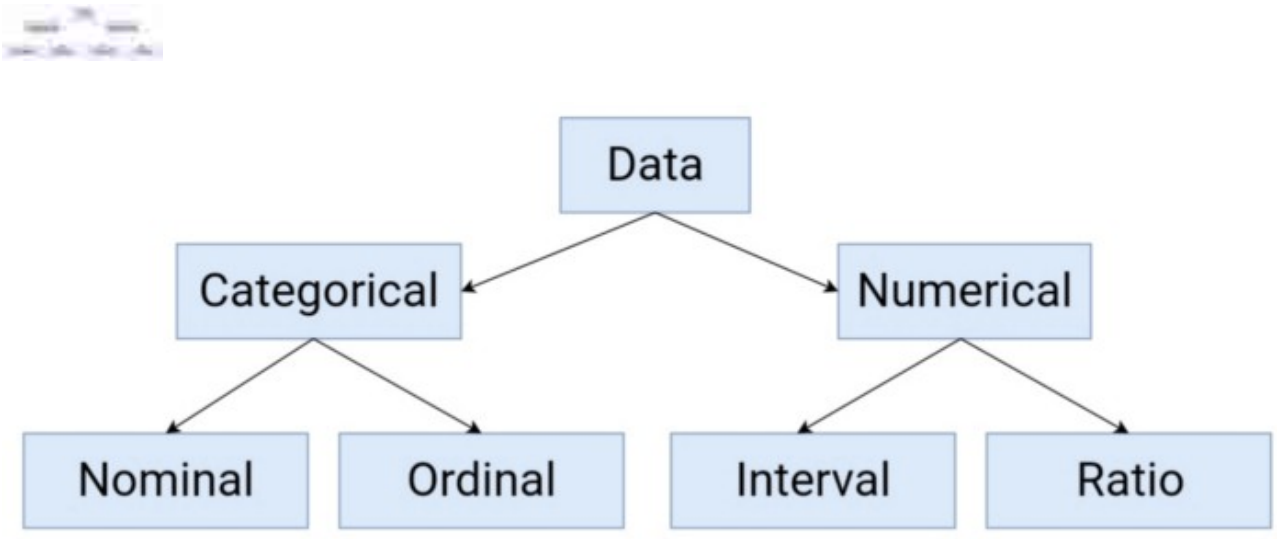
What is Data Preprocessing?
When we talk about data, we usually think of some large datasets with huge number of rows and columns. While that is a likely scenario, it is not always the case — data could be in so many different forms: Structured Tables, Images, Audio files, Videos etc..
Machines don't understand free text, image or video data as it is, they understand 1s and 0s. So it probably won't be good enough if we put on a slideshow of all our images and expect our machine learning model to get trained just by that!

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

Features in Machine Learning
A dataset can be viewed as a collection of *data objects*, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities.
Data objects are described by a number of *features*, that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc.. Features are often called as variables, characteristics, fields, attributes, or dimensions.
As per Wikipedia,
A feature is an individual measurable property or characteristic of a phenomenon being observed

For instance, color, mileage and power can be considered as features of a car. There are different types of features that we can come across when we deal with data.



Statistical Data Types

Features can be:

- **Categorical :** Features whose values are taken from a defined set of values. For instance, days in a week : {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} is a category because its value is always taken from this set. Another example could be the Boolean set : {True, False}
- **Numerical :** Features whose values are continuous or integer-valued. They are represented by numbers and possess most of the properties of numbers. For instance, number of steps you walk in a day, or the speed at which you are driving your car at.

Nominal	Ordinal
Categorical variables without any implied order	Categorical variables with a natural implied order but the scale of difference is not defined
Example : A new car model comes in these colors : Black, Blue, White, Silver	Example : Sizes of clothes has a natural order : Extra Small < Small < Medium < Large < Extra Large - But this does not mean Large - Medium = Medium - Small

Interval	Ratio
Numeric variabes with a defnied unit of measurement, so the differences between values are meaningful	Numeric variables with a defined unit of measurement but both differences and ratios are meaningful
Examples : Calender Dates, Temperature in Celsius or Farhenheit	Examples : Temperature in Kelvin, Monetary quantities, Counts, Age, Mass, Length, Electrical Current

Now that we have gone over the basics, let us begin with the steps of Data Preprocessing. Remember, not all the steps are applicable for each problem, it is highly dependent on the data we are working with, so maybe only a few steps might be required with your dataset. Generally they are :

- Data Quality Assessment
- Feature Aggregation
- Feature Sampling
- Dimensionality Reduction
- Feature Encoding

Data Quality Assessment

Because data is often taken from multiple sources which are normally not too reliable and that too in different formats, more than half our time is consumed in dealing with data quality issues when working on a machine learning problem. It is simply unrealistic to expect that the data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process. Let's go over a few of them and methods to deal with them :

1.**Missing values** :

It is very much usual to have missing values in your dataset. It may have happened during data collection, or maybe due to some data validation rule, but regardless missing values must be taken into consideration.

- Eliminate rows with missing data :
  Simple and sometimes effective strategy. Fails if many objects have missing values. If a feature has mostly missing values, then that feature itself can also be eliminated.
- 1
- Null=1.5
- Independent(x), Dependent(y)
- 2
- Estimate missing values :
  If only a reasonable percentage of values are missing, then we can also run simple interpolation methods to fill in those values. However, most common method of dealing with missing values is by filling them in with the mean, median or mode value of the respective feature.

2. **Inconsistent values** :

We know that data can contain inconsistent values. Most probably we have already faced this issue at some point. For instance, the 'Address' field contains the 'Phone number'. It may be due to human error or maybe the information was misread while being scanned from a handwritten form.

- It is therefore always advised to perform data assessment like knowing what the data type of the features should be and whether it is the same for all the data objects.

3. **Duplicate values** :

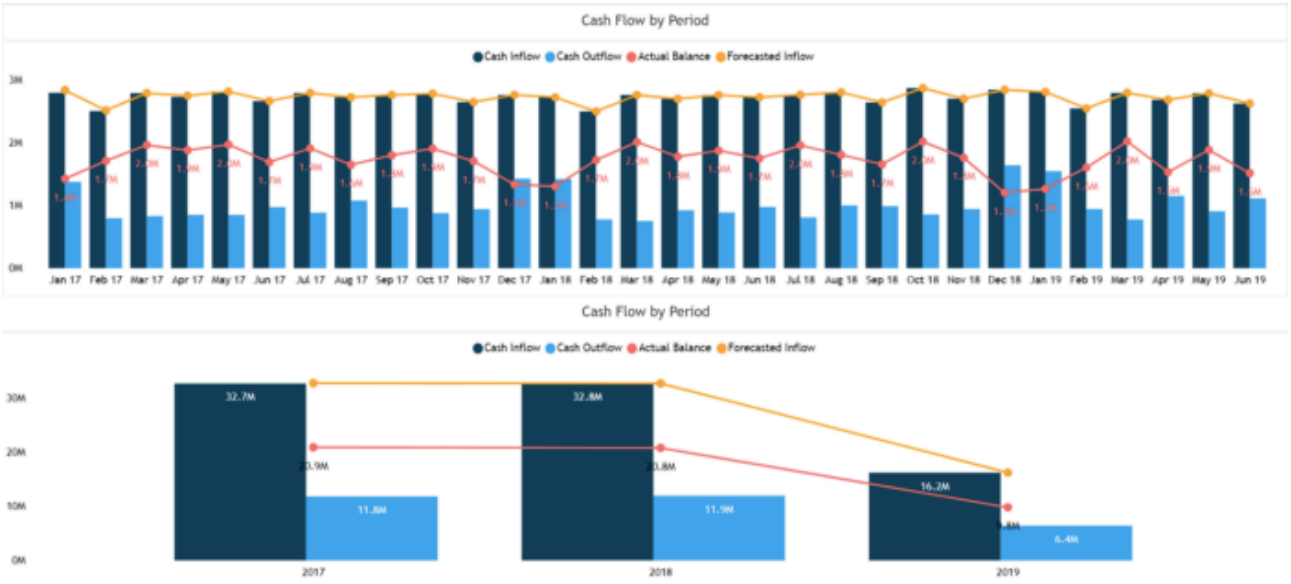
A dataset may include data objects which are duplicates of one another. It may happen when say the same person submits a form more than once. The term deduplication is often used to refer to the process of dealing with duplicates.

- In most cases, the duplicates are removed so as to not give that particular data object an advantage or *bias*, when running machine learning algorithms.

Feature Aggregation

Feature Aggregations are performed so as to take the aggregated values in order to put the data in a better perspective. Think of transactional data, suppose we have day-to-day transactions of a product from recording the daily sales of that product in various store locations over the year. Aggregating the transactions to single store-wide monthly or yearly transactions will help us reducing hundreds or potentially thousands of transactions that occur daily at a specific store, thereby reducing the number of data objects.

- This results in reduction of memory consumption and processing time
- Aggregations provide us with a high-level view of the data as the behaviour of groups or aggregates is more stable than individual data objects



Aggregation from Monthly to Yearly

Feature Sampling

Sampling is a very common method for selecting a subset of the dataset that we are analyzing. In most cases, working with the complete dataset can turn out to be too expensive considering the memory and time constraints. Using a sampling algorithm can help us reduce the size of the dataset to a point where we can use a better, but more *expensive*, machine learning algorithm.

The key principle here is that the sampling should be done in such a manner that the sample generated should have approximately the same properties as

the original dataset, meaning that the sample is *representative*. This involves choosing the correct sample size and sampling strategy.
Simple Random Sampling dictates that there is an equal probability of selecting any particular entity. It has two main variations as well :

- **Sampling without Replacement** : As each item is selected, it is removed from the set of all the objects that form the total dataset.
- **Sampling with Replacement** : Items are not removed from the total dataset after getting selected. This means they can get selected more than once.

Data Sampling (ISTOCK.COM/KKOLOSOV)

Although Simple Random Sampling provides two great sampling techniques, it can fail to output a representative sample when the dataset includes object types which vary drastically in ratio. This can cause problems when the sample needs to have a proper representation of all object types, for example, when we have an *imbalanced* dataset.

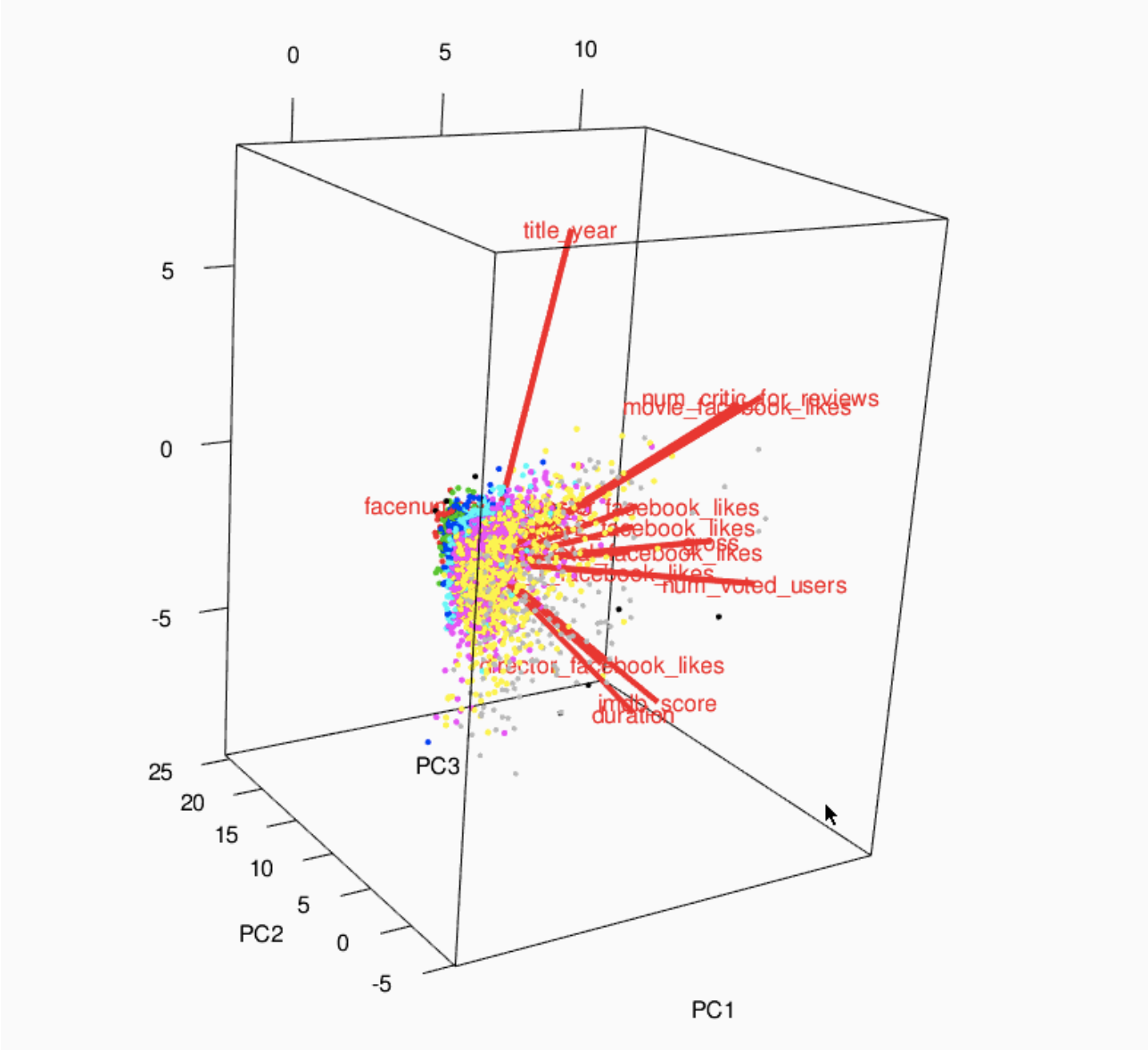
An Imbalanced dataset is one where the number of instances of a class(es) are significantly higher than another class(es), thus leading to an imbalance and creating rarer class(es).

It is critical that the rarer classes be adequately represented in the sample. In these cases, there is another sampling technique which we can use, called *Stratified Sampling*, which begins with predefined groups of objects. There are different versions of Stratified Sampling too, with the simplest version suggesting equal number of objects be drawn from all the groups even

though the groups are of different sizes. For more on sampling check out this article by
Team AV
.

A Data Scientist's Guide to 8 Types of Sampling Techniques
Overview Sampling is a popular statistical concept - learn how it works in this article We will also talk about eight…
www.analyticsvidhya.com

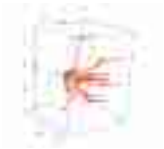
Dimensionality Reduction
Most real world datasets have a large number of features. For example, consider an image processing problem, we might have to deal with thousands of features, also called as *dimensions*. As the name suggests, dimensionality reduction aims to reduce the number of features - but not simply by selecting a sample of features from the *feature-set*, which is something else — Feature Subset Selection or simply Feature Selection.



Conceptually, *dimension* refers to the number of geometric planes the dataset lies in, which could be high so much so that it cannot be visualized with pen and paper. More the number of such planes, more is the complexity of the dataset.

The Curse of Dimensionality

This refers to the phenomena that generally data analysis tasks become significantly harder as the dimensionality of the data increases. As the dimensionality increases, the number planes occupied by the data increases thus adding more and more sparsity to the data which is difficult to model and visualize.



Representation of components in different spaces

What dimension reduction essentially does is that it maps the dataset to a lower-dimensional space, which may very well be to a number of planes which can now be visualized, say 2D. The basic objective of techniques which are used for this purpose is to reduce the dimensionality of a dataset by creating new features which are a combination of the old features. In other words, the higher-dimensional feature-space is mapped to a lower-dimensional feature-space. Principal Component Analysis and Singular Value Decomposition are two widely accepted techniques.

A few major benefits of dimensionality reduction are :

- Data Analysis algorithms work better if the dimensionality of the dataset is lower. This is mainly because irrelevant features and noise have now been eliminated.
- The models which are built on top of lower dimensional data are more understandable and explainable.
- The data may now also get easier to visualize!
  Features can always be taken in pairs or triplets for visualization purposes, which makes more sense if the featureset is not that big.

Feature Encoding

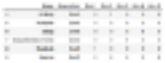
As mentioned before, the whole purpose of data preprocessing is to *encode* the data in order to bring it to such a state that the machine now understands it.

Feature encoding is basically performing transformations on the data such that it can be easily accepted as input for machine learning algorithms while still retaining its original meaning.

Rain-(l,m,h)=>low,moderate an dhigh(0,1)

There are some general norms or rules which are followed when performing feature encoding. For Continuous variables :

- **Nominal** : Any one-to-one mapping can be done which retains the meaning. For instance, a permutation of values like in One-Hot Encoding.
- **Ordinal** : An order-preserving change of values. The notion of small, medium and large can be represented equally well with the help of a new function, that is, <new_value = f(old_value)> - For example, {0, 1, 2} or maybe {1, 2, 3}.



	Name	Generation	Gen 1	Gen 2	Gen 3	Gen 4	Gen 5
4	Octillery	Gen 2	0	1	0	0	0
5	Helioptile	Gen 6	0	0	0	0	0
6	Dialga	Gen 4	0	0	0	1	0
7	DeoxysDefense Forme	Gen 3	0	0	1	0	0
8	Rapidash	Gen 1	1	0	0	0	0
9	Swanna	Gen 5	0	0	0	0	1

One-hot encoding of the data

For Numeric variables:

- **Interval** : Simple mathematical transformation like using the equation <new_value = a*old_value + b>, a and b being constants. For example, Fahrenheit and Celsius scales, which differ in their Zero values size of a unit, can be encoded in this manner.
- **Ratio** : These variables can be scaled to any particular measures, of course while still maintaining the meaning and ratio of their values. Simple mathematical transformations work in this case as well, like the transformation <new_value = a*old_value>. For, length can be measured in meters or feet, money can be taken in different currencies.

Train / Validation / Test Split

After feature encoding is done, our dataset is ready for the exciting machine learning algorithms!

But before we start deciding the algorithm which should be used, it is always advised to split the dataset into 2 or sometimes 3 parts. Machine Learning algorithms, or any algorithm for that matter, has to be first trained on the data distribution available and then validated and tested, before it can be deployed to deal with real-world data.

Training data : This is the part on which your machine learning algorithms are actually trained to build a model. The model tries to *learn* the dataset

and its various characteristics and intricacies, which also raises the issue of Overfitting v/s Underfitting.

Validation data : This is the part of the dataset which is used to validate our various model fits. In simpler words, we use validation data to choose and improve our model hyperparameters. The model does not *learn* the validation set but uses it to get to a better state of hyperparameters.

Test data : This part of the dataset is used to test our model hypothesis. It is left untouched and unseen until the model and hyperparameters are decided, and only after that the model is applied on the test data to get an accurate measure of how it would perform when deployed on real-world data.

Data Split into parts

Split Ratio : Data is split as per a *split ratio* which is highly dependent on the type of model we are building and the dataset itself. If our dataset and model are such that a lot of training is required, then we use a larger chunk of the data just for training purposes (usually the case) — For instance, training on textual data, image data, or video data usually involves thousands of features!

If the model has a lot of hyperparameters that can be tuned, then keeping a higher percentage of data for the validation set is advisable. Models with less number of hyperparameters are easy to tune and update, and so we can keep a smaller validation set.

Like many other things in Machine Learning, the split ratio is highly dependent on the problem we are trying to solve and must be decided after taking into account all the various details about the model and the dataset in hand.