# Feature and Label

## Feature

Any one of the input column (independent variable) used in training ML algorithm or model

## Label

The label is the target column (dependant variable) which the trained model has to predict, for a new input

## Example analogy to remember functionality of feature and label

A television has many **features**, like remote support,wifi, android, based on these features we will decide whether to buy the Television or not, here buying is the **label**

So above **features** are acting as **features** and decision of **buying** (which we want to predict) is acting as **label**

# ML model drift

Important from (model) deployment and usability point (inferencing)

Model drift: a change in behavior of ML model

three types of model drift: 1: concept drift (drift that causes due to change in target variable or dependent variable) 2: data drift (drift caused due to change in independent variables or features) 3: upstream data changes (operational changes in data pipeline)

the first 2 are majorly discussed

## Concept drift

Statistical change in target variable, the definition (way of) predicting a target variable has changed

### Ex:

a ml model which predicts the distance of a car that is infront of our car while driving, the model does this by looking at the 2 tail lights the car has. If the new cars start having one wide horizontal light instead of 2 tail lights, that means the target has changed by definition

This comes as model drift under concept drift category

## Data drift

Any drift that is caused by changing independent variables (or features) is called data drift

### Ex

If a ml model takes opinions of boys in to consideration and gets build upon it and after the model is generated, and tries to predict the opinion of a girl

## Upstream data changes

Changes that occur due to transformations in data pipeline

When a feature that is being generated before is no longer generated

### Ex:

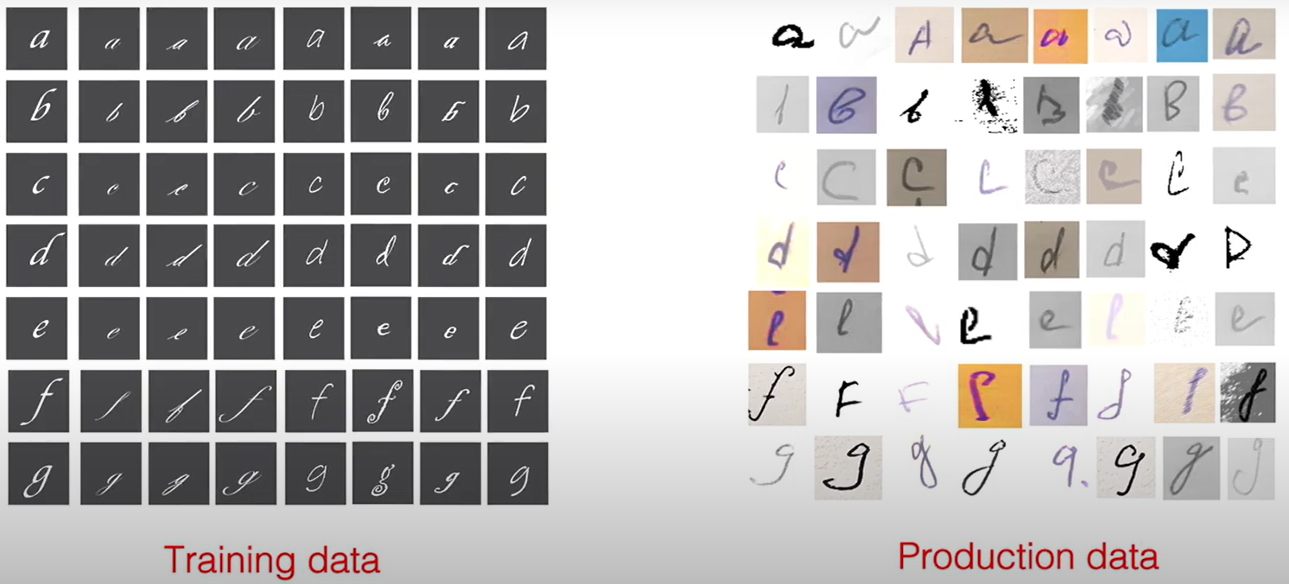
In an customer grievance detection model, if a particular option is removed, then model can no longer generate derived features from that existing feature

# Training serving skew

When the model is trained on one type data and is tested or predicted on other type of data

## Ex

When a model is trained on generated data and used to predict real world data, then comes the training serving skew



# Cardinality

The value of a categorical variable is selected from a group of categories (also called as labels).

Ex: a person’s married status can be unmarried, married or divorced, here the group of categories are married, unmarried or divorced

The **cardinality** is the count of different variables.

For the above example- the cardinality is 3

## Problems due to cardinality

* Uneven distribution of categorical data between train and test sets
  + when a categorical column has high cardinality that means it has more options, chances are there that some of the categorical variable values are not even present in train data
  + ex: if the dataset is for population of india, there are good chance that a minority woman category might not be present in the train dataset
  + to solve this problem, there is something called stratified split afaik
* overfitting in tree-based algorithms (decision tree, adaboost)
  + categorical columns of high cardinality dominate categorical columns with low cardinality
    - ex: pincode categorical column has dominance over gender categorical column
* problem in getting accurate data as, data per label gets reduced (if you have high cardinality)
* operational problems

# fit, transform, fit\_transform

## what is meant by transform:

transform simply means that you are applying a set of rules on a data to change it in our desired form

ex: standardscale, imputer, PCA, MinMaxScaler

## fit

when you perform fit on a data (you get the value that is a result of the operation performed on the data, ex: when you perform fit (using mean method) on values 1,2,3 then the mean value is stored as 2 in the fit object)

## transform

when you perform transform, you need a fit object ( which has already performed fit operation on data) you call transform on the data using the fit object.

## fit\_transform

fit\_transform is the combination of fit and transform discussed above

## how to decide, when to use fit,transform fit\_transform

### use fit,transform when

you want to use fit that is calculated on one data to transform another data, then you need to perform fit and transform separately

#### ex scenario:

1: if you want to perform fit on train data and use the same fit object to transform train and test data

2: if you want to fill the null values of second row of a matrix using the mean of the first column of a matrix

### Use fit\_transform when

you want to use fit and transform on the same data and you don’t have the need to use fit on other data later on

# outliers

as the name suggests it lies out from the group

a data point or an observation which stands out or does not abide by the behavior of majority percentage by a considerable gap

outliers are dangerous when you don’t have a system in place to explain why those data points are outliers, below are 2 scenarios which will help you to decide whether to remove outliers or not

scenario 1: in a data where we have students marks and also hours they spent studying to fetch the marks. In this scenario there might be students who study for a very few hours and get very good marks, we can add another column called IQ which will let those students studying less hours still getting high marks as they have more IQ compared to students (who are possibly with low IQ) studying more hours. In this case we don’t have need to remove outliers

scenario 2: you are collecting people data to decide their percapita income, a person gave age as 300 mistakenly with income of 3CR. If there is no way to fill in the approximate age in such scenario we have to remove the outliers

## algorithms on which outliers effect exist

one of above 2 scenarios has to be implemented if you are using an algorithm which uses weight based approach

few weight based approaches listed below are

* ada boost
* linear regression
* logistic regression
* deep learning

## how to treat outliers

1. trimming
2. capping
3. treating as missing values or discretization

## how to detect outliers

* normal distribution (reverse bell shaped curve)
* skewed distribution (box plot)
* other distributions (ignoring observations (data points) beyond a certain percentile region)

## techniques for outlier detection and removal

* z score treatment (normal distribution)
* IQR based Filtering (skewed distrbution)
* Percentile based filtering
* Winsorization

# Continuous vs discrete variables

* Types of variables:
* Continuous variable: if a variable can take any value between its maximum and minimum range
* Discrete variable: if a variable is countable in finite amount of time. (you can measure it)

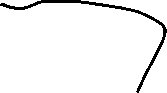
If there is a race, number of people in race is discrete, the time taken by an individual to complete a race is continuous variable

* Regression will deal with **continuous data** or **real valued output** (because it has to predict the output value which is not classified in to set of categories like **classification**), classification will deal with discrete data

Example:



* Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.



* **Classification** will deal with discrete output.
* We could turn the above example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

# Hyper parameters

Before we get in to hyper parameters, we need to understand model parameters, in machine learning model parameters are **intrinsic** to the trained model, i.e., after a model is trained with train data, it will form something like an quadratic equation (y=ax2+bx+c) where you will put in some (test) data and some variables that are generated by training the model with train data and form the parameters(a,b) so that when you pass a new x it can include (y=ax2+bx+c) and find the target variable (y) , a similar type of approach is implemented in regression

Model Parameter: Instrinsic configuration to the model and whose value is defined by the data. For example (1.) in case of linear regression modeling, slope(m) and constant(c) are parameter of the model. (2.) In case of ANN, weight(w) is parameter of ANN-Model.

But hyper parameters are extrinsic to the trained model or normal model, i.e., hyper parameters does not depend on model and infact they are heuristic (self discovering) and they are the main reasons of letting discover the parameters mentioned in the para above. Hyperparameters are set by a practitioner (ex: data scientist)

Simply put:

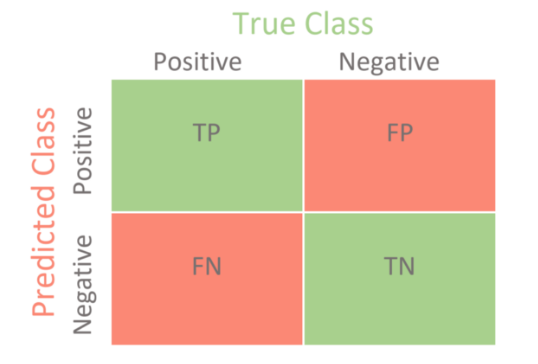
a parameter that the model assigns for itself is model parameter (based on the data it feeds upon), a parameter that is set extrinsically (by a practitioner ex: data scientist) to derive model parameters is hyper parameter

or a parameter which is manually set is a hyperparameter

# Model Quality

Model quality is a metric which sees how well a machine learning model generalizes (generates predictions) to unseen data

# Confusion matrix deductions



* Below are the terms used in confusion matrix related calculations:
* W.r.t True Class (actual values of predictions)
  + Positive labels = TP+FN
  + Negative labels = TN+FP
* W.r.t Predicted Class i.e., (predictions by model)
  + Positive predictions = TP+FP
  + Negative predictions = FN + TN
* Other terms
  + Accuracy = (TP + TN) / (TP + TN + FP + FN)
  + Precision = TP / (TP+FP)
  + Recall = TP / (TP+FN)
    - measures how often the model correctly predicts the cases that should receive a positive outcome
    - Recall is nothing but the true positive rate (see true class -> positives (verticaly)), i.e., the less the False negatives, the more the recall rate.
  + Specificity = TN / (TN+FP)
    - measures how often the model correctly predicts a negative outcome
    - Similar to the process of recall, specificity is to calculate True Negatives, (see true class -> negatives -> vertically)
  + Acceptance = TP/(TP+FP)
  + Rejection = TN / (TN+FN)

## Specificity and sensitivity:

True positives: the count of predictions which are predicted by the ML as positive and are actually positive

False negative: the count of predictions which are predicted by the model as negative and are actually negative

True positives and True negatives can be achieved by plotting confusion matrix

### Specificity

A metric used to evaluate the models ability to predict TRUE NEGATIVES of **each available category**

### Sensitivity

It is used as a metric to evaluate the models ability to predict TRUE POSITIVES of **each available category**

# Sagemaker

## Sagemaker clarify bias monitor

A monitoring service which will **monitor bias drift in model prediction** on a regular basis

## Sagemaker clarify EXPLAINABILITY monitor

Features explain the model to make predictions

Sagemaker clarify explainability monitor **monitors predictions for attribution drift** on a regular basis

# Ground truth

The fundamental teaching given to a machine learning model about truth and false with detailed explanation of **system** (theory and rules to let model understand our mission or purpose) behind it.

In a set of images of cats dogs and cheetas

1. If the system is set to find cats in the above set of images under the principle of deciding an animal as cat based on its biological origin then cheetas should be tagged as cats
2. If the system is set to find cats in above set of images under the principle of deciding an animal as cat based on its friendly behavior and pet nature then cheetas should not be tagged as cats

**Based on the system, ground truth should differ, so that ML model can learn accurately and also can predict better results**

# Types of data used to build a ML model

There are three types of data (afaik):

1. Train data: data which is used to teach the model about data and helps it in understanding how to make predictions after it is deployed
2. Validation data: predominantly used to evaluate the model while tuning hyperparameters and data preparation
3. Test data: used when you want to evaluate the final tuned model for comparing it with other versions of models