UNIT I
INTRODUCTION TO MACHINE LEARNING

Machine Learning vs Statistical Modelling, Applications of Machine Learning, Supervised vs Unsupervised Learning, Supervised Learning Classification, Unsupervised Learning Classification, Python libraries suitable for Machine Learning.

Definition of Machine learning:

Well posed learning problem: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."(Tom Michel)

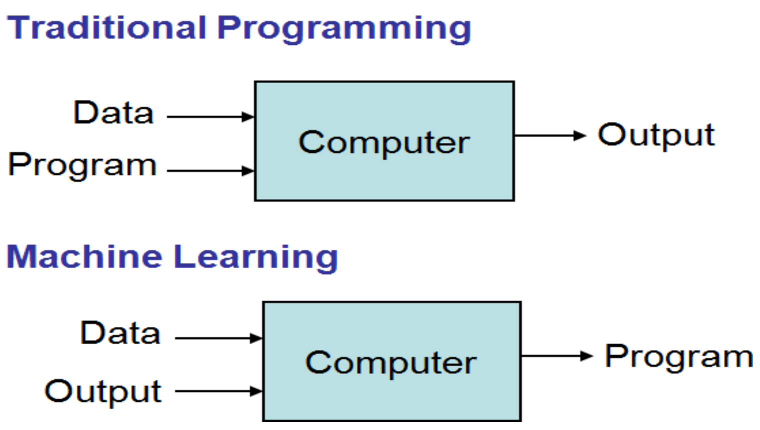
"Field of study that gives computers the ability to learn without being explicitly programmed".

<u>Learning</u> = Improving with experience at some task

- Improve over task T,
- with respect to performance measure P,
- based on experience E.

E.g., Learn to play checkers

- T : Play checkers
- P : % of games won in world tournament
- E: opportunity to play against self



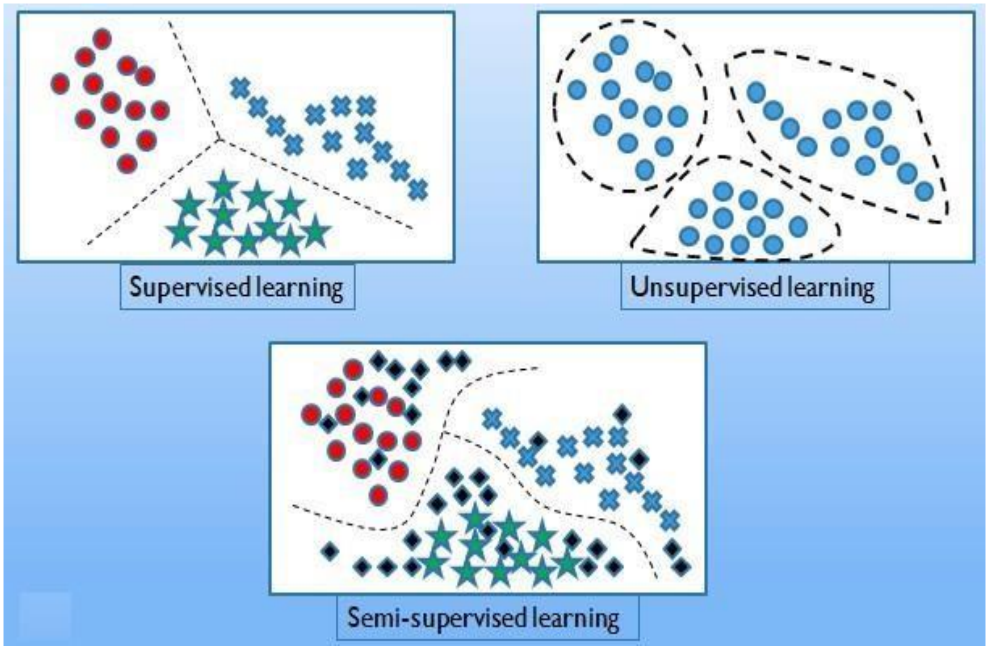
Examples of tasks that are best solved by using a learning algorithm:

- Recognizing patterns:
  - Facial identities or facial expressions
  - Handwritten or spoken words
  - Medical images
- Generating patterns:
  - Generating images or motion sequences (demo)
- Recognizing anomalies:
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant or unusual sound in your car engine.
- Prediction:
  - Future stock prices or currency exchange rates
- The web contains a lot of data. Tasks with very big datasets often use machine learning
  - especially if the data is noisy or non-stationary.
- Spam filtering, fraud detection:
  - The enemy adapts so we must adapt too.
- Recommendation systems:
  - Lots of noisy data. Million dollar prize!
- Information retrieval:
  - Find documents or images with similar content.
- Data Visualization:
  - Display a huge database in a revealing way

Types of learning algorithms:

- Supervised learning
  - Training data includes desired outputs. Examples include,
    - Prediction
    - Classification (discrete labels), Regression (real values)

- Unsupervised learning
  - Training data does not include desired outputs, Examples include,
    - Clustering
    - Probability distribution estimation
    - Finding association (in features)
    - Dimension reduction
- Semi-supervised learning
  - Training data includes a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions
    - Policies: what actions should an agent take in a particular situation
    - Utility estimation: how good is a state (→used by policy)
  - No supervised output but delayed reward
  - Credit assignment problem (what was responsible for the outcome)
  - Applications:
    - Game playing
    - Robot in a maze
    - Multiple agents, partial observability, ...



<u>Hypothesis Space</u>

- One way to think about a supervised learning machine is as a device that explores a "hypothesis space".
  - Each setting of the parameters in the machine is a different hypothesis about the function that maps input vectors to output vectors.
  - If the data is noise-free, each training example rules out a region of hypothesis space.
  - If the data is noisy, each training example scales the posterior probability of each point in the hypothesis space in proportion to how likely the training example is given that hypothesis.
- The art of supervised machine learning is in:
  - Deciding how to represent the inputs and outputs
  - Selecting a hypothesis space that is powerful enough to represent the relationship between inputs and outputs but simple enough to be searched.

<u>Generalization</u>

- The real aim of supervised learning is to do well on test data that is not known during learning.
- Choosing the values for the parameters that minimize the loss function on the training data is not necessarily the best policy.
- We want the learning machine to model the true regularities in the data and to ignore the noise in the data.
  - But the learning machine does not know which regularities are real and which are accidental quirks of the particular set of training examples we happen to pick.
- So how can we be sure that the machine will generalize correctly to new data?

<u>Training set, Test set and Validation set</u>

- Divide the total dataset into three subsets:
  - Training data is used for learning the parameters of the model.
  - Validation data is not used of learning but is used for deciding what type of model and what amount of regularization works best.
  - Test data is used to get a final, unbiased estimate of how well the network works. We expect this estimate to be worse than on the validation data.

- We could then re-divide the total dataset to get another unbiased estimate of the true error rate.

Learning Associations:

- Basket analysis:

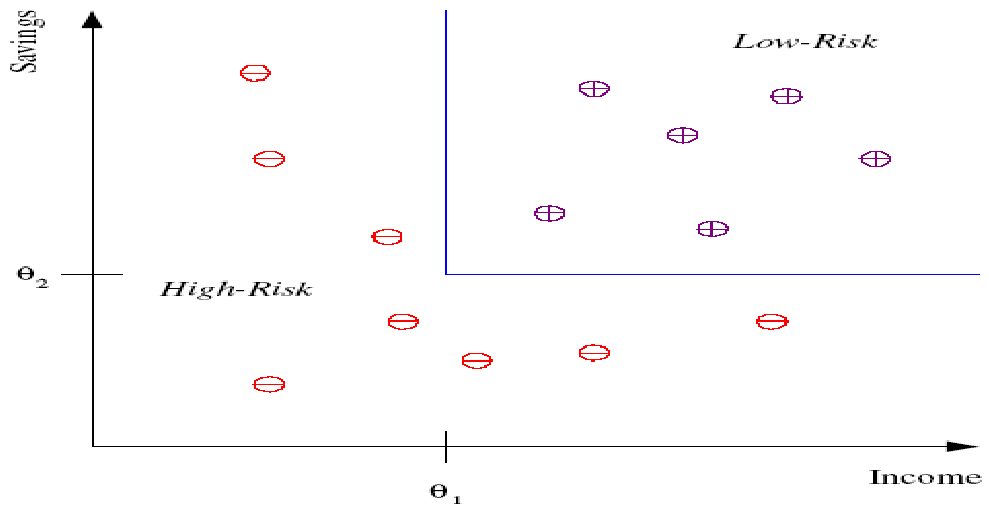
  P (*Y* | *X*) probability that somebody who buys *X* also buys *Y* where *X* and *Y* are products/services.

  Example: *P* (chips | beer) = 0.7 // 70 percent of customers who buy beer also buy chips.

We may want to make a distinction among customers and toward this, estimate P(Y|X,D) where D is the set of customer attributes, for example, gender, age, marital status, and so on, assuming that we have access to this information. If this is a bookseller instead of a supermarket, products can be books or authors. In the case of a Web portal, items correspond to links to Web pages, and we can estimate the links a user is likely to click and use this information to download such pages in advance for faster access.

Classification:

- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their income and savings
- Discriminant: IF income > θ1 AND savings > θ2 THEN low-risk ELSE high-risk



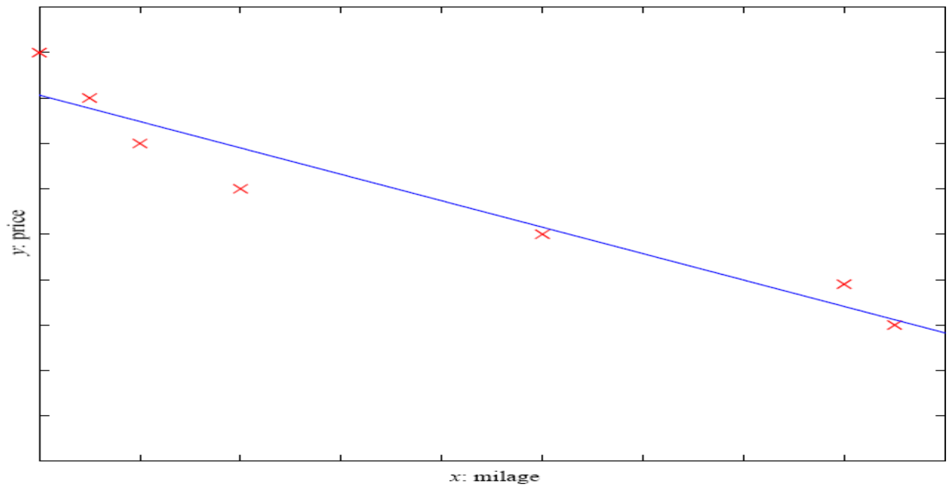
Prediction - Regression:

- Example: Predict Price of a used car
- x : car attributes

  y : price

  $$y = g(x \mid \theta)$$

  where, *g* ( ) is the model, θ are the parameters



A training dataset of used cars and the function fitted. For simplicity, mileage is taken as the only input attribute and a linear model is used.

Supervised Learning:

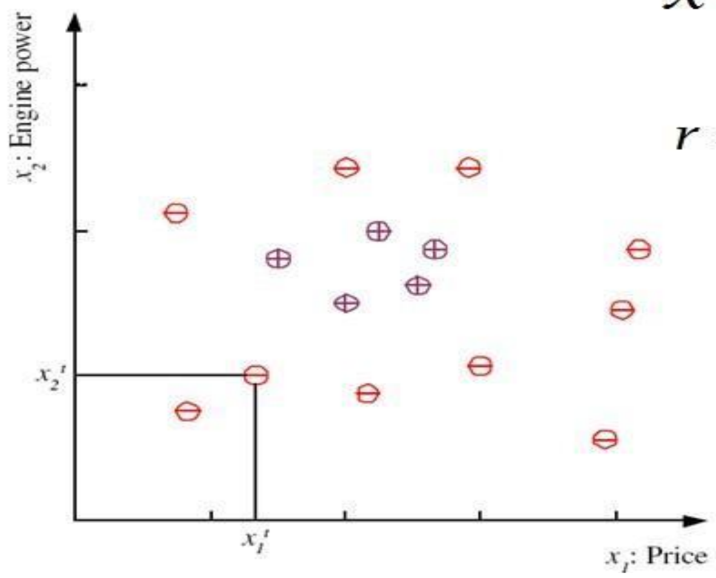
Learning a Class from Examples:

- Class C of a “family car”
  - Prediction: Is car x a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

  Positive (+) and negative (–) examples
- Input representation:

  x_1: price, x_2 : engine power

Training set $\mathcal{X}$



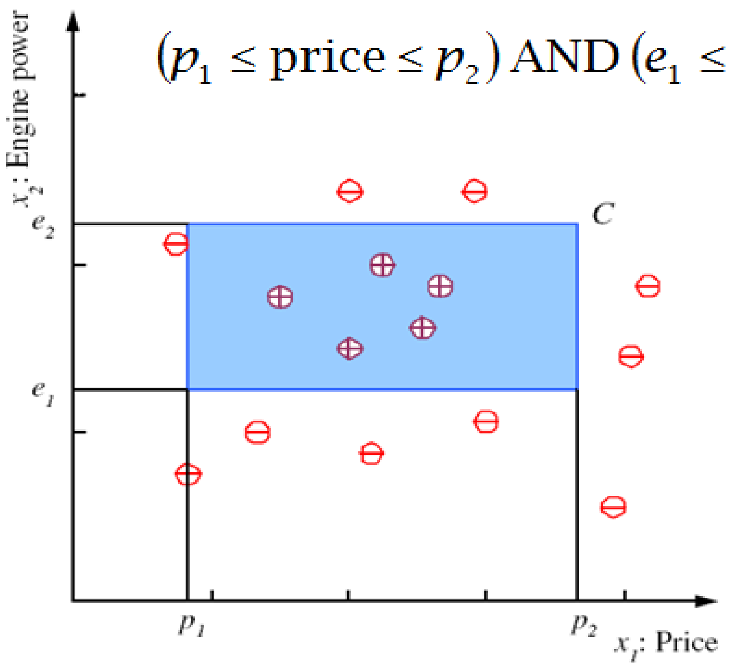
$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$$

$$r = \begin{cases} 1 \text{ if } \boldsymbol{x} \text{ is positive} \\ 0 \text{ if } \boldsymbol{x} \text{ is negative} \end{cases}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

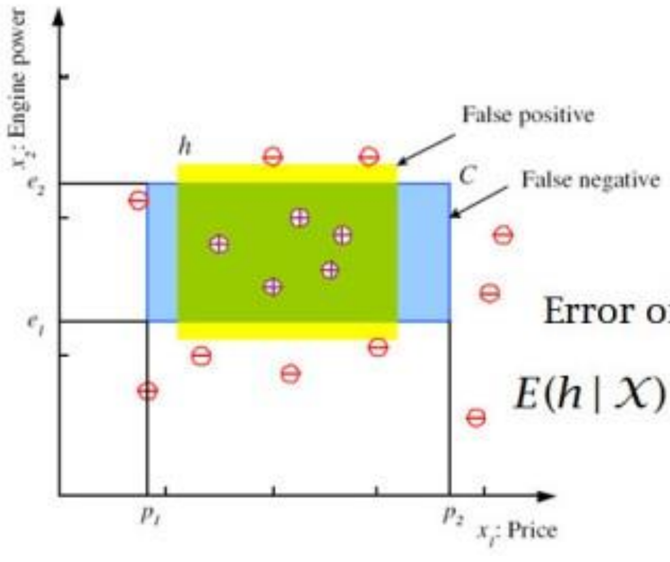
Class C

$$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$$



Hypothesis class $\mathcal{H}$

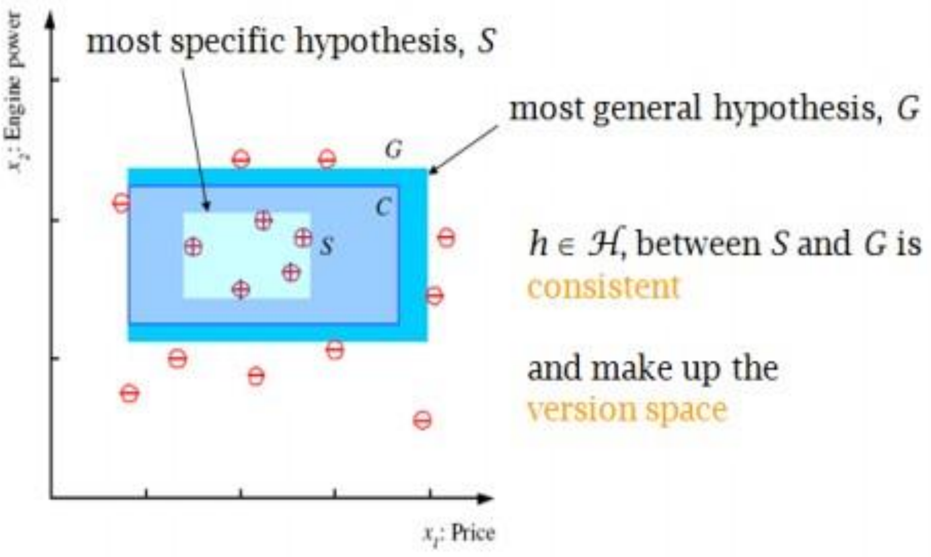
$$h(\boldsymbol{x}) = \begin{cases} 1 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as positive} \\ 0 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as negative} \end{cases}$$



Error of h on $\mathcal{H}$

$$E(h \mid \mathcal{X}) = \sum_{t=1}^{N} 1\left(h\left(\boldsymbol{x}^t\right) \neq r^t\right)$$

S, G, and the Version Space



$h \in \mathcal{H}$, between S and G is consistent

and make up the version space

Probably Approximately Correct (PAC):

- Cannot expect a learner to learn a concept exactly.
- Cannot always expect to learn a close approximation to the target concept
- Therefore, the only realistic expectation of a good learner is that with high probability it will learn a close approximation to the target concept.
- In Probably Approximately Correct (PAC) learning, one requires that given small parameters ε and δ, with probability at least (1- δ) a learner produces a hypothesis with error at most ε.
- The reason we can hope for that is the Consistent Distribution assumption.

PAC Learnability

- Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H.
- C is PAC learnable by L using H if
  - for all f ϵ C,
  - for all distributions D over X, and fixed 0< ε , δ < 1,
- L, given a collection of m examples sampled independently according to D produces
  - with probability at least (1- δ) a hypothesis h ϵ H with error at most ε, ($Error_D = Pr_D[f(x) : = h(x)]$)
- where m is polynomial in 1/ ε, 1/ δ, n and size(H)
- C is efficiently learnable if L can produce the hypothesis in time polynomial in 1/ ε, 1/ δ, n and size(H)
- We impose two limitations:
- Polynomial sample complexity (information theoretic constraint)
  - Is there enough information in the sample to distinguish a hypothesis h that approximate f ?
- Polynomial time complexity (computational complexity)
  - Is there an efficient algorithm that can process the sample and produce a good hypothesis h ?

- To be PAC learnable, there must be a hypothesis h □ H with arbitrary small error for every f ∈ C. We generally assume H ⊇ C. (Properly PAC learnable if H=C)

Occam's Razor:

Claim: The probability that there exists a hypothesis $h \in H$ that
(1) is consistent with m examples and
(2) satisfies error(h) > ε ($\text{Error}_D(h) = \text{Pr}_{x \in D}[f(x) \neg= h(x)]$)
is less than $|H|(1-\varepsilon)^m$.

Proof: Let h be such a bad hypothesis.
- The probability that h is consistent with one example of f is

$$\Pr_{x \in D}[f(x) = h(x)] < 1 - \varepsilon$$

- Since the m examples are drawn independently of each other,
The probability that h is consistent with m example of f is less than $(1-\varepsilon)^m$

- The probability that *some* hypothesis in H is consistent with m examples is less than $|H|(1-\varepsilon)^m$

We want this probability to be smaller than δ, that is:

$$|H|(1-\varepsilon)^m < \delta$$

$$\ln(|H|) + m \ln(1-\varepsilon) < \ln(\delta)$$

(with $e^{-x} = 1-x+x^2/2+\dots$; $e^{-x} > 1-x$; $\ln(1-\varepsilon) < -\varepsilon$; gives a safer δ)

$$\boxed{m > \frac{1}{\varepsilon}\{\ln(|H|) + \ln(1/\delta)\}}$$

(gross over estimate)
It is called Occam's razor, because it indicates a preference towards small hypothesis spaces

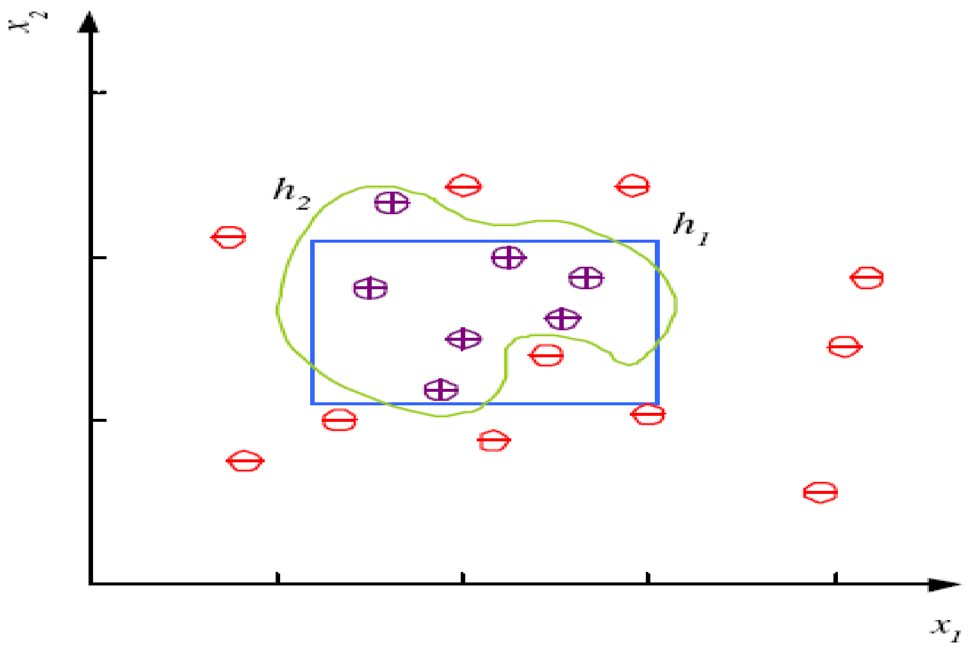
Noise and Model Complexity:

Noise is any unwanted anomaly in the data and due to noise, the class may be more difficult to learn and zero error may be infeasible with a simple hypothesis class. There are several interpretations of noise:

- There may be imprecision in recording the input attributes, which may shift the data points in the input space.
- There may be errors in labeling the data points, which may relabel positive instances as negative and vice versa. This is sometimes called teacher noise.
- There may be additional attributes, which we have not taken into account, that affect the label of an instance. Such attributes may be hidden or latent in that they may be unobservable. The effect of these neglected attributes is thus modeled as a random component and is included in "noise."

Use the simpler one because

- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower variance - Occam's razor)



Learning Multiple Classes:

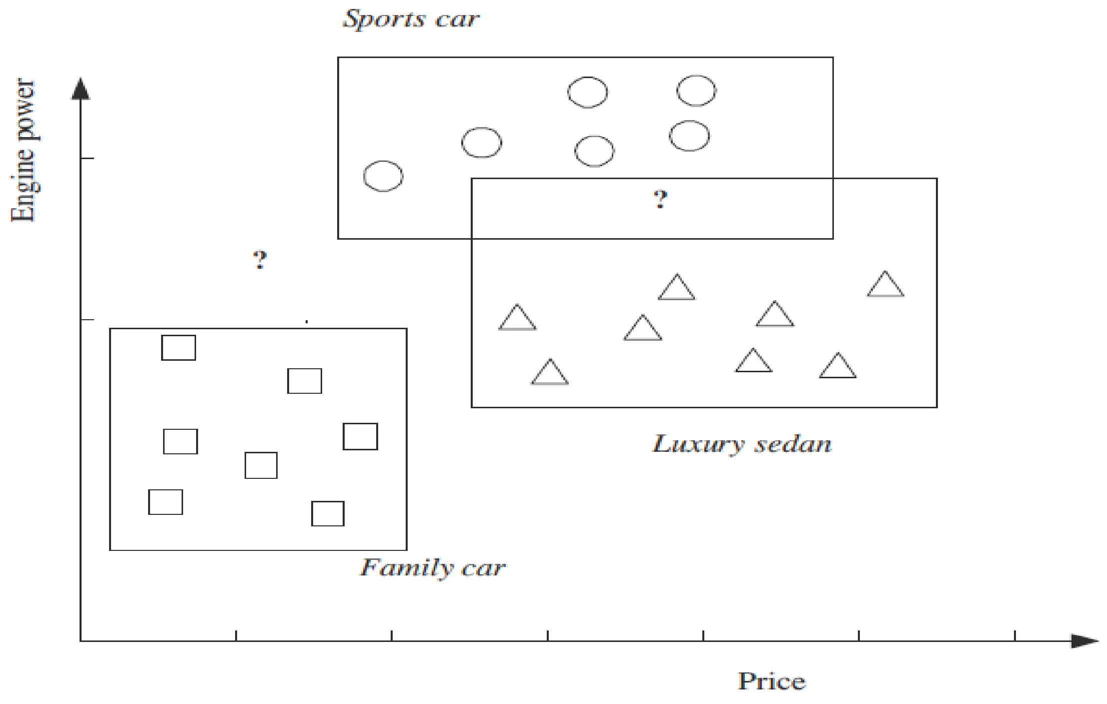
In our example of learning a family car, we have positive examples belonging to the class family car and the negative examples belonging to all other cars. This is a *two-class* problem. In the general case, we have *K* classes denoted as C*i*, *i* = 1, . . . , *K*, and an input instance belongs to one and exactly one of them. The training set is now of the form,

$$\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^{N}$$

$$r_i^t = \begin{cases} 1 \text{ if } \boldsymbol{x}^t \in C_i \\ 0 \text{ if } \boldsymbol{x}^t \in C_j,\ j \neq i \end{cases}$$

Train hypotheses
$h_i(\boldsymbol{x})$, $i = 1, \dots, K$:

$$h_i(\boldsymbol{x}^t) = \begin{cases} 1 \text{ if } \boldsymbol{x}^t \in C_i \\ 0 \text{ if } \boldsymbol{x}^t \in C_j,\ j \neq i \end{cases}$$



There are three classes: family car, sports car, and luxury sedan. There are three hypotheses induced, each one covering the instances of one class and leaving outside the instances of the other two classes .'?' are reject regions where no, or more than one, class is chosen.

Model Selection & Generalization:

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than C or f
- Underfitting: H less complex than C or f

Triple Trade-Off:

- There is a trade-off between three factors (Dietterich, 2003):
    1. Complexity of H, *c* (H),
    2. Training set size, *N*,
    3. Generalization error, *E*, on new data

       As *N* increases, E decreasesAs *c* (H), first *E* decreases and then *E* increases

Cross-Validation:

- To estimate generalization error, we need data unseen during training. We split the data as
    - Training set (50%)
    - Validation set (25%)
    - Test (publication) set (25%)
- Resampling when there is few data

Dimensions of a Supervised Learner:

- Model : $g(\mathbf{x} \mid \theta)$
- Loss function: $E(\theta \mid \mathcal{X}) = \sum_t L(r^t, g(\mathbf{x}^t \mid \theta))$
- Optimization procedure:

$$\theta^* = \arg\min_\theta E(\theta \mid \mathcal{X})$$

Applications of Machine learning

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:

1. Image Recognition:

Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, **Automatic friend tagging suggestion**:

Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's **face detection** and **recognition algorithm**.

It is based on the Facebook project named "**Deep Face**," which is responsible for face recognition and person identification in the picture.

2. Speech Recognition

While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google assistant**, **Siri**, **Cortana**, and **Alexa** are using speech recognition technology to follow the voice instructions.

3. Traffic prediction:

If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

- **Real Time location** of the vehicle form Google Map app and sensors
- **Average time has taken** on past days at the same time.

Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4. Product recommendations:

Machine learning is widely used by various e-commerce and entertainment companies such as **Amazon**, **Netflix**, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.

As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

5. Self-driving cars:

One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering:

Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

Some machine learning algorithms such as **Multi-Layer Perceptron**, **Decision tree**, and **Naïve Bayes classifier** are used for email spam filtering and malware detection.

7. Virtual Personal Assistant:

We have various virtual personal assistants such as **Google assistant**, **Alexa**, **Cortana**, **Siri**. As the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.

These virtual assistants use machine learning algorithms as an important part.

These assistant record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

8. Online Fraud Detection:

Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as **fake accounts**, **fake ids**, and **steal money** in the middle of a transaction. So to detect this, **Feed Forward Neural network** helps us by checking whether it is a genuine transaction or a fraud transaction.

For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

9. Stock Market trading:

Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's **long short term memory neural network** is used for the prediction of stock market trends.

10. Medical Diagnosis:

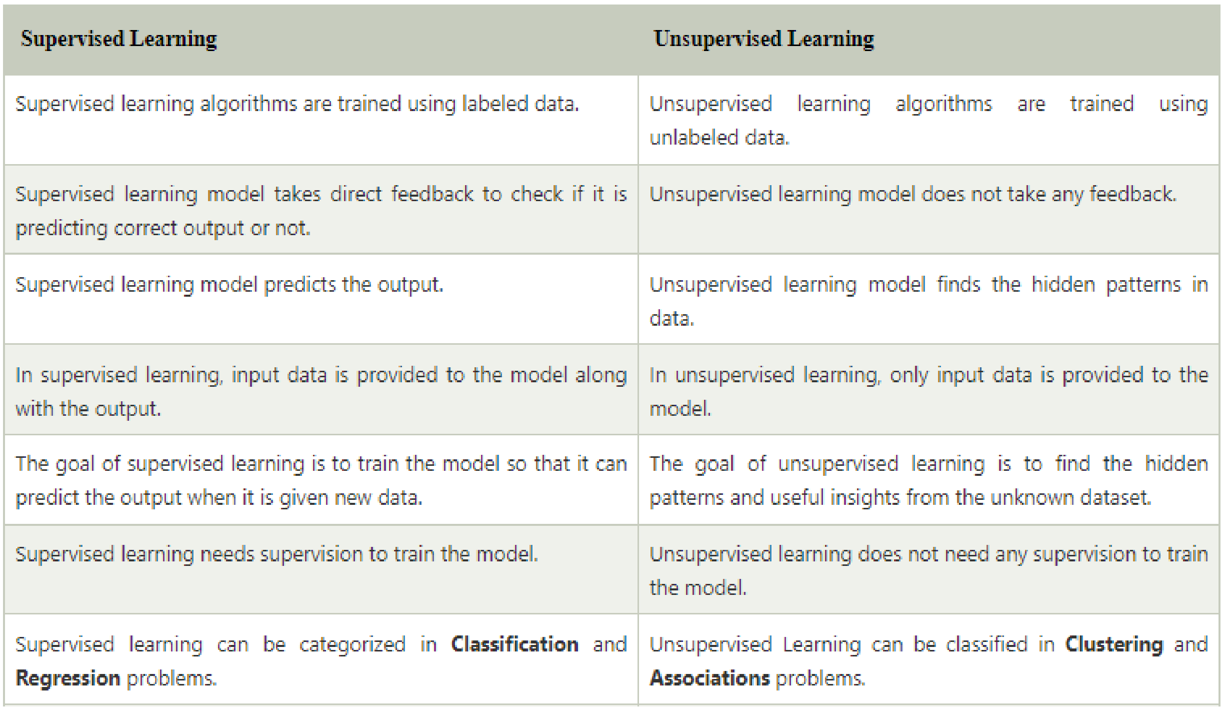
In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain.

It helps in finding brain tumors and other brain-related diseases easily.
11. Automatic Language Translation:

Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.

The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

Supervised learning Vs Unsupervised Learning



Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** problems.	Unsupervised Learning can be classified in **Clustering** and **Associations** problems.

Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

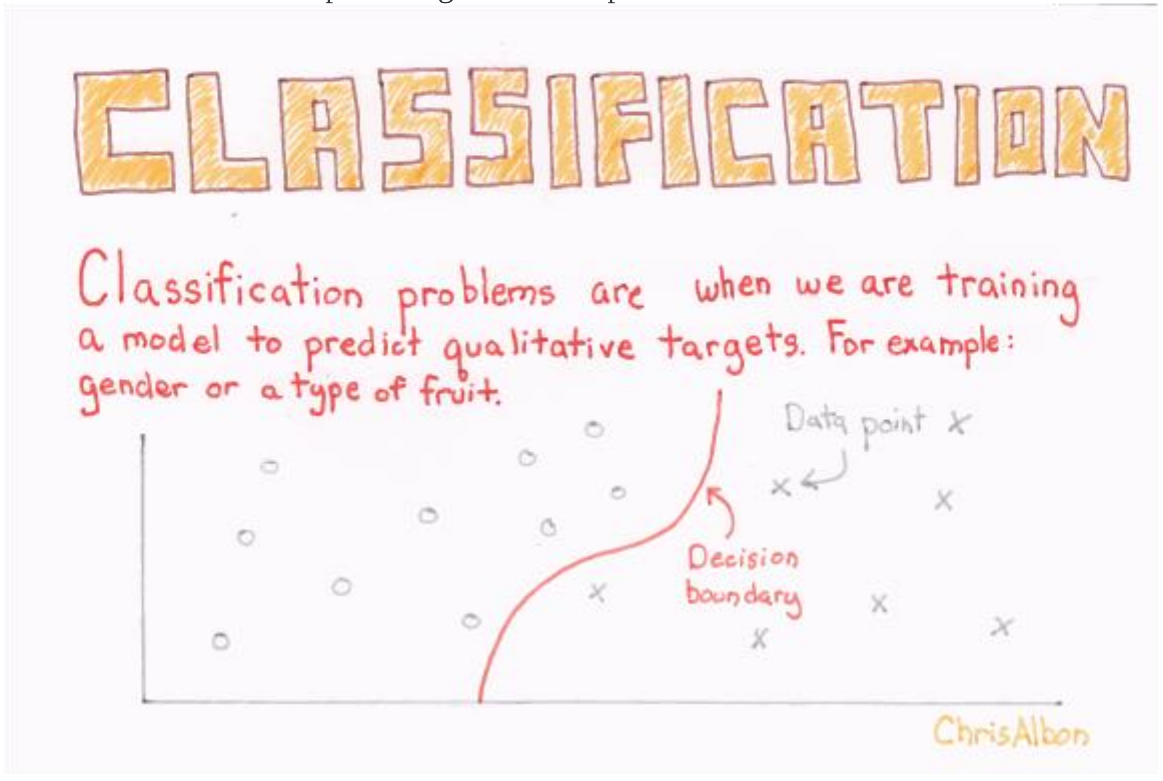
Supervised Learning

In supervised learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data by associating patterns to the unlabeled new data.
Supervised learning can be divided into two categories: classification and regression.

Classification

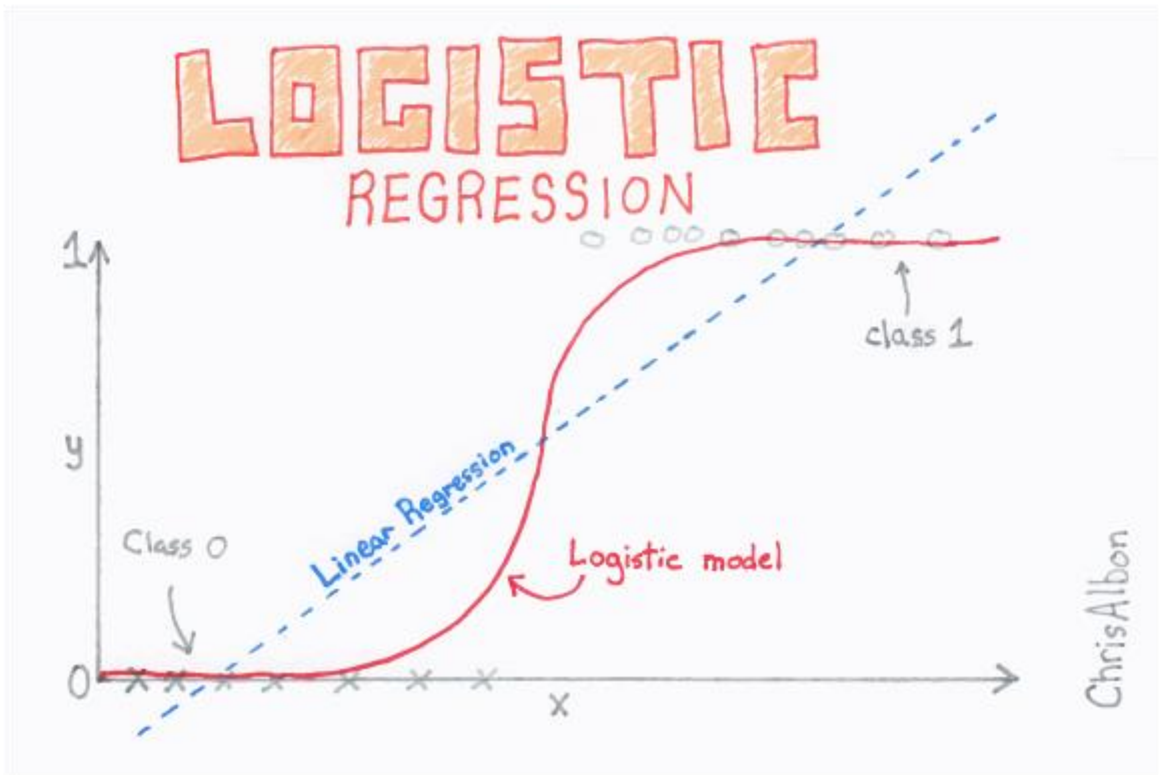
Classification is a technique for determining which class the dependent belongs to based on one or more independent variables.

Classification is used for predicting discrete responses.


CLASSIFICATION
Classification problems are when we are training a model to predict qualitative targets. For example: gender or a type of fruit.
Data point
Decision boundary
ChrisAlbon

1. LOGISTIC REGRESSION

Logistic regression is kind of like linear regression, but is used when the dependent variable is not a number but something else (e.g., a "yes/no" response). It's called regression but performs classification based on the regression and it classifies the dependent variable into either of the classes.



Logistic regression is used for prediction of output which is binary, as stated above. For example, if a credit card company builds a model to decide whether or not to issue a credit card to a customer, it will model for whether the customer is going to "default" or "not default" on their card.

$$y = b_0 + b_1 x$$

Linear Regression

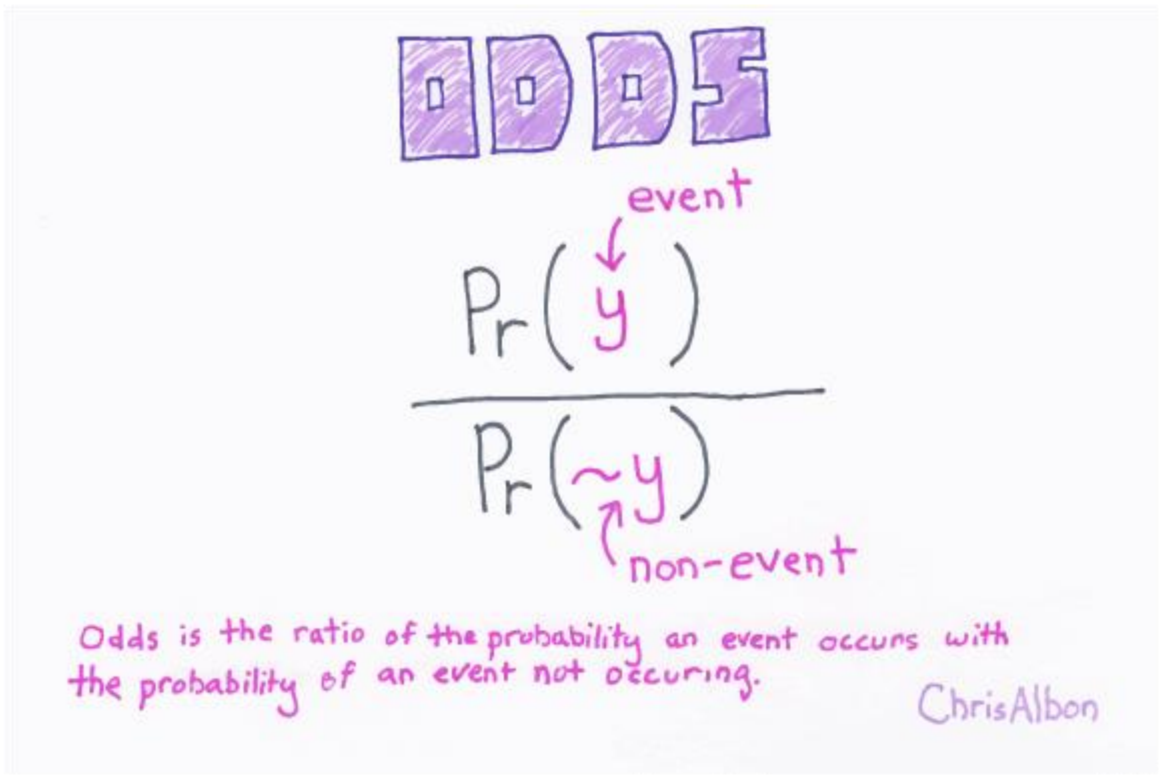
Firstly, linear regression is performed on the relationship between variables to get the model. The threshold for the classification line is assumed to be at 0.5.

$$p = \frac{1}{1 + e^{-y}}$$

Logistic Sigmoid Function

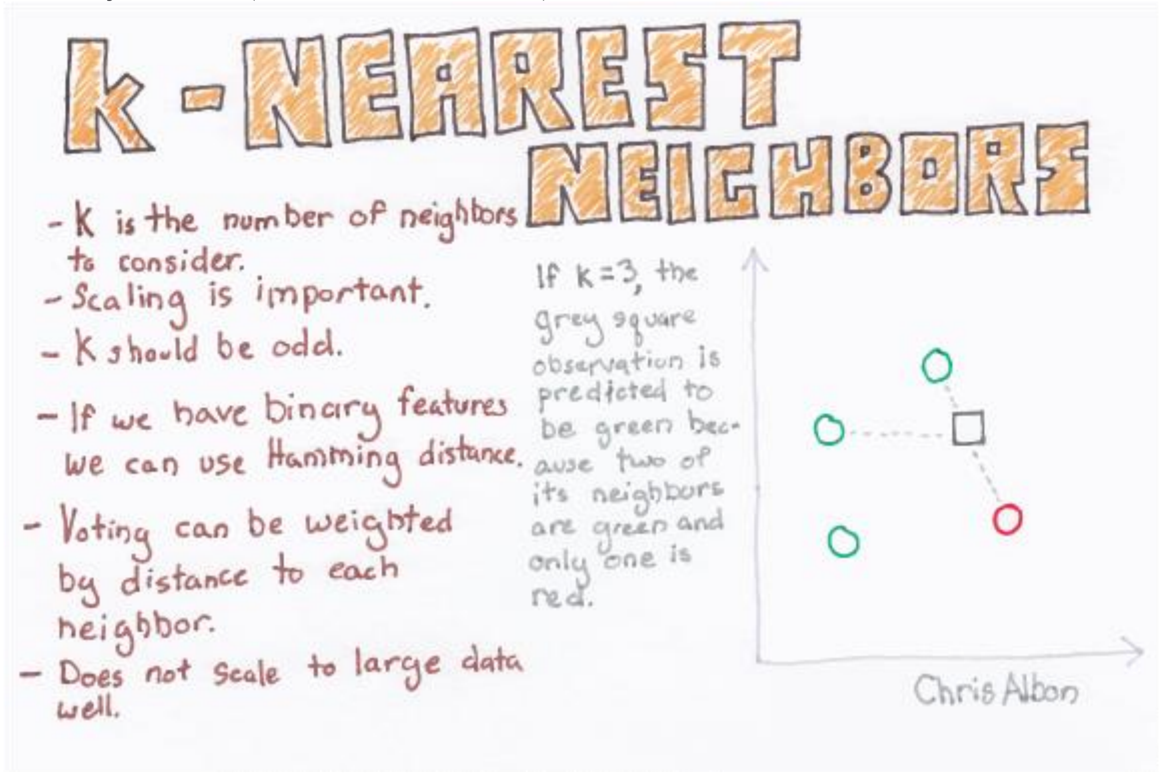
Logistic function is applied to the regression to get the probabilities of it belonging in either class.

It gives the log of the probability of the event occurring to the log of the probability of it not occurring. In the end, it classifies the variable based on the higher probability of either class.


ODDS
event
Pr(y)
Pr(~y)
non-event
Odds is the ratio of the probability an event occurs with the probability of an event not occuring.
Chris Albon

2. K-NEAREST NEIGHBORS (K-NN)

K-NN algorithm is one of the simplest classification algorithms and it is used to identify the data points that are separated into several classes to predict the classification of a new sample point. K-NN is a non-parametric, lazy learning algorithm. It classifies new cases based on a similarity measure (i.e., distance functions).


k-NEAREST NEIGHBORS
- K is the number of neighbors to consider.
- Scaling is important.
- K should be odd.
- If we have binary features we can use Hamming distance.
- Voting can be weighted by distance to each neighbor.
- Does not scale to large data well.
If k=3, the grey square observation is predicted to be green because two of its neighbors are green and only one is red.
Chris Albon

DOES K-NN LEARN

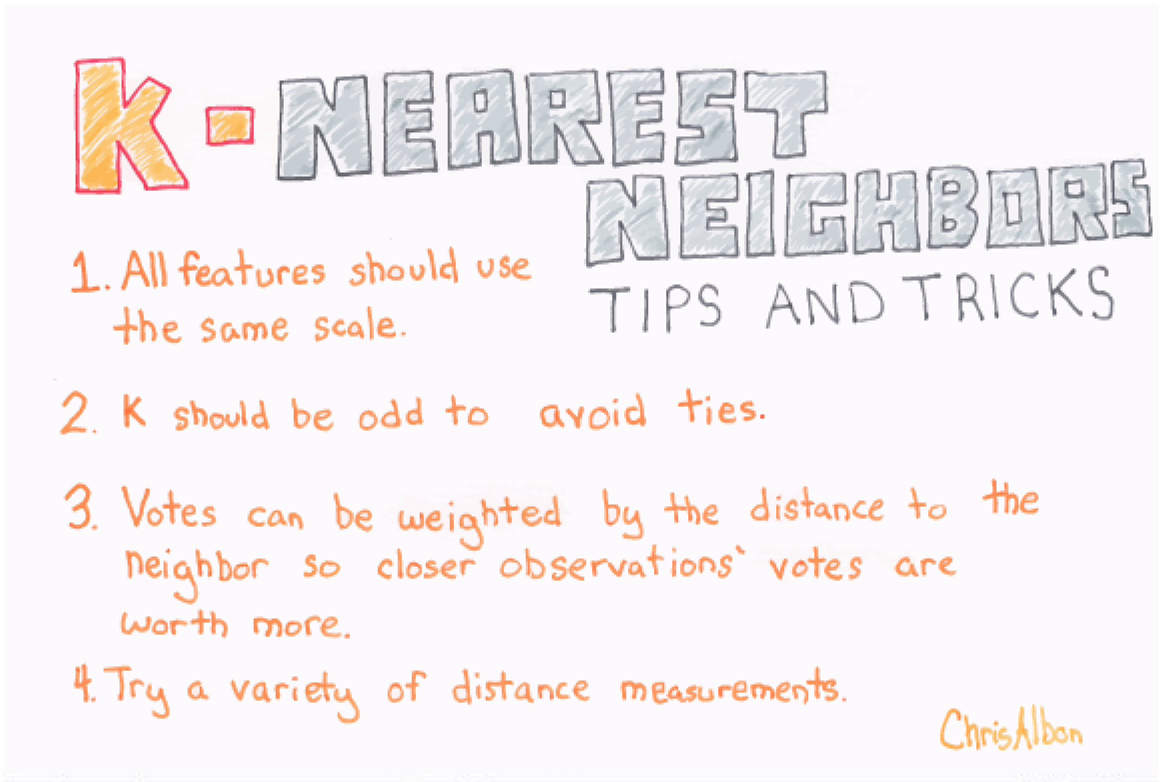
K-nearest neighor does not "learn" per-se. It is lazy and just memorizes the data.

Chris Albon

DOES K-NN LEARN

K-nearest neighor does not "learn" per-se. It is lazy and just memorizes the data.

Chris Albon



K-NEAREST NEIGHBORS

TIPS AND TRICKS

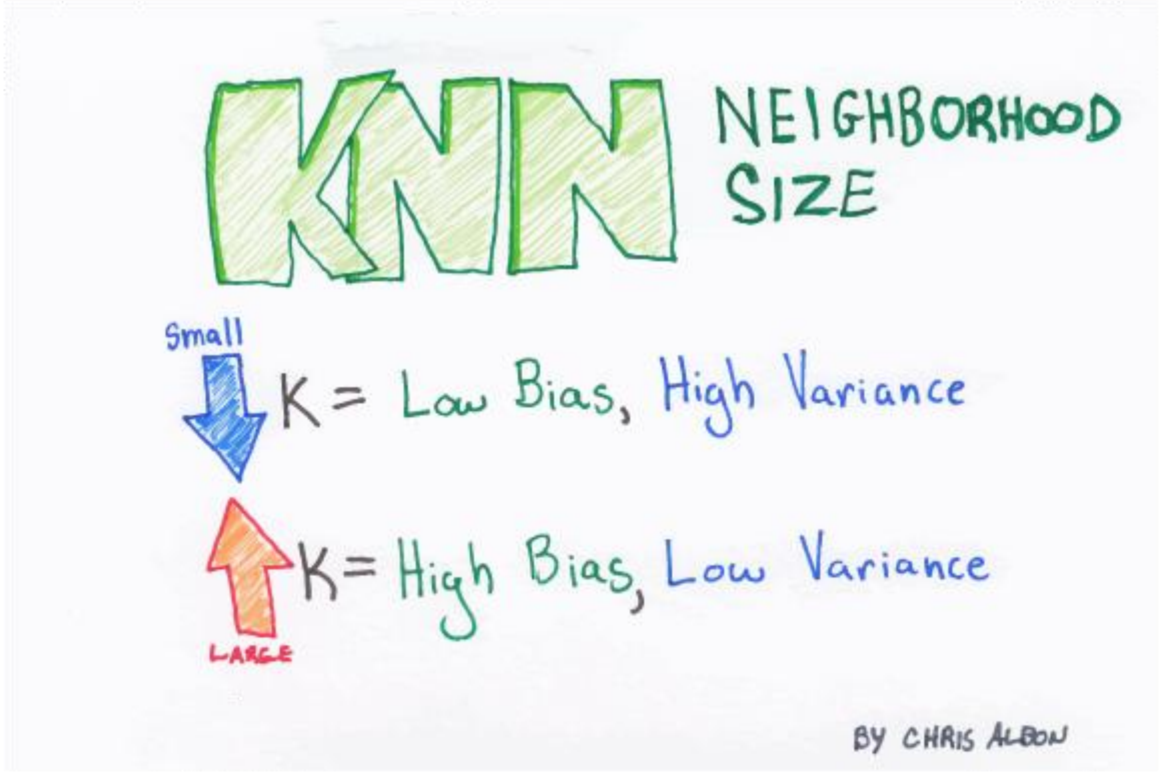
1. All features should use the same scale.

2. K should be odd to avoid ties.

3. Votes can be weighted by the distance to the neighbor so closer observations' votes are worth more.

4. Try a variety of distance measurements.

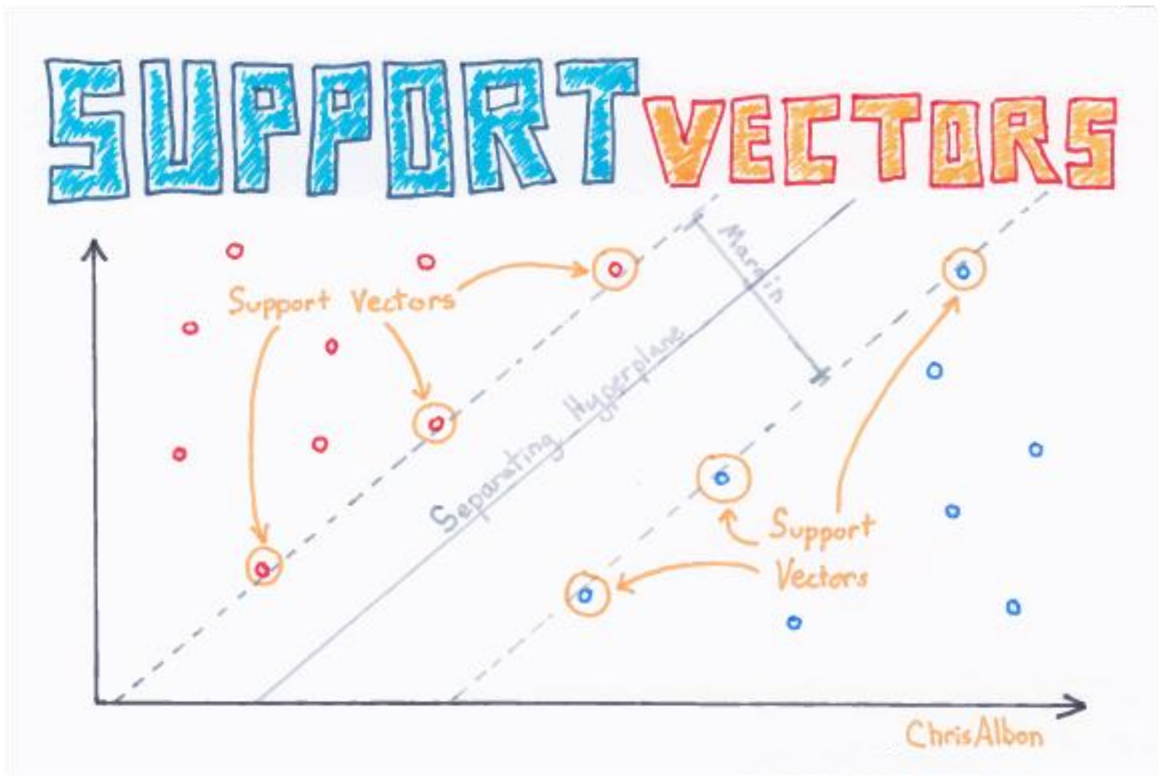
ChrisAlbon



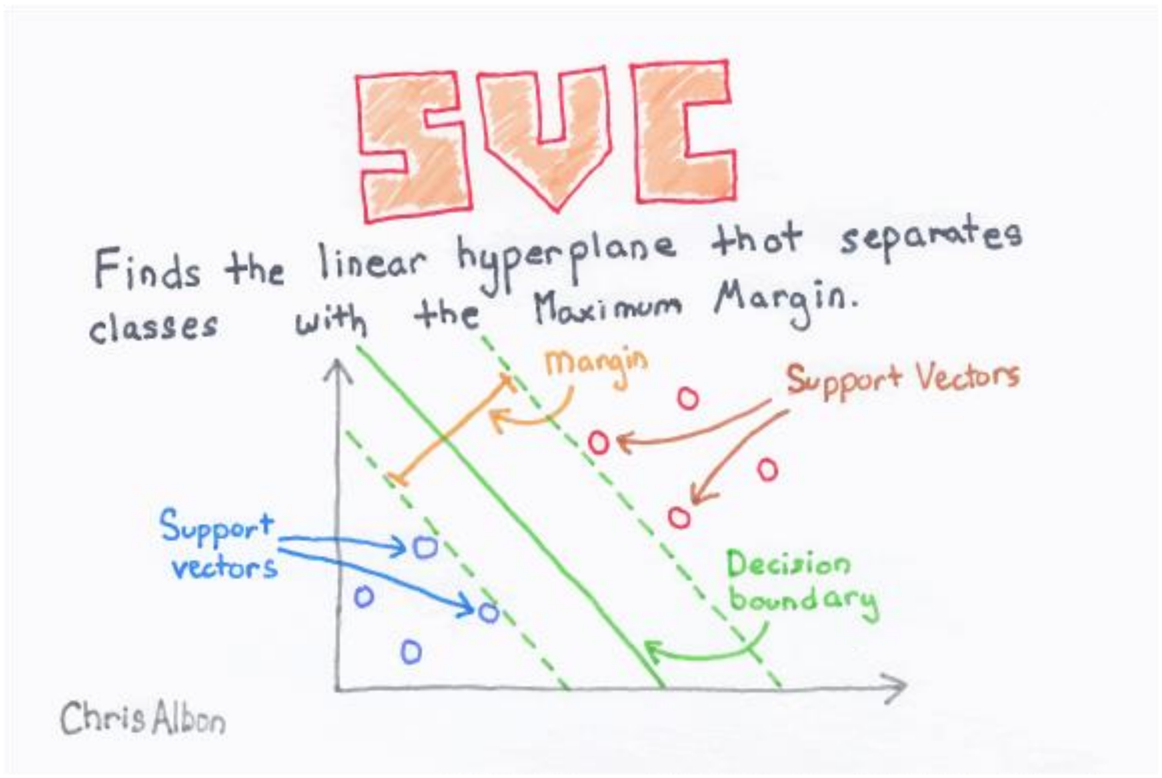
K-NN works well with a small number of input variables (*p*), but struggles when the number of inputs is very large.

3. SUPPORT VECTOR MACHINE (SVM)

Support vector is used for both regression and classification. It is based on the concept of decision planes that define decision boundaries. A decision plane (hyperplane) is one that separates between a set of objects having different class memberships.



It performs classification by finding the hyperplane that maximizes the margin between the two classes with the help of support vectors.


SVC
Finds the linear hyperplane that separates classes with the Maximum Margin.
Margin
Support Vectors
Support vectors
Decision boundary
Chris Albon

The learning of the hyperplane in SVM is done by transforming the problem using some linear algebra (i.e., the example above is a linear kernel which has a linear separability between each variable).

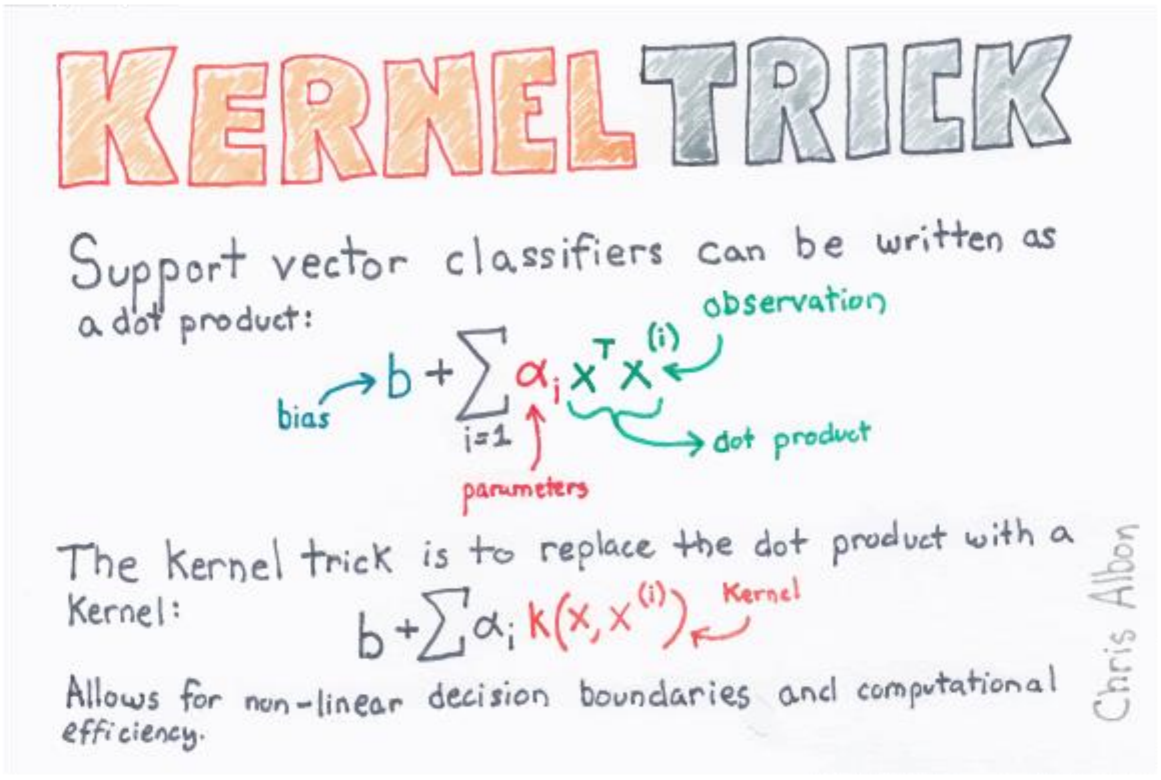
For higher dimensional data, other kernels are used as points and cannot be classified easily. They are specified in the next section.

Kernel SVM
Kernel SVM takes in a kernel function in the SVM algorithm and transforms it into the required form that maps data on a higher dimension which is separable.
Types of kernel function are:

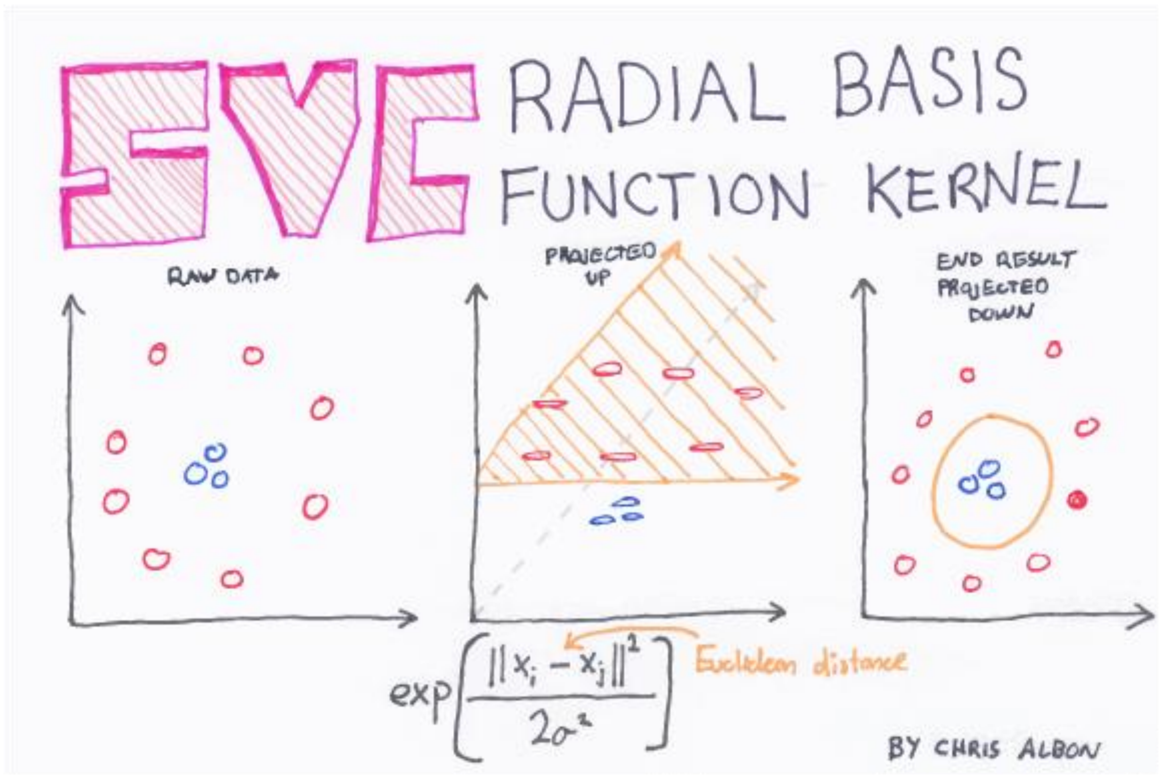
$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma \,|\, \mathbf{X}_i - \mathbf{X}_j \,|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

Type of kernel functions

1. Linear SVM is the one we discussed earlier.
2. In polynomial kernel, the degree of the polynomial should be specified. It allows for curved lines in the input space.
3. In the radial basis function (RBF) kernel, it is used for non-linearly separable variables. For distance, metric squared Euclidean distance is used. Using a typical value of the parameter can lead to overfitting our data. It is used by default in sklearn.
4. Sigmoid kernel, similar to logistic regression is used for binary classification.


KERNEL TRICK
Support vector classifiers can be written as a dot product:
observation
bias
$b + \sum_{i=1} \alpha_i x^T x^{(i)}$
dot product
parameters
The kernel trick is to replace the dot product with a kernel:
$b + \sum \alpha_i k(x, x^{(i)})$
Kernel
Allows for non-linear decision boundaries and computational efficiency.
Chris Albon

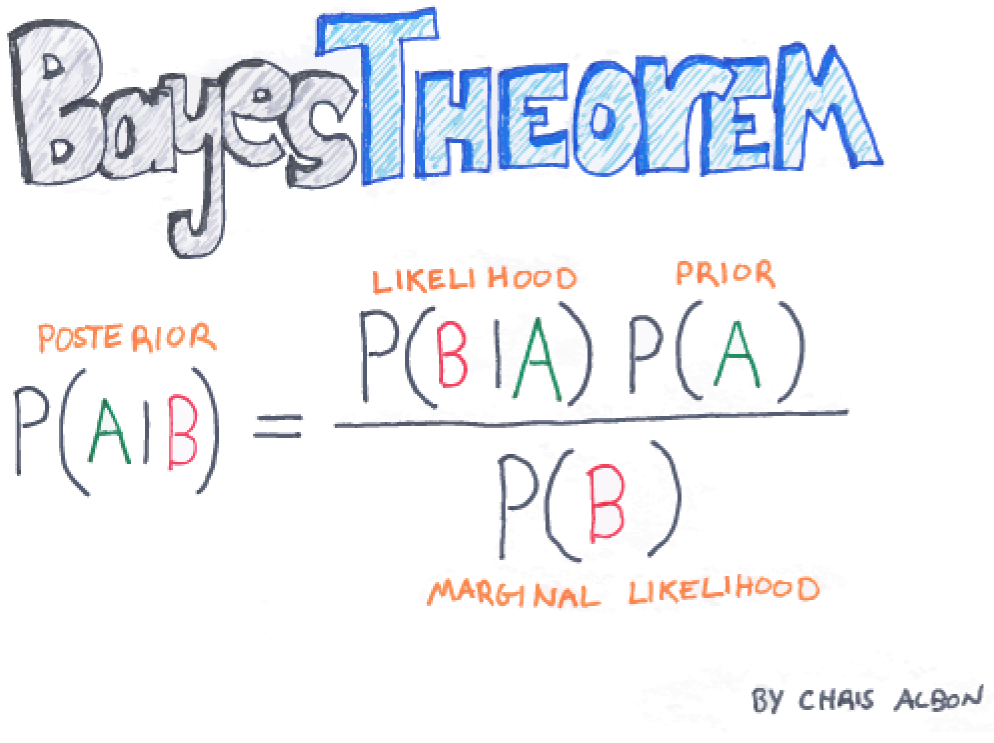
Kernel trick uses the kernel function to transform data into a higher dimensional feature space and makes it possible to perform the linear separation for classification.
Radial Basis Function (RBF) Kernel
The RBF kernel SVM decision region is actually also a linear decision region. What RBF kernel SVM actually does is create non-linear combinations of features to uplift the samples onto a higher-dimensional feature space where a linear decision boundary can be used to separate classes.


SVC
RADIAL BASIS FUNCTION KERNEL
RAW DATA
PROJECTED UP
END RESULT PROJECTED DOWN
$\exp\left[\frac{\|x_i - x_j\|^2}{2\sigma^2}\right]$
Euclidean distance
BY CHRIS ALBON

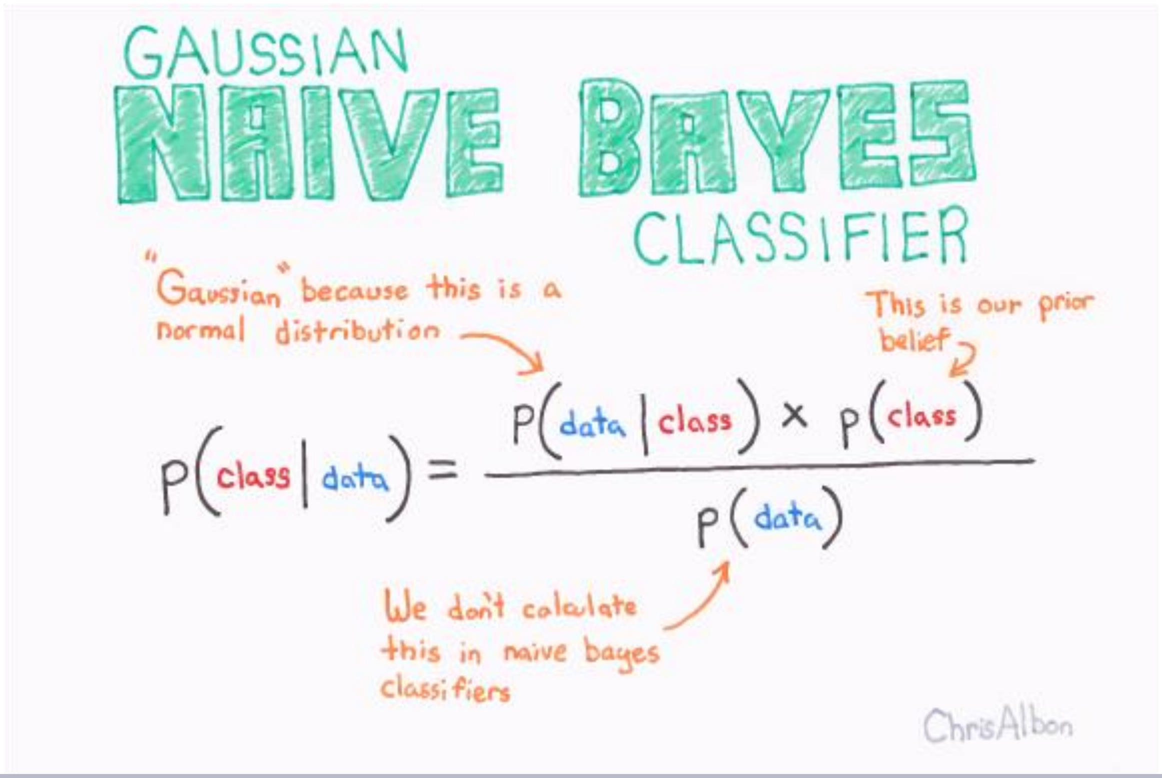
So, the rule of thumb is: use linear SVMs for linear problems, and nonlinear kernels such as the RBF kernel for non-linear problems.

4. NAIVE BAYES

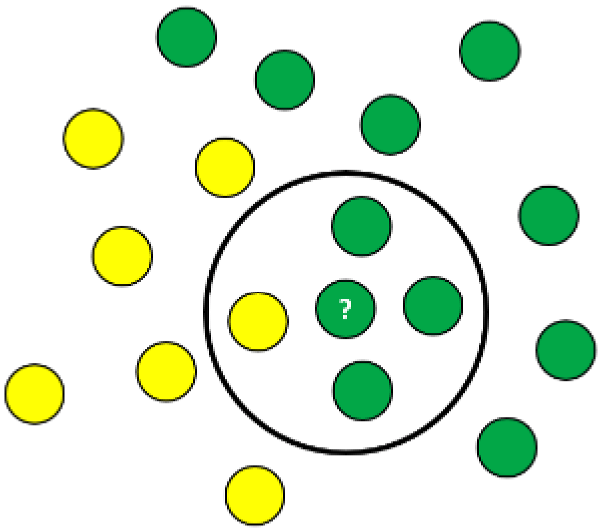
The naive Bayes classifier is based on Bayes' theorem with the independence assumptions between predictors (i.e., it assumes the presence of a feature in a class is unrelated to any other feature). Even if these features depend on each other, or upon the existence of the other features, all of these properties independently. Thus, the name naive Bayes.



Based on naive Bayes, Gaussian naive Bayes is used for classification based on the binomial (normal) distribution of data.


GAUSSIAN
NAIVE BAYES
CLASSIFIER
"Gaussian" because this is a normal distribution
This is our prior belief
P(class | data) = P(data | class) × p(class) / P(data)
We don't calculate this in naive bayes classifiers
ChrisAlbon

- P(class|data) is the posterior probability of class(target) given predictor(attribute). The probability of a data point having either class, given the data point. This is the value that we are looking to calculate.
- P(class) is the prior probability of class.
- P(data|class) is the likelihood, which is the probability of predictor given class.
- P(data) is the prior probability of predictor or marginal likelihood.



NB Classification Example

Steps
1. Calculate Prior Probability
P(class) = Number of data points in the class/Total no. of observations
P(yellow) = 10/17
P(green) = 7/17
2. Calculate Marginal Likelihood
P(data) = Number of data points similar to observation/Total no. of observations
P(?) = 4/17
The value is present in checking both the probabilities.

3. Calculate Likelihood
P(data/class) = Number of similar observations to the class/Total no. of points in the class.
P(?/yellow) = 1/7
P(?/green) = 3/10
4. Posterior Probability for Each Class

$$p(class/data) = \frac{P(data/class) * P(class)}{P(data)}$$

$$P(yellow/?) = \frac{1/7 * 7/17}{4/17} = 0.25$$

$$P(green/?) = \frac{3/10 * 10/17}{4/17} = 0.75$$

5. Classification

$$P(class1/data) > P(class2/data)$$

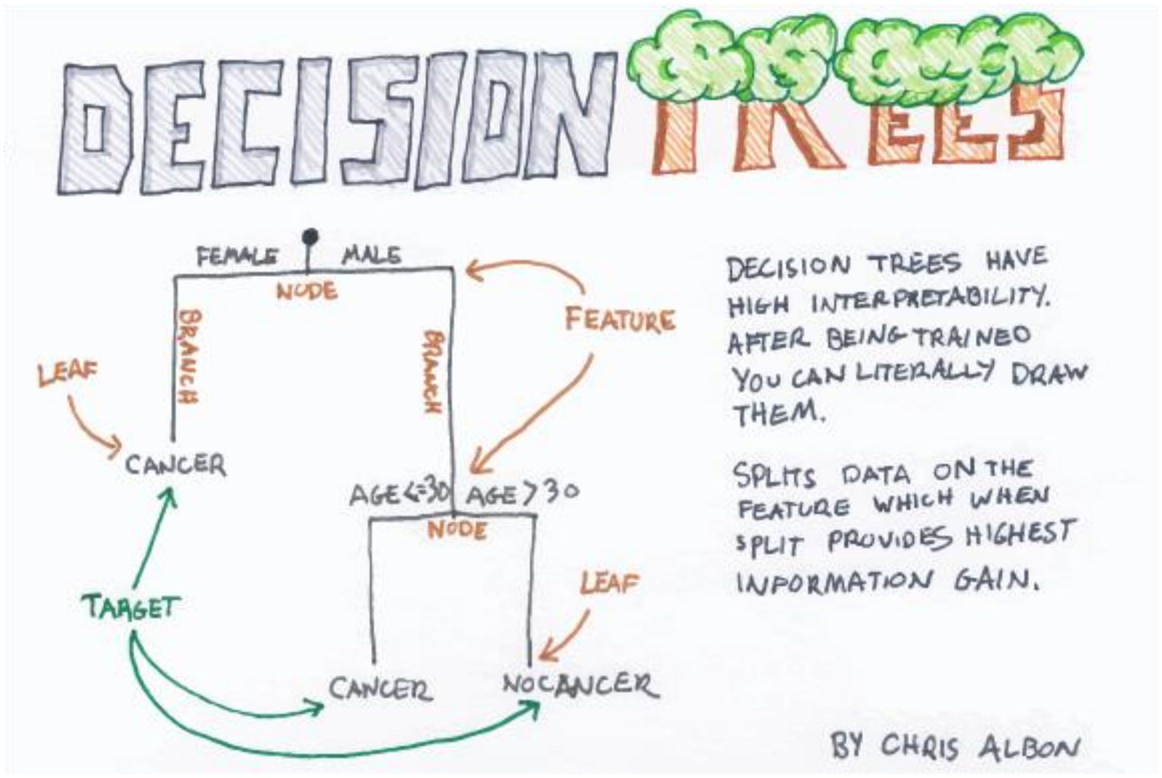
$$P(green/?) > P(yellow/?)$$

The higher probability, the class belongs to that category as from above 75% probability the point belongs to class green.

Multinomial, Bernoulli naive Bayes are the other models used in calculating probabilities. Thus, a naive Bayes model is easy to build, with no complicated iterative parameter estimation, which makes it particularly useful for very large datasets.

5. DECISION TREE CLASSIFICATION

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. It follows Iterative Dichotomiser 3(ID3) algorithm structure for determining the split.


DECISION TREES
FEMALE
MALE
NODE
BRANCH
BRANCH
FEATURE
LEAF
CANCER
AGE<=30
AGE>30
NODE
LEAF
TARGET
CANCER
NOCANCER
DECISION TREES HAVE HIGH INTERPRETABILITY. AFTER BEING TRAINED YOU CAN LITERALLY DRAW THEM.
SPLITS DATA ON THE FEATURE WHICH WHEN SPLIT PROVIDES HIGHEST INFORMATION GAIN.
BY CHRIS ALBON

Entropy and information gain are used to construct a decision tree.

Entropy
Entropy is the degree or amount of uncertainty in the randomness of elements. In other words, it is a measure of impurity**.**

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

Entropy

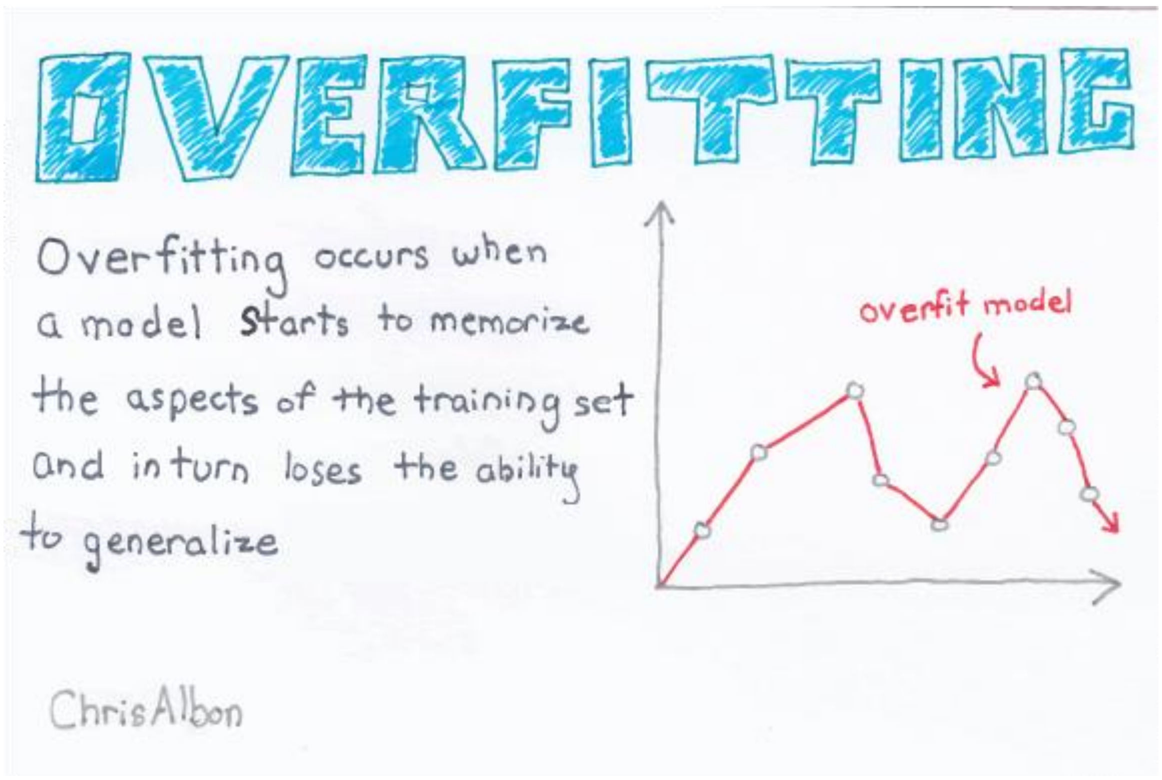
Intuitively, it tells us about the predictability of a certain event. Entropy calculates the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero, and if the sample is equally divided it has an entropy of one.

Information Gain
Information gain measures the relative change in entropy with respect to the independent attribute. It tries to estimate the information contained by each attribute. Constructing a decision tree is all about finding the attribute that returns the highest information gain (i.e., the most homogeneous branches).

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

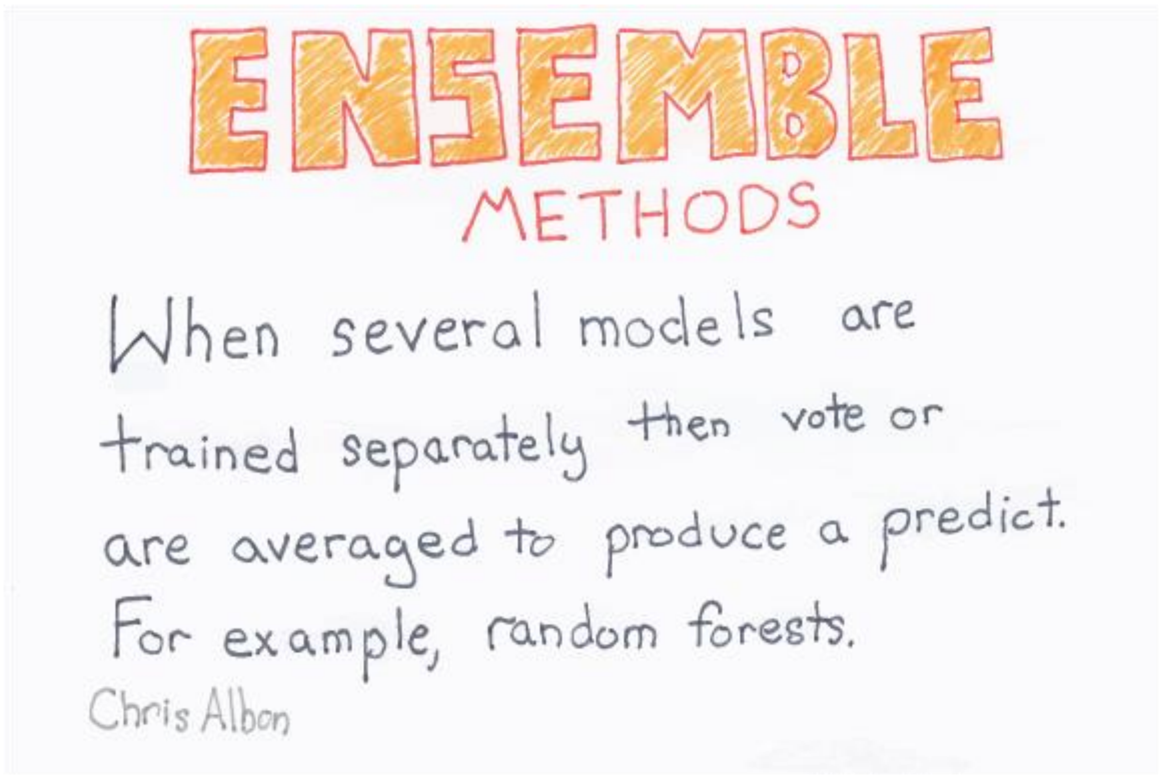
Where *Gain(T, X)* is the information gain by applying feature *X*. *Entropy(T)* is the entropy of the entire set, while the second term calculates the entropy after applying the feature *X*. Information gain ranks attributes for filtering at a given node in the tree. The ranking is based on the highest information gain entropy in each split.

The disadvantage of a decision tree model is overfitting, as it tries to fit the model by going deeper in the training set and thereby reducing test accuracy.


OVERFITTING
Overfitting occurs when a model starts to memorize the aspects of the training set and in turn loses the ability to generalize
overfit model
ChrisAlbon
Overfitting in decision trees can be minimized by pruning nodes.

Ensemble Methods for Classification

An ensemble model is a *team of models*. Technically, ensemble models comprise several supervised learning models that are individually trained and the results merged in various ways to achieve the final prediction. This result has higher predictive power than the results of any of its constituting learning algorithms independently.

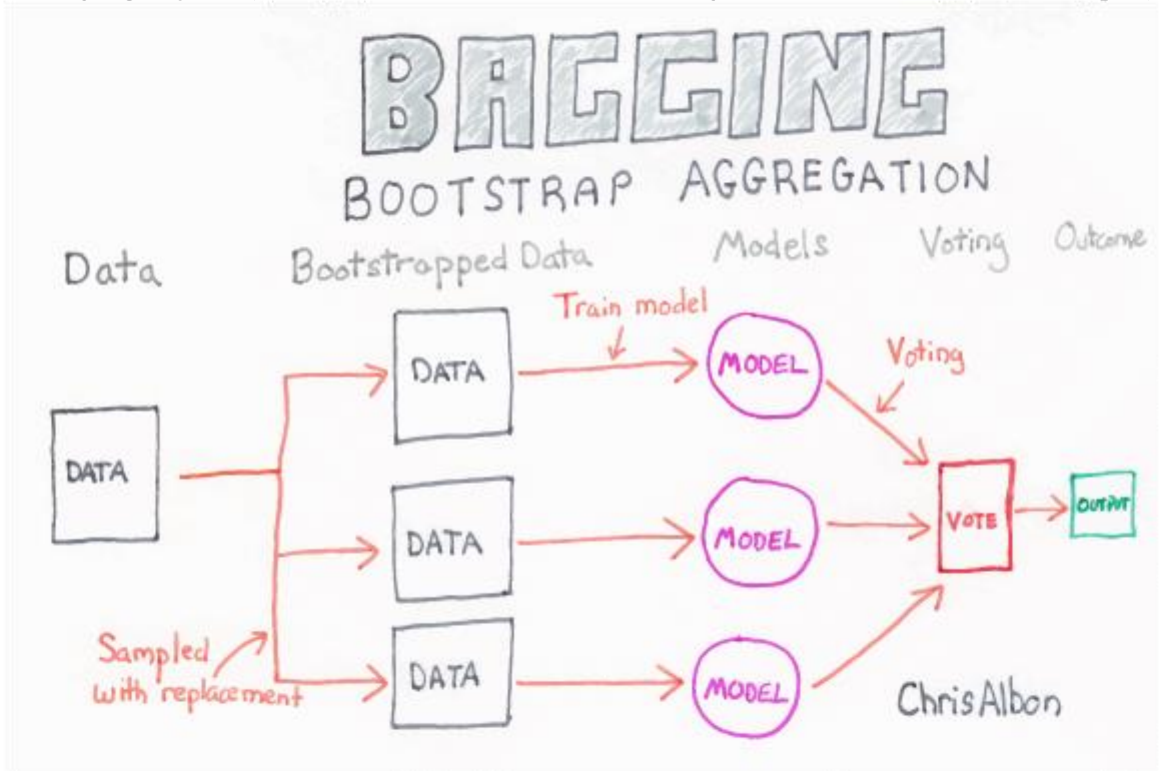

ENSEMBLE
METHODS

When several models are trained separately then vote or are averaged to produce a predict. For example, random forests.

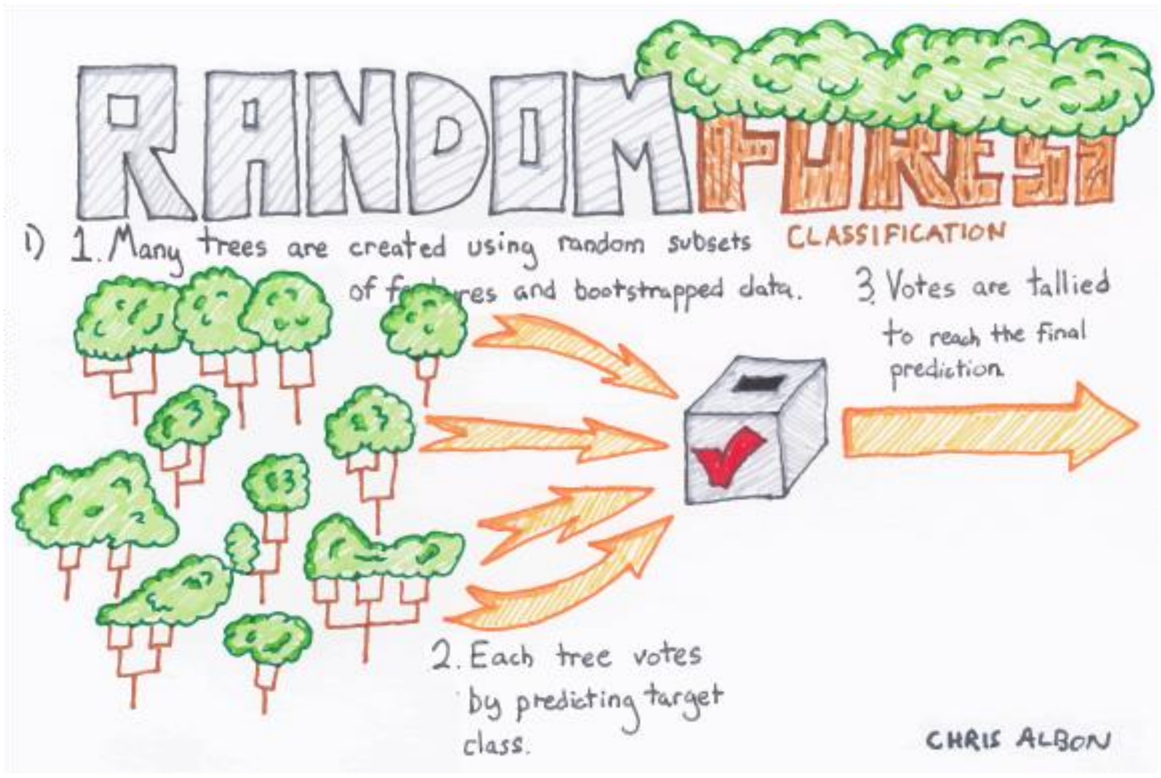
Chris Albon

1. RANDOM FOREST CLASSIFICATION

Random forest classifier is an ensemble algorithm based on bagging i.e bootstrap aggregation. Ensemble methods combines more than one algorithm of the same or different kind for classifying objects (i.e., an ensemble of SVM, naive Bayes or decision trees, for example.)



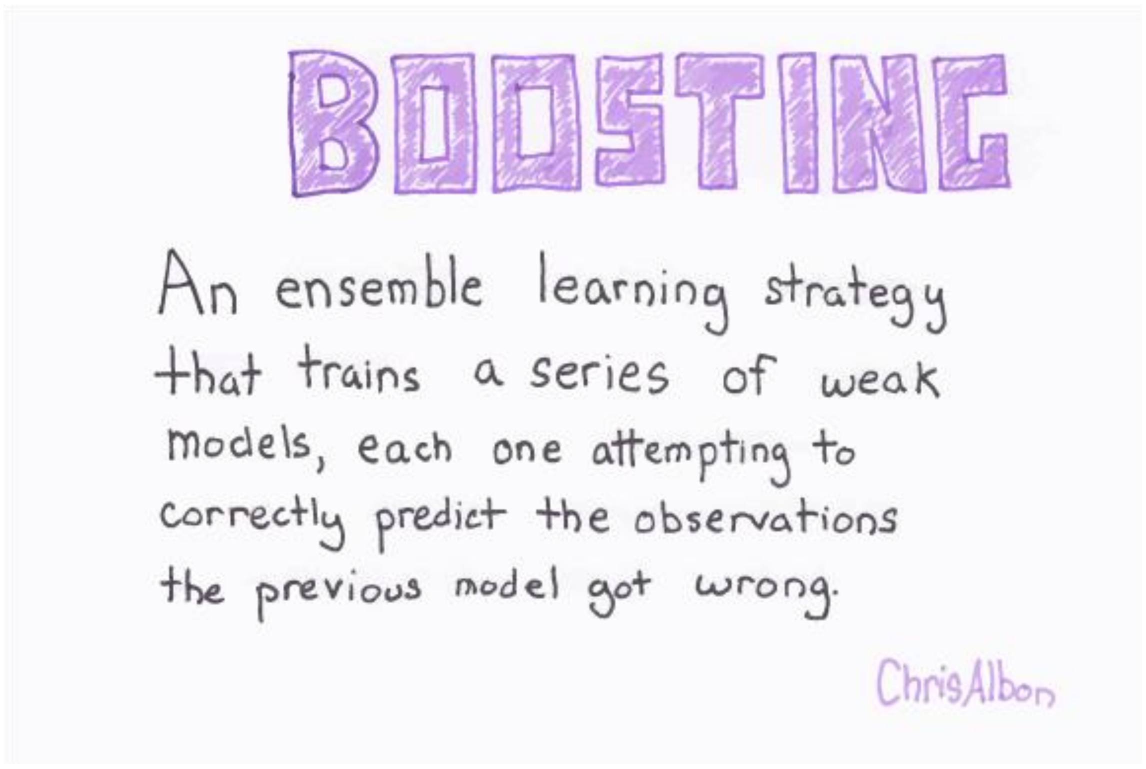
The general idea is that a combination of learning models increases the overall result selected.


RANDOM FOREST
CLASSIFICATION
1) 1. Many trees are created using random subsets of features and bootstrapped data.
2. Each tree votes by predicting target class.
3. Votes are tallied to reach the final prediction.
CHRIS ALBON

Deep decision trees may suffer from overfitting, but random forests prevent overfitting by creating trees on random subsets. The main reason is that it takes the average of all the predictions, which cancels out the biases.Random forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

2. GRADIENT BOOSTING CLASSIFICATION

Gradient boosting classifier is a boosting ensemble method. Boosting is a way to combine (ensemble) weak learners, primarily to reduce prediction bias. Instead of creating a pool of predictors, as in bagging, boosting produces a cascade of them, where each output is the input for the following learner. Typically, in a bagging algorithm trees are grown in parallel to get the average prediction across all trees, where each tree is built on a sample of original data. Gradient boosting, on the other hand, takes a sequential approach to obtaining predictions instead of parallelizing the tree building process. In gradient boosting, each decision tree predicts the error of the previous decision tree—thereby *boosting* (improving) the error (gradient).



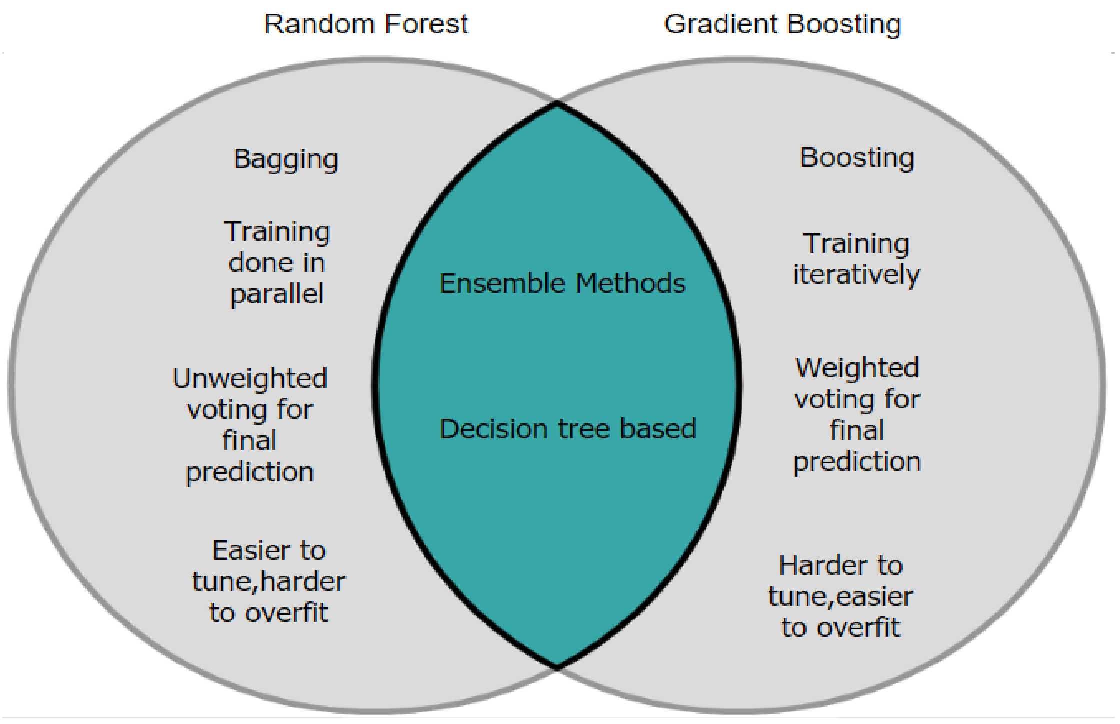
BOOSTING

An ensemble learning strategy that trains a series of weak models, each one attempting to correctly predict the observations the previous model got wrong.

ChrisAlbon

Working of Gradient Boosting

1. Initialize predictions with a simple decision tree.
2. Calculate residual (actual-prediction) value.
3. Build another shallow decision tree that predicts residual based on all the independent values.
4. Update the original prediction with the new prediction multiplied by learning rate.
5. Repeat steps two through four for a certain number of iterations (the number of iterations will be the number of trees).


Random Forest
Gradient Boosting
Bagging
Training done in parallel
Unweighted voting for final prediction
Easier to tune,harder to overfit
Ensemble Methods
Decision tree based
Boosting
Training iteratively
Weighted voting for final prediction
Harder to tune,easier to overfit

Difference between RF & GB

Unsupervised classification is where the outcomes (groupings of pixels with common characteristics) are based on the software analysis of an image without the user providing sample classes. The computer uses techniques to determine which pixels are related and groups them into classes. The user can specify which algorism the software will use and the desired number of output classes but otherwise does not aid in the classification process. However, the user must have knowledge of the area being classified when the groupings of pixels with common characteristics produced by the computer have to be related to actual features on the ground (such as wetlands, developed areas, coniferous forests, etc.).

Python libraries suitable for Machine Learning.

Python machine learning libraries have grown to become the most preferred language for machine learning algorithm implementations. Let's have a look at the main Python libraries used for machine learning.

Top Python Machine Learning Libraries
1) NumPy
NumPy is a well known general-purpose array-processing package. An extensive collection of high complexity mathematical functions make NumPy powerful to process large multi-dimensional arrays and matrices. NumPy is very useful for handling linear algebra, Fourier transforms, and random numbers. Other libraries like TensorFlow uses NumPy at the backend for manipulating tensors.
With NumPy, you can define arbitrary data types and easily integrate with most databases. NumPy can also serve as an efficient multi-dimensional container for any generic data that is in any datatype. The key features of NumPy include powerful N-dimensional array object, broadcasting functions, and out-of-box tools to integrate C/C++ and Fortran code.
2) SciPy
With machine learning growing at supersonic speed, many Python developers were creating **python libraries for machine learning**, especially for scientific and analytical computing. Travis Oliphant, Eric Jones, and Pearu Peterson in 2001 decided to merge most of these bits and pieces codes and standardize it. The resulting library was then named as SciPy library.
The current development of the SciPy library is supported and sponsored by an open community of developers and distributed under the free BSD license.
The SciPy library offers modules for linear algebra, image optimization, integration interpolation, special functions, Fast Fourier transform, signal and image processing, Ordinary Differential Equation (ODE) solving, and other computational tasks in science and analytics.
The underlying data structure used by SciPy is a multi-dimensional array provided by the NumPy module. SciPy depends on NumPy for the array manipulation subroutines. The SciPy library was built to work with NumPy arrays along with providing user-friendly and efficient numerical functions.
3) Scikit-learn
In 2007, **David Cournapeau** developed the Scikit-learn library as part of the Google Summer of Code project. In 2010 INRIA involved and did the public release in January 2010. Skikit-learn was built on top of two Python libraries – NumPy and SciPy and has become the most popular Python machine learning library for developing machine learning algorithms.
Scikit-learn has a wide range of supervised and unsupervised learning algorithms that works on a consistent interface in Python. The library can also be used for data-mining and data analysis. The main machine learning functions that the Scikit-learn library can handle are classification, regression, clustering, dimensionality reduction, model selection, and preprocessing.
4) Theano
Theano is a **python machine learning library** that can act as an optimizing compiler for evaluating and manipulating mathematical expressions and matrix calculations. Built on

NumPy, Theano exhibits a tight integration with NumPy and has a very similar interface. Theano can work on Graphics Processing Unit (GPU) and CPU.
Working on GPU architecture yields faster results. Theano can perform data-intensive computations up to 140x faster on GPU than on a CPU. Theano can automatically avoid errors and bugs when dealing with logarithmic and exponential functions. Theano has built-in tools for unit-testing and validation, thereby avoiding bugs and problems.
5) TensorFlow
TensorFlow was developed for Google's internal use by the Google Brain team. Its first release came in November 2015 under Apache License 2.0. TensorFlow is a popular computational framework for creating **machine learning models**. TensorFlow supports a variety of different toolkits for constructing models at varying levels of abstraction.
TensorFlow exposes a very stable Python and C++ APIs. It can expose, backward compatible APIs for other languages too, but they might be unstable. TensorFlow has a flexible architecture with which it can run on a variety of computational platforms CPUs, GPUs, and TPUs. TPU stands for Tensor processing unit, a hardware chip built around TensorFlow for machine learning and artificial intelligence.
6) Keras
Keras has over 200,000 users as of November 2017. Keras is an open-source library used for neural networks and machine learning. Keras can run on top of TensorFlow, Theano, Microsoft Cognitive Toolkit, R, or PlaidML. Keras also can run efficiently on CPU and GPU.
Keras works with neural-network building blocks like layers, objectives, activation functions, and optimizers. Keras also have a bunch of features to work on images and text images that comes handy when writing Deep Neural Network code.
Apart from the standard neural network, Keras supports convolutional and recurrent neural networks.
7) PyTorch
PyTorch has a range of tools and libraries that support computer vision, machine learning, and natural language processing. The PyTorch library is open-source and is based on the Torch library. The most significant advantage of PyTorch library is it's ease of learning and using.
PyTorch can smoothly integrate with the python data science stack, including NumPy. You will hardly make out a difference between NumPy and PyTorch. PyTorch also allows developers to perform computations on Tensors. PyTorch has a robust framework to build computational graphs on the go and even change them in runtime. Other advantages of PyTorch include multi GPU support, simplified preprocessors, and custom data loaders.
8) Pandas
Pandas are turning up to be the most popular Python library that is used for data analysis with support for fast, flexible, and expressive data structures designed to work on both "relational" or "labeled" data. Pandas today is an inevitable library for solving practical, real-world data analysis in Python. Pandas is highly stable, providing highly optimized performance. The backend code is purely written in C or Python.
The two main types of data structures used by pandas are :
Series (1-dimensional)
DataFrame (2-dimensional)
These two put together can handle a vast majority of data requirements and use cases from most sectors like science, statistics, social, finance, and of course, analytics and other areas of engineering.
Pandas support and perform well with different kinds of data including the below :

Tabular data with columns of heterogeneous data. For instance, consider the data coming from the SQL table or Excel spreadsheet.
Ordered and unordered time series data. The frequency of time series need not be fixed, unlike other libraries and tools. Pandas is exceptionally robust in handling uneven time-series data
Arbitrary matrix data with the homogeneous or heterogeneous type of data in the rows and columns

Any other form of statistical or observational data sets. The data need not be labeled at all. Pandas data structure can process it even without labeling.

9) Matplotlib

Matplotlib is a data visualization library that is used for 2D plotting to produce publication-quality image plots and figures in a variety of formats. The library helps to generate histograms, plots, error charts, scatter plots, bar charts with just a few lines of code.

It provides a MATLAB-like interface and is exceptionally user-friendly. It works by using standard GUI toolkits like GTK+, wxPython, Tkinter, or Qt to provide an object-oriented API that helps programmers to embed graphs and plots into their applications.

TEXT / REFERENCE BOOKS

1. Chris Albon : Machine Learning with Python Cookbook , O"Reilly Media, Inc.2018
2. Stephen Marsland, “Machine Learning – An Algorithmic Perspective”, Second Edition, Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014
3. Tom M. Mitchell, Machine Learning, India Edition 2013, McGraw Hill Education
4. Machine Learning: The art and Science of algorithms that make sense of data, Peter Flach, Cambridge University Press, 2012
5. EthemAlpaydın, Introduction to machine learning, second edition, MIT press.
6. T. Hastie, R. Tibshirani and J. Friedman, “Elements of Statistical Learning”, Springer Series , 2nd edition
7. Sebastian Raschka, "Python Machine Learning",Second Edition.Packt Publication

UNIT II CLASSIFIERS

Classification, K- nearest neighbour, Decision Trees, Implementing Decision Tree, building a Tree, Random Forests - Working of Random Forest, Pros and Cons of Random Forest, Naiver Bayes, building model Using Naiver Bayes.

KNN:

In statistics, the k-nearest neighbors algorithm (k-NN) is a non parametric classification method first developed by Evelyn Fix and Joseph Hodges in 1951 and later expanded by Thomas Cover.

It is used for classification and regression. In both cases, the input consists of the k closest training examples in data set. The output depends on whether k-NN is used for classification or regression.

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm.

It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data.

Working of KNN Algorithm

K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following −

3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 − Now, based on the distance value, sort them in ascending order.

3.3 − Next, it will choose the top K rows from the sorted array.