type of data you are dealing with, the latency you require, budget and various other factors.



upGrad

Data Preparation

Once you have the necessary data stored at a particular location, the next step is to put it in the right format. Put simply, let's say you are cooking a dish:

1. The first step is to obtain all the necessary ingredients necessary to cook the dish.
2. The next step is to prepare the ingredients before cooking the dish.

Data preparation is the process of organising or structuring an unprocessed data set so that it can be used for analysis. The first and foremost step in data preparation includes exploring the data set.

During the exploratory phase, two aspects are stressed upon:

1. **Data consistency**: This involves checking whether or not the data is consistent throughout the data set. For example, a customer's transaction data collected across different countries needs to be in a specific format before it can be analysed for insights.
2. **Data integrity**: You need to check whether or not the features in the data set have permissible values. For example, let's say the age column has values higher than 200; this means there is definitely a particular discrepancy in the data set. You are expected to correct/modify it to make it usable for analysis.

Once these checks are complete and specific issues in the data set, such as missing values, outliers and incorrect values, are identified, the next step is to rectify the the issues.

Data cleaning involves **handling missing values and outlier values** in the data set. First, you check whether the missing values are at random places across the entire data set or are limited to a part of the data set with specific properties. Once you have identified this, you may simply discard that particular part of the data set or choose any of the following methods to impute those values:

1. Mean/median/mode imputation
2. Random sample imputation
3. Replacing with an arbitrary value
4. Missing value indicator
5. Multivariate imputation

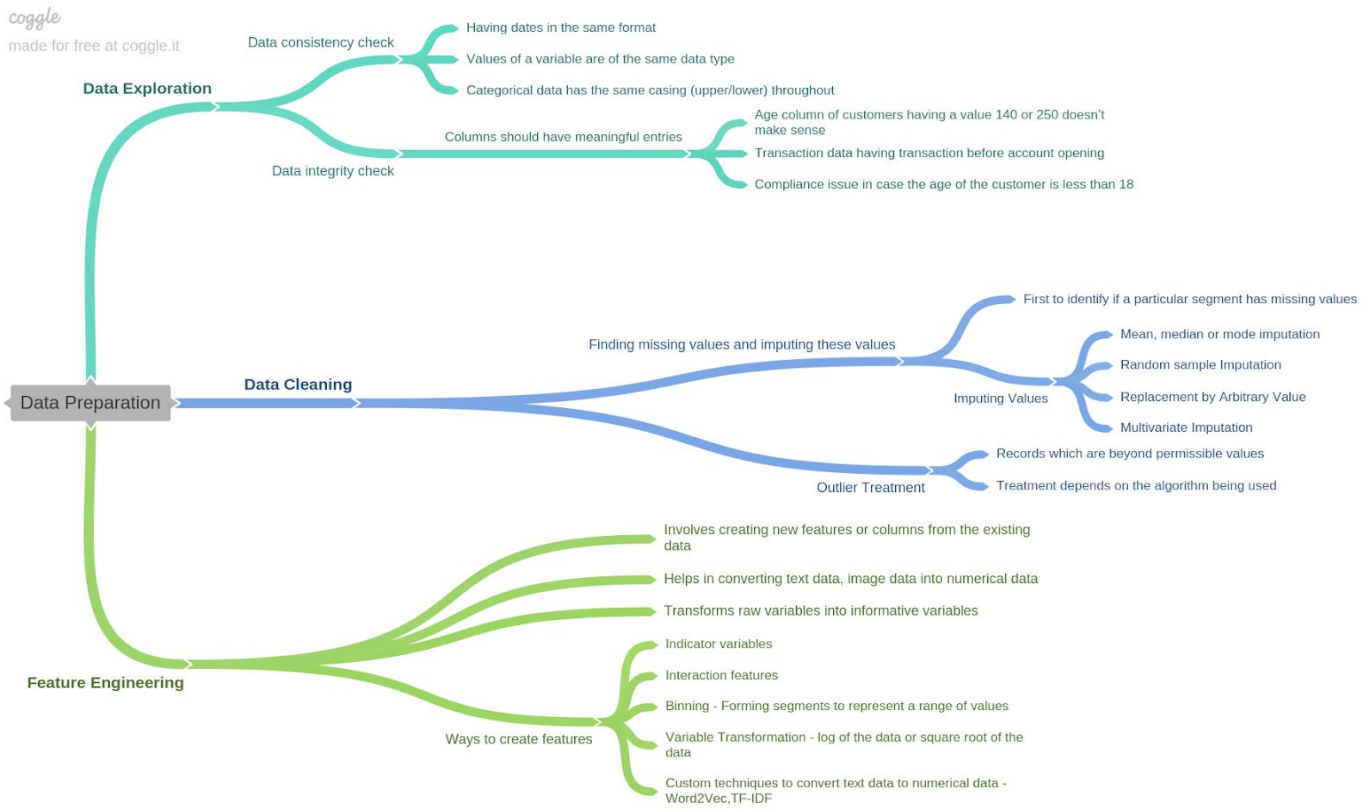
Once you have performed the necessary data cleaning, the next step would be to create new features that best represent the data. This process is known as feature engineering.

Feature engineering is the process of using domain knowledge to create features from the data. These features can increase the predictive power of machine learning algorithms. For example, consider retail store data, which has columns for customer name, product, amount and time of transaction. From this data set, you can create new features such as:

1. **Recency**: Measuring how recently a customer visited the store or made a purchase
2. **Frequency**: Measuring the frequency of the transactions done by customers
3. **Monetary**: Measuring the amount of time spent by the customers on their purchases

A few techniques to create new features include:

1. **Indicator variables**
2. **Interaction features**
3. **Binning**: A new feature representing a range of values
4. **Variable transformation**: The log of a variable
5. **Custom features**: To convert text data to numerical data:
    - Word2Vec
    - TF-IDF
    - Count vectoriser
    - n-grams



Model Development

Model development is a process that involves developing and validating models to arrive at the best solution for the problem at hand. The different steps involved here are:

1. Data Partitioning:

This involves the creation of multiple subsets of data, namely:

- **Training data set:** This subset of data is used to explore ways to create relevant features that can be useful for developing a model.
- **Test data set:** This subset of data is used to assess the overall model performance and compare the performance across all potential algorithms.
- **Validation data set:** The primary purpose of this data is parameter tuning. (For now, understand that parameter tuning is a step in the model building phase that helps improve the performance of the model).

Partitioning ensures that the model performs well on unseen data, as it is possible that the model predicts extremely well on training data but not on test data.

2. Model Selection:

Model selection is crucial to building good machine learning models. Here, you identify the best algorithm from a set of applicable algorithms. In the previous session, you learnt about different types of algorithms such as regression, classification and clustering algorithms. Classification algorithms include algorithms such as logistic regression, decision tree, random forest and many more. Model selection involves finding the best-performing algorithm for a particular data set or the problem at hand.

3. Model Evaluation:

Model evaluation estimates the error in the selected model, i.e., how well the selected model performs on unseen data. Selecting the appropriate metric is an important aspect of model selection. Depending on the problem at hand, there are a host of metrics that can be used.

Some metrics for regression include:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Mean absolute error (MAE)
- R squared (R^2)
- Adjusted R squared (R^2)

Some of the metrics for classification include:

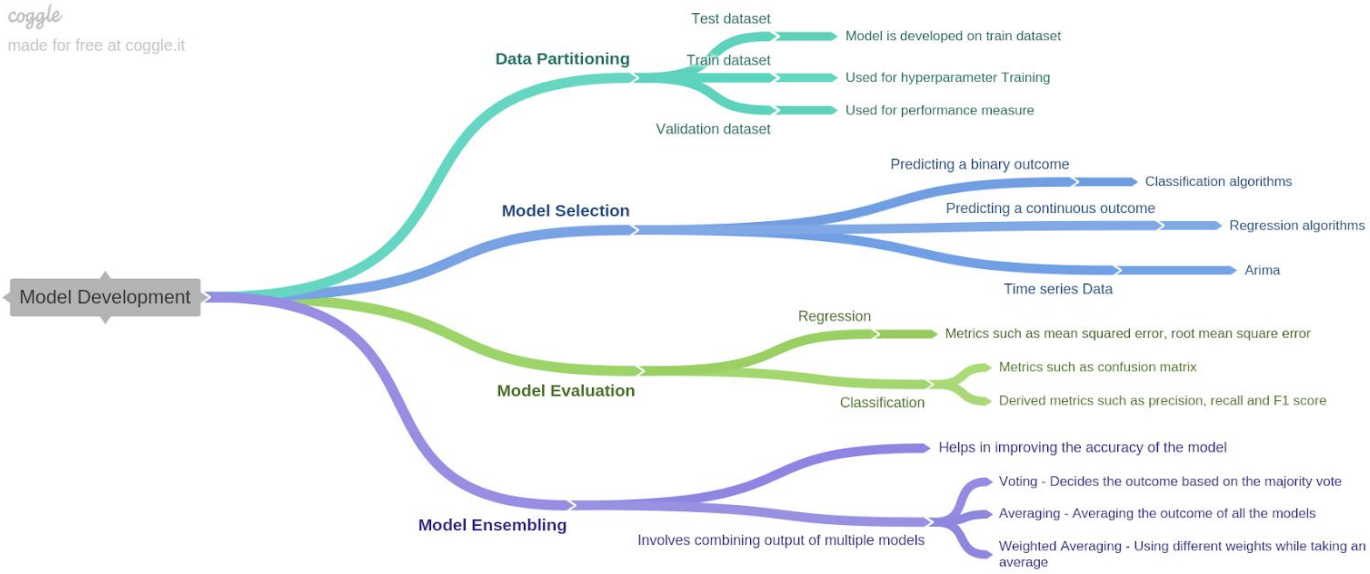
- Confusion or error matrix
- Accuracy
- Recall or Sensitivity or TPR (true positive rate)
- Precision
- Specificity or TNR (true negative rate)
- F1-Score
- Area under the receiver operating characteristic (ROC) curve (AUC)
- Logarithmic loss

4. Model Ensembling

Ensemble learning, which is when multiple models are combined, helps in reducing the error in the model. The reasoning behind using ensembles is that the sum of individual approaches is better than each approach implemented alone. In real-world scenarios, it could be very challenging to train models to generalise on a data set, as it could contain many underlying distributions. One model would work well on a particular aspect of the data, whereas another may work well on other aspects. Ensemble learning can help provide a solution that can work on all aspects of the data. Some ensemble techniques include:

- **Voting**: It involves deciding the prediction based on the class that gets a majority vote.

- **Averaging**: It involves using the average prediction from all the models to come up with the final prediction.
- **Weighted averaging**: It is similar to average, but it offers the flexibility to use different weights while combining the output of different algorithms.



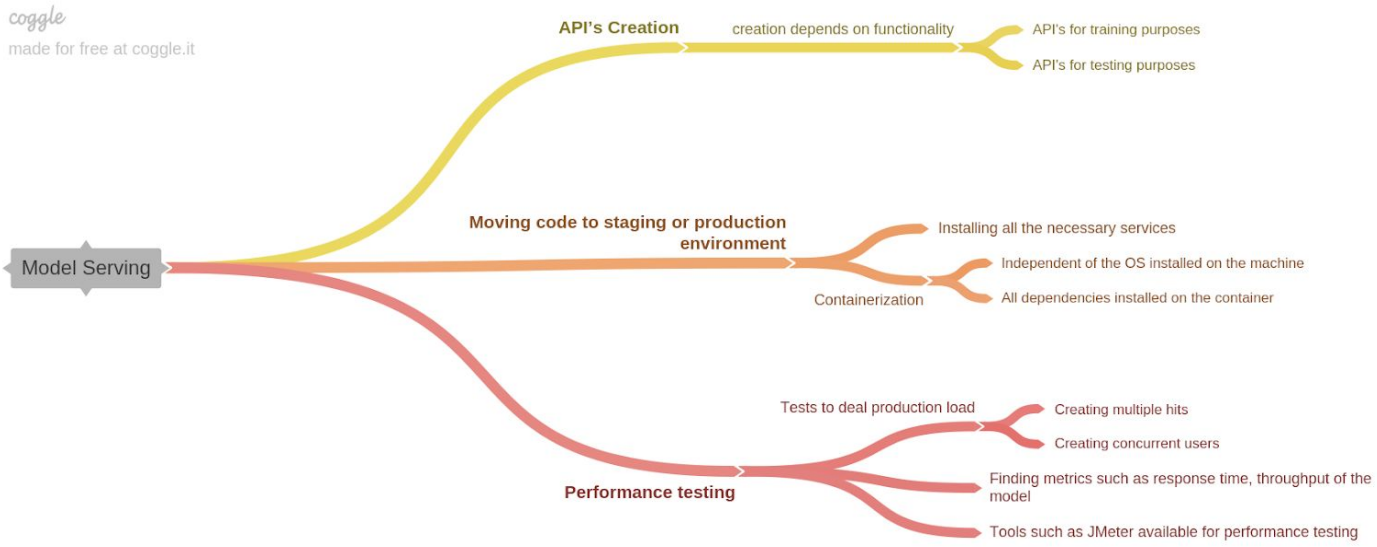
Model Serving

The model built in the previous step now needs to be served or taken to the end user.

Model serving involves the following steps:

1. Exposing the built models as **application programming interfaces (APIs)**. An API is a software construct that facilitates communication between two components by defining the format of input and output messages. If a software engineer wants to use the machine learning model, then they need to call the API in a predefined format:
    - For example, let's say that for a particular food delivery application, you have built a model to predict the ETA of the delivery agent. Now, you need to put this model back into the application so that the end user can benefit from it. The tech stack under which the website is built can be JavaScript, and the machine learning model can be constructed entirely in Python or PySpark. For the application to communicate with the model, you need to expose the model as an API.
2. Once this application of the machine learning model is built, the next step involves moving the code from the local/development environment to the staging/production environment. A key aspect here is that you need to create an environment similar to the development environment. This can be achieved using the technique of **containerisation**.

3. The last step is **performance testing**, where you simulate similar real-time scenarios. For example, a website with regular traffic of 100 users per minute may suddenly witness a surge to 1,000 users per minute during a sale. Performance testing involves simulating such unexpected loads and checking metrics such as response time and concurrency.



Model Monitoring

Once you have developed the model and deployed it successfully, the last step would be to track its performance continuously. This is done in the model monitoring phase.

Model monitoring plays an essential part in **making sure the model's performance doesn't deteriorate over time**. Also, it helps identify some of the bugs in the system faster. Production systems can face unexpected issues, and so it is crucial to detect anomalies as early as possible and take corrective action. It gives business users an overview of the metrics that are impacted by the system.

The steps involved in model monitoring are:

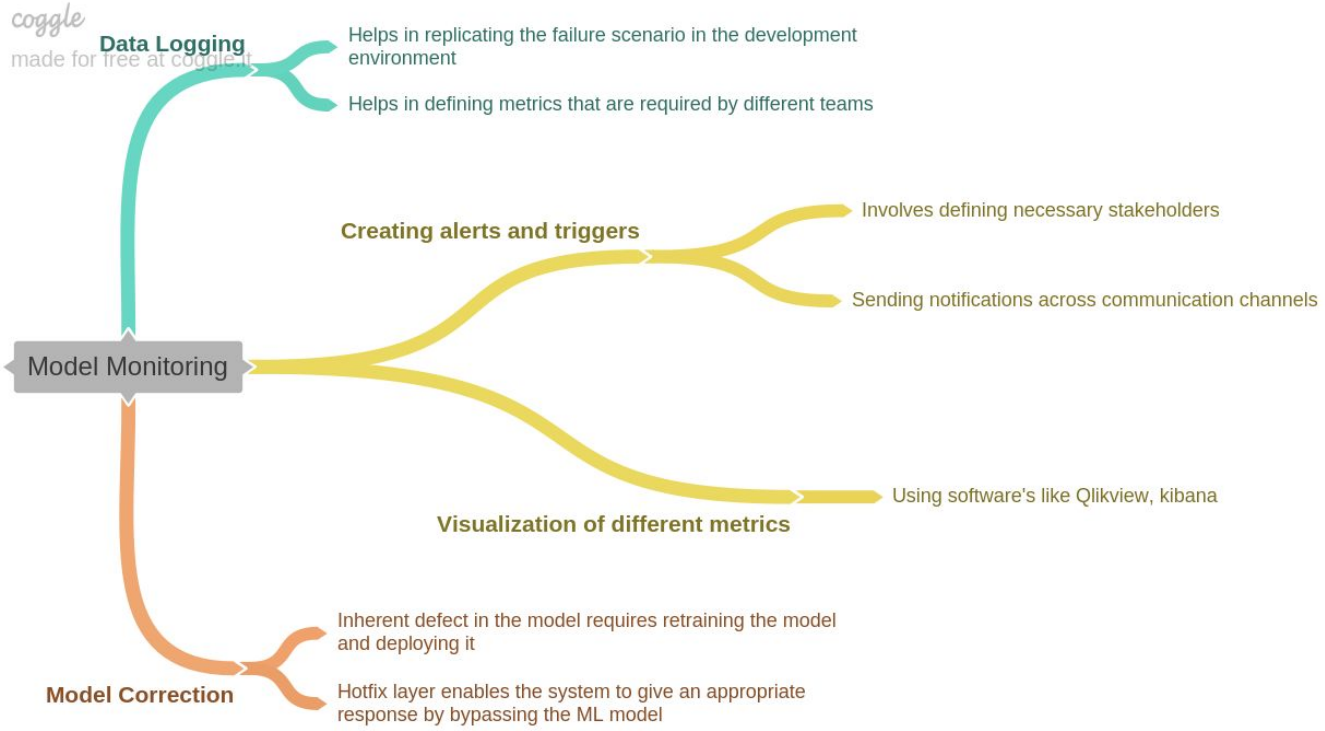
1. **Data logging**: All requests and responses are logged in corresponding data storage systems. This helps in debugging issues, as one can easily replicate the issues in the dev environment.

2. **Reporting**: Once you have defined the metrics, the next step is to provide a UI or dashboard for all stakeholders to access and obtain timely updates. Different visualisation tools such as Tableau, QlikView and PowerBI can be used to design a dashboard that can display all the metrics..

3. **Alerting**: An important step is to set up alerts to notify stakeholders about problems in the system. Alerts are a set of rules created using a combination of metrics that are critical and indicate problems with the system.

4. **Resolution**: Once the problem is detected, the next course of action would be to:

    - Have a 'hot-fix' layer, which enables the system to provide appropriate output by bypassing the machine learning model.
    - Re-train the model.



upGrad