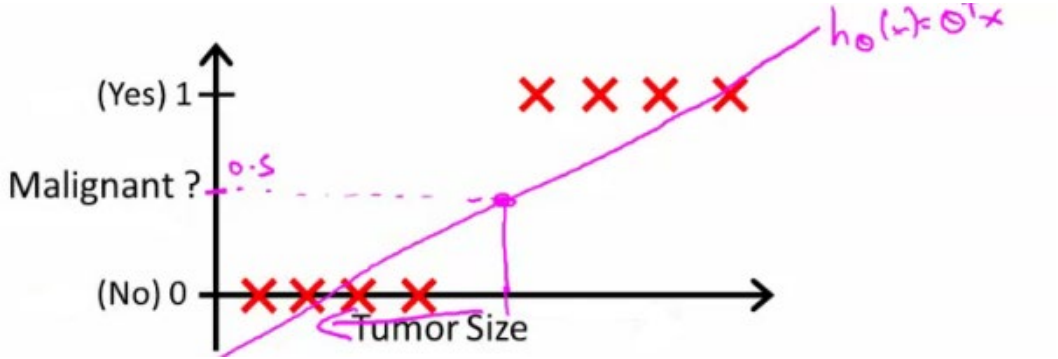
Details of ML part 2

- Start with **binary class problems**

How do we develop a classification algorithm?

- Tumour size vs malignancy (0 or 1)
- We *could* use linear regression
  - Then threshold the classifier output (i.e. anything over some value is yes, else no)
  - In our example below linear regression with thresholding seems to work



•We can see above this does a reasonable job of stratifying the data points into one of two classes
  - But what if we had a single Yes with a very small tumour
  - This would lead to classifying all the existing yeses as nos
•Another issues with linear regression
  - We know Y is 0 or 1
  - Hypothesis can give values large than 1 or less than 0
•So, logistic regression generates a value where is always either 0 or 1
  - Logistic regression is a **classification algorithm** - don't be confused

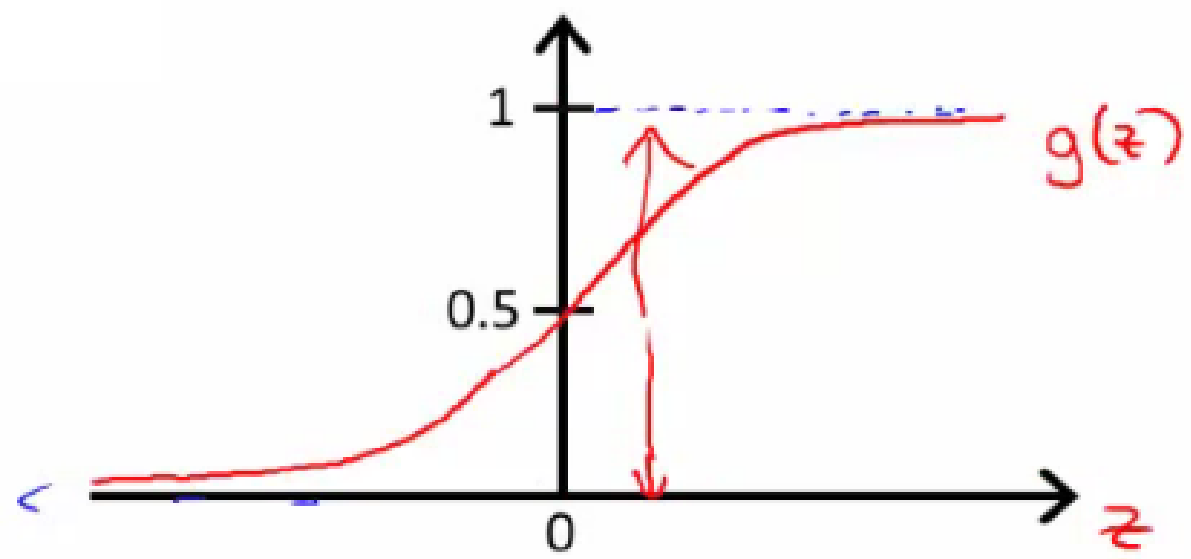
Hypothesis representation

•What function is used to represent our hypothesis in classification
•We want our classifier to output values between 0 and 1
  - When using linear regression we did $h_\theta(x) = (\theta^T x)$
  - For classification hypothesis representation we do $h_\theta(x) = g((\theta^T x))$
    - Where we define g(z)
      - z is a real number
      - This is the **sigmoid function**, or the **logistic function**
    - If we combine these equations we can write out the hypothesis as

$$g(z) = \frac{1}{1+e^{-z}}$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

•How does the sigmoid function look like



•Crosses 0.5 at the origin, then flattens out]

- Asymptotes at 0 and 1

•Interpreting hypothesis output

When our hypothesis ($h_\theta(x)$) outputs a number, we treat that value as the estimated probability that y=1 on input x

- Example
  - If X is a feature vector with $x_0 = 1$ (as always) and x_1 = tumourSize
  - $h_\theta(x) = 0.7$
    - Tells a patient they have a 70% chance of a tumor being malignant

$h_\theta(x) = P(y=1|x ; \theta)$

- What does this mean?
  - Probability that y=1, given x, parameterized by θ

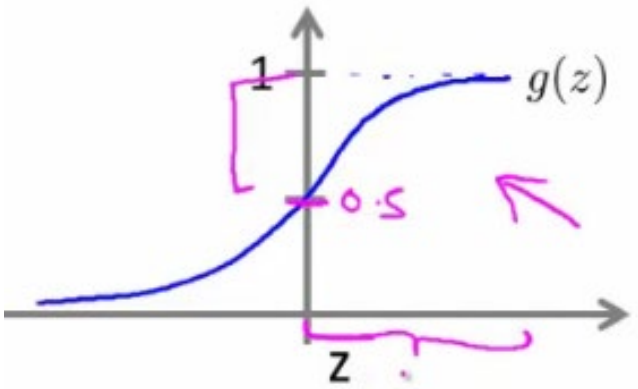
•Since this is a binary classification task we know y = 0 or 1

- So the following must be true
  - $P(y=1|x ; \theta) + P(y=0|x ; \theta) = 1$
  - $P(y=0|x ; \theta) = 1 - P(y=1|x ; \theta)$

Decision boundary

•This gives a better sense of what the hypothesis function is computing

- One way of using the sigmoid function is;
  - When the probability of y being 1 is greater than 0.5 then we can predict y = 1
  - Else we predict y = 0
- When is it exactly that $h_\theta(x)$ is greater than 0.5?
  - Look at sigmoid function
    - g(z) is greater than or equal to 0.5 when z is greater than or equal to 0
  - So if z is positive, g(z) is greater than 0.5
    - $z = (\theta^T x)$
  - So when
    - $\theta^T x >= 0$
  - Then $h_\theta >= 0.5$

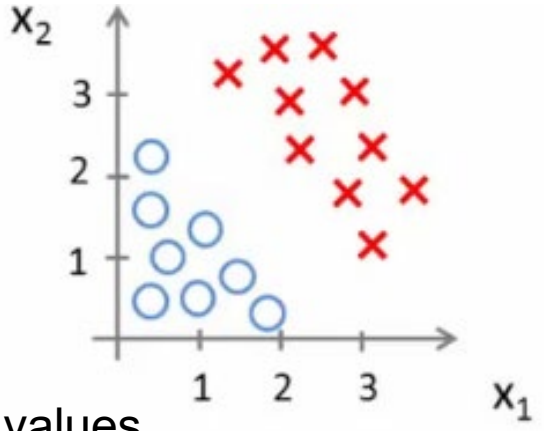


•So what we've shown is that the hypothesis predicts y = 1 when $\theta^T x >= 0$

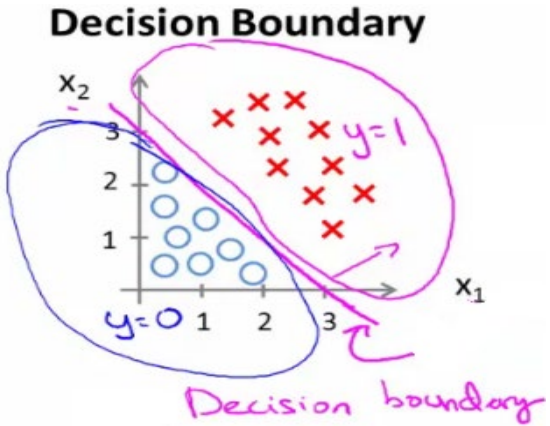
- The corollary of that when $\theta^T x <= 0$ then the hypothesis predicts y = 0
- Let's use this to better understand how the hypothesis makes its predictions

Consider,

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$



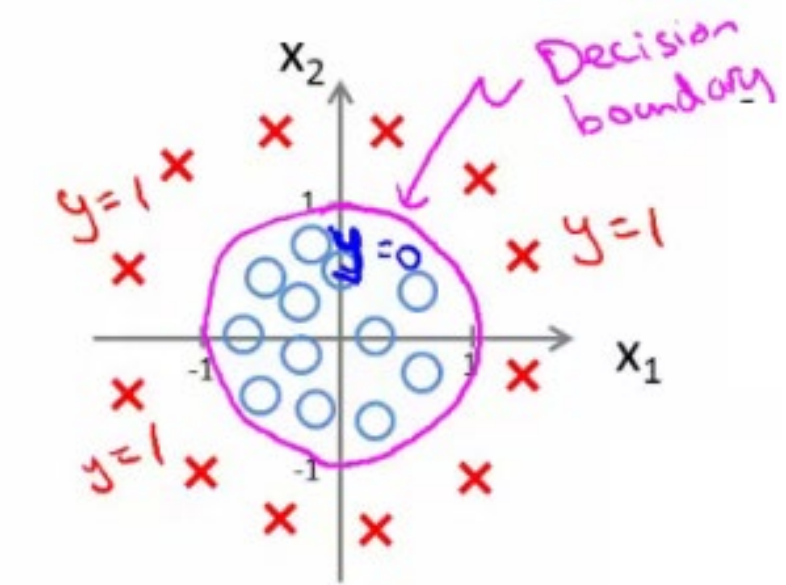
- So, for example
  - $\theta_0 = -3$
  - $\theta_1 = 1$
  - $\theta_2 = 1$
- So our parameter vector is a column vector with the above values
  - So, θ^T is a row vector = [-3,1,1]
- What does this mean?
  - The z here becomes $\theta^T x$
  - We predict "y = 1" if
    - $-3x_0 + 1x_1 + 1x_2 >= 0$
    - $-3 + x_1 + x_2 >= 0$
- We can also re-write this as
  - If $(x_1 + x_2 >= 3)$ then we predict y = 1
  - If we plot
    - $x_1 + x_2 = 3$ we graphically plot our **decision boundary**



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2 + \theta_4 x_2^2)$

- Say θ^T was [-1,0,0,1,1] then we say;
- Predict that "y = 1" *if*
  - $-1 + x_1^2 + x_2^2 >= 0$

    or
  - $x_1^2 + x_2^2 >= 1$
- If we plot $x_1^2 + x_2^2 = 1$



Cost function for logistic regression

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Linear regression uses the following function to determine θ

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

A convex logistic regression cost function

•To get around this we need a different, convex Cost() function which means we can apply gradient descent

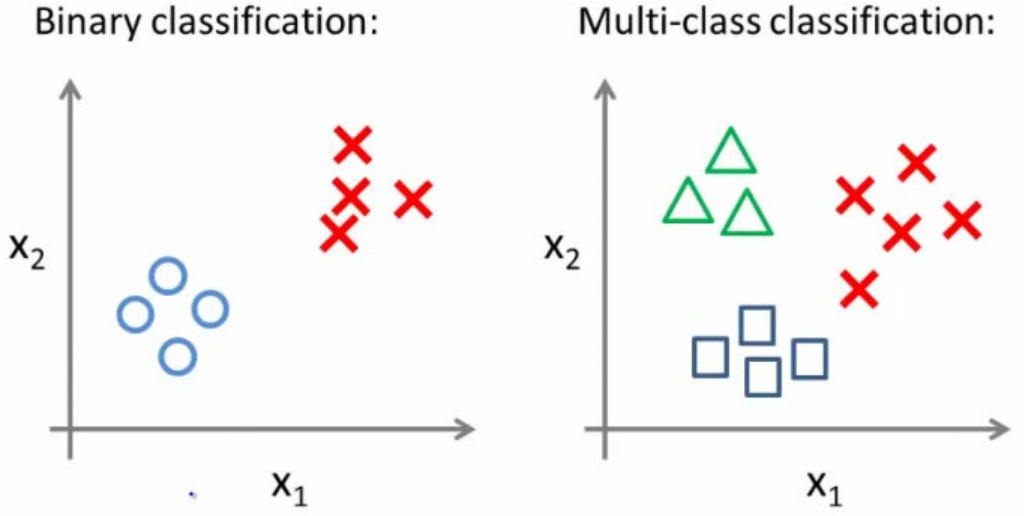
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

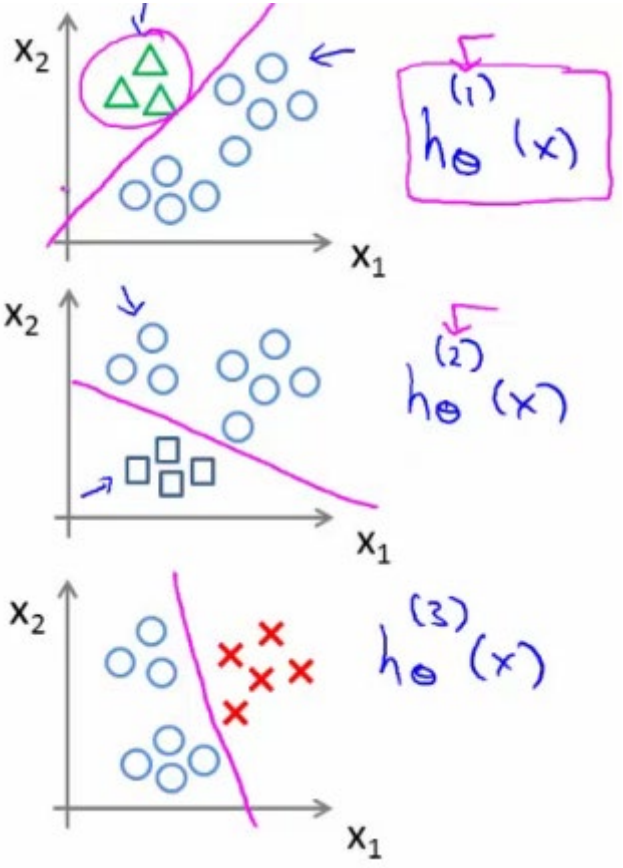
Multiclass classification problems

- Getting logistic regression for multiclass classification using **one vs. all**
- Multiclass - more than yes or no (1 or 0)
  - Classification with multiple classes for assignment



•Given a dataset with three classes, how do we get a learning algorithm to work?
  - Use one vs. all classification make binary classification work for multiclass classification

•One vs. all classification
  - Split the training set into three separate binary classification problems
    - i.e. create a new fake training set
      - Triangle (1) vs crosses and squares (0) $h_\theta^1(x)$
        - $P(y=1 \mid x_1; \theta)$
      - Crosses (1) vs triangle and square (0) $h_\theta^2(x)$
        - $P(y=1 \mid x_2; \theta)$
      - Square (1) vs crosses and square (0) $h_\theta^3(x)$
        - $P(y=1 \mid x_3; \theta)$



•Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that y = i

•On a new input, *x* to make a prediction, pick the class *i* that maximizes the probability that $h_\theta^{(i)}(x) = 1$

K-Nearest Neighbors

X: Independent variable

Y: Dependent variable

	region	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	44	1	9	64	4	5	0	0	2	1
1	3	33	1	7	136	5	5	0	0	6	4
2	3	52	1	24	116	1	29	0	1	2	3
3	2	33	0	12	33	2	0	0	1	1	1
4	2	30	1	9	30	1	2	0	0	4	3
5	2	39	0	17	78	2	16	0	1	1	3
6	3	22	1	2	19	2	4	0	1	5	2
7	2	35	0	5	76	2	10	0	0	3	4
8	3	50	1	7	166	4	31	0	0	5	?

Value	Label
1	Basic Service
2	E-Service
3	Plus Service
4	Total Service

This algorithm classifies cases based on their similarity to other cases.

In K-Nearest Neighbors, data points that are near each other are said to be neighbors.

K-Nearest Neighbors is based on this paradigm.

Similar cases with the same class labels are near each other.
Thus, the distance between two cases is a measure of their dissimilarity.
There are different ways to calculate the similarity or conversely,
the distance or dissimilarity of two data points.
For example, this can be done using Euclidean distance.

the K-Nearest Neighbors algorithm works as follows.

- pick a value for K.
- calculate the distance from the new case hold out from each of the cases in the dataset.
- search for the K-observations in the training data that are nearest to the measurements of the unknown data point.
- predict the response of the unknown data point using the most popular response value from the K-Nearest Neighbors.

There are two parts in this algorithm that might be a bit confusing.

- First, how to select the correct K
- second, how to compute the similarity between cases

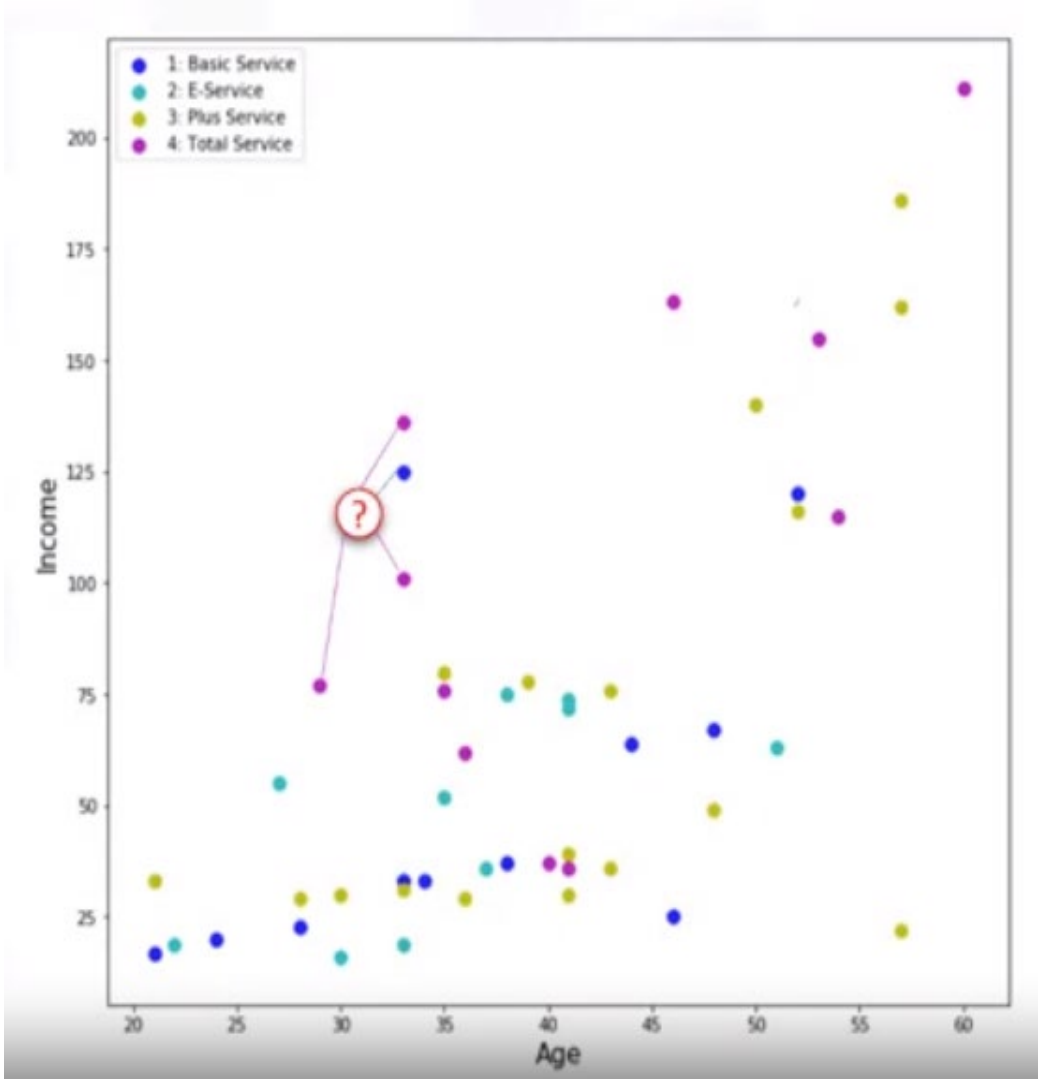
How to select the correct K

As mentioned, K and K-Nearest Neighbors is the number of nearest neighbors to examine.

It is supposed to be specified by the user. So, how do we choose the right K?

Assume that we want to find the class of the customer noted as question mark on the chart.

What happens if we choose a very low value of K?
Let's say, K equals one.
The first nearest point would be blue, which is class one.
This would be a bad prediction, since more of the points around it are magenta or class four.



In fact, since its nearest neighbor is blue we can say that we capture the noise in the data or we chose one of the points that was an anomaly in the data.

A low value of K causes a highly complex model as well, which might result in overfitting of the model.

It means the prediction process is not generalized enough to be used for out-of-sample cases.

Out-of-sample data is data that is outside of the data set used to train the model.

In other words, it cannot be trusted to be used for prediction of unknown samples. It's important to remember that overfitting is bad, as we want a general model that works for any data, not just the data used for training.

Now, on the opposite side of the spectrum, if we choose a very high value of K such as K equals 20,
then the model becomes overly generalized.

So, how can we find the best value for K?

The general solution is to reserve a part of your data for testing the accuracy of the model.
Once you've done so, choose K equals one and then use the training part for modeling and calculate the accuracy of prediction using all samples in your test set.

Repeat this process increasing the K and see which K is best for your model.
For example, in our case,

K equals four will give us the best accuracy.

Calculating the similarity/distance in a multi-dimensional space

Customer 1		
Age	Income	Education
54	190	3

Customer 2		
Age	Income	Education
50	200	8

$$\text{Dis}\,(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

$$= \sqrt{(54-50)^2+(190-200)^2+(3-8)^2} = 11.87$$

Advantages of KNN

1. No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g. SVM, Linear Regression etc.

2. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

3. KNN is very easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Disadvantages of KNN

1. Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.

2. Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.

3.Sensitive to noisy data, missing values and outliers: KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers.

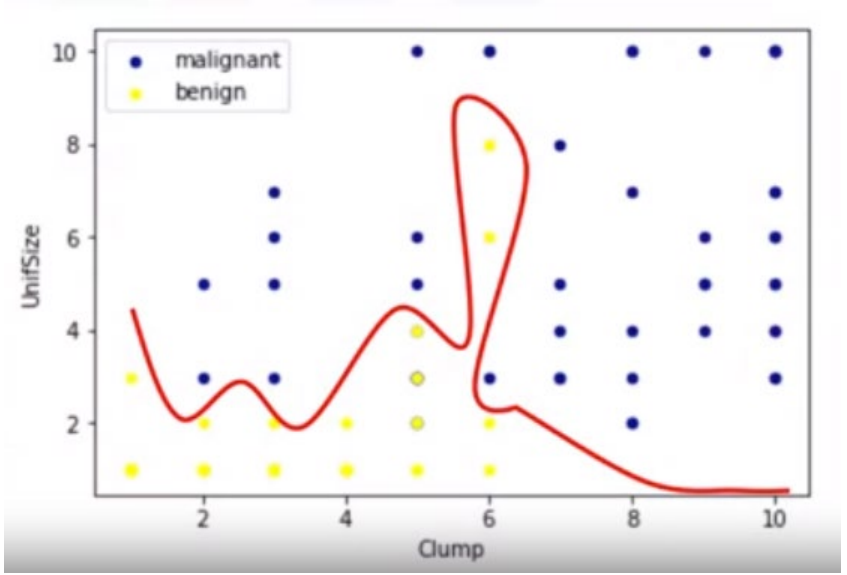
SUPPORT VECTOR MACHINE(SVM)

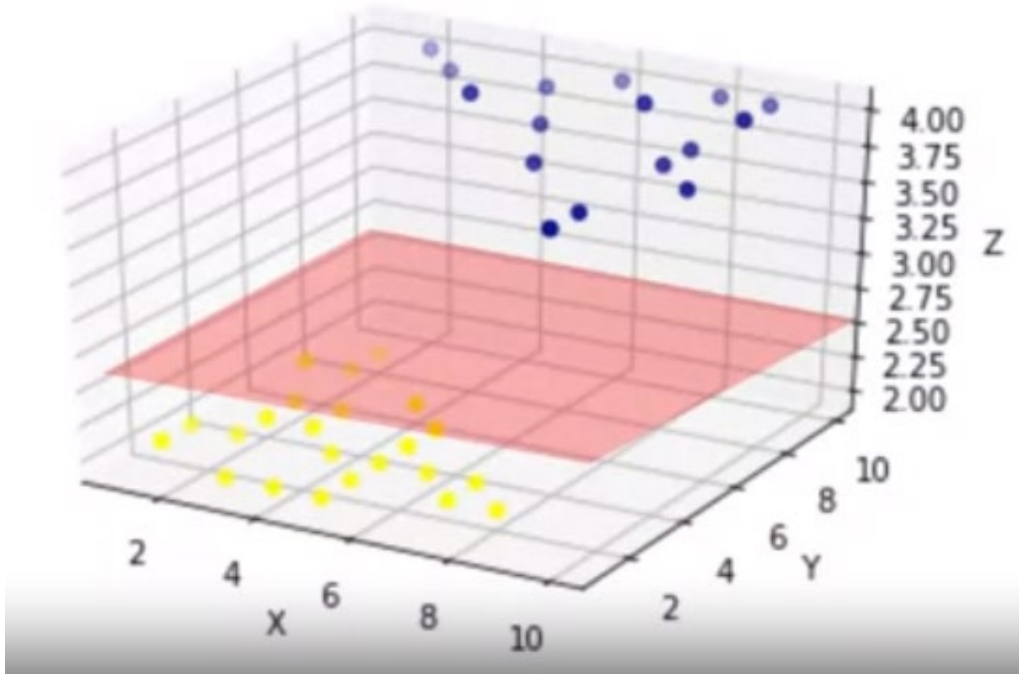
A Support Vector Machine is a supervised algorithm that can classify cases by finding a separator.

SVM works by first mapping data to a high dimensional feature space so that data points can be categorized, even when the data are not linearly separable.

Then, a separator is estimated for the data. The data should be transformed in such a way that a separator could be drawn as a hyperplane.

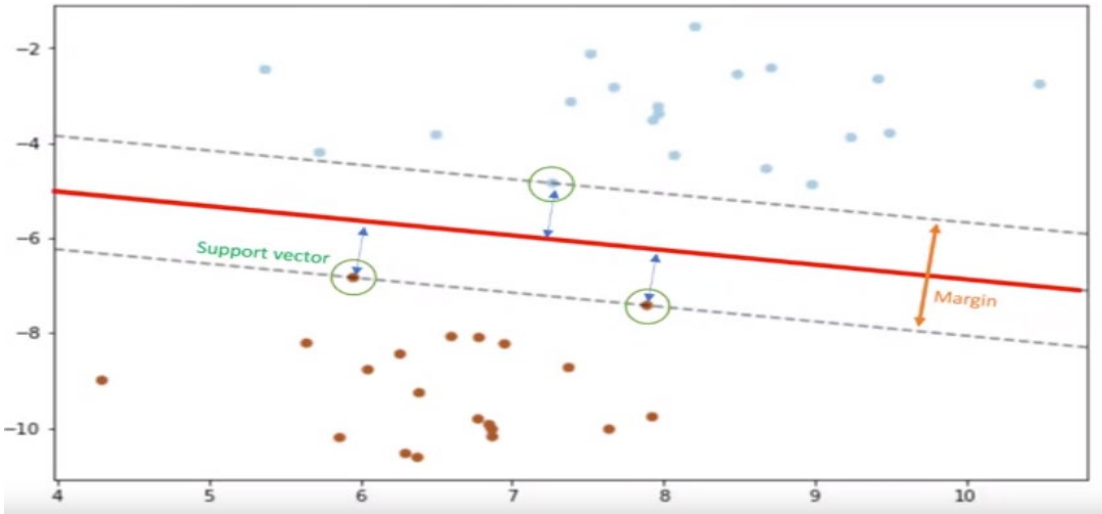
Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
5	1	1	1	2	1	3	1	1	benign
5	4	4	5	7	10	3	2	1	benign
3	1	1	1	2	2	3	1	1	malignant
6	8	8	1	3	4	3	7	1	benign
4	1	1	3	2	1	3	1	1	benign
8	10	10	8	7	10		7	1	malignant
1	1	1	1	2	10	3	1	1	benign
2	1	2	H	2	1	3	1	1	benign
2	1	1	1	2	1	1	1	5	benign
4	2	1	1	2	1	2	1	1	benign





Therefore, the SVM algorithm outputs an optimal hyperplane that categorizes new examples.

SVMs are based on the idea of finding a hyperplane that best divides a data set into two classes as shown here.
As we're in a two-dimensional space, you can think of the hyperplane as a line that linearly separates the blue points from the red points.



ADVANTAGES
- Accurate in high dimension place
- Memory efficient

DISADVANTAGES
- Small datasets
- Prone to overfitting

APPLICATIONS
- Image Recognition
- Spam detection

Naive Bayes Classifiers

collection of classification algorithms

Principle of Naive Bayes Classifier:

- A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

- Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Example:

Let us take an example to get some better intuition. Consider the problem of playing golf. The dataset is represented as right table.

	OUTLOOK	TEMPERATURE	HUMIDITY	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

- We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries. If we take the first row of the dataset, we can observe that is not suitable for playing golf if the outlook is rainy, temperature is hot, humidity is high and it is not windy. We make two assumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high. Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to play golf or not.
- According to this example, Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- The variable **y** is the class variable(play golf), which represents if it is suitable to play golf or not given the conditions.
  Variable **X** represent the parameters/features.

- **X** is given as, $X = (x_1, x_2, x_3, ..., x_n)$
- Here x_1,x_2….x_n represent the features, i.e they can be mapped to outlook, temperature, humidity and windy. By substituting for **X** and expanding using the chain rule we get,

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

- Now, you can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remain static. Therefore, the denominator can be removed and a proportionality can be introduced.

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

- In our case, the class variable(**y**) has only two outcomes, yes or no. There could be cases where the classification could be multivariate. Therefore, we need to find the class **y** with maximum probability.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

- Using the above function, we can obtain the class, given the predictors.

- We need to find $P(x_i \mid y_j)$ for each x_i in X and y_j in y. All these calculations have been demonstrated in the tables below:

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

- So, in the figure above, we have calculated $P(x_i \mid y_j)$ for each x_i in X and y_j in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

- Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.
- So now, we are done with our pre-computations and the classifier is ready!
- Let us test it on a new set of features:

```
today = (Sunny, Hot, Normal, False)
```

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

and

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

Now, since

$$P(Yes|today) + P(No|today) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

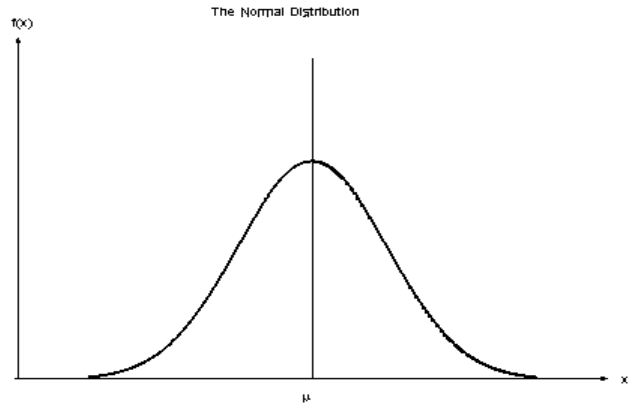
So, prediction that golf would be played is 'Yes'.

Types of Naive Bayes Classifier:

- **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.

- **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.

- **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.



Gaussian Distribution(Normal Distribution)

Conclusion:

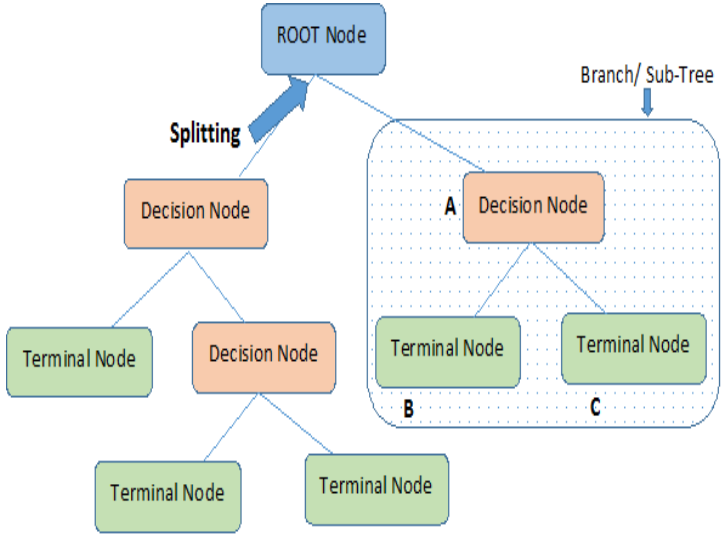
Naive Bayes algorithms are mostly used in sentiment analysis, spam filtering, recommendation systems etc. They are fast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent. In most of the real life cases, the predictors are dependent, this hinders the performance of the classifier.

Decision Tree

Classification Algorithm

- Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems..
- Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.
- We can represent any boolean function on discrete attributes using the decision tree.
- **Types of decision trees**
- Categorical Variable Decision Tree: Decision Tree which has categorical target variable then it called as categorical variable decision tree.
- Continuous Variable Decision Tree: Decision Tree which has continuous target variable then it is called as Continuous Variable Decision Tree.

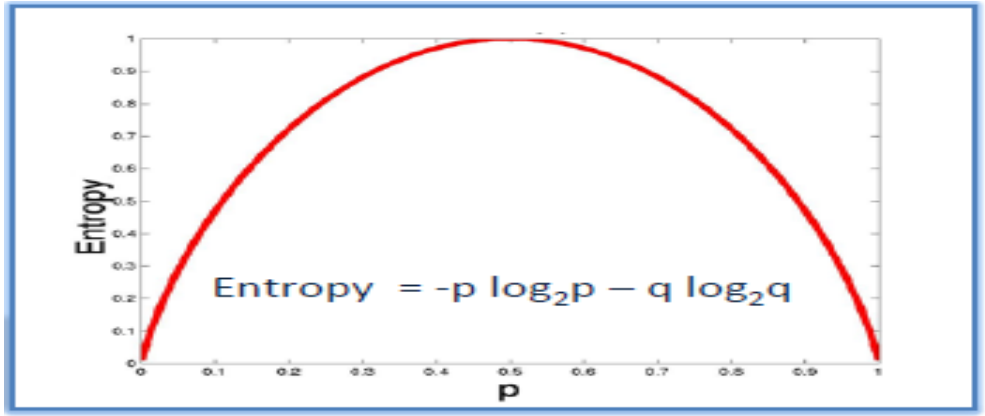
- **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
- **Leaf/ Terminal Node:** Nodes with no children (no further split) is called Leaf or Terminal node.
- **Pruning:** When we reduce the size of decision trees by removing nodes (opposite of Splitting), the process is called pruning.
- **Branch / Sub-Tree:** A sub section of decision tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.



Note:- A is parent node of B and C.

Entropy

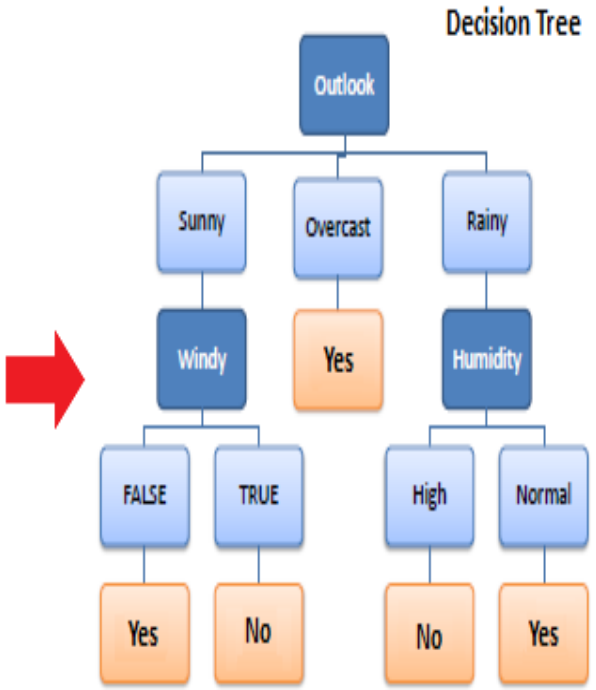
- Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Decision tree algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Example:

Predictors				Target
Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



- To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:
- a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

- b) Entropy using the frequency table of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)*__E__(3,2) + **P**(Overcast)*__E__(4,0) + **P**(Rainy)*__E__(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Information gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

$$\text{Entropy(PlayGolf)} = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94$$

- Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

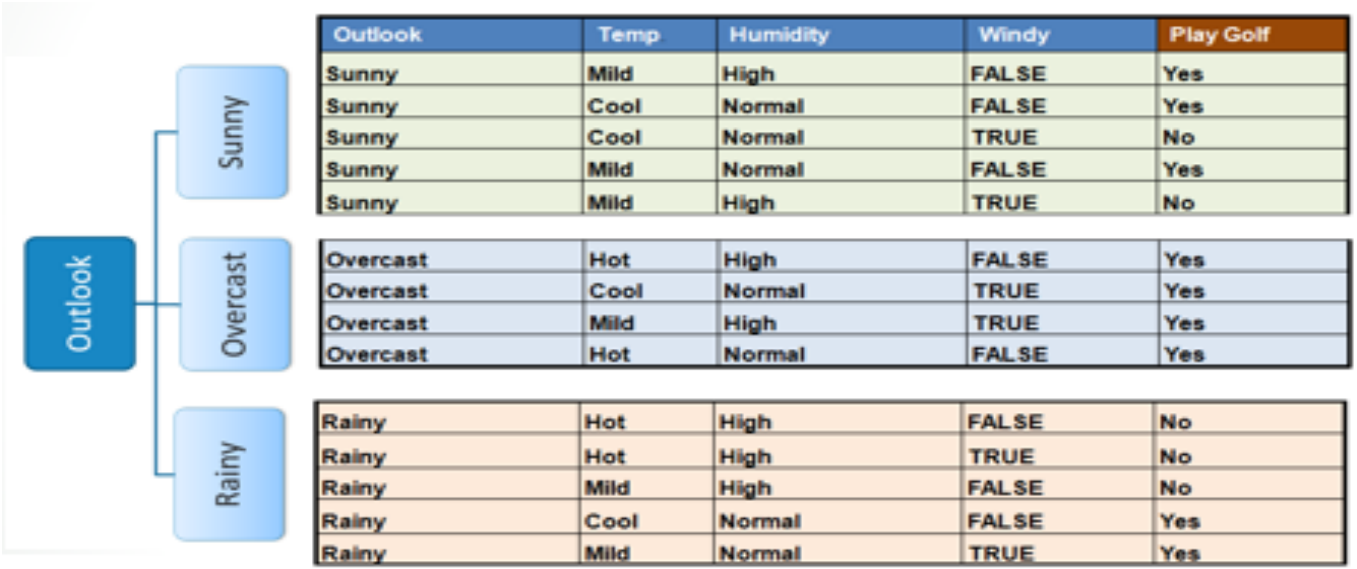
$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

- *Step 3*: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



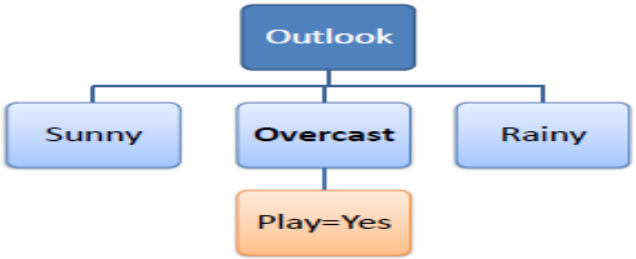
Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes

Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

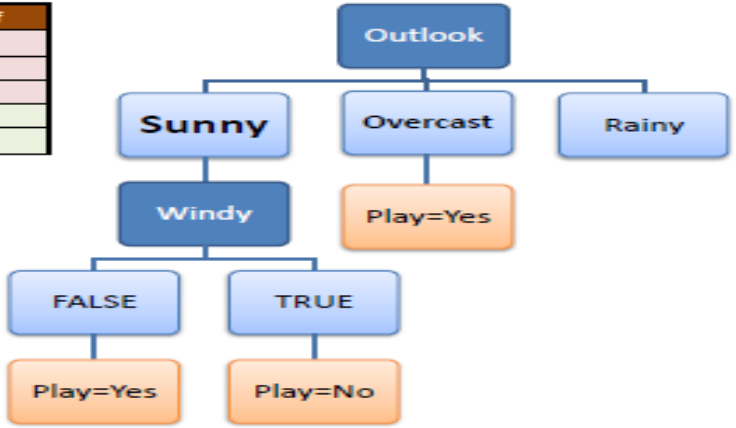
- *Step 4a*: A branch with entropy of 0 is a leaf node

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



- *Step 4b*: A branch with entropy more than 0 needs further splitting

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



- *Step 5*: The algorithm is run recursively on the non-leaf branches, until all data is classified.
- Decision Tree to Decision Rules
- A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

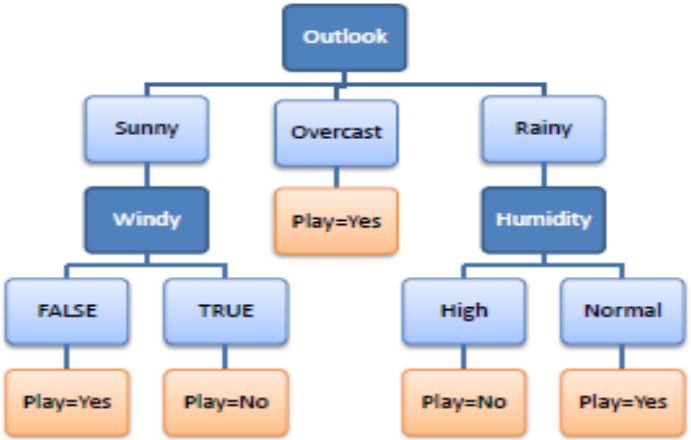
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



- **Limitations to Decision Trees**
- Decision trees tend to have high variance when they utilize different training and test sets of the same data, since they tend to overfit on training data. This leads to poor performance on unseen data. Unfortunately, this limits the usage of decision trees in predictive modeling.
- To overcome these problems we use ensemble methods, we can create models that utilize underlying(weak) decision trees as a foundation for producing powerful results and this is done in Random Forest Algorithm

Random forest

- **Definition**:
- Random forest algorithm is a supervised classification algorithm Based on Decision Trees, also known as random decision forests, are a popular ensemble method that can be used to build predictive models for both classification and regression problems.
- Ensemble we mean(In Random Forest Context), Collective Decisions of Different Decision Trees. In RFT(Random Forest Tree), we make a prediction about the class, not simply based on One Decision Trees, but by an (almost) Unanimous Prediction, made by '**K'** Decision Trees.
- **Construction**:
- 'K' Individual Decision Trees are made from given Dataset, by randomly dividing the Dataset and the Feature Subspace by process called as **Bootstrap Aggregation**(Bagging), which is process of random selection with replacement. Generally 2/3rd of the Dataset (row-wise 2/3rd) is selected by bagging, and On that Selected Dataset we perform what we call is Attribute Bagging.

- Now **Attribute Bagging** is done to select 'm' features from given M features,(this Process is also called Random Subspace Creation.) Generally value of 'm' is square-root of M. Now we select say, 10 such values of m, and then Build 10 Decision Trees based on them, and test the 1/3rd remaining Dataset on these(10 Decision Trees).We would then Select the Best Decision Tree out of this. And Repeat the whole Process 'K' times again to build such 'K' decision trees.
- **Classification:**
- Prediction in Random Forest (a collection of 'K' Decision Trees) is truly ensemble ie, For Each Decision Tree, Predict the class of Instance and then return the class which was predicted the most often.

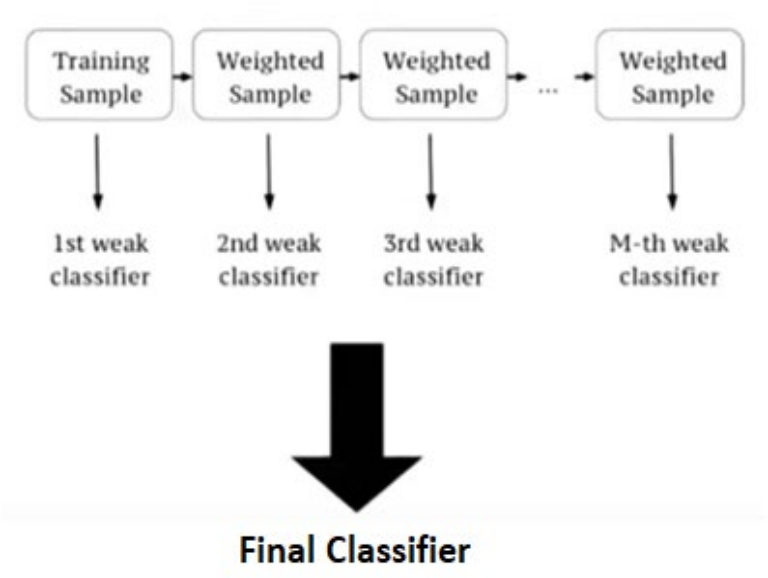
Final Thoughts

- Random Forest Classifier being ensembled algorithm tends to give more accurate result. This is because it works on principle,
- Number of weak estimators when combined forms strong estimator.
- Even if one or few decision trees are prone to a noise, overall result would tend to be correct.

XGBOOST

- XGBoost is an implementation of gradient boosted decision trees designed for speed and performance

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results.



Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

Final value of the classifier is the class which is repeated more number of times among all classifier for classification problem and for regression problem final value is the average of all the regressor values got in sequential trees