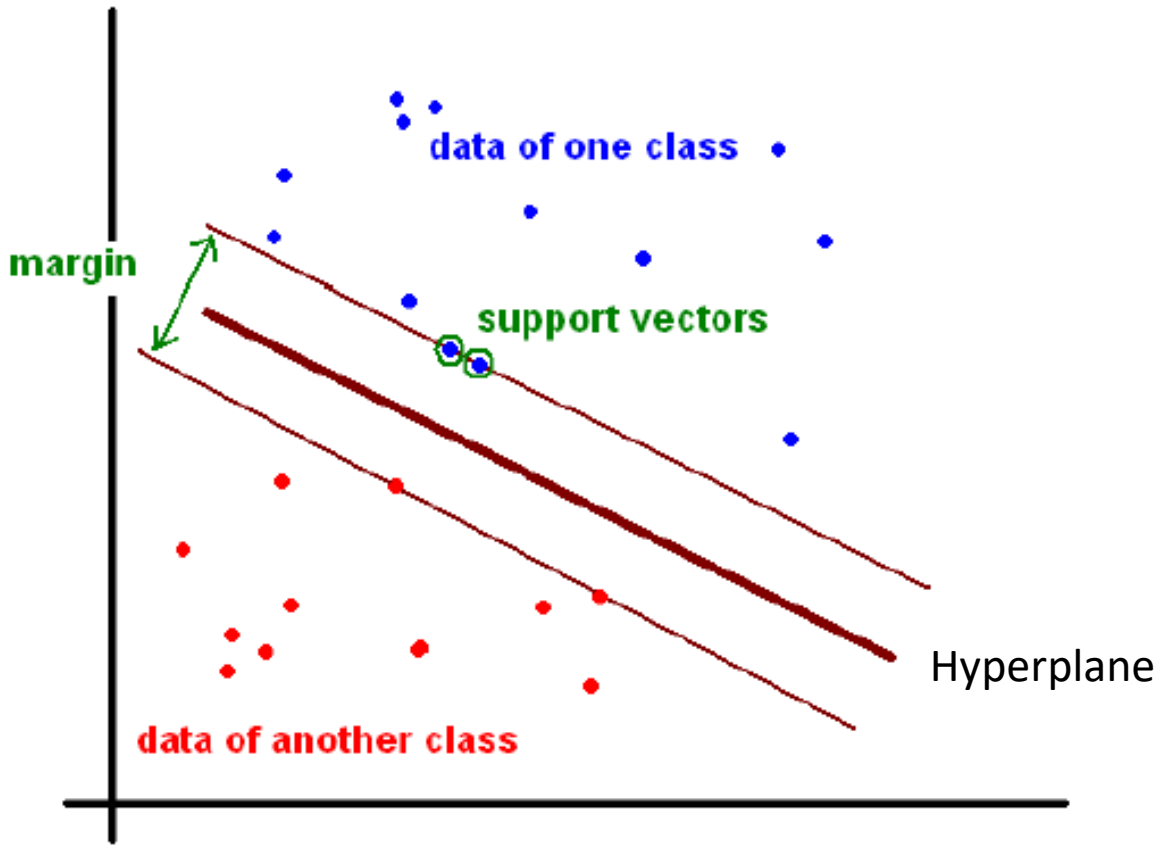
SVM, KNN, Naïve Bayes,

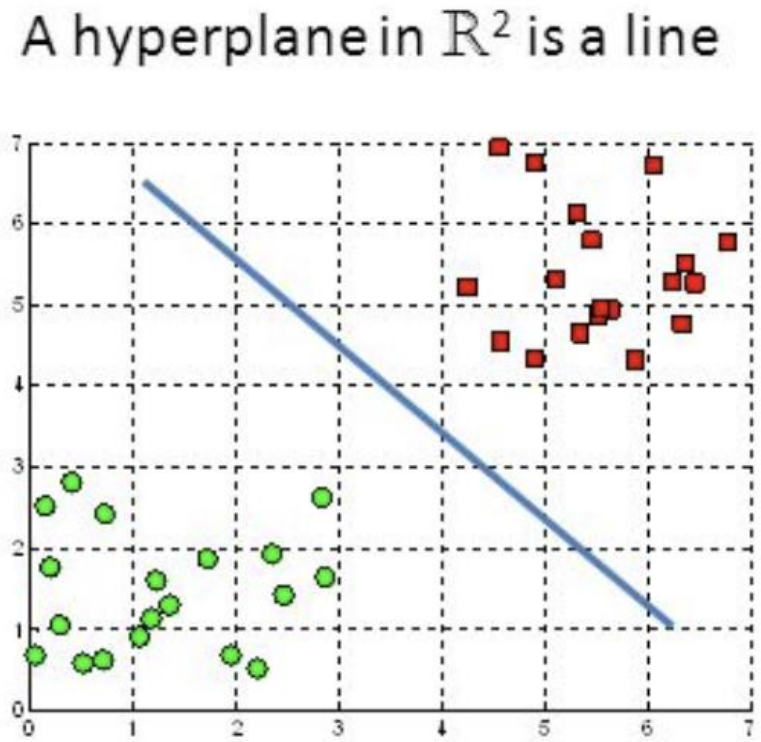
Support Vector Machines (SVM)

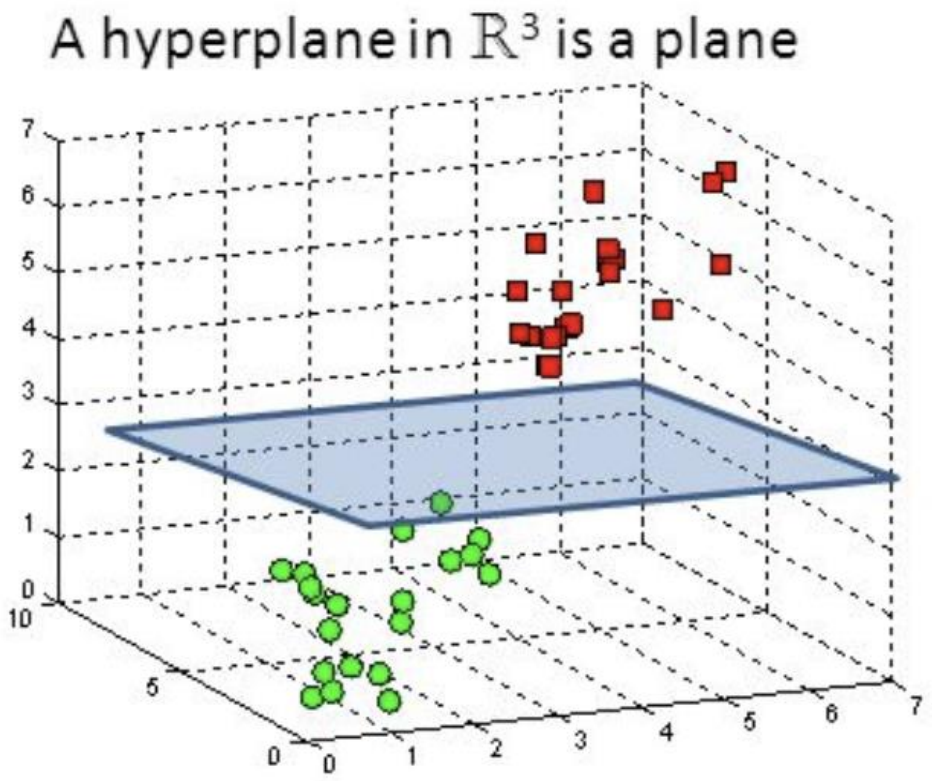
Linearly-separable data, binary classification



- **Goal:** we want to find the hyperplane (i.e. decision boundary) linearly separating our classes.
- A hyperplane splits the input variable space

Hyperplane





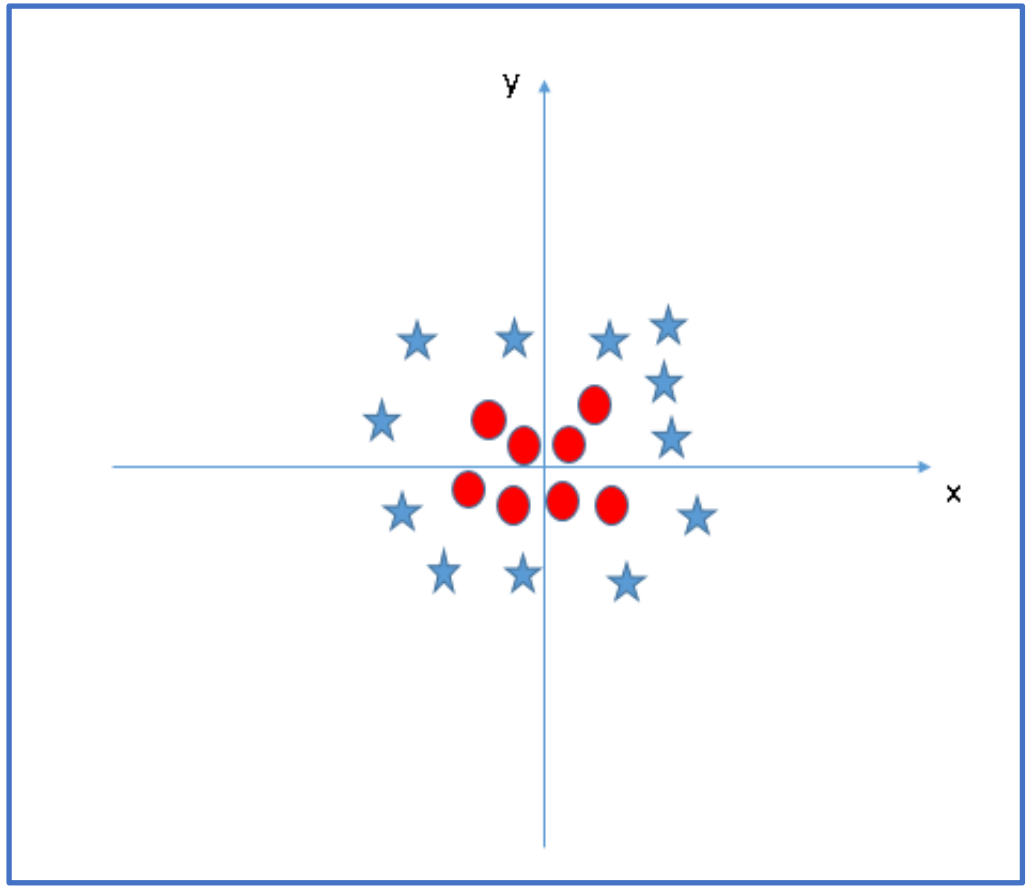
- Hyperplanes are decision boundaries that help classify the data points
- Data points falling on either side of the hyperplane can be attributed to different classes
- The dimension of the hyperplane depends upon the number of features
  - If the number of input features is 2, then the hyperplane is just a line
  - If the number of input features is 3, then the hyperplane becomes a two-dimensional plane

Maximal Margin, Support vectors

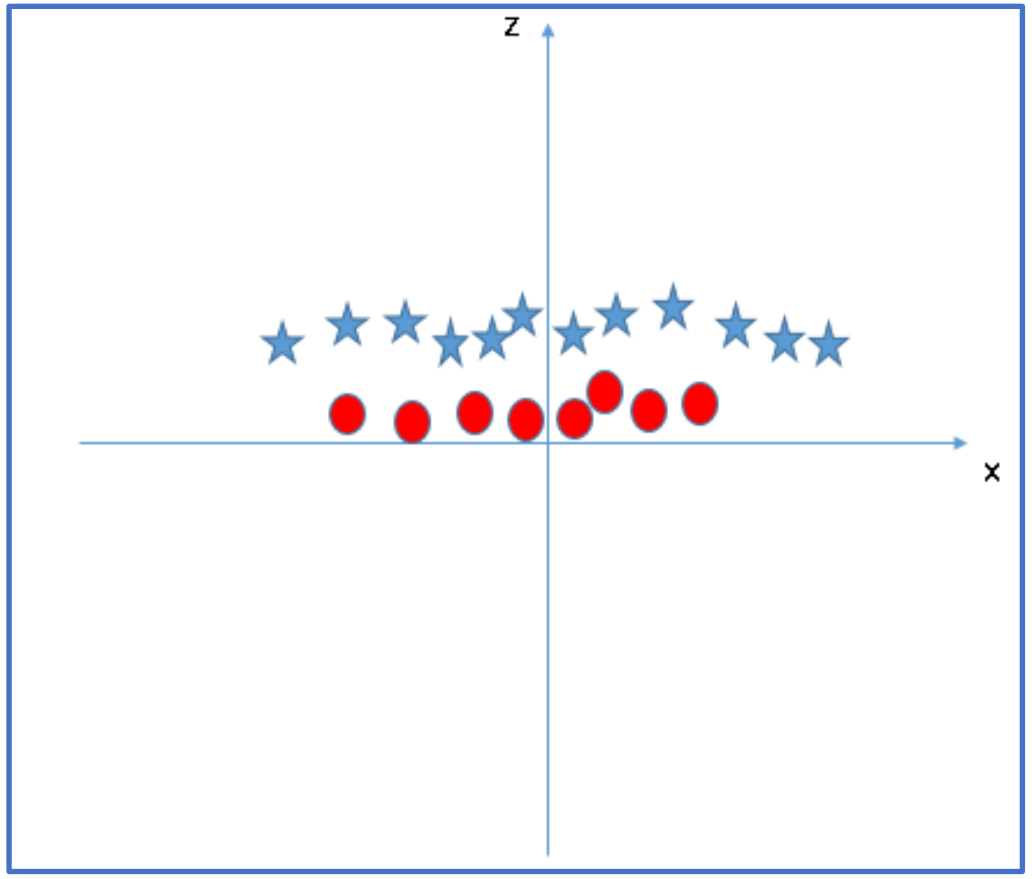
- Maximal Margin
  - The distance between the line and the closest data points is referred to as the margin.
  - The best or optimal line that can separate the two classes is the line that has the largest margin
  - This is called the Maximal-Margin hyperplane.
- Support Vectors
  - The margin is calculated as the perpendicular distance from the line to only the closest points
  - Only these points are relevant in defining the line and in the construction of the classifier.
  - These points are called the support vectors. They support or define the hyperplane.
- The hyperplane is learned from training data using an optimization procedure that maximizes the margin.

Kernel Function

In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes?

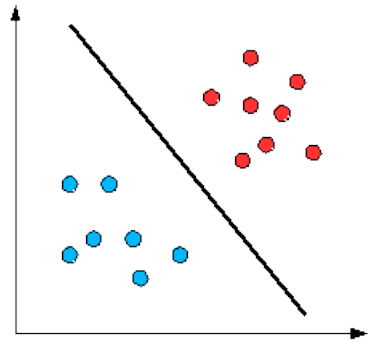


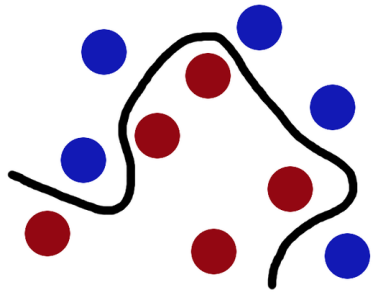
SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



Kernel Function

- In machine learning, a “kernel” is usually used to refer to the kernel trick, a method of using a linear classifier to solve a non-linear problem
- It entails transforming linearly inseparable data like to linearly separable ones
- The kernel function is what is applied on each data instance to map the original non-linear observations into a higher-dimensional space in which they become separable.





MIT Opencourseware
Prof. Patrick Winston – Support Vector Machines

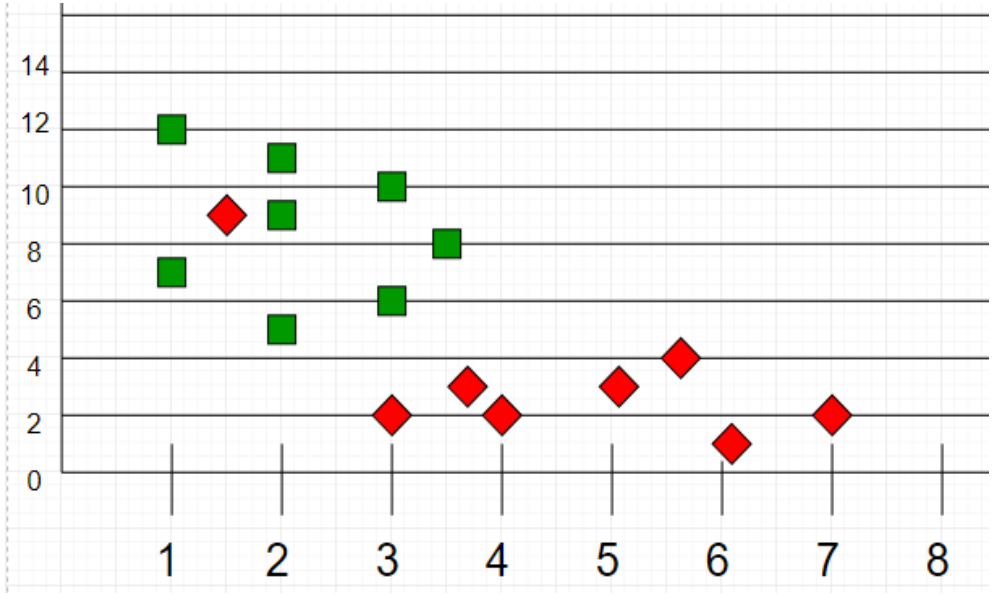
https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-16-learning-support-vector-machines/

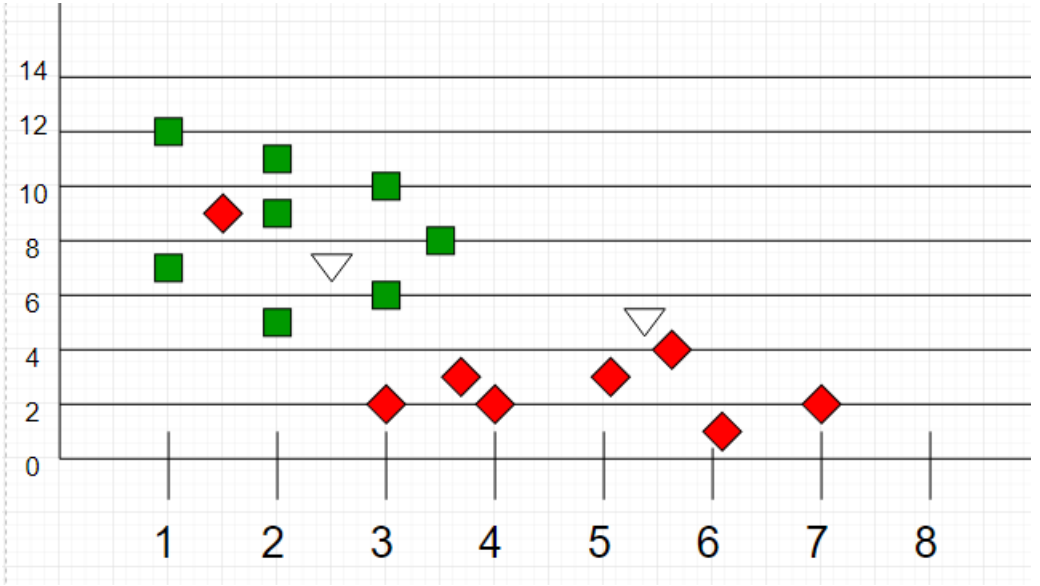
K Nearest Neighbors Classification

K-Nearest Neighbours

x1	x2	y
1	7	0
2	5	0
2	9	0
2	11	0
1	12	0
3	6	0
3	10	0
4	8	0
1.5	9	1
3	2	1
3.75	3	1
4	2	1
5	3	1
5.75	4	1
6.1	1	1
7	2	1

- Lets say we have data as specified in this table
- If we plot these points on a graph, we may be able to locate some clusters or groups
- This means a point close to a cluster of points classified as ‘Red’ has a higher probability of getting classified as ‘Red’
- Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbors belong to
- Intuitively, we can see that the first point (2.5, 7) should be classified as ‘Green’ and the second point (5.5, 4.5) should be classified as ‘Red’.





K Nearest Neighbour

- The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other
- KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics — calculating the distance between points on a graph
- The straight-line distance (also called the Euclidean distance) is a popular and familiar choice
- $\sqrt{a^2 + b^2}$ - Euclidean distance
- $c^2 = (x_A - x_B)^2 + (y_A - y_B)^2$

$$c = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

The KNN Algorithm

- Load the data
- Initialize K to your chosen number of neighbors
- For each example in the data
  - Calculate the distance between the query example and the current example from the data.
  - Add the distance and the index of the example to an ordered collection
- Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
- Pick the first K entries from the sorted collection
- Get the labels of the selected K entries
- Return the mode of the K labels

KNN Advantages and Disadvantages

- **Advantages**
  - The algorithm is simple and easy to implement.
  - There's no need to build a model, tune several parameters, or make additional assumptions.
  - The algorithm is versatile. It can be used for classification, regression
- **Disadvantages**
  - The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase
  - Does not work with categorical data

KNN

- When using KNN ensure that the data is standardized

Naïve Bayes

Joint Probability

- Joint probability is simply the likelihood that two events will happen at the same time.
- Couple conditions.
  - One is that events X and Y must happen at the same time. Throwing two dice would be an example of that.
  - The other is that events *X* and *Y* must be **independent** of each other. Our dice roll is again a good example of independent events
- P(X,Y) = P(X) * P(Y)

Conditional Probabilities

- What is the probability that two cards drawn at random from a deck of playing cards will both be aces?
- Once the first card chosen is an ace, the probability that the second card chosen is also an ace is called the conditional probability of drawing an ace
- Symbolically, we write this as: P(ace on second draw | an ace on the first draw)
- If Events A and B are not independent, then P(A and B) = P(A) x P(B|A).
- Applying this to the problem of two aces, the probability of drawing two aces from a deck is 4/52 x 3/51 = 1/221.

Bayes' theorem:

$$P(A/B) = \frac{P(B/A)\ P(A)}{P(B)}$$

Given $B = (B_1, B_2, \ldots B_n)$

$$P(A/B) = P(B_1/A) * P(B_2/A) \cdots * P(B_n/A) * \frac{P(A)}{P(B)}$$

Assumption: Each predictor variable is independent of each other

$B_1, B_2 \ldots B_n$ independent of each other

Sno	outlook	temper	humidi	windy	Play Golf
0	Rainy	Hot	High	FALSE	No
1	Rainy	Hot	High	TRUE	No
2	Overcast	Hot	High	FALSE	Yes
3	Sunny	Mild	High	FALSE	Yes
4	Sunny	Cool	Normal	FALSE	Yes
5	Sunny	Cool	Normal	TRUE	No
6	Overcast	Cool	Normal	TRUE	Yes
7	Rainy	Mild	High	FALSE	No
8	Rainy	Cool	Normal	FALSE	Yes
9	Sunny	Mild	Normal	FALSE	Yes
10	Rainy	Mild	Normal	TRUE	Yes
11	Overcast	Mild	High	TRUE	Yes
12	Overcast	Hot	Normal	FALSE	Yes
13	Sunny	Mild	High	TRUE	No

P(temp=cool | play golf=TRUE) = 3/9

today = (Sunny, Hot, Normal, False)

P(Play Golf = Yes | today)

Using Bayes' Theorem:

P(Yes | today) = P(SunnyOutlook | Yes) * P(Hot Temp | Yes) * P(Normal Humidity | Yes) * P(Not Windy | Yes) * P(Yes) / P(today)

P(No | today) = P(SunnyOutlook | No) * P(Hot Temp | No) * P(Normal Humidity | No) * P(Not Windy | No) * P(No) / P(today)

P(SunnyOutlook | Yes)	0.333333		P(SunnyOutlook | No)	0.4
P(Hot Temp | Yes)	0.222222		P(Hot Temp | No)	0.4
P(NormalHumidity | Yes)	0.666667		P(NormalHumidity |No)	0.2
P(Not Windy | Yes)	0.666667		P(Not Windy | No)	0.4
P(Yes)	0.642857		P(No)	0.357143

P(Yes | today) = 0.032922 * 0.642857 / P(today)

P(No | today) = 0.0128 * 0.357143 / P(today)

But, P(Yes | today) + P(No | today) = 1

Hence: 0.021164 / P(today) + 0.004571 / P(today) = 1

P(today) = 0.025735

P(Yes | today)	=	0.822368
P(No | today)	=	0.177632

Naïve Bayes : Discrete vs Continuous features

- The method that we discussed above is applicable for discrete data
- In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature
- The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of values of each feature.
- In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution (**also called Normal distribution)
- When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values

Naïve Bayes types

- **Gaussian Naive Bayes:**
  - When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.
- **Multinomial Naive Bayes:**
  - This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the **frequency of the words** present in the document.
- **Bernoulli Naive Bayes:**
  - This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

Naïve Bayes – Pros and Cons

- Decision tree , Random Forest or any tree based algorithm are better as they do not care about correlation between independent variables
- In real world - most of our data will have correlation among independent variables
- Naive Baye's algorithm is most susceptible to the multicollinearity

- Naive bayes algorithm is typically not data hungry (dataset is small)
- The Naïve Bayes models are interpretable models