

Naïve Bayes' Classifier

Parametric, Multi-Class Classification Technique

About

Naive Bayes performs well when we have multiple classes (several categories to classify into) and working with text classification.

It is simple and if the conditional independence assumption actually holds (all outcomes are mutually exclusive), a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data.

It requires less model training time. Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

Suitable for doing POCs and making real time predictions.

The NB models cannot represent complex behavior so it won't get into over-fitting.

Things to Understand

Naive Bayes works best when you have small training data set, relatively small number of features (dimensions/ independent variables). If you have huge feature list, the model may not give you accuracy, because the likelihood would be distributed and may not follow the Gaussian or other distribution.

Another condition for Naive Bayes to work is that features should be independent of each other. Try to analyze how each features are related to each other, if they are they affecting the occurrence of each other. If they are mutually exclusive, Naive Bayes can give you good result.

Use of Probability

- **Class Probabilities:** The probabilities of each class (every outcome in the dependent variable) in the training dataset.
- **Conditional Probabilities:** The conditional probabilities of each input value given each class value

Bayes' Theorem

Bayes' theorem is a way to figure out conditional probability. Conditional probability is the probability of an event happening, given that it has some relationship to one or more other events. For example, your probability of getting a parking space is connected to the time of day you park, where you park, and what conventions are going on at any time. Bayes' theorem is slightly more nuanced. In a nutshell, it gives you the actual probability of an **event** given information about **tests**.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

A diagram illustrating Bayes' Theorem. The equation is centered:
$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$
 Four blue arrows point from text labels to parts of the equation: 'Likelihood' points to $P(x | c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c | x)$, and 'Predictor Prior Probability' points to $P(x)$.

Likelihood

Class Prior Probability

$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$

Posterior Probability

Predictor Prior Probability

Likelihood

$$P(X_1 \dots X_n | Y) = \prod_{i=1}^n P(X_i | Y)$$

Here, X represents the attributes or features, and Y is the response variable. Now, P(X|Y) becomes equal to the products of, probability distribution of each attribute X given Y.

Bayes' Theorem Example #1

You might be interested in finding out a patient's probability of having liver disease if they are an alcoholic. "Being an alcoholic" is the **test** (kind of like a litmus test) for liver disease.

- **A** could mean the event "Patient has liver disease." Past data tells you that 10% of patients entering your clinic have liver disease. $P(A) = 0.10$.
- **B** could mean the litmus test that "Patient is an alcoholic." Five percent of the clinic's patients are alcoholics. $P(B) = 0.05$.
- You might also know that among those patients diagnosed with liver disease, 7% are alcoholics. This is your **B|A**: the probability that a patient is alcoholic, given that they have liver disease, is 7%.

Bayes' theorem tells you:

$$P(A|B) = (0.07 * 0.1)/0.05 = 0.14$$

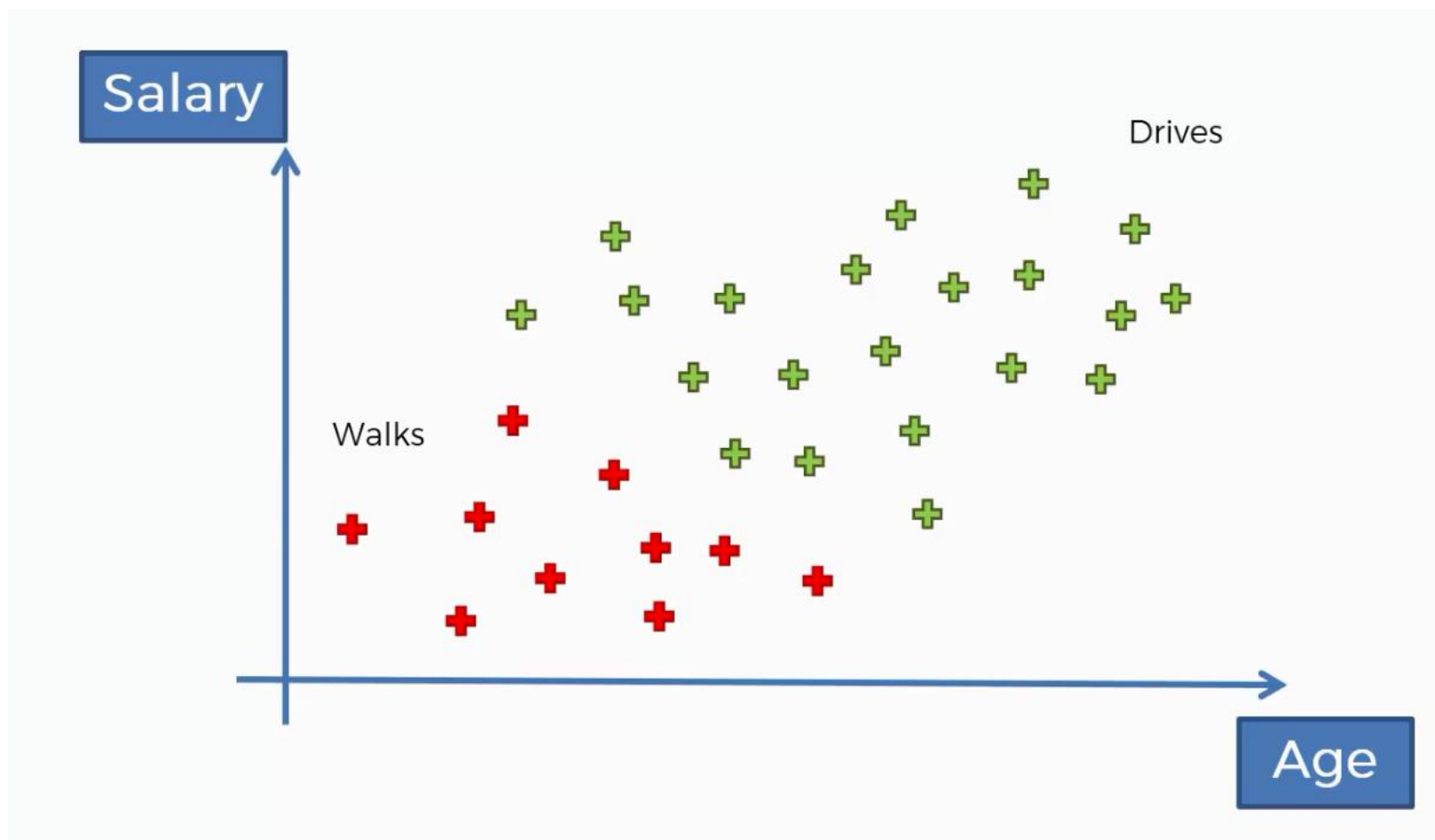
In other words, if the patient is an alcoholic, their chances of having liver disease is 0.14 (14%). This is a large increase from the 10% suggested by past data. But it's still unlikely that any particular patient has liver disease.

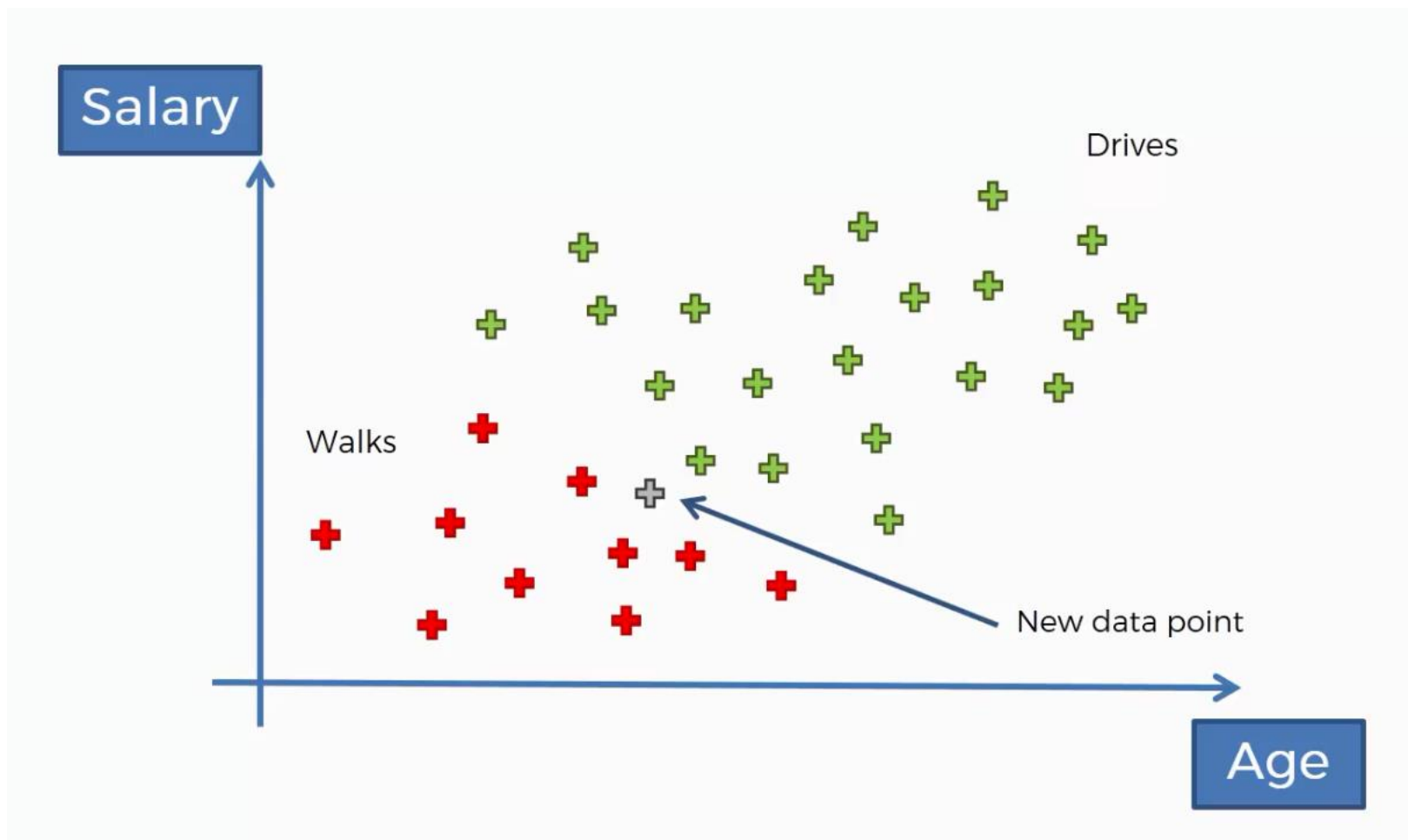
Example Problem

Given a dataset of people and their mode of commute, we will try to figure out the mode of commute of another person.

Every person in the dataset has attributes – Age and Salary.

Our goal is to analyse these attributes and determine the mode of commute for someone outside this group of people.





Steps to solving this example problem:

- Find the probability of the person Walking
- Find the probability of the person Driving
- Compare the probabilities. The higher one wins.

Step 1

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

#4

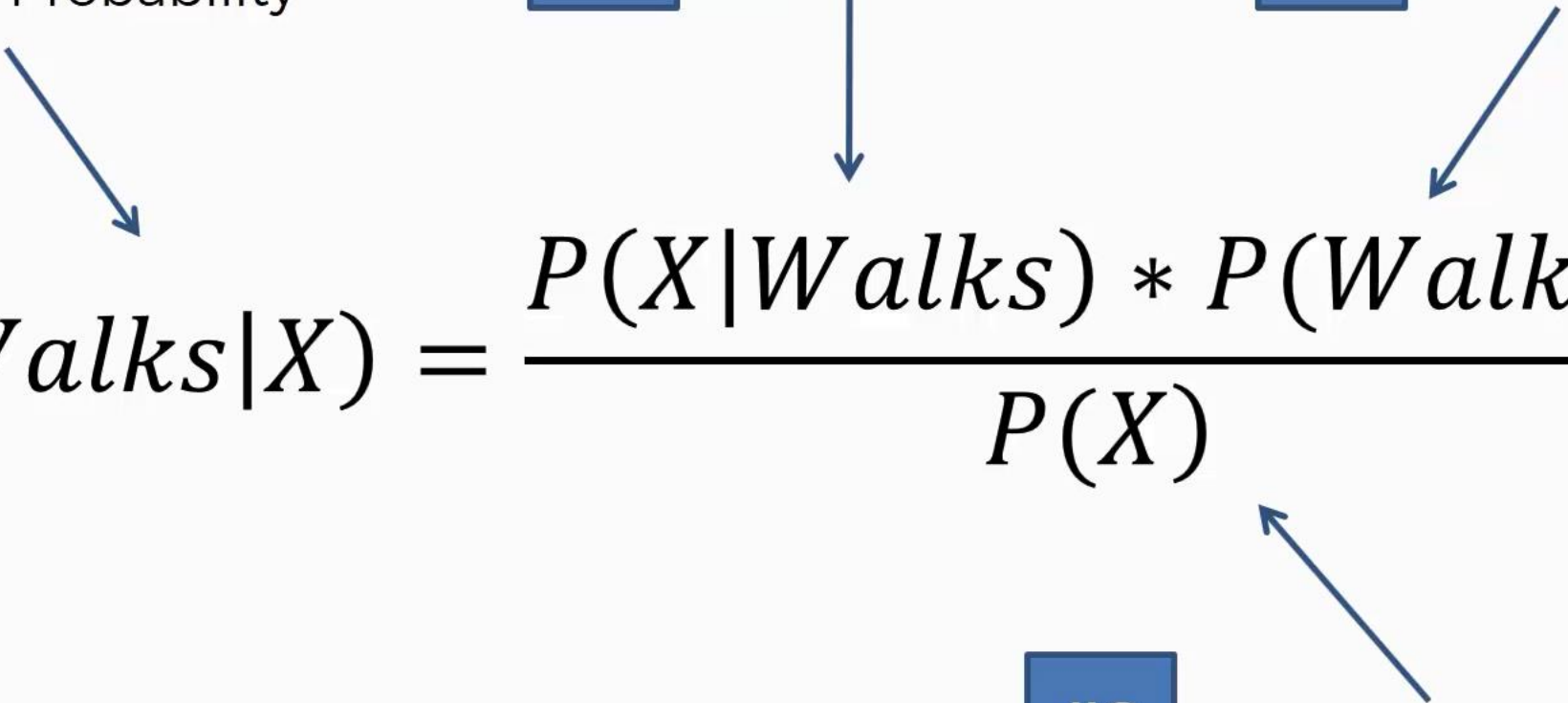
Posterior Probability

#3

Likelihood

#1

Prior Probability


$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

#2

Marginal Likelihood

Step 2

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

#4

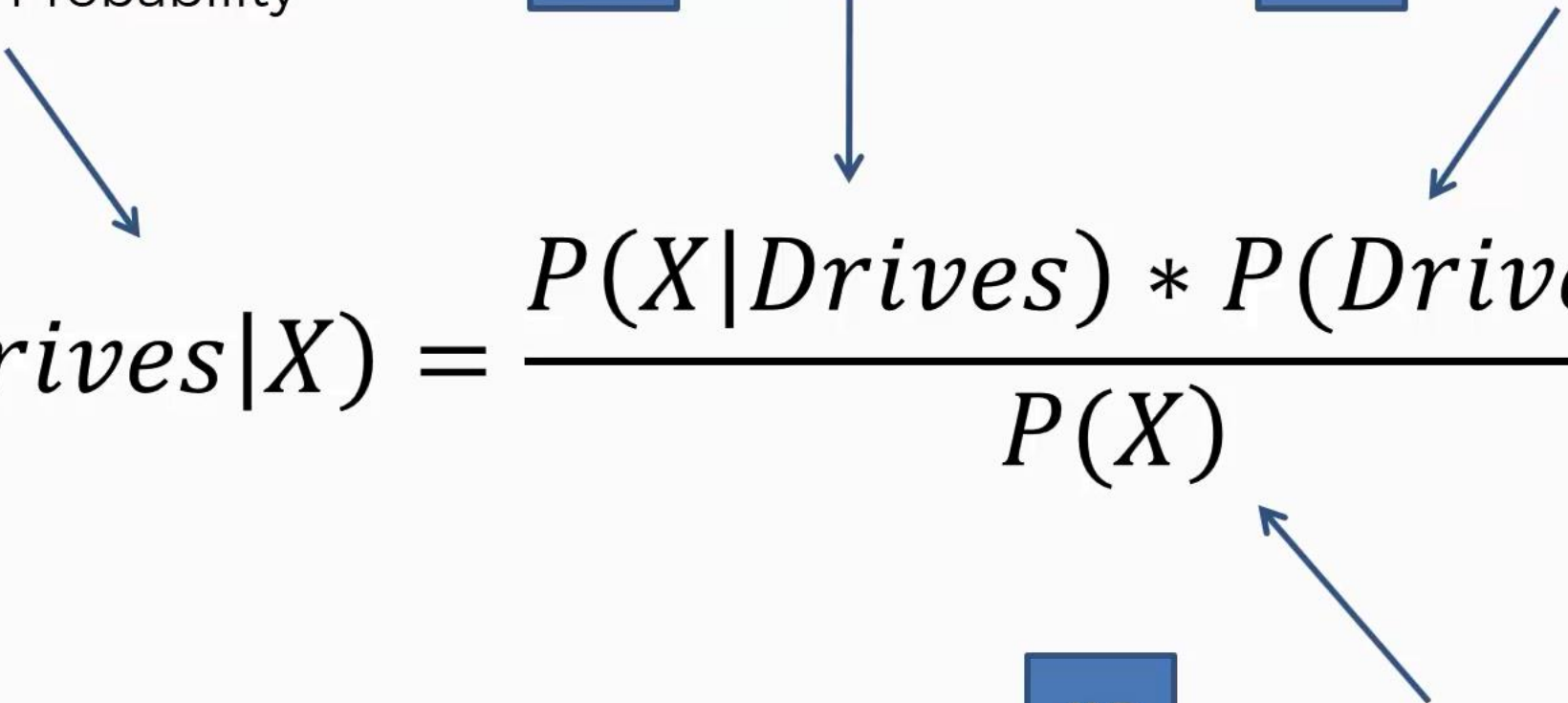
Posterior Probability

#3

Likelihood

#1

Prior Probability


$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

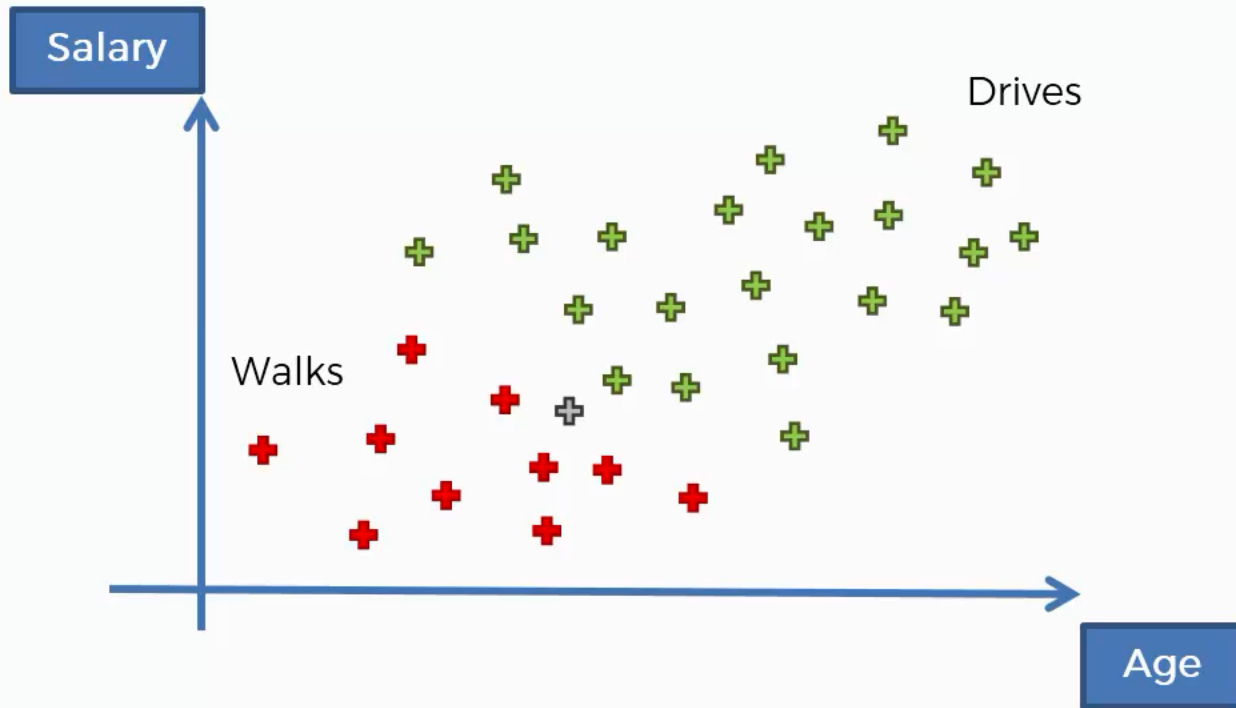
#2

Marginal Likelihood

Step 3

$P(Walks|X)$ v. s. $P(Drives|X)$

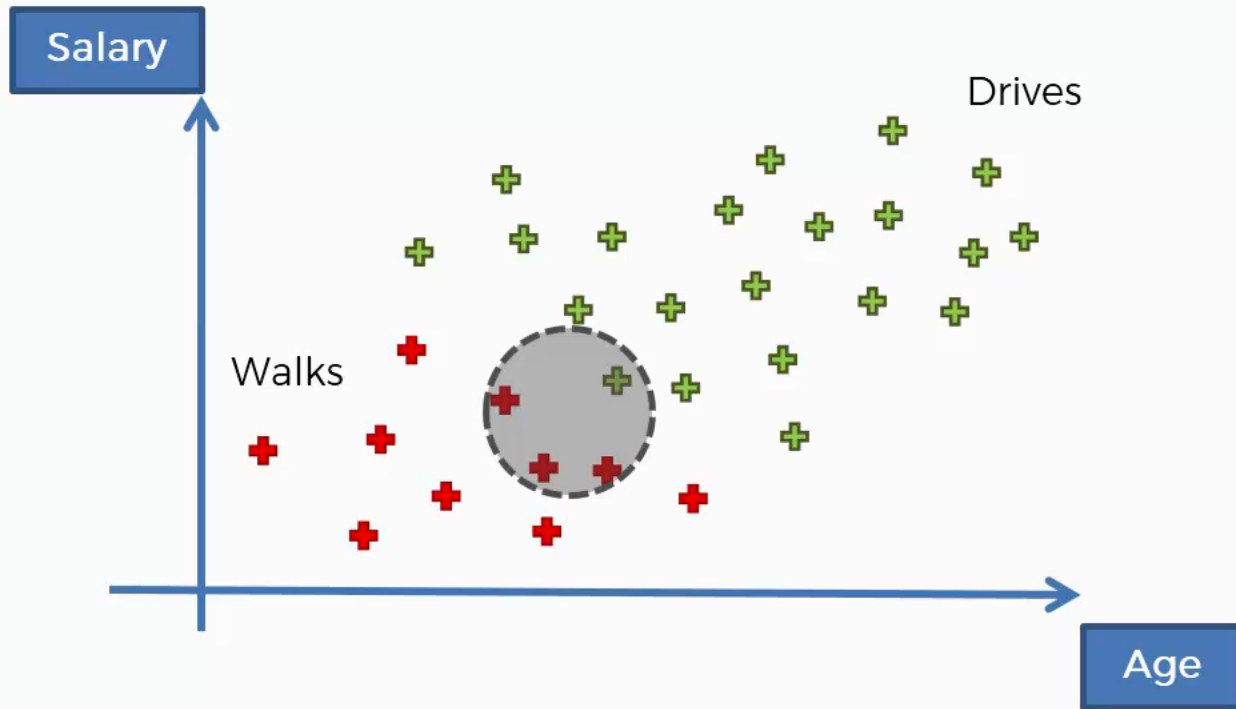
Lets find the Estimation using a Naïve Bayes Classifier



#1. $P(\text{Walks})$

$$P(\text{Walks}) = \frac{\text{Number of Walkers}}{\text{Total Observations}}$$

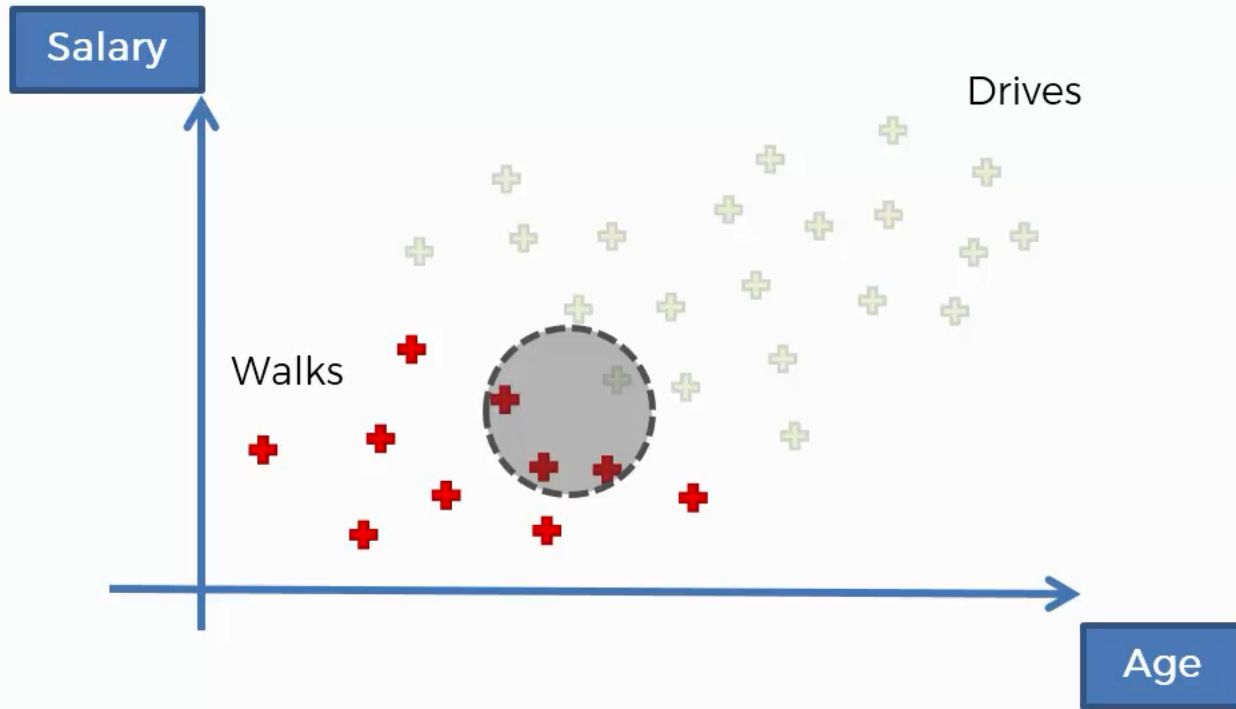
$$P(\text{Walks}) = \frac{10}{30}$$



#2. $P(X)$

$$P(X) = \frac{\text{Number of Similar Observations}}{\text{Total Observations}}$$

$$P(X) = \frac{4}{30}$$



#3. $P(X|Walks)$

*Number of Similar
Observations*

$$P(X|Walks) = \frac{\text{Among those who Walk}}{\text{Total number of Walkers}}$$

$$P(X|Walks) = \frac{3}{10}$$

#4

Posterior Probability

✓ #3

Likelihood

✓ #1

Prior Probability

$$P(Walks|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$$

✓ #2

Marginal Likelihood

#4

Posterior Probability

#3

Likelihood

#1

Prior Probability

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

#2

Marginal Likelihood

#4

Posterior Probability

✓ #3

Likelihood

✓ #1

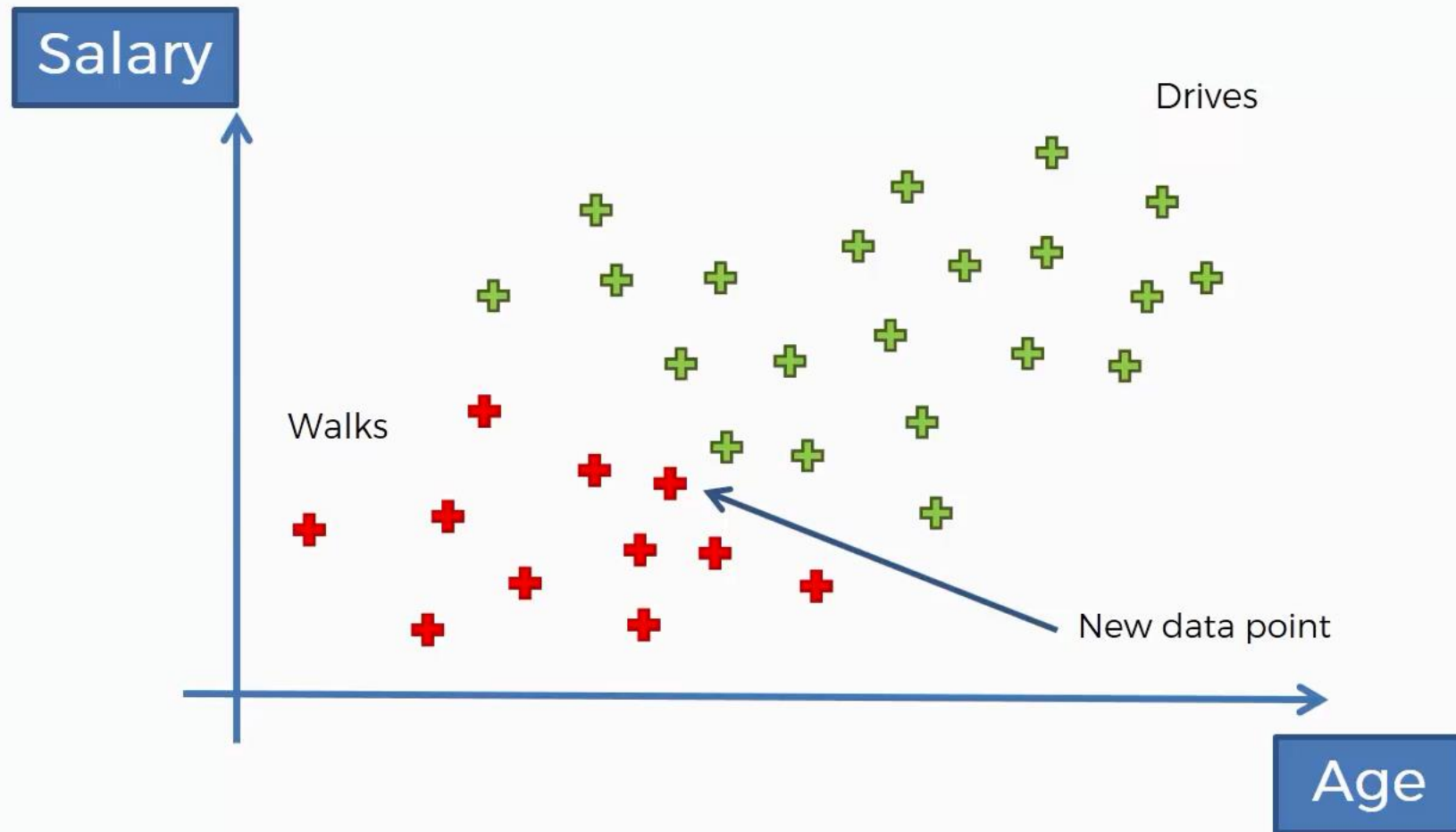
Prior Probability

$$P(Drives|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

✓ #2

Marginal Likelihood

$$P(Walks|X) > P(Drives|X)$$



Gaussian Naive Bayes

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution.

This extension of Naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

Gaussian Inputs: If the input variables are real-valued, a Gaussian distribution is assumed. In which case the algorithm will perform better if the univariate distributions of your data are Gaussian or near-Gaussian. This may require removing outliers (e.g. values that are more than 3 or 4 standard deviations from the mean).

Naïve Bayes Models in Python's sklearn

There are three types of Naive Bayes model under scikit learn library:

- [Gaussian](#): It is used in classification and it assumes that features follow a normal distribution.
- [Multinomial](#): It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".
- [Bernoulli](#): The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Improving the Naive Bayes Model

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like `alpha=1` for smoothing, `fit_prior=[True|False]` to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the feature selection.
- You might think to apply some classifier combination technique like ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.
- If the input variables are real-valued, a Gaussian distribution is assumed. In which case the algorithm will perform better if the univariate distributions of your data are Gaussian or near-Gaussian. This may require removing outliers (e.g. values that are more than 3 or 4 standard deviations from the mean).

Applications of Naïve Bayes Classifier

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and [Collaborative Filtering](#) together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not